Netkiller Java 手札

陈景峰 著



Netkiller Java 手札（版）

目录

自述

1. 写给读者
2. 作者简介
3. 如何获得文档
4. 打赏（Donations）
5. 联系方式

1. Java
    1. JVM
        1.1. Almalinux / RockyLinux
        1.2. CentOS 8 Java 14
        1.3. Java 版本切换
        1.4. 安装 Java 6
            HeapDumpOnOutOfMemoryError
        1.5. java-1.8.0-openjdk
        1.6. docker 环境
        1.7. java - Launches a Java application.
            java 9~11
            -verbose:class 显示载入jar文件
            java.io.tmpdir
            显示版本号
            列出java模块
        1.8. jar
        1.9. jdeps - Java class dependency analyzer.
        1.10. JShell
            /help 显示帮助信息
            退出命令
        1.11. jlink
    2. Java 相关命令
        2.1. jps
    3. System

    3.1. user.dir
    3.2. java.io.tmpdir
    3.3. 打印当前 Java 文件的默认编码
    3.4. 自定义
    3.5. System.in 标准输入(Stdin)
4. exec 运行shell
5. 数据类型
    5.1. var 本地变量类型推断
    5.2. Integer
        前面补零
    5.3. String
        查找字符重现的位置
        行数统计
        复制字符串
        随机字符串
        字符串替换处理
            正则表达式查找与替换
            利用正则快速转换时间格式
            replaceAll
            字符串处理，删除中文以外的字符
            取出字符串中的中文字符
        substring
        string to timestamp
        String.strip
        文本块
        分割字符串
    5.4. 类型转换
        Long to String
    5.5. Date
        SimpleDateFormat
        Timestamp
        TimeZone
        String to Date
        比较两个日期与时间
        Calendar
        getToday

Yesterday
ISO 8601
LocalDateTime
ZonedDateTime
5.6. Array
for each
Array to String

5.7. float
5.8. double
String to double
百分数转Double
Double转百分数
5.9. BigDecimal
Convert BigDecimal Object to double value
去除末尾多余的0
禁用科学计数法
移动小数点位置
5.10. StringBuffer
5.11. enum
5.12. byte 类型
string2byte
byte[] to String
BigInteger2byte
int to byte array
byte array to int
byte2char
longToByte64
byte64ToLong
short2byte
byte8ToDouble
byte4ToFloat
无符号 byte
byte to hex
byte[] to hex
连接两个 byte[]

List<Byte> to byte[]
5.13. 布尔型 Boolean
6. 流程控制
6.1. Switch
yield
7. 异常处理
7.1. 抛出异常
8. 数据结构
8.1. List
静态 List
List.of()
List.copyOf()
String[] to List
Stream.toList() 方法
containsAll
stream().allMatch()
随机 List
8.2. ArrayList
初始化
判断元素是否存在
循环打印
ArrayList to Array
ArrayList to String
Array to List
List to Array
ArrayList forEach
ArrayList stream()
ArrarList 转换为 string[]
string 转换为 ArrayList
ArrayList 转换为 string
string[] 转换为 ArrarList
合并 List<byte[]> ArrayList<byte[]>
8.3. Set 转为 List
Set.of()
Set to Array
8.4. Map

初始化
    static map
    Collections 初始化 Map
    使用 Collectors.toMap（）初始化 Map
    使用Map.Entry流初始化 Map
    Map.of()
    Map.ofEntries()
HashMap
    遍历 HashMap
    遍历map中的键
    遍历map中的值
    通过键取值
    使用 Iterator 遍历 HashMap
LinkedHashMap
遍历数据
迭代器
Map forEach
随机取值

8.5. Queue

阻塞队列
    LinkedBlockingQueue
Deque 双端队列
    LinkedList
数据转换

8.6. Stream

Stream.of
Stream.ofNullable
filter
map
limit/skip
sorted
distinct
forEach
count
collect
takeWhile 和 dropWhile

List to Stream
混合使用的例子
Collectors.teeing()
8.7. Optional
of() 为非null的值创建一个Optional。
ofNullable() 为指定的值创建一个Optional，如果指定的值为null，则返回一个空的Optional。
isPresent 如果值存在返回true，否则返回false。
ifPresent() 如果Optional实例有值执行 lambda 表达式
get() 返回值
orElse 如果有值则将其返回，否则返回指定的其它值。
orElseGet与orElse方法类似，区别在于得到的默认值从 Supplier 返回。
orElseThrow 如果有值则将其返回，否则抛出supplier接口创建的异常
map() 方法用来对Optional实例的值执行一系列操作
flatMap()
filter() 通过传入限定条件过滤Optional值
stream()
or()
example
Optional 与 Map
判断 Object 是否为 null
9. Network
9.1. URL
9.2. 获取IP地址何机器名
10. JDBC
10.1. 安装 JDBC 包
10.2. MySQL
10.3. Oracle
SID
SERVICE_NAME
TNS

Oracle RAC Cluster
Oracle JDBC Demo

10.4. FAQ

java.sql.SQLRecoverableException: IO Error: The Network Adapter could not establish the connection
Exception in thread "main" java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver

11. Util

11.1. Properties 处理 *.properties 文件

打开 properties 文件

文件方式打开
输入流

propertyNames()
keySet()
entrySet()
方法中返回 Properties

getResourceAsStream()
store
实现国际化

11.2. Logging

console

11.3. BASE64
11.4. Locale 国际化
11.5. ResourceBundle
11.6. Scanner
11.7. UUID
11.8. Arrays.equals 判断两个数组是否相等
11.9. Random 随机字符串

取 0-n 范围内随机数
指定随机数范围

11.10. ArrayBlockingQueue
11.11. CRC32

- 11.12. 正则表达式
  - 正则查找
  - 正则替换
  - 字符串分割
- 11.13. java.util.concurrent
  - TimeUnit
  - FutureTask
  - CompletableFuture
    - runAsync 创建没有返回值的异步任务
    - supplyAsync 创建带有返回值的异步任务。
    - thenRun / thenRunAsync
    - thenAccept / thenAcceptAsync
    - thenApply / thenApplyAsync
    - runAsync / thenAccept / thenApply 区别
    - whenComplete
    - 超时处理
    - 按顺序执行
    - 获取结果
    - 异常处理
- 11.14. java.util.stream
  - IntStream

12. IO
- 12.1. 取出文件名中的扩展名
  - getAbsolutePath() 获取绝对路径
  - 创建目录 mkdir()
- 12.2. 临时文件
- 12.3. FileWriter 文本写入文件
- 12.4. BufferedWriter
- 12.5. inputStream.transferTo()
- 12.6. InputStreamReader
- 12.7. 获得 Resource 下文件路径
- 12.8. PrintWriter
- 12.9. OutputStreamWriter
- 12.10. FileOutputStream
- 12.11. FileInputStream
- 12.12. Scanner

12.13. 二进制文件
理解二进制文件
byte 类型
boolean 布尔型
Long 型
chat 类型
UTF 字符串
Short 类型
float 单精度浮点类型
double 数据类型
二进制文件操作演示
所有类型演示一遍
检查文件是否是 png 文件
13. Reflection 反射
13.1. 获得所有变量
13.2. 批量赋值
13.3. 方法操作
获得所有方法
set/get 方法
static 方法调用
14. Java 线程
14.1. 多线程 Lambda 表达式
14.2. 实现异步执行
14.3. 继承 Thread 类实现多线程
设置线程名称
14.4. 实现 Runnable 接口
14.5. 线程同步
14.6. java 线程池
newCachedThreadPool
固定线程池(newFixedThreadPool)
Executors.newScheduledThreadPool
14.7. ThreadLocal
14.8. InheritableThreadLocal
14.9. Java 协程
15. java 脚本引擎
15.1. Maven

15.2. Helloworld
15.3. 运行脚本文件
15.4. 变量传递
15.5. 全局变量与局部变量定义
15.6. 调用脚本中的函数或方法
15.7. 脚本编译
15.8. jjs - Invokes the Nashorn engine.
16. Crypto
16.1. MD5
16.2. AES
16.3. AES/CBC/PKCS5PADDING
16.4. DES
17. java.security
17.1. 列出 Java 支持的数字摘要算法
17.2. 计算文件的 MD5，SHA 等 HASH 值
2. Build Tools
1. Apache Ant
1.1. 安装 ant
1.8
1.10.1
1.2. ANT
ant.project.name
定义
1.3. Project
property
ant
environment
1.4. path
1.5. copy
1.6. javac
1.7. condition
1.8. exec
sshexec
1.9. if
1.10. macrodef
Git

Rsync
SSH
maven
1.11. Javascript
1.12. mail
1.13. basename
1.14. 创建文件
1.15. FAQ
warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last; set to false for repeatable builds
调试 exec
2. Apache Ivy
2.1. Ivy Install
source code
apt-get
2.2. Test example
3. Apache Maven
3.1. 安装 Maven
CentOS 8 安装 Maven
Ubuntu
一键安装
apache-maven-3.8.2
mvnd
3.2. Maven 命令
切换 JAVA 版本
参数
-s 指定 settings.xml 文件
多线程
help
archetype:create
clean
compile
多线程编译
编译测试代码
test

- package
- install
  - install-file
- war
- exec
- dependency
  - build-classpath
  - dependency:tree 显示包依赖树
  - copy-dependencies 导出依赖包
  - analyze 查看未被使用的包
  - sources 下载源码
- jar
- 构建装配Maven Assembly
- 加密密码
- help:describe

3.3. settings.xml 配置

- Maven 仓库
- 镜像配置

3.4. pom.xml

- properties
  - java.version
- 常用的POM属性
- repositories 仓库配置
  - 默认仓库
  - 阿里云仓库
- dependencies
- dependencyManagement
- build
  - finalName
  - sourceDirectory
  - resources 文件处理
- plugins
  - 跳过Unit test
  - maven-shade-plugin

3.5. Maven Module

- Parent

        公共项目 common
        常规项目
        现在测试效果
    3.6. 依赖管理
        创建依赖模块
        引用依赖管理
    3.7. plugins
        maven-compiler-plugin
        maven-war-plugin
        maven-antrun-plugin
            echo 打印调试信息
        maven-install-plugin
        maven-surefire-plugin
        maven-deploy-plugin
            deploy:deploy-file
        maven-jar-plugin
        maven-dependency-plugin
        spring-boot-maven-plugin
        tomcat8-maven-plugin
        docker-maven-plugin
    3.8. 应用案例
        并行开发解决不同环境包引用
4. Gradle 5
    4.1. 安装 Gradle
        CentOS
        Mac
    4.2. Example
    4.3. gradle 命令
        tasks 列出任务
    4.4. build.gradle
        repositories
        dependencies
        jar
        Task
    4.5. gradle.properties
        列出 properties

自定义 gradle.properties

System.properties

5. JitPack - Easy to use package repository for Git
6. Artifactory
    6.1. Artifactory Web UI
    6.2. build.gradle

3. Servlet
    1. Example
    2. Session
    3. HttpServletRequest
    4. Filter
        4.1. web.xml
        4.2. Filter 类
    5. Listener
        5.1. web.xml
        5.2. NewsListener 类
        5.3. NewsTask 类
        5.4. JSP 中心显示
    6. JSP
        6.1. 注释
        6.2. pageContext
            queryString
        6.3. request
            Form
        6.4.
            sendRedirect
        6.5. cookie
        6.6. session
        6.7. page
            Session
        6.8. trimDirectiveWhitespaces
        6.9. include
        6.10. jsp
            jsp:forward
        6.11. error-page

6.12. JSP 编程
随机数
6.13. FAQ
http://www.netkiller.cn/test.html;jsessionid=7D25CE666FF437F2094AA945E97CEB37
7. JSTL(JavaServer Pages Standard Tag Library)
7.1. c:set
c:remove
7.2. c:out
7.3. c:url
7.4. c:redirect
7.5. c:import
7.6. c:if
boolean
7.7. c:choose
7.8. c:forEach
List 处理
Map 处理
7.9. empty 判断是否为空
7.10. JSTL fmt Tag setBundle Example
fmt:message
Language Package
fmt:message 变量

8. WebSocket
8.1. Server
8.2. Client
I. Spring Framework
4. Spring 开发环境
1. Java 开发环境
2. 安装 Spring Tool Suite
3. Dashboard
4. Spring Initializr - Bootstrap your application
5. Spring Boot
1. Spring Boot Quick start
1.1. 创建项目

    1.2. pom.xml
    1.3. Controller
2. Springboot with Maven
    2.1. resource
    2.2. Maven run
    2.3. Spring Boot maven 插件 build-image
    2.4. 生成项目信息
3. SpringApplication
    3.1. 运行 Spring boot 项目
        Linux systemd
        传统 init.d 脚本
        编译用于Tomcat的 War
    3.2. @SpringBootApplication
        排除 @EnableAutoConfiguration 加载项
    3.3. 获取 Resources 目录中的静态文件
    3.4. @EnableAutoConfiguration
    3.5. @ComponentScan
    3.6. @EntityScan 实体扫描
    3.7. @EnableJpaRepositories
    3.8. 启动和销毁
    3.9. 打印环境变量
    3.10. CharacterEncodingFilter
    3.11. 隐藏 Banner
    3.12. 实体与仓库扫描
    3.13. 列出 Beans
    3.14. Tomcat 端口
    3.15. 配置项设定
    3.16. spring.profiles.active
    3.17. @Profile("dev") / @ActiveProfiles("dev")
    3.18. 设置默认时区
4. 如何优雅停止 Springboot 运行
    4.1. 准备工作
    4.2. kill 命令演示
    4.3. 容器中如何优雅关闭 Springboot
    4.4. 写入PID文件
5. Properties 配置文件

5.1. application.properties 配置文件

application.properties 参考

启动指定参数

--spring.config.location 指定配置文件

--spring.profiles.active 切换配置文件

加载排除

PID FILE

banner 关闭

server

优雅关闭

端口配置

Session 配置

cookie

error 路径

压缩传输

ssl

logging

内嵌 tomcat server

server.tomcat.basedir

access.log

charset

servlet

上传限制

server.servlet.context-path

JSON 输出与日期格式化

SMTP 相关配置

Redis

MongoDB

MySQL

Oracle

default_schema

datasource

velocity

Security 相关配置

MVC 配置

Kafka 相关配置
5.2. Properties 文件
禁用命令行注入环境变量
@Value 注解
@EnableConfigurationProperties 引用自定义 *.properties 配置文件
手工载入 *.properties 文件
spring.profiles.active 参数切换配置文件
SpringApplicationBuilder.properties() 方法添加配置项
5.3. 参数引用
5.4. 默认值
5.5. 产生随机数
随机数
5.6. 多行字符串
5.7. 注入多值属性 arrays, list, set
5.8. containsProperty 读取配置文件
5.9. @PropertySource 注解载入 properties 文件
5.10. List 列表类型
5.11. Map类型
5.12. Binder
5.13. 加密 application.properties 中的敏感内容
6. Spring boot with Logging
6.1. 配置日志文件
日志输出级别
Spring boot 2.1 以后的版本不打印 Mapped 日志问题
禁止控制台输出日志
定制日志格式
彩色输出
6.2. 打印日志
lombok
6.3. logback 配置详解
标准输出
禁止 logback 日志输出
指定Class过滤日志

configuration 属性配置
contextName 设置上下文名称
property 设置变量
encoder 日志格式设置
RollingFileAppender
    按日期分割日志
    按照文件尺寸分割日志
日志过滤
标准输出
MDC
日志写入 MongoDB
日志发送给 logstash
    logstash 配置
    Java 项目
    通过 tags 区分日志文件
        logstash pipeline 配置
        logback-spring.xml 配置
        打印日志
fluentd
    Maven 依赖
    安装 fluent-bit
    配置 logback-spring.xml
    查看 fluent 输出
Loki4j Logback
    Maven
    logback.xml

6.4. Log4j2 + Gelf + Logstash

Maven 配置
log4j2.xml 配置
Java 测试代码
Logstash 配置
测试结果
Log4j2 更多技巧
    多环境配置
    控制 class 日志输出级别
    日志输出级别

读取系统变量/环境变量
读取 spring 配置
变量默认值

6.5. 日志报警
Logstash 配置
监控 SpringBootApplication 的启动和退出
6.6. Spring boot with ELK(Elasticsearch + Logstash + Kibana)
TCP 方案
Redis 方案
Kafka 方案
Other
7. Undertow
7.1. Maven 依赖
7.2. Application
7.3. 相关配置
8. Spring boot with Jetty
9. Spring boot with HTTP2 SSL
9.1. 生成自签名证书
9.2. application.properties 配置文件
9.3. 启动 Spring boot
9.4. restTemplate 调用实例
9.5. HTTP2
10. Spring boot with Webpage
10.1. Maven
10.2. application.properties
10.3. Application
10.4. IndexController
10.5. src/main/webapp/WEB-INF/jsp/index.jsp
10.6. 集成模板引擎
11. Spring boot with Velocity template
11.1. Maven
11.2. Resource
11.3. Application
11.4. RestController

    11.5. Test
12. Spring boot with Thymeleaf
    12.1. Maven
    12.2. application.properties
    12.3. Controller
    12.4. HTML5 Template
13. Spring boot with MongoDB
    13.1. Maven
    13.2. Application
    13.3. MongoTemplate
    13.4. Repository
14. Spring boot with MySQL
    14.1. Maven
    14.2. Resource
    14.3. Application
    14.4. JdbcTemplate
    14.5. CrudRepository
15. Spring boot with Oracle
    15.1. Maven
    15.2. application.properties
    15.3. Application
    15.4. CrudRepository
    15.5. JdbcTemplate
    15.6. Controller
16. Spring boot with PostgreSQL
    16.1. pom.xml
    16.2. application.properties
    16.3. Application
    16.4. CrudRepository
    16.5. JdbcTemplate
    16.6. Controller
    16.7. Test
17. Spring boot with Elasticsearch
    17.1. Maven
    17.2. Application
    17.3. application.properties

17.4. Domain
17.5. ElasticsearchRepository
18. Spring boot with Elasticsearch TransportClient
18.1. Maven
18.2. Application
18.3. application.properties
18.4. ElasticsearchConfiguration
18.5. RestController
19. Spring boot with Apache Hive
19.1. Maven
19.2. application.properties
19.3. Configuration
19.4. CURD 操作实例
20. Spring boot with Phoenix
20.1. Maven
20.2. application.properties
20.3. Configuration
21. Spring boot with Datasource
21.1. Master / Slave 主从数据库数据源配置
application.properties
配置主从数据源
选择数据源
21.2. 多数据源配置
21.3. JPA 多数据源
22. 连接池配置
22.1. org.apache.tomcat.jdbc.pool.DataSource
22.2. druid
加密数据库密码
22.3. c3p0 - JDBC3 Connection and Statement Pooling
22.4. dbcp2
22.5. bonecp
22.6. HikariPool
23. Spring boot with Redis
23.1. Spring boot with Redis
maven

application.properties
JUnit
Controller
23.2.
获取过期时间
列表操作
23.3. Redis Pub/Sub
Redis配置类
订阅和发布类
消息发布演示
23.4. Spring Redis Lock
Maven 依赖
配置锁
使用方法
23.5. Sprint boot with Redisson
Springboot 3.x
Springboot 2.1
24. Spring boot with RabbitMQ(AMQP)
24.1. maven
24.2. RabbitMQConfig
24.3. 生产者
24.4. 消费者
25. Spring boot with Apache Kafka
25.1. 安装 kafka
25.2. maven
25.3. Spring boot Application
25.4. EnableKafka
25.5. KafkaListener
25.6. 测试
25.7. 完整的发布订阅实例
Consumer
Producer
Test
25.8. Spring cloud with Kafka
Application 主文件
资源配置文件

application.properties
bootstrap.properties
Git 仓库
启用 kafka
消息发布主程序
26. Spring boot with Scheduling
26.1. Application.java
26.2. Component
26.3. 查看日志
26.4. 计划任务控制开关
26.5. @Scheduled 详解
每3秒钟一运行一次
凌晨23点运行
26.6. Timer 例子
26.7. ScheduledExecutorService 例子
27. Spring boot with Swagger
27.1. Swagger3
27.2. Swagger2
Maven 文件
SpringApplication
EnableSwagger2
RestController
27.3. @Api()
27.4. @ApiOperation()
27.5. @ApiResponses
27.6. @ApiModel 实体类
28. Spring boot with knife4j
28.1. maven
28.2. Knife4jConfiguration
28.3. application.properties
29. Spring boot with lombok
29.1. @Builder
29.2. @Slf4j 注解
30. Spring boot with Docker
30.1. 通过 Docker 命令构建镜像
手工编译镜像

Dockerfile 放在 src/main/docker/Dockerfile 下
通过参数指定 Springboot 文件
SPRING_PROFILES_ACTIVE 指定配置文件
推送镜像到仓库
30.2. 通过 Maven 构建 Docker 镜像
Maven + Dockerfile 方案一
Maven + Dockerfile 方案二
Maven 不使用 Dockerfile 文件
推送镜像
30.3. [ERROR] No plugin found for prefix 'dockerfile' in the current project and in the plugin groups [org.apache.maven.plugins, org.codehaus.mojo] available from the repositories [local (/Users/neo/.m2/repository), central (https://repo.maven.apache.org/maven2)] -> [Help 1]
30.4. curl: (35) LibreSSL SSL_connect: SSL_ERROR_SYSCALL in connection to localhost:8888
31. Spring boot with Docker stack
31.1. 编译 Docker 镜像
31.2.
32. Spring boot with Kubernetes
32.1. Kubernetes 编排脚本
32.2. 部署镜像
33. Spring boot with command line
33.1. Maven
33.2. CommandLineRunner 例子
33.3. ApplicationRunner 例子
34. Spring Boot Actuator
34.1. Maven 依赖
34.2. 与 Spring Boot Actuator 有关的配置
禁用HTTP端点
安全配置
修改 actuator 地址
关机

34.3. actuator 接口
34.4. 健康状态
健康状态
34.5. info 配置信息
34.6. beans 信息
34.7. caches
34.8. conditions
34.9. configprops 配置文件
34.10. env 环境变量
34.11. logfile 日志
34.12. threaddump 线程信息
34.13. 计划任务
34.14. metrics
34.15. 控制器映射 URL
34.16. 自定义监控指标
35. String boot with RestTemplate
35.1. RestTemplate Example
pom.xml
web.xml
springframework.xml
RestController
POJO
在控制器中完整实例
测试
35.2. GET 操作
返回字符串
传递 GET 参数
35.3. POST 操作
postForObject
传递对象
传递数据结构 MultiValueMap
postForEntity
35.4. PUT 操作
35.5. Delete 操作
35.6. 上传文件
35.7. HTTP Auth

Client
35.8. PKCS12
35.9. Timeout 超时设置
JRE 启动参数设置超时时间
RestTemplate timeout with SimpleClientHttpRequestFactory
@Configuration 方式
36. SpringBootTest
36.1. Maven 依赖
36.2. 测试类
Junit基本注解介绍
36.3.
Assert.assertEquals 判断相等
Assert.assertTrue
36.4. JPA 测试
36.5. TestRestTemplate
36.6. Controller单元测试
36.7. WebTestClient
37. Spring boot with Aop
37.1. Aspect
Maven
Pojo 类
Service 类
Aspect 类
控制器
Application
测试
38. Spring boot with starter
38.1. 实现 starter
Maven pom.xml 依赖包
配置文件处理
自动配置文件
启用 starter 的自定义注解
38.2. 引用 starter
Maven pom.xml 引入依赖
通过注解配置 starter

测试运行结果
39. SpringBoot Admin
    39.1. 依赖
    39.2. 启用 Springboot Admin
    39.3. Nginx 跨域
40. Spring boot with Grafana
    40.1. Springboot 集成 InfluxDB
    40.2. InfluxDB
41. Spring Boot with Prometheus
    41.1. Maven 依赖
    41.2. application.properties 配置文件
    41.3. 启动类
    41.4. 测试
    41.5. 控制器监控
    41.6. 自定义埋点监控
        拦截器
        计数器元件
        配置类
        测试埋点效果
42. Spring boot with Git version
    42.1. CommonRestController 公共控制器
    42.2. VersionRestController 测试控制器
    42.3. 创建 .gitattributes 文件
43. Spring boot with Session share
    43.1. Redis
        Maven
        application.properties
        Application
    43.2. 测试 Session
    43.3. JDBC
    43.4. Springboot 2.1
44. Spring boot with Caching
    44.1. maven
        Redis
    44.2. 启用 Cache
    44.3. @Cacheable 的用法

SpEL表达式
排除 null 结果
44.4. @CachePut 用法
44.5. 清空缓存
44.6. @Caching
44.7. 解决Expire 和 TTL 过期时间
45. Spring boot with Email
45.1. Maven
45.2. Resource
45.3. POJO
45.4. RestController
45.5. Test
46. Spring boot with Hessian
46.1. Maven
46.2. Application
46.3. HessianServiceExporter
46.4. Service
46.5. RestController
47. Spring boot with Async
47.1. Callable 实现异步
47.2. WebAsyncTask 实现异步
47.3. DeferredResult 实现异步访问
47.4. SimpleAsyncTaskExecutor
配置线程池
47.5. ThreadPoolTaskExecutor 自定义线程池
最简单的配置
队列
47.6. 定义多个线程池
47.7. 自定义线程池
ThreadPoolExecutor
注入自定义线程池bean
47.8. 设置线程名称
47.9. 线程池监控
48. Springboot with Ethereum (web3j)
48.1. Maven
48.2. application.properties

48.3. TestRestController
48.4. 测试
49. Java Record 新特性
49.1. Record 替代 POJO 类
49.2. Record 作为 Properties
49.3. Record 作为实体类
49.4. Record 作为 Service
49.5. Record 作为 Controller
50. Springboot 接入阿里云
50.1. 阿里云 OSS - STS进行临时授权访问获取 HTTPS 地址
6. Spring MVC
1. @EnableWebMvc
1.1. CORS 跨域请求
1.2. Spring MVC CORS with WebMvcConfigurerAdapter
2. @Controller
2.1. @RequestMapping
@RequestMapping("/")
映射多个URL
匹配通配符
headers
2.2. @GetMapping
2.3. @PostMapping
2.4. @RequestBody
原始数据
@RequestBody 传递 List
传递 Map 数据
获取 JSONObject 数据
2.5. RequestMapping with Request Parameters - @RequestParam
HTTP GET
HTTP POST
@RequestParam 传递特殊字符串
传递日期参数
上传文件

@RequestParam - POST 数组
默认值
是否非必须
用 Map 接收 From 数据

2.6. @RequestHeader - 获取 HTTP Header 信息
@RequestHeader 从 Http 头中获取变量

2.7. RequestMapping with Path Variables - @PathVariable
URL 参数传递
默认值
URL 传递 Date 类型
处理特殊字符
@PathVariable 注意事项

2.8. @ModelAttribute

2.9. @ResponseBody
直接返回HTML

2.10. @ResponseStatus 设置 HTTP 状态

2.11. @CrossOrigin
maxAge

2.12. @CookieValue - 获取 Cookie 值

2.13. @SessionAttributes

2.14. ModelAndView
变量传递
ModelMap 传递多个变量
redirect
ArrayList
HashMap
传递对象

2.15. HttpServletRequest / HttpServletResponse
HttpServletResponse
HttpServletRequest

3. @RestController
3.1. 上传文件
3.2. 返回实体
3.3. JSON
3.4. 处理原始 RAW JSON 数据

- 3.5. 返回 JSON 对象 NULL 专为 "" 字符串
- 3.6. XML
- 3.7. 兼容传统 json 接口
- 3.8. 上传文件
- 3.9. Spring boot with csv
- 3.10. Problem Details [RFC 7807]
    - ResponseEntity
        - status
- 3.11. Jackson
    - Jackson 相关配置
    - @JsonIgnore 返回json是不含有该字段
    - @JsonFormat 格式化 json 时间格式
        - 日期格式化
        - 时区
        - 枚举
        - 枚举
    - @JsonComponent
    - Object to Json
    - Json To Object
- 3.12. synchronized
- 3.13. SSE

4. View
    - 4.1. 配置静态文件目录
    - 4.2. 添加静态文件目录
    - 4.3. Using Spring’s form tag library
        - css
            - cssClass
            - cssStyle
            - cssErrorClass
        - cssClass
    - 4.4. Thymeleaf
        - Maven pom.xml
        - Spring 配置
        - controller
        - HTML5 Template
        - thymeleaf 渲染表格

URL 链接
拆分字符串
日期格式化
4.5. FreeMarker
5. Service
5.1. Application
5.2. 定义接口
5.3. 实现接口
5.4. 调用 Service
5.5. context.getBean 调用 Service
5.6. AopContext
5.7. 变量共享
6. i18n 国际化
6.1. 在 appliction.properties 中配置启用 i18n
6.2. 创建语言包文件
6.3. 控制器重引用语言包
6.4. 参数传递
7. 校验器(Validator)
7.1. 常规用法
定义校验器
获取 BindingResult 结果
测试校验效果
7.2. 自定义注解
定义校验器注解接口
实现接口
注解用法
测试注解
8. Interceptor
8.1. Session 拦截
8.2. Token 拦截
8.3. 拦截器获取PathVariable变量
9. FAQ
9.1. o.s.web.servlet.PageNotFound
9.2. HTTP Status 500 - Handler processing failed; nested exception is

java.lang.NoClassDefFoundError: javax/servlet/jsp/jstl/core/Config
9.3. 同时使用 Thymeleaf 与 JSP
9.4. 排除静态内容
9.5. HTTP Status 406
9.6. Caused by: java.lang.IllegalArgumentException: Not a managed type: class common.domain.Article
9.7. {"error":"unauthorized","error_description":"Full authentication is required to access this resource"}
10. RestClient
10.1. 创建 RestClient
10.2. Get 操作
10.3. Post Json
10.4. HTTP Authorization Basic
10.5.
7. WebFlux framework
1. Getting Started
1.1. Maven
1.2. Application
1.3. RestController
1.4. 测试
2. WebFlux 与 SprintMVC 有什么不同?
2.1. 实验程序
2.2. 实验结果
3. WebFlux Router
3.1. Component 原件
3.2. 路由配置
3.3. Thymeleaf
模板引擎 Thymeleaf 依赖
application.properties 相关的配置
Webflux 控制器
Tymeleaf 视图
3.4. Webflux Redis
Maven Redis 依赖

Redis 配置
Config
Service

3.5. Webflux Mongdb
Maven 依赖
Repository
Service
控制器
3.6. Mono
3.7. Flux 返回多条数据
3.8. SSE
一次性事件
周期性事件
SSE 完整的例子
SSE Client 订阅实例
3.9. WebClient
配置 WebClient
@Controller/@RestController 实例
Get 请求实例
URI 参数
查询参数
Post 操作演示
Post 表单数据
上传文件
设置 HTTP 头
websocket
同步阻塞等待结果
4. Webflux 安全
4.1. Token 拦截器
4.2. JWT
4.3. spring-boot-starter-security
5. 常见问题
5.1. The Java/XML config for Spring MVC and Spring WebFlux cannot both be enabled, e.g. via

@EnableWebMvc and @EnableWebFlux, in the same application.
5.2. @EnableWebFluxSecurity 与 @EnableReactiveMethodSecurity 不生效
5.3. webflux netty 不支持 Content-Type: application/x-www-form-urlencoded

8. Spring Data
    1. Spring Data with Redis
        1.1. 集成 Redis XML 方式
            pom.xml
            springframework-servlet.xml
            Controller
            index.jsp
            测试
        1.2. RedisTemplate
            stringRedisTemplate 基本用法
            设置缓存时间
            字符串截取
            追加字符串
            设置键的字符串值并返回其旧值
            increment
            删除 key
            返回字符串长度
            如果key不存便缓存。
            缓存多个值 /获取多个值 multiSet / multiGet
            List
                rightPush
                rightPushAll
                rightPushIfPresent
                leftPush
                leftPushAll
                range
            SET 数据类型
                返回集合中的所有成员
                取出一个成员
                随机获取无序集合中的一个元素

随机获取 n 个成员（存在重复数据）
随机获取 n 个不重复成员
在两个 SET 间移动数据
成员删除
返回集合数量
判断元素是否在集合成员中
对比两个集合求交集
对比两个集合求交集，然后存储到新的 key 中
合并两个集合，并去处重复数据
合并两个集合去重复后保存到新的 key 中
计算两个合集的差集
计算两个合集的差集，然后保存到新的 key 中
遍历 SET 集合

有序的 set 集合

Hash

put
putAll
从键中的哈希获取给定hashKey的值
delete
确定哈希hashKey是否存在
从哈希中获取指定的多个 hashKey 的值
只有hashKey不存在时才能添加值
获取整个Hash
获取所有key
通过 hashKey 获取所有值
值加法操作
遍历 Hash 表

过期时间未执行

setBit / getBit 二进制位操作

存储 Json 对象

集成 RedisTemplate 定义新类
JsonRedisTemplate
配置 Redis
测试

1.3. Spring Data Redis - Repository Examples
@EnableRedisRepositories 启动 Redis 仓库
定义 Domain 类
Repository 接口
测试代码
2. Spring Data with MongoDB
2.1. Example Spring Data MongoDB
pom.xml
springframework-servlet.xml
POJO
Controller
查看测试结果
条件查询
2.2. MongoDB 多数据源
Maven
Application 禁止自动配置 MongoDB
application.properties 新增配置项
MongoDB 配置类
创建 Document 关系映射类
测试控制器
测试
2.3. @Document
指定表名
@Id
@Version
@Field 定义字段名
@Indexed
普通索引
唯一索引
索引排序方式
稀疏索引
索引过期时间设置
@CompoundIndex 复合索引
普通复合索引
唯一复合索引
@TextIndexed

@GeoSpatialIndex 地理位置索引
@Transient 丢弃数据，不存到 mongodb
@DBRef 做外外键引用
Article 类
Hypermedia 类
MongoRepository
RestController
运行结果
@DateTimeFormat
@NumberFormat
在 @Document 中使用 Enum 类型
在 @Document 中定义数据结构 List/Map
GeoJson 数据类型

2.4. MongoRepository

扫描仓库接口
findAll()
deleteAll()
save()
count()
exists() 判断是否存在
existsById()
findByXXXX
findAll with OrderBy
order by boolean 布尔型数据排序
findAll with Sort
FindAll with Pageable
PageRequest - springboot 1.x 旧版本
StartingWith 和 EndingWith
Between
Before / After
@Query

2.5. mongoTemplate

Save 保存
Insert
updateFirst 修改符合条件第一条记录
updateMulti 修改符合条件的所有

查找并保存
upsert - 修改符合条件时如果不存在则添加
删除
查找一条数据
查找所有数据
Query
    翻页
    between
Criteria
    is
    Regex 正则表达式搜索
    lt 和 gt
    exists()
    包含
Update
    set
    追加数据
    更新数据
    删除数据
    inc
    update.addToSet
BasicUpdate
Sort
Query + PageRequest
newAggregation
创建索引
子对象操作
    List 类型

2.6. GeoJson 反序列化

2.7. FAQ

location object expected, location array not in correct format; nested exception is com.mongodb.MongoWriteException: location object expected, location array not in correct format

3. Spring Data with MySQL

3.1. 选择数据库表引擎
3.2. 声明实体
    @Entity 声明实体
    @Table 定义表名
        catalog
        schema
        uniqueConstraints
    @Id 定义主键
    @Column 定义字段：
        字段长度
        浮点型
        创建于更新控制
        TEXT 类型
        整形数据类型
    非数据库字段
    @Lob 注解属性将被持久化为 Blog 或 Clob 类型
    @NotNull 不能为空声明
    @Temporal 日期定义
    创建日期
        CreatedDate
        与时间日期有关的 hibernate 注解
            设置默认时间
            创建时间
            更新时间
        数据库级别的默认创建日期时间定义
        数据库级别的默认创建日期与更新时间定义
        最后修改时间
    @DateTimeFormat 处理日期时间格式
    Enum 枚举数据类型
        实体中处理 enum 类型，存储字符串
        实体中处理 enum 类型，存储序号
        数据库枚举类型
        自定义枚举value属性
    SET 数据结构

JSON 数据类型
索引
普通索引
唯一索引
复合索引
嵌入
@Embeddable / @Embedded
@AttributeOverrides 定义字段名称
创建复合主键
外键
@JoinColumn
@OneToOne
案例一

does not define an IdClass
共享主键
null identifier
OneToMany 一对多
ManyToMany 多对多
外键级联操作
CascadeType.PERSIST
CascadeType.REMOVE
外键级联删除
MySQL ON DELETE CASCADE
@JoinTable
多对多实例
@OrderBy
@ElementCollection
外键名称
@JsonIgnore
@EnableJpaAuditing 开启 JPA 审计功能
注释 @Comment
@Pattern 数据匹配
实体继承

3.3. Repository

CrudRepository
    批量保存
JpaRepository
PagingAndSortingRepository
    Pageable
    解决 PagingAndSortingRepository 没有 save 等方法的问题
    @PageableDefault 分页
findByXXX
    传 Boolean 参数
    Eunm 传递枚举参数
count 操作
delete 删除操作
OrderBy
GreaterThan
Sort 排序操作操作
Pageable 翻页操作
    PageRequest.of
@DynamicInsert 与 @DynamicUpdate
继承已存在的 Repository

3.4. TransactionTemplate

3.5. JPQL @Query
    @Modifying 更新/删除
    事务 @Transactional
        删除更新需要 @Transactional 注解
        回滚操作
        private、default、protected 和 final 不支持事物
        Service 注意事项
        需要 @Service 注解配合使用

    参数传递
    原生 SQL
    @Query 与 Pageagble
    返回指定字段
    返回指定的模型

Collection
原生SQL查询
Sort
锁 @Lock
4. EntityManager
5. Spring Data with JdbcTemplate
5.1. execute
5.2. queryForInt
5.3. queryForLong
5.4. queryForObject
返回整形与字符型
查询 Double 类型数据库
返回日期
返回结果集
通过 "?" 向SQL传递参数
RowMapper 记录映射
5.5. queryForList
Iterator 用法
for 循环
forEach 用法
5.6. queryForMap
5.7. query
ResultSet
ResultSetExtractor
RowMapper
5.8. queryForRowSet
5.9. update
5.10.
5.11. 实例参考
参数传递技巧
6. Spring Data with Elasticsearch
6.1. 内嵌 Elasticsearch
Maven
src/main/resources/application.properties
Domain Class
ElasticsearchRepository

SearchRestController
测试
6.2. 集群模式
6.3. Document
6.4. Elasticsearch 删除操作
6.5. FAQ
java.lang.IllegalStateException: Received message from unsupported version: [2.0.0] minimal compatible version is: [5.0.0]
7. Spring boot with Data restful
7.1. Maven
8. Apache ShardingSphere
8.1. 微服务集群环境，雪花算法出现重复ID
方案一、配置实现
方案二、代码实现
9. Spring Data FAQ
9.1. No identifier specified for entity
9.2. Oracle Date 类型显示日期和时间
9.3. java.lang.ClassCastException: java.lang.Long cannot be cast to java.lang.Integer
9.4. Executing an update/delete query; nested exception is javax.persistence.TransactionRequiredException: Executing an update/delete query
9. Spring Security
1. Spring Security with HTTP Auth
1.1. 默认配置
1.2. 设置用户名和密码
1.3. 禁用 Security
2. Spring boot with Spring security
2.1. Maven
2.2. Reource
2.3. Application
2.4. WebSecurityConfigurer
2.5. RestController
2.6. 测试

- 2.7. Spring + Security + MongoDB
  - Account
  - AccountRepository
  - WebSecurityConfiguration
3. Spring Boot with Web Security
  - 3.1. EnableWebSecurity
  - 3.2. Web静态资源
  - 3.3. 正则匹配
  - 3.4. 登陆页面，失败页面，登陆中页面
  - 3.5. CORS
  - 3.6. X-Frame-Options 安全
4. 访问控制列表（Access Control List，ACL）
  - 4.1. antMatchers
  - 4.2. HTTP Auth
  - 4.3. Rest
  - 4.4. hasRole
  - 4.5. hasAnyRole()
  - 4.6. withUser
    - 添加用户
    - 添加多个用户，并指定角色
    - 获取当前用户
5. Spring Authorization Server
  - 5.1. Oauth2 协议
    - token
    - grant_type
    - 授权码授权模式（Authorization Code Grant）
    - 密码模式（Resource Owner Password Credentials Grant）
    - 客户端凭证模式（Client Credentials Grant）
    - 刷新 TOKEN 方式
  - 5.2. Maven 依赖
  - 5.3. Spring cloud with Oauth2
    - authorization_code
      - 验证服务器器

- 测试
- Spring boot with Oauth2 - Password
  - Maven
  - Password tools
  - Server
    - Maven
    - application.properties
    - EnableAuthorizationServer
    - EnableResourceServer
    - Entity Table
    - UserRepository
    - UserService
    - TestRestController
    - 数据库初始化
    - Test
  - Spring boot with Oauth2 RestTemplate
    - Maven
    - OAuth2ClientConfiguration.java
    - Application.java
    - application.properties
    - Controller
    - Test
- Spring boot with Oauth2 jwt
  - Maven
  - Authorization Server
  - Resource Server
  - Web Security
  - 插入数据
  - 使用 CURL 测试 JWT
  - 测试 Shell
  - refresh_token
- Spring boot with Oauth2 jwt 非对称证书
  - 创建证书
  - Authorization Server

Resource Server
Apple iOS 访问 Oauth2
Oauth2 客户端

application.yml
SpringApplication
WebSecurityConfigurer
TestController
Android Oauth2 + Jwt example
RestTemplate 使用 HttpClient
Maven
SpringBootApplication
ClientRestController
Test
自签名证书信任问题
Principal
SecurityContextHolder 对象
资源服务器配置
access()

Client
Overriding Spring Boot 2.0 Auto-configuration
Oauth2 常见问题
修改 /oauth/token 路径
password 认证方式静态配置用户列表

10. Spring Cloud
    1. Spring Cloud 相关的 application.properties 配置
        1.1. 启用或禁用 bootstrap
        1.2. bootstrap.properties 配置文件
    2. Spring Cloud Config
        2.1. Maven 项目 pom.xml 文件
        2.2. Server
Maven config 模块
Application

application.properties
Git 仓库
测试服务器
2.3. Client
Maven pom.xml
Application
bootstrap.properties
测试 client
2.4. Config 高级配置
仓库配置
分支
basedir
HTTP Auth
本地git仓库
native 本地配置
Config server 用户认证
Server 配置
application.properties
Maven
测试是否生效
Client 配置
加密敏感数据
Spring Cloud Config JDBC Backend
Maven pom.xml
数据库表结构
Config 服务器
application.properties
2.5. Old
Server (Camden.SR5)
Client (Camden.SR5)
3. Spring Cloud Consol
3.1. Spring Cloud Consul 配置
3.2. Maven 父项目
3.3. Consul 服务生产者
Maven
application.properties

SpringApplication
TestController
3.4. Consul 服务消费者
Maven
application.properties
SpringApplication
TestController
3.5. Openfeign
Maven
application.properties
SpringApplication
Feign 接口
TestController
4. Spring Cloud Netflix
4.1. Eureka Server
Maven
Application
application.properties
检查注册服务器
4.2. Eureka Client
Maven
Application
RestController
application.properties
测试
4.3. Feign client
Maven
Application
interface
application.properties
测试
fallback
4.4. 为 Eureka Server 增加用户认证
Maven
application.properties
Eureka Client

Feign Client
4.5. Eureka 配置项
/eureka/apps
Eureka instance 配置项
Eureka client 配置项
Eureka Server配置项
4.6. ribbon

LoadBalancerClient 实例
application.properties
LoadBalancerClient 获取服务器列表
Ribbon 相关配置
内置负载均衡策略
4.7. 获取 EurekaClient 信息
4.8. Zuul
Maven
EnableZuulProxy
application.yml
负载均衡配置
5. Openfeign
5.1. Openfeign 扫描包配置
5.2. 用户认证
5.3. 应用实例
5.4. 配置连接方式
httpclient
okhttp
5.5. 配置手册
6. Spring Cloud Gateway
6.1. Gateway 例子
Maven
SpringApplication
application.yml
RouteLocator 方式
6.2. 路由配置
转发操作
URL 参数

7. Spring Cloud Stream
8. Spring Cloud Bus
9. Spring Cloud Sleuth
    9.1. logback 安装
10. Spring Cloud with Kubernetes
    10.1. Config
        Maven 依赖
        Spring Cloud 配置文件
        程序文件
            SpringBootApplication 启动文件
            配置类
            控制器
        Kubernetes 编排脚本
        测试
    10.2. 注册发现
        Maven 父项目
        provider
            Maven 依赖
            Springboot 启动类
            控制器
            application.properties 配置文件
            Kubernetes provider 编排脚本
        consumer
            Maven 依赖
            Springboot 启动类
            控制器
            FeignClient 接口
            application.properties 配置文件
            Kubernetes consumer 编排脚本
        测试

11. Spring Cloud Alibaba
    11.1. 安装 Nacos
        Docker 安装 Nacos
        Kubernetes 安装 Nacos
        IP限制，白名单

防火墙配置
11.2. Kubernetes 部署微服务
pom.xml 中加入 docker 插件
容器启动脚本
构建 docker 镜像
编排 kubernetes 容器
启动指定 nacos
11.3. Nacos 配置中心/注册中心代码实例
Maven
SpringBootApplication
ConfigController
配置文件
11.4. FAQ
禁用 Nacos
禁止注册
Failed to bind properties under 'server.tomcat.basedir' to java.io.File:
不读取 bootstrap.yaml 文件
WARN [com.alibaba.nacos.client.naming:177] [,] - out of date data received, old-t: 1665711914993, new-t: 1665711902390
User limit of inotify instances reached or too many open files
开启权限
ERROR Whitelabel

12. FAQ
12.1. Cannot execute request on any known server
12.2. @EnableDiscoveryClient与@EnableEurekaClient 区别
12.3. Feign请求超时
12.4. 已停止的微服务节点注销慢或不注销
12.5. Feign 启动出错 PathVariable annotation was empty on param 0.

12.6. Feign 提示 Consider defining a bean of type 'common.feign.Cms' in your configuration.
12.7. Load balancer does not have available server for client
12.8. Eureka Client (Dalston.SR1)
Maven
Application
RestController
application.properties
测试
12.9. Config Server(1.3.1.RELEASE)
Server
Maven
Application
application.properties
Git 仓库
测试服务器
Client
Maven pom.xml
Application
bootstrap.properties
测试 client
12.10. feign.RetryableException: Read timed out executing
11. Tomcat Spring 运行环境
1. Maven
2. Spring Boot Quick start
2.1. 创建项目
2.2. pom.xml
2.3. Controller
3. Spring MVC configuration
4. Tomcat
5. 集成 Mybatis
5.1. pom.xml
5.2. properties
5.3. dataSource

        5.4. SqlSessionFactory
        5.5. Mapper 扫描
        5.6. Mapper 单一class映射
        5.7. Service
        5.8. 测试实例
12. 杂项 Miscellaneous
    1. URL 拼装/解析
    2. ServletUriComponentsBuilder
    3. URL 路径相关
13. FAQ
    1. org.hibernate.dialect.Oracle10gDialect does not support identity key generation
    2. No identifier specified for entity
    3. Could not read document: Invalid UTF-8 middle byte 0xd0
    4. java.sql.SQLRecoverableException: IO Error: The Network Adapter could not establish the connection
    5. Field javaMailSender in cn.netkiller.rest.EmailRestController required a bean of type 'org.springframework.mail.javamail.JavaMailSender' that could not be found.
    6. org.postgresql.util.PSQLException: FATAL: no pg_hba.conf entry for host "172.16.0.3", user "test", database "test ", SSL off
    7. Spring boot 怎样显示执行的SQL语句
    8. Cannot determine embedded database driver class for database type NONE
    9. Spring boot / Spring cloud 时区差8个小时
    10. @Value 取不到值
    11. Spring boot 2.1.0
    12. Field authenticationManager in cn.netkiller.oauth2.config.AuthorizationServerConfigurer required a bean of type 'org.springframework.security.authentication.AuthenticationManager' that could not be found.

13. 打印 Bean 信息
14. The dependencies of some of the beans in the application context form a cycle
15. no main manifest attribute, in /srv/job-admin.jar

14. MyBatis
    1. Mybatis 入门
    2. 接口注解
15. Apache Struts
    1. struts.xml
        1.1. include
    2. Struts Tags
        2.1. property
        2.2. set
        2.3. url
        2.4. s:include
        2.5. s:action
        2.6. HTML Form
            form
            textfield
            s:hidden
            select
        2.7. iterator
        2.8. if elseif else
    3. Action
        3.1. redirect
        3.2. redirectAction
        3.3. JSON
            enableGZIP 压缩传输
            excludeProperties 排除 Properties

        3.4. 传递 Timestamp 变量
    4. Ajax + JSON
        4.1. GET/POST JSON
    5. Json 内容展示
        5.1. 禁止方法
        5.2. 格式化日期

        5.3. 重命名变量名
        5.4. org.apache.struts2.json
    6. Interceptor
        6.1. Session
    7. Action 中使用线程
    8. 日志
    9. FAQ
        9.1. Struts 怎样判断用户来自电脑还是移动设备
16. Apache Tiles
    1. 配置 Tiles
        1.1. Maven
        1.2. web.xml
    2. Template 配置模板
    3. Struts tiles
17. Play
18. Log
    1. Logback
        1.1. Maven 包
        1.2. Example
    2. slf4j
    3. log4j
        3.1. 安装 Log4j
            手工安装
            Maven
        3.2. log4j 环境变量
        3.3. Log4j Example
        3.4. log4j.properties
19. JSON (JavaScript Object Notation)
    1. javax.json.*
        1.1. Json 编码
        1.2. Json 解码
        1.3. URL获取Json
    2. Jackson
        2.1. ObjectToJSON
        2.2. JSONToObject
        2.3. JsonNode

    3. org.json
        3.1. JSONArray forEach
    4. Google Json
        4.1. json 转 map
        4.2. LinkedHashMap 转 Json
20. AMQP(Advanced Message Queuing Protocol)
    1. Send and Recv
    2. direct
21. NoSQL
    1. MongoDB
        1.1. pom.xml
        1.2. 插入操作
        1.3. 读取操作
22. Elasticsearch API
    1. Client
    2. insert
    3. Get
    4. delete
    5. Search
    6. Query 查询
        6.1. match all 匹配所有数据
        6.2. match 匹配查询
        6.3. match phrase 短语精准匹配
    7. Filter 过滤
        7.1. term
        7.2. range
    8. Sorting
    9. 返回 Source 字段
    10. Count
    11. Example 范例
        11.1. Spring boot 案例
    12. FAQ
        12.1. 显示查询 JSON 字符串
23. Jersey - RESTful Web Services in Java.
    1. Client 2.x
        1.1. Maven 版本

    1.2. GET 操作
    1.3. GET + Auth 用户认证
  2. Client 1.x
    2.1. Jersey + Auth + HTTP2 + SSL
24. Apache HttpComponents
  1. org.apache.commons.lang3
    1.1. HTML 标签处理
    1.2. StringUtils.join 使用特定字符链接字符串
    1.3. RandomStringUtils
  2. commons-text
    2.1. 禁止转译 json
  3. Apache HttpClient
    3.1. Maven
    3.2. HTTP POST 操作
      Post Data
      POST RAW 数据
      POST GBK 编码得数据
    3.3. HTTPS
      Get https 接口
      POST json 数据
    3.4. HTTP/2
    3.5. Java11
      sync get
      async get
      post form
    3.6. Host name 'api.netkiller.cn' does not match the certificate subject provided
    3.7. HttpStatus
    3.8.
25. Cache
  1. java memcached client
  2. Jedis
    2.1. 认证
    2.2. jedis.keys
  3. Ehcache
26. Kafka

1. 安装 Kafka 环境
2. Maven
3. 启动 kafka
4. 入门例子
    4.1. 订阅例子
    4.2. 发布例子
5. 线程例子

27. Software Development Kit
1. JAVE(Java Audio Video Encoder)
2. Google
    2.1. com.google.gson
        map 处理
        POJO
        toJson
        fromJson
        JsonParser
        Exmaple 范例
            Map to Json
        Exmaple 范例
            Map to Json
        处理复杂的类型
    2.2. Guava
        maven
        删除不可显示的字符
3. Mahout
    3.1. 推荐系统
        Maven pom.xml
        推荐程序
        数据文件
4. Hessian
5. quartz-scheduler
6. Redisson

II. Android 9 Pie
28. Android Studio
1. 卸载 Android Studio
2. 代码格式化

3. 设置兼容最低SDK版本
4. SDK Tools
    4.1. 接受 License
    4.2. 查看 SDK 列表
    4.3. 按照 Android SDK
5. 命令行操作
6. adb 命令
    6.1. 获得 root 权限
    6.2. 设备管理
    6.3. Shell
        网络相关
            查看 IP 地址
            无线 IP 地址
            Mac 地址
        内存信息
        查看硬件与系统属性
    6.4. 设备 ID
        获取变量
        设置变量
            显示/关闭虚拟键
    6.5. 查看安卓版本
        产品型号
    6.6. Logcat
    6.7. 上传文件
    6.8. 下载文件
    6.9. 安卓 .apk bk
    6.10. 屏幕尺寸
        查看 dpi
        设置 dpi
    6.11. dump 系统信息
        电池信息
    6.12. 解锁
    6.13. 蓝牙管理
29. AndroidManifest.xml
    1. SDK 版本配置
    2. 开启网络

3. 文件存储权限
4. 相机权限
5. GPS 定位权限
6. 全屏-无标题
7. 设置为默认开机启动
8. 开机启动
9. 默认横屏
10. 禁止屏幕旋转变化

30. 设备
    1. 环境变量
        1.1. 扩展存储
        1.2. 下载缓存目录
        1.3. 数据目录
    2. 配置文件
        2.1. *.properties 文件
        2.2. 再 AndroidManifest.xml 使用 meta-data element 定义
        2.3. 再 build.gradle 文件中配置 productFlavors
        2.4. 从 assets 目录读取配置文件
            配置文件例子
    3. 设备信息
    4. Physical density
    5. 声卡
        5.1. 播放
        5.2. 录音
        5.3. 查看声卡信息
        5.4. /proc/asound 设备信息
        5.5. 查看声卡当前占用设备
        5.6. tinymix 设置声卡参数
        5.7. 麦克风阵列调试
            录音测试

31. Activity
    1. 定义 UI
    2. 隐藏虚拟键
    3. 显式四种跳转方式
        3.1. startActivity()

4. 定时关闭
5. 恢复触发
6. 返回触发
7. 保持屏幕常开
8. 标题栏添加返回按钮
9. Activity 间数据传递
    9.1. Intent 方式
    9.2. Bundle 方式
    9.3. Flag 属性
        在 Service，BroadcastReceiver 中切换 View
    9.4. 返回值
10. intentActivityResultLauncher 跳转
11. startActivityForResult 替代方案
    11.1. 返回值
12. Activity 关闭
    12.1. 退出 App
13. App 间跳转
14. Res 资源
    14.1. 通过名称查找 layout ID
    14.2. 查找 drawable 资源 ID
    14.3. 获取 color 颜色 ID
    14.4. 获取 array.xml 文件下某个字段的 ID
    14.5. 获取 style.xml 文件下的某个样式的 id

32. Fragment
    1. 启动 Fragment
    2. 关闭 Fragment
    3. 在 Fragment 中使用 findViewById
    4. 在 Fragment 中使用 Intent 跳转
    5. Fragment 中调用 getPackageManager()
    6. 在 Fragment 中使用 runOnUiThread
    7. Fragment 中调用 findViewById
    8. 替换 FrameLayout
    9. Fragment 接收 BroadcastReceiver 广播
33. Resources
    1. strings.xml
        1.1.

1.2.
1.3. 获取 Resource

34. Palette 视觉设计
    1. 父容器定位
    2. 样式布局
        2.1. 对齐布局
        2.2. LinearLayout
            外边距设置
            内边距设置
            水平剧中
        2.3. FrameLayout
            FrameLayout 事件穿透
            叠加层
        2.4. 动画
        2.5. 声音波形图
    3. Widgets
        3.1. ImageView
            剧中效果
            ImageView 显示 URL 图片
            唱片播放效果（旋转PNG图片）
                UI 布局
                旋转动画效果文件
                启动旋转效果
        3.2. TextClock
        3.3. 进度条
    4. Containers
        4.1. CardView
            实现圆角 ImageView
    5. Legacy
        5.1. GardView
        5.2. GridView
    6. 渐变背景色
    7. 屏幕
        7.1. 尺寸
        7.2. 屏幕触摸事件 onTouch(View view, MotionEvent motionEvent)

        屏幕触摸事件 onTouchEvent(MotionEvent event)
    7.3. 手势事件
36. Schedule 计划任务
    1. 延迟执行
    2. Time 和 TimerTask 定时刷新
    3. 使用 Runnable 和 Handler 实现定时执行
    4. 循环执行
    5. TimerTask 实现循环播放
    6. TimerTask 更新 UI
37. Internationalization i18n with Android (国际化)
    1. 创建国际化文件
    2. strings.xml 文件
    3. 翻译语言
    4. 引用国际化文件
    5. 切换语言
38. 存储
    1. 存储目录
    2. SharedPreferences
        2.1. 操作模式
        2.2. 保存数据
        2.3. 读取数据
        2.4. 通过 key 查询数据是否存在
        2.5. 删除数据
        2.6. 清空数据
        2.7. 对象存储
        2.8. SharedPreferences 读取物理存储文件
    3. SD Card
        3.1. SD Card 状态
        3.2. Android 11 申请 sdcard 权限
39. 网络
    1. Wifi 配置
    2. OkHttp - An HTTP & HTTP/2 client for Android and Java applications
        2.1. Gradle

    2.2. AndroidManifest.xml 开启网络访问权限
    2.3. okhttp 默认是 HTTPS 开启 HTTP
    2.4. GET
        URL 组装
        Get 异步调用
    2.5. POST
        POST Form Data
        POST RAW JSON
        数据流提交
    2.6. HTTP PUT 请求
    2.7. http header 相关设置
        设置 HTTP 头
        Cookie 管理
        禁用缓存
        设置缓存 max-age
        强制缓存
    2.8. HTTP Base Auth
    2.9. HttpUrl.Builder 组装 URL 地址参数
    2.10. Android Activity Example
    2.11. Android Oauth2 + Jwt example
    2.12. HTTP/2
    2.13. 异步更新 UI
    2.14. WebSocket Client
40. 相机与相册
    1. manifest 文件
    2. layout
    3. Activity
    4. LED flash 做手电筒
41. 麦克风与录音
    1. 开启麦克风和SD卡权限
    2. layout
    3. Activity
42. 多媒体开发
    1. MediaPlayer
        1.1. 播放Raw下的元数据
        1.2. 播放assets文件夹中的音乐

        1.3. 播放互联网音乐
        1.4. 使用单例模式
        1.5. 设置速度，快进播放
    2. VideoView 开发
        2.1. 播放网络视频
        2.2. MediaController 添加翻页事件
        2.3. 静音播放视频
        2.4. 更新进度条
        2.5. 完整的例子
        2.6. 循环播放
        2.7. 静音播放
    3. SoundPool
    4. 音量控制
    5. SurfaceView
    6. Vitamio
43. 定位
    1. GPS + 网络 定位
        1.1. manifest 权限配置
        1.2. layout
        1.3. Activity
    2. 只从 GPS 获取定位
44. 电话
    1. SIM 卡状态
    2. 通信录与拨打电话
    3. 发送短信
45. 消息广播
    1. 动态注册
    2. 静态注册
        2.1. 电源管理
        2.2. 接收不到消息
    3. 自定义用户消息广播
    4. 本地广播
    5. 动态监听广播
    6. 广播重复接收
    7. 指定静态广播接收者
    8. 异步执行广播

46. Service 服务
    1. Service的基本用法
        1.1. manifest 文件
        1.2. 创建 Service
        1.3. Layout 代码
        1.4. Activity 代码
    2. Service 中启动线程
    3. Service 和 Activity 通信
        3.1. Layout
        3.2. Service
        3.3. Activity
    4. Service 和 Toast
    5. Service 中启动 Activity
    6. Service 中更新 UI
47. Notification 通知中心
    1. 文本通知
    2. 添加点击操作
48. NFC (Near field communication)
    1. AndroidManifest.xml 文件配置
    2. Loyout 文件
    3. Activity 文件
49. 图形开发
    1. Paint
    2. AnimationDrawable
50. 下载管理
    1. 从 URL 下来文件
    2. 安装 APK
    3. 下载后接收广播通知
51. Android 多线程
    1. GPIO
52. EventBus
    1. 添加 EventBus 依赖到项目Gradle文件
    2. 快速开始一个演示例子
        2.1. 创建 MessageEvent 类
        2.2. Layout
        2.3. Activity

    3. Sticky Events
        3.1. MainActivity
        3.2. StickyActivity
        3.3. MessageEvent
        3.4. 删除粘性事件
    4. 线程模型
    5. 配置 EventBus
    6. 事件优先级
    7. 捕获异常事件
53. Android MQTT
    1. build.gradle 添加依赖包
    2. AndroidManifest.xml
    3. Android Mqtt v5 例子
54. 安卓开发版
    1. rk3568
        1.1. 声卡
55. 杂项
    1. Sleep
    2. Caused by: java.net.UnknownServiceException: CLEARTEXT communication to 47.100.253.187 not permitted by network security policy
    3. 设计模式
        3.1. 单例模式
    4. Android OS 包
        4.1. 进程ID
        4.2. handler
    5. fastjson android
        5.1. 对象转字符串
        5.2. JsonObject 转对象
        5.3. 字符串 与 json 互转
        5.4. json 转 数组
        5.5. JSON数组转List
        5.6. Map 与 Json 互转
    6. Butter Knife
    7. Android Things
        7.1. GPIO

56. FAQ
1. java.net.UnknownServiceException: CLEARTEXT communication to 192.168.0.185 not permitted by network security policy
2. Caused by: android.os.NetworkOnMainThreadException
3. java.lang.IllegalStateException: Player is accessed on the wrong thread.
4. Manifest merger failed with multiple errors, see logs
5. 从 Android API 30 废弃 setSystemUiVisibility(uiOptions)
57. 讯飞云
1. AIUI
1.1. AIUIPlayer
1.2. 酷我音乐
获取音乐URL
1.3. 控制技能
1.4. 唤醒词
手工唤醒
1.5. 汉字转拼音
2. 讯飞 TTS
2.1. 设置日志输出级别
2.2. 流式语音合成
3. 语音唤醒
3.1. 范例
A. 附录
1. 一致性算法

范例清单

1.1. /etc/profile.d/java.sh
2.1. Maven properties
2.2. 将本地 lib/*.jar 包添加到项目中
2.3. 将本地 src/resources 打包到项目
2.4. Maven parent

2.5. watir-webdriver example
5.1. Spring boot with Velocity template (pom.xml)
5.2. Example Spring boot with Oracle
5.3. RedisTemplate
5.4. Spring boot with Apache kafka.
5.5. Spring boot with Apache kafka.
5.6. Test Spring Kafka
5.7. Spring boot with Email (pom.xml)
8.1. Spring Data Redis Example
8.2. Spring Data MongoDB - springframework-servlet.xml
10.1. Share feign interface.
11.1. MyBatis
25.1. memcached.java
36.1.

Netkiller Java 手札（2024版）

Java, Servlet, JavaBean, Struts, Spring ...

ISBN#

Mr. Neo Chan, 陈景峯(BG7NYT)

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

<netkiller@msn.com>

文档始创于 2015-11-10

电子书最近一次更新于 2024-01-26 00:11:00

版权 © 2015-2022 Netkiller(Neo Chan). All rights reserved.

版权声明

转载请与作者联系，转载时请务必标明文章原始出处和作者信息及本声明。





http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net
微信公众号: netkiller
微信：13113668890 请注明“读者”
QQ：13721218 请注明“读者”
QQ群：128659835 请注明“读者”
知乎专栏 | 多维度架构





2017-11

关于《Netkiller Java 手札》

作者2002年开始在项目中使用Java，各种原因没有留下Java文档，2015因工作需要重新拾起Java并整理本文档。

本电子书重点内容是Spring boot, Spring cloud, Spring data, Spring security

我的系列文档

编程语言

Netkiller Architect 手札	Netkiller Developer 手札	Netkiller Java 手札	Netkiller Spring 手札	Netkiller PHP 手札	Netkiller Python 手札
Netkiller Testing 手札	Netkiller Cryptography 手札	Netkiller Perl 手札	Netkiller Docbook 手札	Netkiller Project 手札	Netkiller Database 手札

致读者

Netkiller 系列手札 已经被 Github 收录，并备份保存在北极地下250米深的代码库中，备份会保留1000年。

Preserving open source software for future generations



The world is powered by open source software. It is a hidden cornerstone of modern civilization, and the shared heritage of all humanity.

The GitHub Arctic Code Vault is a data repository preserved in the Arctic World Archive (AWA), a very-long-term archival facility 250 meters deep in the permafrost of an Arctic mountain.

We are collaborating with the Bodleian Library in Oxford, the Bibliotheca Alexandrina in Egypt, and Stanford Libraries in California to store copies of 17,000 of GitHub's most popular and most-depended-upon projects—open source's "greatest hits"—in their archives, in museum-quality cases, to preserve them for future generations.

https://archiveprogram.github.com/arctic-vault/

自述



Netkiller Java 手札

陈景峰 著





《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是 nickname 从1999 开使用至今，“手札” 是札记，手册的含义。

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随着内容增加，不断修订，开始发布第一版，第二版.....

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用 Microsoft Office Word 慢慢随着文档尺寸增加 Word 开始表现出力不从心。

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习 Docbook SGML。使用Docbook写的第一部电子书是《Netkiller Postfix Integrated Solution》这是Netkiller 系列手札的原型。

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》 的 SGML文件已经不存在; Docbook SGML存在很多缺陷 UTF-8支持不好，转而使用Docbook XML.

目前技术书籍的价格一路飙升，动则￥80，￥100，少则￥50，￥60. 技术书籍有时效性，随着技术的革新或淘汰，大批书记成为废纸垃圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所有我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

1. 写给读者

为什么写这篇文章

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

写作动力:

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

没有内容的章节:

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本章写其它的.

文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.

理论的东西,网上大把,我这里就不写了,需要可以去网上查.

我爱写错别字,还有一些是打错的,如果发现请指正.

文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

写给读者

至读者:

我不知道什么时候，我不再更新文档或者退出IT行业去从事其他工作，我必须给这些文档找一个归宿，让他能持续更新下去。

我想捐赠给某些基金会继续运转，或者建立一个团队维护它。

我用了20年时间坚持不停地写作，持续更新，才有今天你看到的《Netkiller 手扎》系列文档，在中国能坚持20年，同时没有任何收益的技术类文档，是非常不容易的。

有很多时候想放弃，看到外国读者的支持与国内社区的影响，我坚持了下来。

中国开源事业需要各位参与，不要成为局外人，不要让外国人说:中国对开源索取无度，贡献却微乎其微。

我们参与内核的开发还比较遥远，但是进个人能力，写一些文档还是可能的。

系列文档

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札

Netkiller Developer 手札

Netkiller PHP 手札

Netkiller Python 手札

Netkiller Testing 手札

Netkiller Cryptography 手札
Netkiller Linux 手札
Netkiller FreeBSD 手札
Netkiller Shell 手札
Netkiller Security 手札
Netkiller Web 手札
Netkiller Monitoring 手札
Netkiller Storage 手札
Netkiller Mail 手札
Netkiller Docbook 手札
Netkiller Version 手札
Netkiller Database 手札
Netkiller PostgreSQL 手札
Netkiller MySQL 手札
Netkiller NoSQL 手札
Netkiller LDAP 手札
Netkiller Network 手札
Netkiller Cisco IOS 手札
Netkiller H3C 手札
Netkiller Multimedia 手札
Netkiller Management 手札
Netkiller Spring 手札
Netkiller Perl 手札
Netkiller Amateur Radio 手札

2. 作者简介

陈景峯 (ㄔㄣˊ ㄐㄧㄥˇ ㄈㄥ)

Nickname： netkiller | English name: Neo chen | Nippon name: ちん けいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: BG7NYT | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工, Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

成长阶段

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Trubo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作 UCDOS 中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

职业生涯

2001 年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002 年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003 年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入分布式计算团队,目前成绩，工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL，相隔几年发现PHP进步很大。在前台展现方面无人能敌，于是便前台使用PHP，后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机，休息了4个月（其实是找不到工作），关外很难上439.460中继，搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,2009-6-13学车，年底拿到C1驾照

2010 对电子打击乐产生兴趣，计划学习爵士鼓。由于我对Linux热爱，我轻松的接管了公司的运维部，然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜，我们用买服务器纸箱的钱改善伙食。我将40多台服务器安装BOINC做压力测试，获得了中国第二的名次。

2011 平凡的一年，户外运动停止，电台很少开，中继很少上，摄影主要是拍女儿与家人，年末买了一辆山地车

2012 对油笔画产生了兴趣，活动基本是骑行银湖山绿道，

2013 开始学习民谣吉他，同时对电吉他也极有兴趣；最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola

MOTOTRBO XIR P8668，Netkiller 系列手札从Sourceforge向Github迁移；年底对MYSQL UDF，Engine与PHP扩展开发产生很浓的兴趣，拾起遗忘10+年的C，写了几个mysql扩展（图片处理，fifo管道与ZeroMQ），10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴，由于这家钢琴是合成器电钢，里面有打击乐，我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游，对中国道教文化与音乐产生了兴趣，10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣（https://github.com/SheetMusic/Piano），给女儿做了几首钢琴伴奏曲，MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练，经过反复琢磨加上之前学钢琴的乐理知识，终于在02号晚上，打出了简单的基本节奏，迈出了第一步。

2016 对弓箭（复合弓）产生兴趣，无奈天朝法律法规不让玩。每周游泳轻松1500米无压力，年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台，连接Xbox打游戏爽翻了，入Kindle电子书，计划学习蝶泳，果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦，丢弃了多年攒下的家底。11 月开始玩 MMDVM，使用 Yaesu FT-7800 发射，连接MMDVM中继板，树莓派，覆盖深圳湾，散步骑车通联两不误。

2019 卖了常德的房子，住了5次院，哮喘反复发作，决定停止电子书更新，兴趣转到知乎，B站

2020 准备找工作

职业生涯路上继续打怪升级

3. 如何获得文档

下载 **Netkiller 手札 (epub,kindle,chm,pdf)**
EPUB
https://github.com/netkiller/netkiller.github.io/tree/master/download/epub
MOBI
https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi
PDF
https://github.com/netkiller/netkiller.github.io/tree/master/download/pdf
CHM
https://github.com/netkiller/netkiller.github.io/tree/master/download/chm

通过 GIT 镜像整个网站

https://github.com/netkiller/netkiller.github.com.git

$ git clone https://github.com/netkiller/netkiller.github.com.git

镜像下载

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

Yum 下载文档

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

http://netkiller.sourceforge.net/pub/repo/

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
name=Netkiller Free Books
baseurl=http://netkiller.sourceforge.net/pub/repo/
enabled=1
gpgcheck=0
gpgkey=
EOF
```

查找包

```
# yum search netkiller

netkiller-centos.x86_64 : Netkiller centos Cookbook
netkiller-cryptography.x86_64 : Netkiller cryptography
Cookbook
netkiller-docbook.x86_64 : Netkiller docbook Cookbook
netkiller-linux.x86_64 : Netkiller linux Cookbook
netkiller-mysql.x86_64 : Netkiller mysql Cookbook
netkiller-php.x86_64 : Netkiller php Cookbook
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook
netkiller-python.x86_64 : Netkiller python Cookbook
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```

4. 打赏（Donations）

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

银行(Bank)

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

微信（Wechat）



支付宝（Alipay）



PayPal Donations

https://www.paypal.me/netkiller

5. 联系方式

主站 http://www.netkiller.cn/

备用 http://netkiller.github.io/

繁体网站 http://netkiller.sourceforge.net/

联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222

注：请不要问我安装问题！

博客 **Blogger**

知乎专栏 https://zhuanlan.zhihu.com/netkiller

LinkedIn: http://cn.linkedin.com/in/netkiller

OSChina: http://my.oschina.net/neochen/

Facebook: https://www.facebook.com/bg7nyt

Flickr: http://www.flickr.com/photos/bg7nyt/

Disqus: http://disqus.com/netkiller/

solidot: http://solidot.org/~netkiller/

SegmentFault: https://segmentfault.com/u/netkiller

Reddit: https://www.reddit.com/user/netkiller/

Digg: http://www.digg.com/netkiller

Twitter: http://twitter.com/bg7nyt

weibo: http://weibo.com/bg7nyt

Xbox club

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部 netkiller 欢迎加入。

Radio

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

MMDVM Hotspot:

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

第 1 章 Java

1. JVM

1.1. Almalinux / RockyLinux

安装 Java 1.8 JRE

```
dnf install java-1.8.0-openjdk
```

安装 Java 1.8 JDK

```
dnf install java-1.8.0-openjdk-devel
```

当我们安装 maven 的时候会自动安装 Java 11，需要使用下面方法切换默认Java版本

```
[root@cloud ~]# alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-
1.8.0-openjdk-1.8.0.342.b07-1.el9_0.x86_64/jre/bin/java)
*+ 2           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-1.el9_0.x86_64/bin/java)
```

```
Enter to keep the current selection[+], or type selection
number: 1
[root@cloud ~]# alternatives --config javac

There are 2 programs which provide 'javac'.

  Selection    Command
-----------------------------------------------
   1           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-
1.8.0-openjdk-1.8.0.342.b07-1.el9_0.x86_64/bin/javac)
*+ 2           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-
openjdk-11.0.16.0.8-1.el9_0.x86_64/bin/javac)

Enter to keep the current selection[+], or type selection
number: 1
```

```
[root@cloud ~]# javac -version
javac 1.8.0_342

[root@cloud ~]# java -version
openjdk version "1.8.0_342"
OpenJDK Runtime Environment (build 1.8.0_342-b07)
OpenJDK 64-Bit Server VM (build 25.342-b07, mixed mode)
```

1.2. CentOS 8 Java 14

安装 jre

```
[root@localhost ~]# dnf install java-latest-openjdk
```

安装 jdk

```
[root@localhost ~]# dnf install java-latest-openjdk-devel
```

默认是 jdk1.8，使用 alternatives 切换 java 默认版本为 java 14

```
[root@localhost ~]# java -version
openjdk version "1.8.0_262"
OpenJDK Runtime Environment (build 1.8.0_262-b10)
OpenJDK 64-Bit Server VM (build 25.262-b10, mixed mode)


[root@localhost ~]# alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           java-latest-openjdk.x86_64 (/usr/lib/jvm/java-
14-openjdk-14.0.2.12-1.rolling.el8.x86_64/bin/java)
*+ 2           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-
1.8.0-openjdk-1.8.0.262.b10-0.el8_2.x86_64/jre/bin/java)

Enter to keep the current selection[+], or type selection
number: 1


[root@localhost ~]# java -version
openjdk version "14.0.2" 2020-07-14
OpenJDK Runtime Environment 20.3 (build 14.0.2+12)
OpenJDK 64-Bit Server VM 20.3 (build 14.0.2+12, mixed mode,
sharing)
```

1.3. Java 版本切换

查看配置

```
root@netkiller ~/ops (master) [SIGINT]# alternatives --list
libnssckbi.so.x86_64    auto    /usr/lib64/pkcs11/p11-kit-
trust.so
soelim                  auto    /usr/bin/soelim.groff
iptables                auto    /usr/sbin/iptables-nft
ebtables                auto    /usr/sbin/ebtables-nft
arptables               auto    /usr/sbin/arptables-nft
cifs-idmap-plugin       auto    /usr/lib64/cifs-
utils/cifs_idmap_sss.so
man                     auto    /usr/bin/man.man-db
man.7.gz                manual  /usr/share/man/man7/man.man-
pages.7.gz
nc                      auto    /usr/bin/ncat
ld                      auto    /usr/bin/ld.bfd
mvn                     auto    /usr/share/maven/bin/mvn
gradle                  auto    /srv/gradle-7.5.1/bin/gradle
java                    manual  /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.362.b09-2.el9_1.x86_64/jre/bin/java
jre_openjdk             auto    /usr/lib/jvm/java-11-openjdk-
11.0.18.0.10-2.el9_1.x86_64
jre_11                  auto    /usr/lib/jvm/java-11-openjdk-
11.0.18.0.10-2.el9_1.x86_64
jre_11_openjdk          auto    /usr/lib/jvm/jre-11-openjdk-
11.0.18.0.10-2.el9_1.x86_64
jre_1.8.0               auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.362.b09-2.el9_1.x86_64/jre
jre_1.8.0_openjdk       auto    /usr/lib/jvm/jre-1.8.0-openjdk-
1.8.0.362.b09-2.el9_1.x86_64
javac                   manual  /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.362.b09-2.el9_1.x86_64/bin/javac
java_sdk_openjdk        auto    /usr/lib/jvm/java-11-openjdk-
11.0.18.0.10-2.el9_1.x86_64
java_sdk_11             auto    /usr/lib/jvm/java-11-openjdk-
11.0.18.0.10-2.el9_1.x86_64
java_sdk_11_openjdk     auto    /usr/lib/jvm/java-11-openjdk-
11.0.18.0.10-2.el9_1.x86_64
java_sdk_1.8.0          auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.362.b09-2.el9_1.x86_64
java_sdk_1.8.0_openjdk  auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.362.b09-2.el9_1.x86_64
```

java 切换

```
root@netkiller ~/ops (master)# alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
*  1           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-
openjdk-11.0.18.0.10-2.el9_1.x86_64/bin/java)
 + 2           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-
1.8.0-openjdk-1.8.0.362.b09-2.el9_1.x86_64/jre/bin/java)

Enter to keep the current selection[+], or type selection
number:
```

javac 切换

```
root@netkiller ~/ops (master)# alternatives --config javac
```

1.4. 安装 Java 6

解压

```
chmod +x jdk-6u1-linux-i586.bin
./jdk-6u1-linux-i586.bin
输入"yes"回车

mv jdk1.6.0_01 /usr/local/
ln -s /usr/local/jdk1.6.0_01/ /usr/local/java
```

/etc/profile.d/java.sh

例 1.1. /etc/profile.d/java.sh

```
#################################################
### Java environment by neo
#################################################
export JAVA_HOME=/usr/local/java
export JRE_HOME=/usr/local/java/jre
export PATH=$PATH:/usr/local/java/bin:/usr/local/java/jre/bin
export
CLASSPATH="./:/usr/local/java/lib:/usr/local/java/jre/lib:/usr/
local/memcached/api/java"
export JAVA_OPTS="-Xms128m -Xmx1024m"
```

HeapDumpOnOutOfMemoryError

```
JAVA_OPTS = "$JAVA_OPTS -XX:+HeapDumpOnOutOfMemoryError"
```

如果针对Tomcat可以在catalina.sh加入

```
if [ "$1" = "debug" ] ; then
JAVA_OPTS = "$JAVA_OPTS -XX:+HeapDumpOnOutOfMemoryError"
```

1.5. java-1.8.0-openjdk

```
# yum install -y java-1.8.0-openjdk
```

1.6. docker 环境

在docker中运行java

```
neo@MacBook-Pro ~ % docker pull openjdk:12-jdk
```

```
```

```
docker run -it openjdk:12-jdk /bin/jshell
```

```
docker run -it openjdk:12-jdk /bin/bash
root@44d1d18351a8:/# java -version
```

1.7. java - Launches a Java application.

java 9~11

直接使用 java 命令运行 *.java 文件

```
java netkiller.java
```

-verbose:class 显示载入jar文件

```
# java -verbose:class hello
[Opened /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Object from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.Serializable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Comparable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.CharSequence from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.String from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.AnnotatedElement from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.GenericDeclaration from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Type from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Cloneable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.System from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Throwable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Error from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ThreadDeath from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Exception from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.RuntimeException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.SecurityManager from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.ProtectionDomain from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.AccessControlContext from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.SecureClassLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ReflectiveOperationException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassNotFoundException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.LinkageError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.NoClassDefFoundError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassCastException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ArrayStoreException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded java.lang.VirtualMachineError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.OutOfMemoryError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StackOverflowError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.IllegalMonitorStateException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.Reference from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.SoftReference from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.WeakReference from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.FinalReference from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.PhantomReference from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Cleaner from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.Finalizer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Runnable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Thread from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Thread$UncaughtExceptionHandler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ThreadGroup from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Map from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Dictionary from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Hashtable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Properties from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.AccessibleObject from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Member from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Field from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Parameter from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded java.lang.reflect.Executable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Method from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Constructor from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.MagicAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.MethodAccessor from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.MethodAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.ConstructorAccessor from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.ConstructorAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.DelegatingClassLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.ConstantPool from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.FieldAccessor from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.FieldAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.UnsafeFieldAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.UnsafeStaticFieldAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.annotation.Annotation from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.CallerSensitive from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandle from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.DirectMethodHandle from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MemberName from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleNatives from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.LambdaForm from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodType from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.BootstrapMethodError from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.CallSite from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.ConstantCallSite from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MutableCallSite from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.VolatileCallSite from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Appendable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.AbstractStringBuilder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StringBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StringBuilder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Unsafe from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.AutoCloseable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.Closeable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.InputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.ByteArrayInputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.File from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URLClassLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URL from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.jar.Manifest from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Launcher from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Launcher$AppClassLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Launcher$ExtClassLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.CodeSource from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StackTraceElement from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.Buffer from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Boolean from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded java.lang.Character from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Number from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Float from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Double from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Byte from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Short from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Integer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Long from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.NullPointerException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ArithmeticException from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.ObjectStreamField from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Comparator from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.String$CaseInsensitiveComparator from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.Guard from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.Permission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.BasicPermission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.RuntimePermission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.AccessController from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.ReflectPermission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.PrivilegedAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded
sun.reflect.ReflectionFactory$GetReflectionFactoryAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.cert.Certificate from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Iterable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded java.util.List from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.RandomAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.AbstractCollection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.AbstractList from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Vector from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Stack from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.ReflectionFactory from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.Reference$Lock from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.Reference$ReferenceHandler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.ReferenceQueue from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.ReferenceQueue$Null from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.ReferenceQueue$Lock from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ref.Finalizer$FinalizerThread from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Map$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Hashtable$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.VM from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Math from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.Charset from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.spi.CharsetProvider from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.FastCharsetProvider from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.StandardCharsets from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.AbstractMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.util.PreHashedMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.StandardCharsets$Aliases from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.StandardCharsets$Classes from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.StandardCharsets$Cache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ThreadLocal from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.atomic.AtomicInteger from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.IncompatibleClassChangeError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.NoSuchMethodError from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.ArrayList from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Set from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.AbstractSet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$EmptySet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$EmptyList from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$EmptyMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$UnmodifiableCollection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$UnmodifiableList from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$UnmodifiableRandomAccessList
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.Reflection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.HashMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.HashMap$Node from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class$3 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class$ReflectionData from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class$Atomic from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.generics.repository.AbstractRepository
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.generics.repository.GenericDeclRepository
```

```
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.generics.repository.ClassRepository from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class$AnnotationData from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.annotation.AnnotationType from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.WeakHashMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassValue$ClassValueMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Modifier from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.LangReflectAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.ReflectAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Arrays from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.HistoricallyNamedCharset from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.Unicode from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.UTF_8 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.ReflectionFactory$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.NativeConstructorAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.DelegatingConstructorAccessorImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StringCoding from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ThreadLocal$ThreadLocalMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ThreadLocal$ThreadLocalMap$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StringCoding$StringDecoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.ArrayDecoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CharsetDecoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded sun.nio.cs.UTF_8$Decoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CodingErrorAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Hashtable$EntrySet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$SynchronizedCollection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$SynchronizedSet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Objects from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Enumeration from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Iterator from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Hashtable$Enumerator from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Runtime from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Version from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileInputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileDescriptor from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaIOFileDescriptorAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileDescriptor$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.SharedSecrets from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.Flushable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.OutputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileOutputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FilterInputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.BufferedInputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded
java.util.concurrent.atomic.AtomicReferenceFieldUpdater from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded
java.util.concurrent.atomic.AtomicReferenceFieldUpdater$Atomi
cReferenceFieldUpdaterImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.PrivilegedExceptionAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded
java.util.concurrent.atomic.AtomicReferenceFieldUpdater$Atomi
cReferenceFieldUpdaterImpl$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.reflect.misc.ReflectUtil from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FilterOutputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.PrintStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.BufferedOutputStream from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.Writer from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.OutputStreamWriter from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.StreamEncoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.security.action.GetPropertyAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.ArrayEncoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CharsetEncoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.UTF_8$Encoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.ByteBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.HeapByteBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.Bits from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.ByteOrder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaNioAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.Bits$1 from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.BufferedWriter from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.DefaultFileSystem from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded java.io.FileSystem from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.UnixFileSystem from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.ExpiringCache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.LinkedHashMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.ExpiringCache$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Enum from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.File$PathStatus from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.file.Watchable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.file.Path from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.StringCoding$StringEncoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassLoader$3 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.ExpiringCache$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.LinkedHashMap$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassLoader$NativeLibrary from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Terminator from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.SignalHandler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Terminator$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Signal from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.NativeSignalHandler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Integer$IntegerCache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.OSEnvironment from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaLangAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.System$2 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.IllegalArgumentException from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Compiler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Compiler$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URLStreamHandlerFactory from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Launcher$Factory from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.security.util.Debug from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassLoader$ParallelLoaders from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.WeakHashMap$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Collections$SetFromMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.WeakHashMap$KeySet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaNetAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URLClassLoader$7 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.StringTokenizer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Launcher$ExtClassLoader$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.MetaIndex from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Readable from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.Reader from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.BufferedReader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.InputStreamReader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileReader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.cs.StreamDecoder from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.CharBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.HeapCharBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CoderResult from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CoderResult$Cache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CoderResult$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.charset.CoderResult$2 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.reflect.Array from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.HashMap$TreeNode from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileInputStream$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.www.ParseUtil from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.BitSet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Locale from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.util.locale.LocaleObjectCache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Locale$Cache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.locks.Lock from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.locks.ReentrantLock from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$Segment from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$Node from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$CounterCell
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$CollectionView
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$KeySetView
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$ValuesView
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.ConcurrentHashMap$EntrySetView
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded sun.util.locale.BaseLocale from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.util.locale.BaseLocale$Cache from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.util.locale.BaseLocale$Key from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.util.locale.LocaleObjectCache$CacheEntry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.Locale$LocaleKey from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.util.locale.LocaleUtils from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.CharacterData from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.CharacterDataLatin1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.Parts from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URLStreamHandler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.www.protocol.file.Handler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaSecurityAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.ProtectionDomain$JavaSecurityAccessImpl
from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaSecurityProtectionDomainAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.ProtectionDomain$2 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.ProtectionDomain$Key from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.Principal from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.HashSet from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.URLClassPath from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.www.protocol.jar.Handler from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Launcher$AppClassLoader$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.SystemClassLoaderAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleImpl from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

```
[Loaded java.lang.invoke.MethodHandleImpl$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.function.Function from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleImpl$2 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleImpl$3 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassValue from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleImpl$4 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassValue$Entry from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassValue$Identity from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.ClassValue$Version from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MemberName$Factory from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleStatics from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.invoke.MethodHandleStatics$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.PostVMInitHook from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.usagetracker.UsageTrackerClient from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.concurrent.atomic.AtomicBoolean from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.usagetracker.UsageTrackerClient$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.usagetracker.UsageTrackerClient$4 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.usagetracker.UsageTrackerClient$3 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FileOutputStream$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.launcher.LauncherHelper from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URLClassLoader$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.util.URLUtil from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.URLClassPath$3 from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.URLClassPath$Loader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.URLClassPath$JarLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.zip.ZipConstants from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.zip.ZipFile from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.JavaUtilZipFileAccess from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.util.zip.ZipFile$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.URLClassPath$FileLoader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Resource from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.URLClassPath$FileLoader$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.ByteBuffered from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.PerfCounter from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Perf$GetPerfAction from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.Perf from /srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.misc.PerfCounter$CoreCounters from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.nio.ch.DirectBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.MappedByteBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.DirectByteBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.LongBuffer from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.nio.DirectLongBufferU from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.PermissionCollection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.Permissions from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.net.URLConnection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.www.URLConnection from
```

```
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.www.protocol.file.FileURLConnection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded sun.net.www.MessageHeader from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FilePermission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FilePermission$1 from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.io.FilePermissionCollection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.AllPermission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.UnresolvedPermission from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.security.BasicPermissionCollection from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded hello from file:/root/java/]
[Loaded sun.launcher.LauncherHelper$FXHelper from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Class$MethodArray from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Void from /srv/jdk1.8.0_60/jre/lib/rt.jar]
Hello
[Loaded java.lang.Shutdown from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
[Loaded java.lang.Shutdown$Lock from
/srv/jdk1.8.0_60/jre/lib/rt.jar]
```

java.io.tmpdir

临时文件目录

```
java -Djava.io.tmpdir=/path/to/tmpdir
```

显示版本号

```
$ java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed
mode)
```

列出java模块

```
neo@MacBook-Pro ~ % java --list-modules
java.activation@10.0.1
java.base@10.0.1
java.compiler@10.0.1
java.corba@10.0.1
java.datatransfer@10.0.1
java.desktop@10.0.1
java.instrument@10.0.1
java.jnlp@10.0.1
java.logging@10.0.1
java.management@10.0.1
java.management.rmi@10.0.1
java.naming@10.0.1
java.prefs@10.0.1
java.rmi@10.0.1
java.scripting@10.0.1
java.se@10.0.1
java.se.ee@10.0.1
java.security.jgss@10.0.1
java.security.sasl@10.0.1
java.smartcardio@10.0.1
java.sql@10.0.1
java.sql.rowset@10.0.1
java.transaction@10.0.1
java.xml@10.0.1
java.xml.bind@10.0.1
java.xml.crypto@10.0.1
java.xml.ws@10.0.1
java.xml.ws.annotation@10.0.1
javafx.base@10.0.1
javafx.controls@10.0.1
```

```
javafx.deploy@10.0.1
javafx.fxml@10.0.1
javafx.graphics@10.0.1
javafx.media@10.0.1
javafx.swing@10.0.1
javafx.web@10.0.1
jdk.accessibility@10.0.1
jdk.aot@10.0.1
jdk.attach@10.0.1
jdk.charsets@10.0.1
jdk.compiler@10.0.1
jdk.crypto.cryptoki@10.0.1
jdk.crypto.ec@10.0.1
jdk.deploy@10.0.1
jdk.deploy.controlpanel@10.0.1
jdk.dynalink@10.0.1
jdk.editpad@10.0.1
jdk.hotspot.agent@10.0.1
jdk.httpserver@10.0.1
jdk.incubator.httpclient@10.0.1
jdk.internal.ed@10.0.1
jdk.internal.jvmstat@10.0.1
jdk.internal.le@10.0.1
jdk.internal.opt@10.0.1
jdk.internal.vm.ci@10.0.1
jdk.internal.vm.compiler@10.0.1
jdk.internal.vm.compiler.management@10.0.1
jdk.jartool@10.0.1
jdk.javadoc@10.0.1
jdk.javaws@10.0.1
jdk.jcmd@10.0.1
jdk.jconsole@10.0.1
jdk.jdeps@10.0.1
jdk.jdi@10.0.1
jdk.jdwp.agent@10.0.1
jdk.jfr@10.0.1
jdk.jlink@10.0.1
jdk.jshell@10.0.1
jdk.jsobject@10.0.1
jdk.jstatd@10.0.1
jdk.localedata@10.0.1
jdk.management@10.0.1
jdk.management.agent@10.0.1
jdk.management.cmm@10.0.1
jdk.management.jfr@10.0.1
```

```
jdk.management.resource@10.0.1
jdk.naming.dns@10.0.1
jdk.naming.rmi@10.0.1
jdk.net@10.0.1
jdk.pack@10.0.1
jdk.packager@10.0.1
jdk.packager.services@10.0.1
jdk.plugin@10.0.1
jdk.plugin.server@10.0.1
jdk.rmic@10.0.1
jdk.scripting.nashorn@10.0.1
jdk.scripting.nashorn.shell@10.0.1
jdk.sctp@10.0.1
jdk.security.auth@10.0.1
jdk.security.jgss@10.0.1
jdk.snmp@10.0.1
jdk.unsupported@10.0.1
jdk.xml.bind@10.0.1
jdk.xml.dom@10.0.1
jdk.xml.ws@10.0.1
jdk.zipfs@10.0.1
oracle.desktop@10.0.1
oracle.net@10.0.1
```

模块所在位置 /Library/Java/JavaVirtualMachines/jdk-10.0.1.jdk/Contents/Home/jmods

```
cd /Library/Java/JavaVirtualMachines/jdk-
10.0.1.jdk/Contents/Home/jmods

neo@MacBook-Pro /Library/Java/JavaVirtualMachines/jdk-
10.0.1.jdk/Contents/Home/jmods % ll -1
java.activation.jmod
java.base.jmod
java.compiler.jmod
java.corba.jmod
java.datatransfer.jmod
java.desktop.jmod
java.instrument.jmod
```

```
java.jnlp.jmod
java.logging.jmod
java.management.jmod
java.management.rmi.jmod
java.naming.jmod
java.prefs.jmod
java.rmi.jmod
java.scripting.jmod
java.se.ee.jmod
java.se.jmod
java.security.jgss.jmod
java.security.sasl.jmod
java.smartcardio.jmod
java.sql.jmod
java.sql.rowset.jmod
java.transaction.jmod
java.xml.bind.jmod
java.xml.crypto.jmod
java.xml.jmod
java.xml.ws.annotation.jmod
java.xml.ws.jmod
javafx.base.jmod
javafx.controls.jmod
javafx.deploy.jmod
javafx.fxml.jmod
javafx.graphics.jmod
javafx.media.jmod
javafx.swing.jmod
javafx.web.jmod
jdk.accessibility.jmod
jdk.aot.jmod
jdk.attach.jmod
jdk.charsets.jmod
jdk.compiler.jmod
jdk.crypto.cryptoki.jmod
jdk.crypto.ec.jmod
jdk.deploy.controlpanel.jmod
jdk.deploy.jmod
jdk.dynalink.jmod
jdk.editpad.jmod
jdk.hotspot.agent.jmod
jdk.httpserver.jmod
jdk.incubator.httpclient.jmod
jdk.internal.ed.jmod
jdk.internal.jvmstat.jmod
```

```
jdk.internal.le.jmod
jdk.internal.opt.jmod
jdk.internal.vm.ci.jmod
jdk.internal.vm.compiler.jmod
jdk.internal.vm.compiler.management.jmod
jdk.jartool.jmod
jdk.javadoc.jmod
jdk.javaws.jmod
jdk.jcmd.jmod
jdk.jconsole.jmod
jdk.jdeps.jmod
jdk.jdi.jmod
jdk.jdwp.agent.jmod
jdk.jfr.jmod
jdk.jlink.jmod
jdk.jshell.jmod
jdk.jsobject.jmod
jdk.jstatd.jmod
jdk.localedata.jmod
jdk.management.agent.jmod
jdk.management.cmm.jmod
jdk.management.jfr.jmod
jdk.management.jmod
jdk.management.resource.jmod
jdk.naming.dns.jmod
jdk.naming.rmi.jmod
jdk.net.jmod
jdk.pack.jmod
jdk.packager.jmod
jdk.packager.services.jmod
jdk.plugin.jmod
jdk.plugin.server.jmod
jdk.rmic.jmod
jdk.scripting.nashorn.jmod
jdk.scripting.nashorn.shell.jmod
jdk.sctp.jmod
jdk.security.auth.jmod
jdk.security.jgss.jmod
jdk.snmp.jmod
jdk.unsupported.jmod
jdk.xml.bind.jmod
jdk.xml.dom.jmod
jdk.xml.ws.jmod
jdk.zipfs.jmod
oracle.desktop.jmod
```

```
oracle.net.jmod
```

1.8. jar

查看包中的文件列表 jar -tf package.war/package.jar

```
$ /srv/java/bin/jar -tf mis.netkiller.cn-0.0.1.war |more
META-INF/
META-INF/MANIFEST.MF
WEB-INF/
WEB-INF/jsp/
WEB-INF/jsp/include/
WEB-INF/jsp/system/
WEB-INF/jsp/banner/
```

1.9. jdeps - Java class dependency analyzer.

包类依赖分析器

```
[net@netkiller lib]$ jdeps jersey-client-1.18.1.jar
jersey-client-1.18.1.jar -> not found
jersey-client-1.18.1.jar ->
/usr/java/jdk1.8.0_73/jre/lib/rt.jar
   com.sun.jersey.api.client (jersey-client-1.18.1.jar)
      -> com.sun.jersey.api.client.async
jersey-client-1.18.1.jar
      -> com.sun.jersey.api.client.config
jersey-client-1.18.1.jar
      -> com.sun.jersey.api.client.filter
jersey-client-1.18.1.jar
      -> com.sun.jersey.client.impl
jersey-client-1.18.1.jar
      -> com.sun.jersey.client.impl.async
jersey-client-1.18.1.jar
```

```
      -> com.sun.jersey.client.proxy
jersey-client-1.18.1.jar
      -> com.sun.jersey.client.urlconnection
jersey-client-1.18.1.jar
      -> com.sun.jersey.core.header                          not
found
      -> com.sun.jersey.core.provider                        not
found
      -> com.sun.jersey.core.reflection                      not
found
      -> com.sun.jersey.core.spi.component                   not
found
      -> com.sun.jersey.core.spi.component.ioc               not
found
      -> com.sun.jersey.core.spi.factory                     not
found
      -> com.sun.jersey.core.util                            not
found
      -> com.sun.jersey.spi                                  not
found
      -> com.sun.jersey.spi.inject                           not
found
      -> com.sun.jersey.spi.service                          not
found
      -> java.io
      -> java.lang
      -> java.lang.annotation
      -> java.lang.reflect
      -> java.net
      -> java.util
      -> java.util.concurrent
      -> java.util.logging
      -> javax.ws.rs.core                                    not
found
      -> javax.ws.rs.ext                                     not
found
   com.sun.jersey.api.client.async (jersey-client-1.18.1.jar)
      -> com.sun.jersey.api.client
jersey-client-1.18.1.jar
      -> java.lang
      -> java.util.concurrent
   com.sun.jersey.api.client.config (jersey-client-1.18.1.jar)
      -> com.sun.jersey.core.util                            not
found
      -> java.lang
```

```
      -> java.util
   com.sun.jersey.api.client.filter (jersey-client-1.18.1.jar)
      -> com.sun.jersey.api.client
jersey-client-1.18.1.jar
      -> com.sun.jersey.core.util                              not
found
      -> java.io
      -> java.lang
      -> java.net
      -> java.nio.charset
      -> java.security
      -> java.util
      -> java.util.logging
      -> java.util.regex
      -> java.util.zip
      -> javax.ws.rs                                           not
found
      -> javax.ws.rs.core                                      not
found
   com.sun.jersey.client.impl (jersey-client-1.18.1.jar)
      -> com.sun.jersey.api.client
jersey-client-1.18.1.jar
      -> com.sun.jersey.core.header                            not
found
      -> java.io
      -> java.lang
      -> java.net
      -> java.util
      -> java.util.concurrent.atomic
      -> javax.ws.rs.core                                      not
found
   com.sun.jersey.client.impl.async (jersey-client-1.18.1.jar)
      -> com.sun.jersey.api.client
jersey-client-1.18.1.jar
      -> com.sun.jersey.api.client.async
jersey-client-1.18.1.jar
      -> java.lang
      -> java.util.concurrent
   com.sun.jersey.client.proxy (jersey-client-1.18.1.jar)
      -> com.sun.jersey.api.client
jersey-client-1.18.1.jar
      -> com.sun.jersey.api.client.async
jersey-client-1.18.1.jar
      -> java.lang
      -> java.util.concurrent
```

```
   com.sun.jersey.client.urlconnection (jersey-client-
1.18.1.jar)
      -> com.sun.jersey.api.client
jersey-client-1.18.1.jar
      -> com.sun.jersey.core.header                          not
found
      -> com.sun.jersey.spi                                  not
found
      -> java.io
      -> java.lang
      -> java.lang.reflect
      -> java.net
      -> java.security
      -> java.util
      -> java.util.logging
      -> javax.net.ssl
      -> javax.ws.rs.core                                    not
found
   com.sun.ws.rs.ext (jersey-client-1.18.1.jar)
      -> com.sun.jersey.core.spi.factory                     not
found
      -> java.lang
      -> javax.ws.rs.core                                    not
found
```

1.10. JShell

JShell，即 Java Shell。从java9开始，java开始引入了 REPL（Read-Eval-Print Loop，读取-求值-输出 循环）工具

```
neo@MacBook-Pro ~ % jshell
|  Welcome to JShell -- Version 12
|  For an introduction type: /help intro

jshell>
```

/help 显示帮助信息

```
jshell> /help
|  Type a Java language expression, statement, or declaration.
|  Or type one of the following commands:
|  /list [<name or id>|-all|-start]
|       list the source you have typed
|  /edit <name or id>
|       edit a source entry
|  /drop <name or id>
|       delete a source entry
|  /save [-all|-history|-start] <file>
|       Save snippet source to a file
|  /open <file>
|       open a file as source input
|  /vars [<name or id>|-all|-start]
|       list the declared variables and their values
|  /methods [<name or id>|-all|-start]
|       list the declared methods and their signatures
|  /types [<name or id>|-all|-start]
|       list the type declarations
|  /imports
|       list the imported items
|  /exit [<integer-expression-snippet>]
|       exit the jshell tool
|  /env [-class-path <path>] [-module-path <path>] [-add-
modules <modules>] ...
|       view or change the evaluation context
|  /reset [-class-path <path>] [-module-path <path>] [-add-
modules <modules>]...
|       reset the jshell tool
|  /reload [-restore] [-quiet] [-class-path <path>] [-module-
path <path>]...
|       reset and replay relevant history -- current or
previous (-restore)
|  /history [-all]
|       history of what you have typed
|  /help [<command>|<subject>]
|       get information about using the jshell tool
|  /set editor|start|feedback|mode|prompt|truncation|format ...
|       set configuration information
|  /? [<command>|<subject>]
```

```
|       get information about using the jshell tool
|  /!
|       rerun last snippet -- see /help rerun
|  /<id>
|       rerun snippets by ID or ID range -- see /help rerun
|  /-<n>
|       rerun n-th previous snippet -- see /help rerun
|
|  For more information type '/help' followed by the name of a
|  command or a subject.
|  For example '/help /list' or '/help intro'.
|
|  Subjects:
|
|  intro
|       an introduction to the jshell tool
|  keys
|       a description of readline-like input editing
|  id
|       a description of snippet IDs and how use them
|  shortcuts
|       a description of keystrokes for snippet and command
completion,
|       information access, and automatic code generation
|  context
|       a description of the evaluation context options for
/env /reload and /reset
|  rerun
|       a description of ways to re-evaluate previously entered
snippets
```

介绍信息

```
jshell> /help intro
|
|                                    intro
|                                    =====
|
|  The jshell tool allows you to execute Java code, getting
```

```
immediate results.
|  You can enter a Java definition (variable, method, class,
etc), like:  int x = 8
|  or a Java expression, like:  x + x
|  or a Java statement or import.
|  These little chunks of Java code are called 'snippets'.
|
|  There are also the jshell tool commands that allow you to
understand and
|  control what you are doing, like:  /list
|
|  For a list of commands: /help
```

退出命令

```
jshell> /exit
|  Goodbye
```

1.11. jlink

使用 jlink 定自己的 jre

```
jlink --module-path jmods --add-modules java.desktop --output
/tmp/jre-16.0.2
```

查看输出结果

```
neo@MacBook-Pro-Neo ~ % ll /tmp/jre-16.0.2
total 8
drwxr-xr-x   4 neo  wheel   128B  8 31 15:08 bin
```

```
drwxr-xr-x   5 neo  wheel   160B  8 31 15:08 conf
drwxr-xr-x   8 neo  wheel   256B  8 31 15:08 include
drwxr-xr-x   7 neo  wheel   224B  8 31 15:08 legal
drwxr-xr-x  38 neo  wheel   1.2K  8 31 15:08 lib
drwxr-xr-x   3 neo  wheel    96B  8 31 15:08 man
-rw-r--r--   1 neo  wheel    93B  8 31 15:08 release
```

查看 release 文件

```
neo@MacBook-Pro-Neo ~ % cat /tmp/jre-16.0.2/release
JAVA_VERSION="16.0.2"
MODULES="java.base java.datatransfer java.xml java.prefs
java.desktop"
```

2. Java 相关命令

2.1. jps

```
参数      说明
-l        输出主类全名或jar路径
-q        只输出LVMID
-m        输出JVM启动时传递给main()的参数
-v        输出JVM启动时显示指定的JVM参数
```

```
neo@MacBook-Pro-M2 ~> jps
6560 Jps
6273 Launcher
6274 TestThread
88034 Eclipse
6404 XMLServerLauncher
97828 org.eclipse.equinox.launcher_1.6.400.v20210924-0641.jar
1176
2891
98015 BootLanguageServerBootApp

neo@MacBook-Pro-M2 ~> jps -m
6273 Launcher /Applications/IntelliJ IDEA
CE.app/Contents/plugins/java/lib/jps-
builders.jar:/Applications/IntelliJ IDEA
CE.app/Contents/plugins/java/lib/jps-builders-
6.jar:/Applications/IntelliJ IDEA
CE.app/Contents/plugins/java/lib/jps-javac-
extension.jar:/Applications/IntelliJ IDEA
CE.app/Contents/lib/util.jar:/Applications/IntelliJ IDEA
CE.app/Contents/lib/util-8.jar:/Applications/IntelliJ IDEA
CE.app/Contents/lib/util_rt.jar:/Applications/IntelliJ IDEA
CE.app/Contents/lib/annotations.jar:/Applications/IntelliJ IDEA
CE.app/Contents/lib/3rd-party-rt.jar:/Applications/IntelliJ
IDEA CE.app/Contents/lib/protobuf.jar:/Applications/IntelliJ
IDEA CE.app/Contents/lib/jps-model.jar:/Applications/IntelliJ
```

```
IDEA
CE.app/Contents/plugins/java/lib/javac2.jar:/Applications/Intel
liJ IDEA
CE.app/Contents/lib/forms_rt.jar:/Applications/IntelliJ IDEA
CE.app/Contents/plugins/java/lib/aether-dependency-
resolver.jar:/Applications/IntelliJ IDEA
CE.app/Contents/lib/idea_rt.jar:/Applications/IntelliJ IDEA
CE.ap
6274 TestThread
6562 Jps -m
88034 Eclipse
6404 XMLServerLauncher
97828 org.eclipse.equinox.launcher_1.6.400.v20210924-0641.jar -
configuration /Users/neo/Library/Application
Support/Code/User/globalStorage/redhat.java/1.17.0/config_mac -
data /Users/neo/Library/Application
Support/Code/User/workspaceStorage/c0c3760503bc4a3db6b66b0155e5
0c5b/redhat.java/jdt_ws
1176
2891
98015 BootLanguageServerBootApp

neo@MacBook-Pro-M2 ~> jps -l
6273 org.jetbrains.jps.cmdline.Launcher
6274 cn.netkiller.TestThread
88034 Eclipse
6564 jdk.jcmd/sun.tools.jps.Jps
6404 org.eclipse.lemminx.XMLServerLauncher
97828 /Users/neo/.vscode/extensions/redhat.java-1.17.0-darwin-
arm64/server/plugins/org.eclipse.equinox.launcher_1.6.400.v2021
0924-0641.jar
1176
2891
98015
org.springframework.ide.vscode.boot.app.BootLanguageServerBootA
pp
```

3. System

```
Java.version 运行时环境版本

java.vendor 运行时环境供应商

java.vendor.url 供应商的 URL

java.home 安装目录

java.vm.specification.version 虚拟机规范版本

java.vm.specification.vendor 虚拟机规范供应商

java.vm.specification.name 虚拟机规范名称

java.vm.version 虚拟机实现版本

java.vm.vendor 虚拟机实现供应商

java.vm.name 虚拟机实现名称

java.specification.version 运行时环境规范版本

java.specification.vendor Java 运行时环境规范供应商

java.specification.name Java 运行时环境规范名称

java.class.version Java 类格式版本号

java.class.path Java 类路径

java.library.path 加载库时搜索的路径列表

java.io.tmpdir 默认的临时文件路径

java.compiler 要使用的 JIT 编译器的名称

java.ext.dirs 一个或多个扩展目录的路径
```

```
os.name 操作系统的名称

os.arch 操作系统的架构

os.version 操作系统的版本

file.separator 文件分隔符（在 UNIX 系统中是“/”）

path.separator 路径分隔符（在 UNIX 系统中是“:”）

line.separator 行分隔符（在 UNIX 系统中是“/n”）

user.name 用户的账户名称

user.home 用户的主目录

user.dir 用户的当前工作目录
```

3.1. user.dir

```
public class Test {

        public static void main(String[] args) {
                System.out.println("Working Directory = " +
System.getProperty("user.dir"));


System.out.println(System.getProperty("user.home"));

System.out.println(System.getProperty("java.version"));

System.out.println(System.getProperty("os.name"));

System.out.println(System.getProperty("java.vendor.url"));
        }
}
```

3.2. java.io.tmpdir

改变java.io.tmpdir的默认值

```
System.setProperty("java.io.tmpdir", "/vat/tmp");

System.out.println(System.getProperty("java.io.tmpdir"));
```

如果你希望从外部修改这个系统属性的话，你可以使用-D这个参数，例如 java -Djava.io.tmpdir=/path/to/tmpdir

```
System.out.println(System.getProperty("java.io.tmpdir"));
```

3.3. 打印当前 Java 文件的默认编码

```
package cn.netkiller.test;

import java.nio.charset.Charset;

public class Test {

        public Test() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {


System.out.println(System.getProperty("file.encoding"));
                System.out.println(Charset.defaultCharset());

        }
```

```
}
```

3.4. 自定义

```
        public static void main(String[] args) {
                try {
                        Oracle oracle = new Oracle();
                        if(System.getProperty("config") ==
null){
                                System.exit(1);
                        }

oracle.openConfig(System.getProperty("config"));
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }

        }
```

上面程序编译打包后运行

```
neo@netkiller:~/project/Oracle$ java -jar -
Dconfig=jdbc.properties target/oracle-0.0.1-SNAPSHOT.jar
```

3.5. System.in 标准输入(Stdin)

标准输入一般用于管道输入，例如：

```
```

```
cat test.txt | java cn.netkiller.ipo.test.StdinToStdout
```

下面的程序例子里从管道输入，并从标准输出打印。

```
package cn.netkiller.ipo.test;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class StdinToStdout {

	public StdinToStdout() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {

		String s = "";
		try {
			BufferedReader stdIn = new
BufferedReader(new InputStreamReader(System.in));

			while ((s = stdIn.readLine()) !=
null) {
				System.out.println(s);
			}

		} catch (Exception e) {
			e.printStackTrace();
		}

	}
}
```

默认 BufferedReader 缓冲区比较小，无法处理大于1000 行的输入，可以通过配置缓冲区来解决。

```
stdin = new BufferedReader(new InputStreamReader(System.in,
"utf-8"), 1024 * 1024 * 1024); // 1G 缓冲区
```

4. exec 运行shell

```
exec(String[] cmdarray, String[] envp, File dir)
Executes the specified command and arguments in a separate
process with the specified environment and working directory.

那个dir就是调用的程序的工作目录，这句其实还是很有用的。

Windows下调用程序
Process proc =Runtime.getRuntime().exec("file.exe");

Linux下调用程序就要改成下面的格式
Process proc =Runtime.getRuntime().exec("./file");

Windows下调用系统命令
String [] cmd={"cmd","/C","copy file1.txt file2.txt"};
Process proc =Runtime.getRuntime().exec(cmd);

Linux下调用系统命令就要改成下面的格式
String [] cmd={"/bin/sh","-c","ln -s file1 file2"};
Process proc =Runtime.getRuntime().exec(cmd);

Windows下调用系统命令并弹出命令行窗口
String [] cmd={"cmd","/C","start copy file1.txt file2.txt"};
Process proc =Runtime.getRuntime().exec(cmd);

Linux下调用系统命令并弹出终端窗口就要改成下面的格式
String [] cmd={"/bin/sh","-c","xterm -e ln -s file1 file2"};
Process proc =Runtime.getRuntime().exec(cmd);

还有要设置调用程序的工作目录就要
Process proc =Runtime.getRuntime().exec("ls",null, new
File("/bin"));
```

5. 数据类型

数据类型的最大值和最小值。

```
基本类型：int 二进制位数：32
包装类：java.lang.Integer
最小值：Integer.MIN_VALUE= -2147483648  (-2的31次方)
最大值：Integer.MAX_VALUE= 2147483647   (2的31次方-1)

基本类型：short 二进制位数：16
包装类：java.lang.Short
最小值：Short.MIN_VALUE=-32768  (-2的15此方)
最大值：Short.MAX_VALUE=32767  (2的15次方-1)

基本类型：long 二进制位数：64
包装类：java.lang.Long
最小值：Long.MIN_VALUE=-9223372036854775808  (-2的63次方)
最大值：Long.MAX_VALUE=9223372036854775807  (2的63次方-1)

基本类型：float 二进制位数：32
包装类：java.lang.Float
最小值：Float.MIN_VALUE=1.4E-45  (2的-149次方)
最大值：Float.MAX_VALUE=3.4028235E38  (2的128次方-1)

基本类型：double 二进制位数：64
包装类：java.lang.Double
最小值：Double.MIN_VALUE=4.9E-324  (2的-1074次方)
最大值：Double.MAX_VALUE=1.7976931348623157E308  (2的1024次方-1)
```

5.1. var 本地变量类型推断

```
var javastack = "javastack";
就等于：
String javastack = "javastack";
```

5.2. Integer

十进制转成十六进制：

```
Integer.toHexString(int i)
```

十进制转成八进制

```
Integer.toOctalString(int i)
```

十进制转成二进制

```
Integer.toBinaryString(int i)
```

十六进制转成十进制

```
Integer.valueOf("FFFF",16).toString()
```

八进制转成十进制

```
Integer.valueOf("876",8).toString()
```

二进制转十进制

```
Integer.valueOf("0101",2).toString()
```

有什么方法可以直接将2,8,16进制直接转换为10进制的吗?

java.lang.Integer类

```
parseInt(String s, int radix)
```

使用第二个参数指定的基数，将字符串参数解析为有符号的整数。

examples from jdk:

```
parseInt("0", 10) returns 0

parseInt("473", 10) returns 473

parseInt("-0", 10) returns 0

parseInt("-FF", 16) returns -255

parseInt("1100110", 2) returns 102

parseInt("2147483647", 10) returns 2147483647

parseInt("-2147483648", 10) returns -2147483648

parseInt("2147483648", 10) throws a NumberFormatException

parseInt("99",throws a NumberFormatException

parseInt("Kona", 10) throws a NumberFormatException

parseInt("Kona", 27) returns 411787
```

例二

```
int i=100;

String binStr=Integer.toBinaryString(i);
String otcStr=Integer.toOctalString(i);
String hexStr=Integer.toHexString(i);
System.out.println(binStr);
```

例二

```
Integer factor = Integer.valueOf(args[0]);

String s;
```

```
s = String.format("%d", factor);
System.out.println(s);

s = String.format("%x", factor);
System.out.println(s);

s = String.format("%o", factor);
System.out.println(s);
```

其他方法:

```
Integer.toHexString(你的10进制数);
```

例如

```
String temp = Integer.toHexString(75);
```

输出temp就为 4b

前面补零

```
public static String frontCompWithZore(int sourceDate,int
formatLength) {

    String newString = String.format("%0"+formatLength+"d",
sourceDate);
    return newString;
}
```

5.3. String

Java 11 增加了一系列的字符串处理方法，如以下所示。

```
// 判断字符串是否为空白
" ".isBlank(); // true
// 去除首尾空格
" Javastack ".strip(); // "Javastack"
// 去除尾部空格
" Javastack ".stripTrailing(); // " Javastack"
// 去除首部空格
" Javastack ".stripLeading(); // "Javastack "
```

查找字符重现的位置

```
package cn.netkiller.test;

public class Test {

        public static void main(String[] args) {

                String string = new
String("http://www.netkiller.cn");

                System.out.println("查找字符 . 第一次出现的位置:
" + string.indexOf('.'));
                System.out.println("从第15个字符位置查找字符 . 出
现的位置:" + string.indexOf('.', 15));
        }
}
```

行数统计

```
"A\nB\nC".lines().count(); // 3
```

复制字符串

```
"Java".repeat(3);// "JavaJavaJava"
```

随机字符串

```
        public String randomString(String chars, int length)
{
                Random rand = new Random();
                StringBuilder buf = new StringBuilder();
                for (int i = 0; i < length; i++) {

buf.append(chars.charAt(rand.nextInt(chars.length())));
                }
                return buf.toString();
        }

        /**
     * 获取4位随机验证码
     * @return
     */
    public static String getValidationCode(){
        return String.valueOf((new Random().nextInt(8999) +
1000));
    }

    /**
     * 获取6位随机验证码
     * @return
     */
    public static String getValidationCode(){
        return String.valueOf((new Random().nextInt(899999) +
```

```
100000));
    }
```

字符串替换处理

```
public class Test {

        public Test() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                System.out.println("2010-09-
11T20:00:30".replace("T", " "));
        }
}
```

```
{"status":0,"message":"","bankcode":"ABOC;IBC;CCTB;ICBC"}
转换后
{\"status\":0,\"message\":\"\",\"bankcode\":\"ABOC;IBC;CCTB;I
CBC\"}
```

```
package test;

public class str {

        public static void main(String[] args) {
```

```
                String jsonString = "
{\"status\":0,\"message\":\"\",\"bankcode\":\"ABOC;IBC;CCTB;I
CBC\"}";
                System.out.println(jsonString);
                System.out.println(jsonString.replace("\"",
"\\\""));
        }

}
```

正则表达式查找与替换

查找特定字符并替换为找到的内容

```
package cn.netkiller.type;

public class ragexTest {

        public ragexTest() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                String str = "<html>Netkiller</html>";
                String regex = "<html>|</html>";
                //运行结果返回 Netkiller
                System.out.println(str.replaceAll(regex,
""));

                // 运行结果返回 Neo

System.out.println("CN/NETKILLER/WWW/Neo_Chen".replaceAll(".*
/(.+)_Chen", "$1"));
        }
}
```

利用正则快速转换时间格式

```
// 20200303 => 2020-03-03
date = date.replace(/(.{4})/, "$1-");
date = date.replace(/(.{7})/, "$1-");
```

用正则处理 Alertmanager 日期 rfc3339 格式 2021-08-23T01:30:30.2206015+08:00，Alertmanager 日期格式中有7位数的毫秒，使用 SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSXXX") 无法正常读取。

```
package cn.netkiller.alertmanager;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.TimeZone;

public class Test {

        public Test() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws
ParseException {
                // TODO Auto-generated method stub
                String rfc3339 = "2021-08-
23T01:30:30.2206015+08:00";
                rfc3339 = rfc3339.replaceAll(".[0-9]{7}",
"");

                System.out.println(rfc3339);
```

```
                System.out.println(rfc3339.replaceAll("([0-9]
{4})-([0-9]{2})-([0-9]{2})T([0-9]{2}):([0-9]{2}):([0-9]{2})+
(.*)", "$1年$2月$3日 $4时$5分$6秒"));
                System.out.println(rfc3339.replaceAll("([0-9]
{4}-[0-9]{2}-[0-9]{2})T([0-9]{2}:[0-9]{2}:[0-9]{2})+(.+)", "日
期: $1 时间: $2"));

                // SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSXXX");
                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssXXX");

simpleDateFormat.setTimeZone(TimeZone.getTimeZone("GMT+8"));

System.out.println(simpleDateFormat.parse(rfc3339).toString()
);

        }

}
```

```
2021-08-23T01:30:30+08:00
2021年08月23日 01时30分30秒
日期: 2021-08-23 时间: 01:30:30
Mon Aug 23 01:30:30 CST 2021
```

replaceAll

替换一组字符串

```
        String str1 = "广东省，福建省，北京市，海淀区，河北省，上海
市";
        str1 = str1.replaceAll("(省|市|区)", "");
        System.out.println("替换多个中文: " + str1);
```

```
        替换多个中文：广东，福建，北京，海淀，河北，上海
```

替换 Ascii

```
                String string = "佛山市123南海区ABC精密abc机械有限公司□□□□□□,";

                String clean = string.replaceAll("\\P{Print}", "");
                System.out.println(clean);
```

字符串处理，删除中文以外的字符

Unicode 码表 https://www.ssec.wisc.edu/~tomw/java/unicode.html

```
        String string = "2017-12-18 netkiller http://www.netkiller.cn -  网站正常";
        System.out.println(string.replaceAll("[^\u4e00-\u9fa5]", ""));
```

取出字符串中的中文字符

Unicode 码表 https://www.ssec.wisc.edu/~tomw/java/unicode.html

```
                String string = "2017-12-18 netkiller http://www.netkiller.cn -  网站正常";
                String regex = "[\u4e00-\u9fa5]";
                System.out.println(string.replaceAll(regex,
```

```
""));
```

substring

```
例如：
String str = "helloword!!!";

System.out.println(str.substring(1,4));

System.out.println(str.substring(3,5));

System.out.println(str.substring(0,4));

将得到结果为：

ell

lo

hell
```

string to timestamp

Timestamp转化为String:

```
                                    SimpleDateFormat df = new
                                    SimpleDateFormat("yyyy-MM-dd
HH:mm:ss"); //定义格式，不显示毫秒
                                    Timestamp now =
                                    new
Timestamp(System.currentTimeMillis());
                                    //获取系统当前时间
                                    String str =
```

```
                                    df.format(now);
```

String转化为Timestamp:

```
                                    SimpleDateFormat df = new
                                    SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                                    String time = df.format(new
                                    Date());
                                    Timestamp ts =
Timestamp.valueOf(time);
```

String.strip

java11对String类新增了strip，stripLeading以及stripTrailing方法，除了strip相关的方法还新增了isBlank、lines、repeat(int)等方法

```
        @Test
    public void testStrip(){
        String text = "  \u2000a  b  ";
        Assert.assertEquals("a  b",text.strip());
        Assert.assertEquals("\u2000a  b",text.trim());
        Assert.assertEquals("a  b  ",text.stripLeading());
        Assert.assertEquals("  \u2000a
b",text.stripTrailing());
    }
```

文本块

```
package cn.netkiller.kubernetes;
```

```
public class Test {

        public Test() {
        }

        public static void main(String[] args) {

                String json = """
                                {
                                "name" : "netkiller",
                                "home" :
"https://www.netkiller.cn/"
                                }

""";
                System.out.println(json);
        }

}
```

分割字符串

```
        String[] string = topic.split("/");
        Log.d("Service", string.length + "");
        for (String str: string) {
            Log.d("Service", str);
        }

                String str2 = new String("www.netkiller.cn");
        for (String retval: str2.split("\\.", 3)){
            System.out.println(retval);
        }

        System.out.println("");
        String str3 = new String("acount=? and uu =? or n=?");
        for (String retval: str3.split("and|or")){
            System.out.println(retval);
        }
```

5.4. 类型转换

Long to String

```
        public class MainClass {

          public static void main(String[] arg) {
            long b = 12L;
            System.out.println(String.valueOf(b));

          }
        }
```

5.5. Date

java.util.Date, java.sql.Date, java.sql.Time, java.sql.Timestamp 区别

```
package cn.netkiller.java.date;

/**
 * Hello world!
 *
 */
public class App {
        public static void main(String[] args) {
                System.out.println("Hello World!");

                // Get standard date and time
                java.util.Date utilDate = new
java.util.Date();
                long javaTime = utilDate.getTime();
```

```
                System.out.println("The Java Date is:" +
utilDate.toString());

                // Get and display SQL DATE
                java.sql.Date sqlDate = new
java.sql.Date(javaTime);
                System.out.println("The SQL DATE is: " +
sqlDate.toString());

                // Get and display SQL TIME
                java.sql.Time sqlTime = new
java.sql.Time(javaTime);
                System.out.println("The SQL TIME is: " +
sqlTime.toString());

                // Get and display SQL TIMESTAMP
                java.sql.Timestamp sqlTimestamp = new
java.sql.Timestamp(javaTime);
                System.out.println("The SQL TIMESTAMP is: " +
sqlTimestamp.toString());
        }
}
```

```
The Java Date is:Thu Aug 24 16:51:57 CST 2017
The SQL DATE is: 2017-08-24
The SQL TIME is: 16:51:57
The SQL TIMESTAMP is: 2017-08-24 16:51:57.234
```

SimpleDateFormat

字符串转日期

```
        try {
```

```
            DateFormat df = new SimpleDateFormat("yyyy-MM-
dd");
            String str = "2018-12-11";
            Date date = df.parse(str);
            System.out.println(date);

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
public static void main(String[] args) {

        DateFormat dateFormat = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
        //get current date time with Date()
        Date date = new Date();
        System.out.println(dateFormat.format(date));

        //get current date time with Calendar()
        Calendar cal = Calendar.getInstance();
        System.out.println(dateFormat.format(cal.getTime()));

}
```

不规范的 RFC3339 日期

```
                String dt = "2021-08-
26T09:17:23.859291804+08:00";
                String rfc3339 = dt.replaceAll("\\.([0-9]+)",
"");
                System.out.println(dt);
                System.out.println(rfc3339);
```

```
                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssXXX");

simpleDateFormat.setTimeZone(TimeZone.getTimeZone("GMT+8"));

System.out.println(simpleDateFormat.parse(rfc3339).toLocaleSt
ring());
```

Timestamp

```
Timestamp timestamp = new
Timestamp(System.currentTimeMillis());

Date date = new Date();
Timestamp timestamp = new Timestamp(date.getTime());
```

TimeZone

```
package cn.netkiller.example;

import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.TimeZone;

public class TimeZoneTest {

        public TimeZoneTest() {
                // TODO Auto-generated constructor stub
        }
```

```
        public static void main(String[] args) {
                // TODO Auto-generated method stub

                SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

                TimeZone timeZone =
TimeZone.getTimeZone("Asia/Harbin");

                Date date = new Date();
                Timestamp timestamp = new
Timestamp(date.getTime());

                System.out.println(timestamp);

                timestamp.setHours(timestamp.getHours()+8);
                System.out.println(timestamp);


simpleDateFormat.setTimeZone(TimeZone.getTimeZone("GMT"));

System.out.println(simpleDateFormat.format(date));


simpleDateFormat.setTimeZone(TimeZone.getTimeZone("Asia/Harbin"
));

System.out.println(simpleDateFormat.format(date));

                Calendar calendar = new GregorianCalendar();
                calendar.setTime(date);
                calendar.setTimeZone(timeZone);

System.out.println(simpleDateFormat.format(calendar.getTime()))
;
        }

}
```

String to Date

```
package cn.netkiller.example;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class StringToDate {

        public StringToDate() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SimpleDateFormat formatter = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                String dateString = "2008-8-8 8:8:8";

                try {

                        Date date =
formatter.parse(dateString);
                        System.out.println(date);

System.out.println(formatter.format(date));

                } catch (ParseException e) {
                        e.printStackTrace();
                }
        }

}
```

比较两个日期与时间

```
package cn.netkiller.example;
```

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateCompare {

	public DateCompare() {
		// TODO Auto-generated constructor stub
	}

	public void fun1() throws InterruptedException {
		Date d1 = new Date();
		Thread.sleep(5000);
		Date d2 = new Date();
		if (d1.before(d2)) {
			System.out.println(String.format("%s
< %s", d1.toString(), d2.toString()));
		} else {
			System.out.println(String.format("%s
> %s", d1.toString(), d2.toString()));
		}
		if (d2.after(d1)) {
			System.out.println(String.format("%s
> %s", d2.toString(), d1.toString()));
		}

		System.out.println(String.format("%s : %s =>
%d", d2.toString(), d1.toString(), d1.compareTo(d2)));
		System.out.println(String.format("%s : %s =>
%d", d1.toString(), d2.toString(), d2.compareTo(d1)));
	}

	public void fun2() throws InterruptedException {

		DateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

		Date date1 = new Date();
		Date date2 = new Date();
		String s1 = dateFormat.format(date1);
		String s2 = dateFormat.format(date2);
		System.out.println(String.format("%s : %s =>
%d", s1, s2, s1.compareTo(s2)));
```

```
                date1 = new Date();
                Thread.sleep(5000);
                date2 = new Date();
                s1 = dateFormat.format(date1);
                s2 = dateFormat.format(date2);
                System.out.println(String.format("%s : %s =>
%d", s1, s2, s1.compareTo(s2)));
                System.out.println(String.format("%s : %s =>
%d", s2, s1, s2.compareTo(s1)));
                System.out.println();
        }

        public void fun3() throws InterruptedException,
ParseException {
                DateFormat formatter = new
SimpleDateFormat("yyyy-MM-dd HH:mm");
                //Date time = formatter.parse("2016-09-27
16:29");
                Date time = formatter.parse("2016-08-09
09:15");
                Date startDate = formatter.parse("2016-08-09
09:15");
                Date endDate = formatter.parse("2016-09-27
16:30");

                if (time.before(startDate) ||
time.after(endDate)) {
                        System.out.println("Skip");
                }
        }

        public static void main(String[] args) throws
InterruptedException {
                // TODO Auto-generated method stub
                DateCompare dateCompare = new DateCompare();
                dateCompare.fun1();
                System.out.println();
                dateCompare.fun2();
                System.out.println();
                dateCompare.fun3();
        }

}
```

Calendar

```
                Calendar cal = Calendar.getInstance();
                int year = cal.get(Calendar.YEAR);
                int month = cal.get(Calendar.MONTH )+1;

                System.out.println(year + "年 " + month +
"月");
```

getToday

```
        public Date getToday(String time) {
                final Calendar cal = Calendar.getInstance();
                DateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd " + time);
                DateFormat fmt = new SimpleDateFormat("yyyy-
MM-dd HH:mm:ss");
                Date date = null;
                try {
                        date =
fmt.parse(dateFormat.format(cal.getTime()));
                } catch (ParseException e) {
                        e.printStackTrace();
                }
                return date;
        }

        private Date addOneDay(Date date, int day) {
                Calendar cal = Calendar.getInstance();
                cal.setTime(date);
                cal.add(Calendar.DATE, day);
                return cal.getTime();
```

```
        }
```

Yesterday

```
package cn.netkiller.date;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class Yesterday {

        public Yesterday() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Yesterday yesterday = new Yesterday();
                System.out.println(yesterday.yesterday());

System.out.println(yesterday.getYesterday("00:00:00"));

System.out.println(yesterday.getYesterday("23:59:59"));
        }
        private Date yesterday() {
            final Calendar cal = Calendar.getInstance();
            cal.add(Calendar.DATE, -1);
            return cal.getTime();
        }

        private Date getYesterday(String time) {
                DateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd "+time);

                DateFormat fmt =new SimpleDateFormat("yyyy-
MM-dd HH:mm:ss");
```

```
                Date date = null;
                try {
                    date =
fmt.parse(dateFormat.format(yesterday()));
                } catch (ParseException e) {
                    e.printStackTrace();
                }
                return date;
        }
}
```

ISO 8601

```
ISO 8601扩展格式 YYYY-MM-DDTHH:mm:ss.sssZ
```

LocalDateTime

```
        LocalDateTime localDateTime = LocalDateTime.of(2016,
1, 1, 13, 55);
        ZonedDateTime zonedDateTime =
localDateTime.atZone(ZoneId.of("Asia/Shanghai"));
        Date date = Date.from(zonedDateTime.toInstant());

        Instant instant =
Instant.ofEpochMilli(date.getTime());
    LocalDateTime ldt = LocalDateTime.ofInstant(instant,
ZoneOffset.UTC);
    Instant instant = ldt.toInstant(ZoneOffset.UTC);
    Date date = Date.from(instant);
```

ZonedDateTime

```
Date.from(java.time.ZonedDateTime.now().toInstant());
```

5.6. Array

一定字符串数组

```
String[] str={"AAA","BBB","CCC"};
String str[]={"AAA","BBB","CCC"};
```

String to Array

```
package cn.netkiller.java;

public class StringToArray {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		String
str="a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z";
		String[] array = null;
		array = str.split(",");
		for(int i=0; i<array.length; i++){
			System.out.println(array[i]);
		}
	}
}
```

for each

```
        public static void main(String[] args) {
                try {

                        for (String arg : args) {
                                System.out.println(arg);
                        }
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }
        }
```

Array to String

```
package cn.netkiller.java;

import java.util.Arrays;

public class ArrayToString {

        public static void main(String[] args) {
                String[] array = {"a", "b", "c", "d", "e",
"f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q",
"r", "s", "t", "u", "v", "w", "x", "y", "z"};
                System.out.println(Arrays.toString(array));

System.out.println(Arrays.toString(array).replaceAll(", |\\
[|\\]", ""));
        }

}
```

```
String[][] deepArray = new String[][] {{"John", "Mary"},
{"Alice", "Bob"}};
System.out.println(Arrays.toString(deepArray));
//output: [[Ljava.lang.String;@106d69c,
[Ljava.lang.String;@52e922]
System.out.println(Arrays.deepToString(deepArray));

Output:
[[John, Mary], [Alice, Bob]]
```

```
String[] array = {"neo", "chen"};
String string = String.join(",", array)
// 输出结果 "neo,chen"
```

5.7. float

float 不能直接做减法运算

```
float a = 77.22f;
float b = 77.2f;

System.out.println((float)a-b);
System.out.println((float)a+b);

输出结果为:
0.020004272
154.42
```

```
package cn.netkiller.example;
```

```
import java.math.BigDecimal;

public class Math {

	public Math() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {

		float a = 77.22f;
		float b = 77.2f;

		System.out.println((float) a + b);
		System.out.println((float) a - b);
		System.out.println((float) a * b);
		System.out.println((float) a / b);

		System.out.println("-------------");

		System.out.println(add(a, b));
		System.out.println(sub(a, b));
		System.out.println(mul(a, b));
		System.out.println(div(a, b));

	}

	public static float add(float v1, float v2) {
		BigDecimal b1 = new
BigDecimal(Float.toString(v1));
		BigDecimal b2 = new
BigDecimal(Float.toString(v2));
		return b1.add(b2).floatValue();
	}

	public static float sub(float v1, float v2) {
		BigDecimal b1 = new
BigDecimal(Float.toString(v1));
		BigDecimal b2 = new
BigDecimal(Float.toString(v2));
		return b1.subtract(b2).floatValue();
	}

	public static float mul(float v1, float v2) {
```

```
                BigDecimal b1 = new
BigDecimal(Float.toString(v1));
                BigDecimal b2 = new
BigDecimal(Float.toString(v2));
                return b1.multiply(b2).floatValue();
        }

        public static float div(float v1, float v2) {
                return div(v1, v2, 5);
        }

        public static float div(float v1, float v2, int
scale) {
                if (scale < 0) {
                        throw new
IllegalArgumentException("The scale must be a positive
integer or zero");
                }
                BigDecimal b1 = new
BigDecimal(Float.toString(v1));
                BigDecimal b2 = new
BigDecimal(Float.toString(v2));
                return b1.divide(b2, scale,
BigDecimal.ROUND_HALF_UP).floatValue();
        }

        public static float round(float v, int scale) {
                if (scale < 0) {
                        throw new
IllegalArgumentException("The scale must be a positive
integer or zero");
                }
                BigDecimal b = new
BigDecimal(Float.toString(v));
                BigDecimal one = new BigDecimal("1");
                return b.divide(one, scale,
BigDecimal.ROUND_HALF_UP).floatValue();
        }
}
```

5.8. double

```
package cn.netkiller.example;

import java.math.BigDecimal;

public class Math {

        public Math() {
                // TODO Auto-generated constructor stub
        }

        public static double add(double v1, double v2) {
                BigDecimal b1 = new
BigDecimal(Double.toString(v1));
                BigDecimal b2 = new
BigDecimal(Double.toString(v2));
                return b1.add(b2).doubleValue();
        }

        public static double sub(double v1, double v2) {
                BigDecimal b1 = new
BigDecimal(Double.toString(v1));
                BigDecimal b2 = new
BigDecimal(Double.toString(v2));
                return b1.subtract(b2).doubleValue();
        }

        public static double mul(double v1, double v2) {
                BigDecimal b1 = new
BigDecimal(Double.toString(v1));
                BigDecimal b2 = new
BigDecimal(Double.toString(v2));
                return b1.multiply(b2).doubleValue();
        }

        public static double div(double v1, double v2) {
                return div(v1, v2, 8);
        }

        public static double div(double v1, double v2, int
```

```
scale) {
                if (scale < 0) {
                        throw new
IllegalArgumentException("The scale must be a positive
integer or zero");
                }
                BigDecimal b1 = new
BigDecimal(Double.toString(v1));
                BigDecimal b2 = new
BigDecimal(Double.toString(v2));
                return b1.divide(b2, scale,
BigDecimal.ROUND_HALF_UP).doubleValue();
        }

        public static double round(double v, int scale) {
                if (scale < 0) {
                        throw new
IllegalArgumentException("The scale must be a positive
integer or zero");
                }
                BigDecimal b = new
BigDecimal(Double.toString(v));
                BigDecimal one = new BigDecimal("1");
                return b.divide(one, scale,
BigDecimal.ROUND_HALF_UP).doubleValue();
        }
}
```

String to double

```
double amount = Double.parseDouble(value);
```

百分数转Double

```
import java.text.NumberFormat;
import java.text.ParseException;
try {

    Double num =
(Double)NumberFormat.getInstance().parse("67.89%"); // 转换的结
果是67.89
    Double num2 =
(Double)NumberFormat.getPercentInstance().parse("67.89%"); //
转换的结果是0.6789

    System.out.println(num);
} catch (ParseException e) {
    e.printStackTrace();
}
```

Double转百分数

```
import java.text.NumberFormat;
import java.text.ParseException;

try {
    NumberFormat percentInstance =
NumberFormat.getPercentInstance();
    percentInstance.setMaximumFractionDigits(2); // 保留小数两
位
    String format = percentInstance.format(0.81247); // 结果是
81.25% , 最后一们四舍五入了

    System.out.println(num);
} catch (ParseException e) {
    e.printStackTrace();
}
```

5.9. BigDecimal

```
package cn.netkiller.example;

import java.math.BigDecimal;

public class BigDecimalTest {

	public BigDecimalTest() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		BigDecimal first = new BigDecimal("1.0");
		BigDecimal second = new BigDecimal("1.77");
		System.out.println(String.format("%s, %s",
first, second));

		if (first.equals(second))

			System.out.println("equals: true");

		else

			System.out.println("equals: false");

		if (first.compareTo(second) == 0)

			System.out.println("compareTo: true");
		else
			System.out.println("compareTo: false");

		BigDecimal zero = new BigDecimal("0");
		BigDecimal one = new BigDecimal("1");
		BigDecimal minus = new BigDecimal("-1");

		if (zero.compareTo(one) < 0)

			System.out.println("比較演算子[ <  ]:
true");
```

```
                if (one.compareTo(one) == 0)
                        System.out.println("比較演算子[ == ]:
true");
                if (zero.compareTo(minus) > 0)
                        System.out.println("比較演算子[ >  ]:
true");
                if (zero.compareTo(minus) >= 0)
                        System.out.println("比較演算子[ >= ]:
true");
                if (zero.compareTo(minus) != 0)
                        System.out.println("比較演算子[ != ]:
true");
                if (zero.compareTo(one) <= 0)
                        System.out.println("比較演算子[ <= ]:
true");
        }
}
```

Convert BigDecimal Object to double value

```
BigDecimal.doubleValue()
```

去除末尾多余的0

```
System.out.println( new
BigDecimal("100.000").stripTrailingZeros().toString());
```

禁用科学计数法

```
有时会输出 1E+2，如果你不希望这种科学计数法输出可以使用
toPlainString() 替代 toString()
System.out.println( new
BigDecimal("100.000").stripTrailingZeros().toPlainString());
```

移动小数点位置

```
package cn.netkiller.example.test;

import java.math.BigDecimal;
import java.math.BigInteger;

public class Test {

        public Test() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                int decimal = 4;

                BigInteger amount =
BigInteger.valueOf(10000000000L);
```

```
                BigDecimal balance = new BigDecimal(amount);
                BigDecimal point = new BigDecimal(0.1 /
Math.pow(10, decimal));
                balance = balance.multiply(point);
                System.out.println(balance);
        }

}
```

发现输出有问题
100000.000000000008180305391403130954586231382563710212707519531250000000000

换种方法

```
package cn.netkiller.example.test;

import java.math.BigDecimal;
import java.math.BigInteger;

public class Test {

        public Test() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                // String i =
Integer.valueOf("0x57c457",16).toString();
                // System.out.println(i);

                int decimal = 6;

                BigInteger amount =
BigInteger.valueOf(10000000000L);

                System.out.println(amount);
```

```
                String tmp = amount.toString();

                String number = new
StringBuffer(tmp).insert(tmp.length() - decimal,
".").toString();
                BigDecimal balance = new BigDecimal(number);

                System.out.println(balance);
        }

}
```

最佳方案

```
                int decimal = 6;


System.out.println(BigDecimal.TEN.pow(decimal));
                BigDecimal balance1 = new BigDecimal("1234");
                BigDecimal value =
balance1.divide(BigDecimal.TEN.pow(decimal));
                System.out.println(value);

                BigDecimal balance2 = new
BigDecimal("12.107");
                BigDecimal value2 =
balance2.multiply(BigDecimal.TEN.pow(decimal)).setScale(0,
RoundingMode.DOWN);
                System.out.println(value2);
```

5.10. StringBuffer

```

```

```
String str = Integer.toString(j);
str = new StringBuilder(str).insert(str.length()-2,
".").toString();

Or if you need synchronization use the StringBuffer with
similar usage :

String str = Integer.toString(j);
str = new StringBuffer(str).insert(str.length()-2,
".").toString();
```

5.11. enum

```
class EnumExample1{

        public enum Season { WINTER, SPRING, SUMMER, FALL }

        public static void main(String[] args) {
                for (Season s : Season.values())
                        System.out.println(s);
        }
}
```

```
```

```
package cn.netkiller.api.util;

public enum HttpMethod {
    GET("GET"), POST("POST"), PUT("PUT"), PATCH("PATCH"),
DELETE("DELETE");

    private String value;
```

```
    private HttpMethod(String value) {
        this.value = value;
    }

    public String toString() {
        return value;
    }
}
```

5.12. byte 类型

string2byte

```
        byte[] bytes = "Helloworld!!! -
http://www.netkiller.cn".getBytes();
```

byte[] to String

```
        byte[] bytes = "Helloworld!!! -
http://www.netkiller.cn".getBytes();
        String str = new String(bytes,
StandardCharsets.UTF_8);
        System.out.println(str);
```

BigInteger2byte

```
BigInteger b= new BigInteger('300');
byte bytes= b.byteValue();
```

int to byte array

```
int a= 120;
byte b= (byte)a;
```

```
private byte[] bigIntToByteArray( final int i ) {
    BigInteger bigInt = BigInteger.valueOf(i);
    return bigInt.toByteArray();
}
```

```
private byte[] intToByteArray ( final int i ) throws
IOException {
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    DataOutputStream dos = new DataOutputStream(bos);
    dos.writeInt(i);
    dos.flush();
    return bos.toByteArray();
}
```

```

```

```
private byte[] intToBytes( final int i ) {
    ByteBuffer bb = ByteBuffer.allocate(4);
    bb.putInt(i);
    return bb.array();
}
```

位移操作

```
private static byte[] intToBytes(final int a) {
    return new byte[] {
        (byte)((data >> 24) & 0xff),
        (byte)((data >> 16) & 0xff),
        (byte)((data >> 8) & 0xff),
        (byte)((data >> 0) & 0xff),
    };
}
```

byte array to int

```
private int convertByteArrayToInt(byte[] intBytes){
    ByteBuffer byteBuffer = ByteBuffer.wrap(intBytes);
    return byteBuffer.getInt();
}
```

```
private int convertByteArrayToInt(byte[] data) {
    if (data == null || data.length != 4) return 0x0;
    // ----------
    return (int)( // NOTE: type cast not necessary for int
            (0xff & data[0]) << 24  |
```

```
            (0xff & data[1]) << 16  |
            (0xff & data[2]) << 8   |
            (0xff & data[3]) << 0
            );
}
```

```
        public static byte[] intToByte32(int num) {
        byte[] arr = new byte[32];
        for (int i = 31; i >= 0; i--) {
            // &1 也可以改为num&0x01,表示取最地位数字.
            arr[i] = (byte) (num & 1);
            // 右移一位.
            num >>= 1;
        }
        return arr;
    }


        public static int byte32ToInt(byte[] arr) {
        if (arr == null || arr.length != 32) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且长度是32!");
        }
        int sum = 0;
        for (int i = 0; i < 32; ++i) {
            sum |= (arr[i] << (31 - i));
        }
        return sum;
    }
```

int array to byte array

```
private byte[] convertIntArrayToByteArray(int[] data) {
        if (data == null) return null;
```

```
        // ----------
        byte[] byts = new byte[data.length * 4];
        for (int i = 0; i < data.length; i++)
            System.arraycopy(convertIntToByteArray(data[i]),
0, byts, i * 4, 4);
        return byts;
    }
```

byte array to int array

```
public int[] convertByteArrayToIntArray(byte[] data) {
        if (data == null || data.length % 4 != 0) return
null;
        // ----------
        int[] ints = new int[data.length / 4];
        for (int i = 0; i < ints.length; i++)
            ints[i] = ( convertByteArrayToInt(new byte[] {
                    data[(i*4)],
                    data[(i*4)+1],
                    data[(i*4)+2],
                    data[(i*4)+3],
            } ));
        return ints;
    }
```

byte2char

```
        byte b1 = 65;
    // char ch = b1;

    char ch = (char) b1;

    System.out.println("byte value: " + b1);             //
```

```
prints 65
    System.out.println("Converted char value: " + ch);   //
prints A (ASCII is 65 for A)
```

```
        public static char byte2ToChar(byte[] arr) {
        if (arr == null || arr.length != 2) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且是2位!");
        }
        return (char) (((char) (arr[0] << 8)) | ((char)
arr[1]));
    }

    public static byte[] charToByte2(char c) {
        byte[] arr = new byte[2];
        arr[0] = (byte) (c >> 8);
        arr[1] = (byte) (c & 0xff);
        return arr;
    }
```

longToByte64

```
    public static byte[] longToByte64(long sum) {
        byte[] arr = new byte[64];
        for (int i = 63; i >= 0; i--) {
            arr[i] = (byte) (sum & 1);
            sum >>= 1;
        }
        return arr;
    }
```

byte64ToLong

```
        public static long byte8ToLong(byte[] arr) {
        if (arr == null || arr.length != 8) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且是8位!");
        }
        return (long) (((long) (arr[0] & 0xff) << 56) |
((long) (arr[1] & 0xff) << 48) | ((long) (arr[2] & 0xff) <<
40)
                        | ((long) (arr[3] & 0xff) << 32) |
((long) (arr[4] & 0xff) << 24)
                        | ((long) (arr[5] & 0xff) << 16) |
((long) (arr[6] & 0xff) << 8) | ((long) (arr[7] & 0xff)));
    }

        public static byte[] longToByte8(long sum) {
        byte[] arr = new byte[8];
        arr[0] = (byte) (sum >> 56);
        arr[1] = (byte) (sum >> 48);
        arr[2] = (byte) (sum >> 40);
        arr[3] = (byte) (sum >> 32);
        arr[4] = (byte) (sum >> 24);
        arr[5] = (byte) (sum >> 16);
        arr[6] = (byte) (sum >> 8);
        arr[7] = (byte) (sum & 0xff);
        return arr;
    }

        public static long byte64ToLong(byte[] arr) {
        if (arr == null || arr.length != 64) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且长度是64!");
        }
        long sum = 0L;
        for (int i = 0; i < 64; ++i) {
            sum |= ((long) arr[i] << (63 - i));
        }
        return sum;
    }
```

short2byte

```
    public static byte[] shortToByte16(short s) {
        byte[] arr = new byte[16];
        for (int i = 15; i >= 0; i--) {
            arr[i] = (byte) (s & 1);
            s >>= 1;
        }
        return arr;
    }

    public static short byte16ToShort(byte[] arr) {
        if (arr == null || arr.length != 16) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且长度为16!");
        }
        short sum = 0;
        for (int i = 0; i < 16; ++i) {
            sum |= (arr[i] << (15 - i));
        }
        return sum;
    }

    public static short byte2ToShort(byte[] arr) {
        if (arr != null && arr.length != 2) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且是2位!");
        }
        return (short) (((short) arr[0] << 8) | ((short)
arr[1] & 0xff));
    }

    public static byte[] shortToByte2(Short s) {
        byte[] arr = new byte[2];
        arr[0] = (byte) (s >> 8);
        arr[1] = (byte) (s & 0xff);
        return arr;
    }
```

```
```

byte8ToDouble

```
        public static double byte8ToDouble(byte[] arr) {
        if (arr == null || arr.length != 8) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且是8位!");
        }
        long l = byte8ToLong(arr);
        return Double.longBitsToDouble(l);
    }

    public static byte[] doubleToByte8(double i) {
        long j = Double.doubleToLongBits(i);
        return longToByte8(j);
    }
```

byte4ToFloat

```
    public static float byte4ToFloat(byte[] arr) {
        if (arr == null || arr.length != 4) {
            throw new IllegalArgumentException("byte数组必须不
为空,并且是4位!");
        }
        int i = byte4ToInt(arr);
        return Float.intBitsToFloat(i);
    }

    public static byte[] floatToByte4(float f) {
        int i = Float.floatToIntBits(f);
        return intToByte4(i);
    }
```

无符号 byte

```
byte b= -120;
int a= bytes & 0xff;
```

```
byte a = (byte)234;
System.out.println(a);

上面的代码，结果是-22，因为java中byte是有符号的，byte范围是-128~127。

如果想输出234，该怎么做呢，首先想到的是将a 赋给大一点的类型，如下：

byte a = (byte)234;
int i = a&0xff;
System.out.println(i);

原因是 0xff是int，占4个字节，a是byte，占1个字节，进行&操作的细节如下：

   00000000 00000000 00000000 11101010    (a)
&
   00000000 00000000 00000000 11111111    (0xff)
----------------------------------------------------------------
--------
=  00000000 00000000 00000000 11101010

结果是int，但是符号位是0，说明是正数，最后就是正整数234.
```

byte to hex

```
byte bv = 10;
String hexString = Integer.toHexString(bv);
```

byte[] to hex

```
byte bytes[] = {(byte)0, (byte)0, (byte)134, (byte)0,
(byte)61};
System.out.println(javax.xml.bind.DatatypeConverter.printHexB
inary(bytes));
```

```
public static String byteArrayToHex(byte[] bytes) {
   StringBuilder sb = new StringBuilder(a.length * 2);
   for(byte b: bytes)
      sb.append(String.format("%02x", b));
   return sb.toString();
}
```

```
BigInteger n = new BigInteger(byteArray);
String hexa = n.toString(16));
```

连接两个 byte[]

```
byte[] one = { 1, 2, 3 };
byte[] two = { 6, 8, 9 };
int length = one.Length + two.Length;
byte[] sum = new byte[length];
one.CopyTo(sum,0);
two.CopyTo(sum,one.Length);
```

```
byte[] one = { 1, 2, 3 };
byte[] two = { 6, 8, 9 };
List<byte> list1 = new List<byte>(one);
List<byte> list2 = new List<byte>(two);
list1.AddRange(list2);
byte[] sum2 = list1.ToArray();
```

List<Byte> to byte[]

```
                List<Byte> byteList = new ArrayList<Byte>();
        byteList.add((byte) 1);
        byteList.add((byte) 2);
        byteList.add((byte) 3);
        byte[] byteArray = Bytes.toArray(byteList);
        System.out.println(Arrays.toString(byteArray));
```

5.13. 布尔型 Boolean

将boolean转换成Boolean

```
Boolean b1 = new Boolean(true);
Boolean b2 = Boolean.valueOf(true);
```

将Boolean转boolean

```
Boolean b3=new Boolean(true);
boolean b4=b3.booleanValue();
```

将String转Boolean

```
String s="true";
Boolean bo1=new Boolean(s);
Boolean bo2=Boolean.valueOf(s);
```

将Boolean转String

```
Boolean boo=new Boolean(true);
String s1=boo. toString();
String s2=Boolean.toString(boo);
String s3=String.valueOf(boo);
```

boolean转String

```
boolean  b =true;
String st1 = String.valueOf(b);
```

String转boolean

```
String str="true";
boolean boo11=Boolean.parseBoolean(str);
Boolean boo12=new Boolean(str);
boolean bool3=boo12.booleanValue();
```

6. 流程控制

6.1. Switch

yield

```
package cn.netkiller.demo;

public class DemoSwitch {

        public DemoSwitch() {

        }

        public static void main(String[] args) {

                var number = 4;
                var operation = "平方";
                var result = switch (operation) {
                case "立方" -> {
                        yield number * number * number;
                }
                case "平方" -> {
                        yield number * number;
                }
                default -> number;
                };
                System.out.println(result);

        }

}
```

不必为break每个 case 块定义一个语句，我们可以简单地使用箭头语法

```
int money = 3;
String cn = switch (money) {
case 1 -> "壹";
case 2 -> "贰";
case 3 -> "叁";
case 4 -> "肆";
case 5 -> "伍";
case 6 -> "陆";
case 7 -> "柒";
case 8 -> "捌";
case 9 -> "玖";
case 10 -> "拾";
default -> "零";
};
System.out.println(cn);
```

7. 异常处理

7.1. 抛出异常

```
throw new NullPointerException();
throw new IOException();
throw new FileNotFoundException("文件不存在");
throw new ArrayIndexOutOfBoundsException("数字范围越界");
```

8. 数据结构

8.1. List

```
Collection
 ├List
 │ ├LinkedList
 │ ├ArrayList
 │ └Vector
 │   └Stack
 └Set
```

```
Collection接口
    Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素（Elements）。一些 Collection允许相同的元素而另一些不行。一些能排序而另一些不行。Java SDK不提供直接继承自Collection的类，Java SDK提供的类都是继承自Collection的“子接口”如List和Set。
    所有实现Collection接口的类都必须提供两个标准的构造函数：无参数的构造函数用于创建一个空的Collection，有一个 Collection参数的构造函数用于创建一个新的Collection，这个新的Collection与传入的Collection有相同的元素。后一个构造函数允许用户复制一个Collection。
    如何遍历Collection中的每一个元素？不论Collection的实际类型如何，它都支持一个iterator()的方法，该方法返回一个迭代子，使用该迭代子即可逐一访问Collection中每一个元素。典型的用法如下：
        Iterator it = collection.iterator(); // 获得一个迭代子
        while(it.hasNext()) {
            Object obj = it.next(); // 得到下一个元素
        }
    由Collection接口派生的两个接口是List和Set。
List接口
    List是有序的Collection，使用此接口能够精确的控制每个元素插入的位置。用户能够使用索引（元素在List中的位置，类似于数组下标）来访问List中的元素，这类似于Java的数组。
和下面要提到的Set不同，List允许有相同的元素。
    除了具有Collection接口必备的iterator()方法外，List还提供一个listIterator()方法，返回一个 ListIterator接口，和标准的Iterator接口相
```

比，ListIterator多了一些add()之类的方法，允许添加，删除，设定元素， 还能向前或向后遍历。

实现List接口的常用类有LinkedList，ArrayList，Vector和Stack。

LinkedList类

LinkedList实现了List接口，允许null元素。此外LinkedList提供额外的get，remove，insert方法在 LinkedList的首部或尾部。这些操作使LinkedList可被用作堆栈（stack），队列（queue）或双向队列（deque）。

注意LinkedList没有同步方法。如果多个线程同时访问一个List，则必须自己实现访问同步。一种解决方法是在创建List时构造一个同步的List：

```
List list = Collections.synchronizedList(new LinkedList(...));
```

ArrayList类

ArrayList实现了可变大小的数组。它允许所有元素，包括null。ArrayList没有同步。

size，isEmpty，get，set方法运行时间为常数。但是add方法开销为分摊的常数，添加n个元素需要O(n)的时间。其他的方法运行时间为线性。

每个ArrayList实例都有一个容量（Capacity），即用于存储元素的数组的大小。这个容量可随着不断添加新元素而自动增加，但是增长算法 并没有定义。当需要插入大量元素时，在插入前可以调用ensureCapacity方法来增加ArrayList的容量以提高插入效率。

和LinkedList一样，ArrayList也是非同步的（unsynchronized）。

Vector类

Vector非常类似ArrayList，但是Vector是同步的。由Vector创建的Iterator，虽然和 ArrayList创建的Iterator是同一接口，但是，因为Vector是同步的，当一个Iterator被创建而且正在被使用，另一个线程改变了 Vector的状态（例如，添加或删除了一些元素），这时调用Iterator的方法时将抛出ConcurrentModificationException，因此必须捕获该异常。

Stack 类

Stack继承自Vector，实现一个后进先出的堆栈。Stack提供5个额外的方法使得Vector得以被当作堆栈使用。基本的push和pop 方法，还有peek方法得到栈顶的元素，empty方法测试堆栈是否为空，search方法检测一个元素在堆栈中的位置。Stack刚创建后是空栈。

Set接口

Set是一种不包含重复的元素的Collection，即任意的两个元素e1和e2都有e1.equals(e2)=false，Set最多有一个null元素。

很明显，Set的构造函数有一个约束条件，传入的Collection参数不能包含重复的元素。

请注意：必须小心操作可变对象（Mutable Object）。如果一个Set中的可变元素改变了自身状态导致Object.equals(Object)=true将导致一些问题。

Map接口

请注意，Map没有继承Collection接口，Map提供key到value的映射。一个Map中不能包含相同的key，每个key只能映射一个 value。Map接口提供3种集合的视图，Map的内容可以被当作一组key集合，一组value集合，或者一组key-value映射。

```
package cn.netkiller.example;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.TreeSet;

public class Test {

	public Test() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		// A A B E F G C D
		String[] array = { "A", "A", "B", "E", "F",
"G", "C", "D" };
		Collection<String> list = new
ArrayList<String>(Arrays.asList(array));
		for (String str : list) {
			System.out.print(str + " ");
		}
		System.out.println();

		// A A B E F G C D
		Collection<String> linkedList = new
LinkedList<String>(Arrays.asList(array));
		for (String str : linkedList) {
			System.out.print(str + " ");
		}
		System.out.println();

		// 无重复，无序 D E F G A B C
		Collection<String> hashSet = new
```

```
HashSet<String>(Arrays.asList(array));
                for (String str : hashSet) {
                        System.out.print(str + " ");
                }
                System.out.println();

                // 无重复 A B C D E F G
                Collection<String> treeSet = new
TreeSet<String>(Arrays.asList(array));
                for (String str : treeSet) {
                        System.out.print(str + " ");
                }
                System.out.println();

                // 无重复 A B E F G C D
                Collection<String> linkedHashSet = new
LinkedHashSet<String>(Arrays.asList(array));
                for (String str : linkedHashSet) {
                        System.out.print(str + " ");

                }

        }

}
```

输出结果

```
A A B E F G C D
A A B E F G C D
A B C D E F G
A B C D E F G
A B E
F G C D
```

静态 List

```
        public static List<String> list = new
ArrayList<String>();
        static {
                list.add("录入");
                list.add("变更");
                list.add("收藏");
                list.add("在售");
                list.add("展出");
        }
```

List.of()

```
List<String> strings = List.of("first", "second");
```

List.copyOf()

```
var list = List.of("Java", "Python", "C");
var copy = List.copyOf(list);
System.out.println(list == copy); // true
```

```
var list = new ArrayList<String>();
var copy = List.copyOf(list);
System.out.println(list == copy); // false
```

String[] to List

```
String[] arr = new String[] {"1", "2"};
List list = Arrays.asList(arr);
```

Stream.toList() 方法

数组转List

```
String[] arrays = {"tom", "jack", "kate"};
List<String> stringList=
Stream.of(arrays).collect(Collectors.toList());
```

```
package cn.netkiller.demo;

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class TestList {

	public TestList() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {

		List<String> list = Arrays.asList("1", "2",
"3");
		// jdk11 之前这样写
```

```
                List<Integer> list1 =
list.stream().map(Integer::parseInt).collect(Collectors.toLis
t());
                System.out.println(list1);
                // jdk16 现在可以这样写
                List<Integer> list2 =
list.stream().map(Integer::parseInt).toList();
                System.out.println(list2);

        }
}
```

containsAll

```
        List<String> inputStringList =
Arrays.asList("Neo|Netkiller|Tom|Jerry".split("|"));
        List<String> wordsList =
Arrays.asList("Neo|Tom".split("|"));

System.out.println(inputStringList.containsAll(wordsList));
```

stream().allMatch()

```
        List<String> inputStringList =
Arrays.asList("Neo|Netkiller|Tom|Jerry".split("|"));
        List<String> wordsList = Arrays.asList("Neo","Tom");

System.out.println(wordsList.stream().allMatch(inputStringLis
t::contains));
```

随机 List

```
List<String> list = Arrays.asList("a","b","c");
int index = (int) (Math.random()* list.size());
System.out.println(list.get(index));
```

8.2. ArrayList

初始化

最常用的初始化方式,先创建一个list1，再给list1赋值。

```
List<String> list1 = new ArrayList<String>();
list1.add("apple");
list1.add("banana");
list1.add("orange");
```

使用一个List来初始化

```
List<String> list2 = new ArrayList<String>
(Arrays.asList("apple", "banana", "orange"));
```

```
List<String> list3 = new ArrayList<String>
(Collections.nCopies(2, "orange"));
```

使用匿名内部类来初始化

```
List<String> list4 = new ArrayList<String>() {
     {
         add("apple");
         add("banana");
         add("orange");
     }
};
```

判断元素是否存在

判断元素是否存在

```
import java.util.ArrayList;

public class arraylist {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ArrayList<String> whitelist = new
ArrayList<String>();
                whitelist.add("Neo");
                whitelist.add("Jam");
                whitelist.add("Sam");

                if (whitelist.contains("Neo")) {
                        System.out.println("Found!");
                }else{
                        System.out.println("Not Found!");
                }
        }
```

```
}
```

循环打印

```
package cn.netkiller.type;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ArrayListExample {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		List<String> list = new ArrayList<String>();
		list.add("Jack");
		list.add("Jet");
		list.add("Jack");
		list.add("Mike");
		list.add("Kitty");
		list.add("Tom");

		//while 循环
		Iterator<String> it = list.iterator();
		while (it.hasNext()) {
			System.out.println(it.next());
		}

		for (Iterator<String> it1 = list.iterator();
it1.hasNext();) {
			System.out.println(it1.next());
		}

		// for 循环
		for (int i = 0; i < list.size(); i++) {
			System.out.println(list.get(i));
		}
```

```
                // for 循环加强版
                for (String i : list) {
                        System.out.println(i);
                }

        }

}
```

ArrayList to Array

ArrayList 转为 Array

```
                String[] array = {"/bin/sh","-c"};
                List<String> list = new ArrayList<String>
(Arrays.asList(array));
            list.add("command");
            list.add("param");

            String[] command = (String[]) list.toArray(new
String[0]);
            System.out.println(Arrays.toString(command));
```

ArrayList to String

```
                List<String> list = new ArrayList<String>();
            list.add("command");
            list.add("param");

            String listString = String.join(", ", list);

            System.out.println(listString);
```

Array to List

```
Arrays.asList(array)
```

List to Array

```
List<String> list = new ArrayList<String>();
list.add("str1");
list.add("str2");

String[] array = (String[]) list.toArray();
System.out.println(array);
```

ArrayList forEach

```
// create an ArrayList which going to
// contains a list of Numbers
ArrayList<Integer> Numbers = new ArrayList<Integer>();

// Add Number to list
Numbers.add(23);
Numbers.add(32);
Numbers.add(45);
Numbers.add(63);

// forEach method of ArrayList and
// print numbers
Numbers.forEach((n) -> System.out.println(n));
```

```
import java.util.*;
public class ArrayListTest {

        public static void main(String[] args)
        {
                // create an ArrayList which going to
                // contains a list of Student names which is
actually
                // string values
                ArrayList<String> students = new
ArrayList<String>();

                // Add Strings to list
                // each string represents student name
                students.add("Ram");
                students.add("Mohan");
                students.add("Sohan");
                students.add("Rabi");

                // print result
                System.out.println("list of Students:");

                // forEach method of ArrayList and
                // print student names
                students.forEach((n) -> print(n));
        }

        // printing student name
        public static void print(String n)
        {
                System.out.println("Student Name is " + n);
        }
}
```

```
List<String> arrayList = new ArrayList<>();
arrayList.add("A");
arrayList.add("B");
arrayList.add("C");
arrayList.add("D");
arrayList.add("E");

for (String item:arrayList){
    System.out.println(item);
}

arrayList.forEach(item->System.out.println(item));

arrayList.forEach(System.out::println);

arrayList.forEach(item->{
    if("C".equals(item)){
        System.out.println(item);
    }
});
```

```
package cn.netkiller;

import java.util.ArrayList;
import java.util.List;

public class Test {

        public Test(int id) {
                this.id = id;
        }

        public int id;
        public String name;

        public void setId(int id) {
                this.id = id;
        }

        public int getId() {
```

```
                return this.id;
        }

        public static void main(String[] args) throws
InterruptedException {
                List<Test> arrayList = new ArrayList<Test>();
                arrayList.add(new Test(1));
                arrayList.add(new Test(2));
                arrayList.add(new Test(3));
                arrayList.add(new Test(4));
                arrayList.add(new Test(5));

                arrayList.forEach(item -> {
                        System.out.println(item.getId());
                        item.setId(item.getId() + 1);
                });

                arrayList.forEach(item -> {
                        System.out.println(item.getId());
                });
        }

}
```

ArrayList stream()

```
arrayList.stream()
        .filter(s-> s.contains("B")||s.contains("C"))
        .forEach(System.out::println);

arrayList.stream()
        .filter(s->s.contains("E"))
        .findFirst().ifPresent(s -> System.out.println(s));
```

ArrarList 转换为 string[]

```
    ArrayList list = new ArrayList();

    list.Add("aaa");

    list.Add("bbb");

    string[] arrString = (string[])list.ToArray(typeof(
string)) ;
```

string 转换为 ArrayList

```
    String str="1,2,3,4,5";
    ArrayList b = new ArrayList( str.split(',') ) ;
```

ArrayList 转换为 string

```
    ArrayList list = new ArrayList();

    list.Add("aaa");

    list.Add("bbb");

    string str= string.Join(",",
(string[])list.ToArray(typeof( string)));
```

string[] 转换为 ArrarList

```
   ArrayList list = new ArrayList(new string[] { "aaa", "bbb"
});
```

合并 List<byte[]> ArrayList<byte[]>

```
        List<byte[]> audios = new ArrayList<byte[]>();

        public static byte[]
concatenateByteArrays(List<byte[]> byteArrays) {
            ByteArrayOutputStream outputStream = new
ByteArrayOutputStream();

            for (byte[] byteArray : byteArrays) {
                try {
                    outputStream.write(byteArray);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            return outputStream.toByteArray();
        }
```

8.3. Set 转为 List

```
                // 将Map Key 转化为List
        List<String> mapKeyList = new ArrayList<String>
(map.keySet());
        System.out.println("mapKeyList:"+mapKeyList);
```

```
        // 将Map Key 转化为List
        List<String> mapValuesList = new ArrayList<String>
(map.values());
        System.out.println("mapValuesList:"+mapValuesList);
```

```
Set<Type> set = new Set<>();
Set<Type> set = new HashSet<>();
```

Set.of()

```
Set<Integer> ints = Set.of(1, 2, 3);
```

Set to Array

Set.toArray(IntFunction)

```
        @Test
    public void testCollectionToArray(){
        Set<String> names = Set.of("Fred", "Wilma", "Barney",
"Betty");
        String[] copy = new String[names.size()];
        names.toArray(copy);
        System.out.println(Arrays.toString(copy));

System.out.println(Arrays.toString(names.toArray(String[]::new)
));
    }
```

8.4. Map

```
Map
 ├Hashtable
 ├HashMap
 └WeakHashMap

 Map接口
    请注意，Map没有继承Collection接口，Map提供key到value的映射。一个Map中不能包含相同的key，每个key只能映射一个 value。Map接口提供3种集合的视图，Map的内容可以被当作一组key集合，一组value集合，或者一组key-value映射。
Hashtable类
    Hashtable继承Map接口，实现一个key-value映射的哈希表。任何非空（non-null）的对象都可作为key或者value。
    添加数据使用put(key, value)，取出数据使用get(key)，这两个基本操作的时间开销为常数。
Hashtable通过initial capacity和load factor两个参数调整性能。通常缺省的load factor 0.75较好地实现了时间和空间的均衡。增大load factor可以节省空间但相应的查找时间将增大，这会影响像get和put这样的操作。
使用Hashtable的简单示例如下，将1，2，3放到Hashtable中，他们的key分别是”one”，”two”，”three”：
        Hashtable numbers = new Hashtable();
        numbers.put(“one”, new Integer(1));
        numbers.put(“two”, new Integer(2));
        numbers.put(“three”, new Integer(3));
    要取出一个数，比如2，用相应的key：
        Integer n = (Integer)numbers.get(“two”);
        System.out.println(“two = ” + n);
    由于作为key的对象将通过计算其散列函数来确定与之对应的value的位置，因此任何作为key的对象都必须实现hashCode和equals方 法。hashCode和equals方法继承自根类Object，如果你用自定义的类当作key的话，要相当小心，按照散列函数的定义，如果两个对象相 同，即obj1.equals(obj2)=true，则它们的hashCode必须相同，但如果两个对象不同，则它们的hashCode不一定不同，如 果两个不同对象的hashCode相同，这种现象称为冲突，冲突会导致操作哈希表的时间开销增大，所以尽量定义好的hashCode()方法，能加快哈希 表的操作。
    如果相同的对象有不同的hashCode，对哈希表的操作会出现意想不到的结果（期待的get方法返回null），要避免这种问题，只需要牢记一条：要同时复写equals方法和hashCode方法，而不要只写其中一个。
```

Hashtable是同步的。

HashMap类

HashMap和Hashtable类似，不同之处在于HashMap是非同步的，并且允许null，即null value和null key。，但是将HashMap视为Collection时（values()方法可返回Collection），其迭代子操作时间开销和HashMap 的容量成比例。因此，如果迭代操作的性能相当重要的话，不要将HashMap的初始化容量设得过高，或者load factor过低。

WeakHashMap类

WeakHashMap是一种改进的HashMap，它对key实行“弱引用”，如果一个key不再被外部所引用，那么该key可以被GC回收。

初始化

```
        Map<String, Object> data = new HashMap<String,
 Object>() {
                {
                        put("name", "neo");
                }
        };
```

static map

```
        private static final Map<String, String> point;
        static {
                point = new HashMap<String, String>();
                point.put("CN", "China");
                point.put("HK", "Hongkong");
                point.put("TW", "Taiwan");
        };
```

```
        public final static Map<String, String> hostMap = new
HashMap<String, String>() {
        {
            put("redis", "127.0.0.1");
            put("solr", "127.0.0.1");
        }
        };


        public final static Map map = new HashMap() {{
            put("key1", "value1");
            put("key2", "value2");
        }};
```

Collections 初始化 Map

```
Map<String, String> emptyMap = Collections.emptyMap();

public static Map<String, String> createSingletonMap() {
    return Collections.singletonMap("username1",
"password1");
}
```

使用 Collectors.toMap（）初始化 Map

数组方式

```
Map<String, String> map = Stream.of(new String[][] {
  { "Hello", "World" },
  { "Neo", "Chen" },
```

```
}).collect(Collectors.toMap(data -> data[0], data ->
data[1]));
```

对象方式

```
 Map<String, Integer> map = Stream.of(new Object[][] {
     { "Neo", 1 },
     { "Netkiller", 2 },
 }).collect(Collectors.toMap(data -> (String) data[0], data -
> (Integer) data[1]));
```

使用 Collectors.collectingAndThen（）初始化不可变的Map

```
Map<String, String> map = Stream.of(new String[][] {
    { "Hello", "World" },
    { "John", "Doe" },
}).collect(Collectors.collectingAndThen(
    Collectors.toMap(data -> data[0], data -> data[1]),
    Collections::<String, String> unmodifiableMap));
```

使用Map.Entry流初始化 Map

使用 Entry 接口的SimpleEntry 实现：

```
Map<String, Integer> map = Stream.of(
  new AbstractMap.SimpleEntry<>("idea", 1),
  new AbstractMap.SimpleEntry<>("mobile", 2))
```

```
  .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```

使用SimpleImmutableEntry 实现:

```
Map<String, Integer> map = Stream.of(
  new AbstractMap.SimpleImmutableEntry<>("idea", 1),
  new AbstractMap.SimpleImmutableEntry<>("mobile", 2))
  .collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue));
```

Map.of()

```
Map<String, String> emptyMap = Map.of();
Map<String, String> singletonMap = Map.of("key1", "value");
Map<String, String> map = Map.of("key1","value1", "key2",
"value2");
```

Map.ofEntries()

```
Map<String, String> map = Map.ofEntries(
  new AbstractMap.SimpleEntry<String, String>("name", "Neo"),
  new AbstractMap.SimpleEntry<String, String>("city",
"Shenzhen"),
  new AbstractMap.SimpleEntry<String, String>("zip",
"518000"),
```

```
  new AbstractMap.SimpleEntry<String, String>("home",
"https://www.netkiller.cn")
);
```

HashMap

遍历 HashMap

```
Map<String, Integer> session = new HashMap<String, Integer>
();

session.put("A",1);
...
...
session.put("Z",26)

for (Map.Entry<String, Integer> entry : session.entrySet()) {
        System.out.println(String.format("%s:%d",
entry.getKey(), entry.getValue()));
}

Map<Integer, Integer> map = new HashMap<Integer, Integer>
();

for (Map.Entry<Integer, Integer> entry : map.entrySet()) {

    System.out.println("Key = " + entry.getKey() + ", Value =
 " + entry.getValue());

}
```

遍历map中的键

```
Map<Integer, Integer> map = new HashMap<Integer, Integer>
();

//遍历map中的键

for (Integer key : map.keySet()) {

    System.out.println("Key = " + key);

}
```

遍历map中的值

```
Map<Integer, Integer> map = new HashMap<Integer, Integer>
();
for (Integer value : map.values()) {
    System.out.println("Value = " + value);
}
```

通过键取值

```
Map<Integer, Integer> map = new HashMap<Integer, Integer>
();

for (Integer key : map.keySet()) {

    Integer value = map.get(key);

    System.out.println("Key = " + key + ", Value = " + value)
;
```

```
}
```

使用 **Iterator** 遍历 **HashMap**

```
Map<Integer, Integer> map = new HashMap<Integer, Integer>();

Iterator<Map.Entry<Integer, Integer>> entries = map.entrySet(
).iterator();

while (entries.hasNext()) {

    Map.Entry<Integer, Integer> entry = entries.next();

    System.out.println("Key = " + entry.getKey() + ", Value =
 " + entry.getValue());

}

Map map = new HashMap();

Iterator entries = map.entrySet().iterator();

while (entries.hasNext()) {

    Map.Entry entry = (Map.Entry) entries.next();

    Integer key = (Integer)entry.getKey();

    Integer value = (Integer)entry.getValue();

    System.out.println("Key = " + key + ", Value = " + value)
;
}
```

LinkedHashMap

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
public class TestLinkedHashMap {

  public static void main(String args[])
  {
   System.out.println("*** LinkedHashMap ***");
   Map<Integer,String> map = new
LinkedHashMap<Integer,String>();
   map.put(6, "apple");
   map.put(3, "banana");
   map.put(2,"pear");

   for (Iterator it =  map.keySet().iterator();it.hasNext();)
   {
    Object key = it.next();
    System.out.println( key+"="+ map.get(key));
   }

   System.out.println("*** HashMap ***");
   Map<Integer,String> map1 = new  HashMap<Integer,String>();
   map1.put(6, "apple");
   map1.put(3, "banana");
   map1.put(2,"pear");

   for (Iterator it =
map1.keySet().iterator();it.hasNext();)
   {
    Object key = it.next();
    System.out.println( key+"="+ map1.get(key));
   }
  }
}
```

遍历数据

```
        Map<Integer, String> map = new HashMap<Integer,
String>(){{
            put(1, "127.0.0.1");
            put(2, "192.168.0.1");
            put(3, "172.16.0.1");
        }};
        for (Integer key : map.keySet()) {
            System.out.println("key= "+ key + ", value= " +
map.get(key));
        }
```

迭代器

```
        Iterator<Map.Entry<Integer, String>> iterator =
map.entrySet().iterator();
        while(iterator.hasNext()) {
            System.out.println(iterator.next());
        }
```

Map forEach

```
Map<String, Integer> items = new HashMap<>();
items.put("A", 10);
items.put("B", 20);
items.put("C", 30);
items.put("D", 40);
items.put("E", 50);
items.put("F", 60);

items.forEach((k,v)->System.out.println("key : " + k + ";
```

```
value : " + v));

//output
key : A value : 10
key : B value : 20
key : C value : 30
key : D value : 40
key : E value : 50
key : F value : 60

items.forEach((k,v)->{
    System.out.println("key : " + k + " value : " + v);
});
```

随机取值

```
        Map<Integer, String> map = new HashMap<Integer,
String>(){{
            put(1, "127.0.0.1");
            put(2, "192.168.0.1");
            put(3, "172.16.0.1");
        }};

        Integer[] keys = map.keySet().toArray(new
Integer[0]);
        Random random = new Random();
        Integer randomKey =
keys[random.nextInt(keys.length)];
        String randomValue = map.get(randomKey);

        System.out.println(randomValue);
```

8.5. Queue

```
        Queue<String> queue = new Queue<String>();
        queue.offer("1");
        queue.offer("2");
        queue.offer("3");
        queue.offer("4");
        System.out.println("当前第一个元素: " + queue.peek());//
取队列第一个元素
        System.out.println("出列第一个元素: " + queue.poll());//
出列第一个元素
        System.out.println("当前第一个元素: " + queue.peek());//
取队列第一个元素
```

阻塞队列

阻塞队列是一个可以阻塞的先进先出集合，比如某个线程在空队列获取元素时、或者在已存满队列存储元素时，都会被阻塞。

BlockingQueue 接口常用的实现类如下:

ArrayBlockingQueue： 基于数组的有界阻塞队列，必须指定大小。
LinkedBlockingQueue： 基于单链表的无界阻塞队列，不需指定大小。
PriorityBlockingQueue： 基于最小二叉堆的无界、优先级阻塞队列。
DelayQueue：基于延迟、优先级、无界阻塞队列。
SynchronousQueue： 基于 CAS 的阻塞队列。

LinkedBlockingQueue

```
        LinkedBlockingQueue<String> voice = new
LinkedBlockingQueue<String>();
    voice.add("第一句，你是哪里的人");
```

```
    voice.add("第二句，你住在哪里啊");
    voice.add("第三句，你叫什么名字");
    voice.add("第四句，你从事什么职业");
    voice.add("第五句，问完了");
```

Deque 双端队列

```
public class Main  {
    public static void main(String[] args) {

        Deque deque = new LinkedList<String>();
        deque.push("one");
        deque.push("two");
        deque.push("three");
        while (deque.peek() != null)
             System.out.println(deque.pop());
    }
}
```

LinkedList

```
public class QueueTest {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList();
        queue.offer("元素A");
        queue.offer("元素B");
        queue.offer("元素C");
        queue.offer("元素D");
        queue.offer("元素E");
        while (queue.size() > 0) {
            String element = queue.poll();
            System.out.println(element);
        }
```

```
    }
}
```

```
package cn.netkiller.test;

import java.util.LinkedList;
import java.util.Queue;

public class QueueTest {
    public static void main(String[] args) {

        Queue<String> queue = new LinkedList<String>();
        //添加元素
        queue.offer("a");
        queue.offer("b");
        queue.offer("c");
        queue.offer("d");
        queue.offer("e");

        for(String q : queue){
            System.out.println(q);
        }
        System.out.println("===");
        System.out.println("poll="+queue.poll()); //返回第一个
元素，并在队列中删除
        for(String q : queue){
            System.out.println(q);
        }
        System.out.println("===");
        System.out.println("element="+queue.element()); //返回
第一个元素
        for(String q : queue){
            System.out.println(q);
        }
        System.out.println("===");
        System.out.println("peek="+queue.peek()); //返回第一个
元素
        for(String q : queue){
            System.out.println(q);
        }
```

```
        //add()和remove()方法在失败的时候会抛出异常(不推荐)
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        LinkedList queue = new LinkedList();
        queue.add(1);
        queue.add(2);
        queue.add(3);

        //FIFO 先进先出 通过removeFirst()验证一下
        while (!queue.isEmpty())
            System.out.println(queue.removeFirst());
    }
}

public class Main {
    public static void main(String[] args) {
        LinkedList queue = new LinkedList();
        queue.add(1);
        queue.add(2);
        queue.add(3);

        // 当作栈区使用 removeLast() 先进后出
        while (!queue.isEmpty())
            System.out.println(queue.removeLast());
    }
}

输出结果
3
2
1
```

数据转换

```
                String str = "酒店坐落于青岛崂山区金融CBD，背靠历史悠久的“海上名山第一”崂山，迎黄海湾峡绵延悠长。风景秀美壮观，于酒店眺望闻名于世的青岛石老人景观，清风朗日浪花缱绻让人沉醉。国际娱乐标杆美高梅集团精心打造334间奢华畅阔客房，成为同城豪华旅居全新乐选之地。酒店拥有五间风格迥异餐厅、酒吧吧及糕点屋，资深大厨呈现五洲美食，惊喜不绝。";
        String[] arr = str.split("。");
        LinkedBlockingQueue<String> voice = new
LinkedBlockingQueue<String>(Arrays.asList(arr));
```

8.6. Stream

Stream.of

```
Stream<String> stream = Stream.of("Hollis", "HollisChuang",
"hollis", "Hello", "HelloWorld", "Hollis");
```

Stream.ofNullable

增加单个参数构造方法，可为null

```
Stream.ofNullable(null).count(); // 0
```

filter

filter 方法用于通过设置的条件过滤出元素。以下代码片段使用 filter 方法过滤掉空字符串：

```
List<String> strings = Arrays.asList("Hollis", "",
"HollisChuang", "H", "hollis");
strings.stream().filter(string ->
!string.isEmpty()).forEach(System.out::println);
```

map

map 方法用于映射每个元素到对应的结果，以下代码片段使用 map 输出了元素对应的平方数：

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
numbers.stream().map( i -> i*i).forEach(System.out::println);
//9,4,4,9,49,9,25
```

limit/skip

limit 返回 Stream 的前面 n 个元素；skip 则是扔掉前 n 个元素。以下代码片段使用 limit 方法保理4个元素：

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
numbers.stream().limit(4).forEach(System.out::println);
//3,2,2,3
```

sorted

sorted 方法用于对流进行排序。以下代码片段使用 sorted 方法进行排序：

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
numbers.stream().sorted().forEach(System.out::println);
//2,2,3,3,3,5,7
```

distinct

distinct主要用来去重，以下代码片段使用 distinct 对元素进行去重：

```
List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);
numbers.stream().distinct().forEach(System.out::println);
//3,2,7,5
```

forEach

Stream 提供了方法 'forEach' 来迭代流中的每个数据。以下代码片段使用 forEach 输出了10个随机数：

```
Random random = new Random();
random.ints().limit(10).forEach(System.out::println);
```

count

count用来统计流中的元素个数。

```
List<String> strings = Arrays.asList("Hollis",
"HollisChuang", "hollis","Hollis666", "Hello", "HelloWorld",
"Hollis");
System.out.println(strings.stream().count());
//7
```

collect

collect就是一个归约操作，可以接受各种做法作为参数，将流中的元素累积成一个汇总结果：

```
List<String> strings = Arrays.asList("Hollis",
"HollisChuang", "hollis","Hollis666", "Hello", "HelloWorld",
"Hollis");
strings  = strings.stream().filter(string ->
string.startsWith("Hollis")).collect(Collectors.toList());
System.out.println(strings);
//Hollis, HollisChuang, Hollis666, Hollis
```

takeWhile 和 dropWhile

增加 takeWhile 和 dropWhile 方法

```
Stream.of(1, 2, 3, 2, 1)
.takeWhile(n -> n < 3)
.collect(Collectors.toList()); // [1, 2]
```

从开始计算，当 n < 3 时就截止

```
Stream.of(1, 2, 3, 2, 1)
.dropWhile(n -> n < 3)
.collect(Collectors.toList()); // [3, 2, 1]
```

List to Stream

```
List<String> strings = Arrays.asList("Hollis",
"HollisChuang", "hollis", "Hello", "HelloWorld", "Hollis");
Stream<String> stream = strings.stream();
```

混合使用的例子

```
List<String> strings = Arrays.asList("Hollis",
"HollisChuang", "hollis", "Hello", "HelloWorld", "Hollis");
Stream s = strings.stream().filter(string -> string.length()
<= 6).map(String::length).sorted().limit(3)
            .distinct();
```

Collectors.teeing()

teeing 收集器已公开为静态方法Collectors::teeing。该收集器将其输入转发给其他两个收集器，然后将它们的结果使用函数合并。

```
package cn.netkiller.demo;

public class Student {
	public String name;
	public int score;

	public Student(String name, int score) {
		this.name = name;
		this.score = score;

	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getScore() {
		return score;
	}

	public void setScore(int score) {
		this.score = score;
	}

	@Override
	public String toString() {
		return "Student [name=" + name + ", score=" +
score + "]";
	}

}
```



```
package cn.netkiller.demo;

import java.util.Arrays;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

public class Test {

        public Test() {
        }

        public static void main(String[] args) {

                List<Student> list = Arrays.asList(new
Student("Neo", 80), new Student("Tom", 60), new
Student("Jerry", 70));
                // 平均分 总分
                String result =
list.stream().collect(Collectors.teeing(Collectors.averagingI
nt(Student::getScore),
Collectors.summingInt(Student::getScore), (s1, s2) -> s1 +
":" + s2));
                // 最低分 最高分
                String result2 =
list.stream().collect(Collectors.teeing(Collectors.minBy(Comp
arator.comparing(Student::getScore)),
Collectors.maxBy(Comparator.comparing(Student::getScore)),
(s1, s2) -> s1.orElseThrow() + ":" + s2.orElseThrow()));
                System.out.println(result);
                System.out.println(result2);
        }

}
```

8.7. Optional

```
Optional.of("javastack").orElseThrow(); // javastack
```

```
// 1
```

of() 为非null的值创建一个Optional。

of方法通过工厂方法创建Optional类。需要注意的是，创建对象时传入的参数不能为null。如果传入参数为null，则抛出NullPointerException 。

```
                Optional<String> name =
Optional.of("netkiller");
                if (name.isPresent()) {
                        // 在Optional实例内调用get()返回已存在的值
                        System.out.println(name.get());// 输出
netkiller
                }
```

传入参数为null，抛出NullPointerException.

```
Optional<String> someNull = Optional.of(null);
```

ofNullable() 为指定的值创建一个Optional，如果指定的值为null，则返回一个空的Optional。

```
                Optional<String> name =
Optional.ofNullable("netkiller");
                if (name.isPresent()) {
```

```
                        // 在Optional实例内调用get()返回已存在的值
                        System.out.println(name.get());// 输出
netkiller
                }

                Optional<String> empty =
Optional.ofNullable(null);
                if (empty.isPresent()) {
                        System.out.println(empty.get());
                }
```

isPresent 如果值存在返回**true**，否则返回**false**。

```
        //isPresent方法用来检查Optional实例中是否包含值
        if (name.isPresent()) {
                System.out.println(name.get());
        }
```

ifPresent() 如果**Optional**实例有值执行 **lambda** 表达式

如果Optional实例有值，调用ifPresent()可以接受接口段或lambda表达式。类似下面的代码:

```
        Optional<String> name =
Optional.ofNullable("netkiller");

        name.ifPresent((value) -> {
                System.out.println("hello " + value);
        });

        name.ifPresent((value) -> {
                System.out.println(value.length());
```

```
        });
```

get() 返回值

如果Optional有值则将其返回，否则抛出NoSuchElementException。

```
                Optional<String> name =
Optional.ofNullable("netkiller");
                System.out.println(name.get());

                Optional<String> empty =
Optional.ofNullable(null);
                try {
                        System.out.println(empty.get());
                } catch (NoSuchElementException e) {
                        System.out.println(e.getMessage());
                }
```

输出内容

```
netkiller
No value present
```

orElse 如果有值则将其返回，否则返回指定的其它值。

如果Optional实例有值则将其返回，否则返回orElse方法传入的参数。示例如下：

```
package cn.netkiller.test;

import java.util.Optional;

public class OptionalTest {

        public OptionalTest() {
                // TODO Auto-generated constructor stub
        }


        public static void main(String[] args) {

                Optional<String> name =
Optional.ofNullable("netkiller");

                Optional<String> empty =
Optional.ofNullable(null);

                System.out.println(name.orElse("There is some
value!"));
                System.out.println(empty.orElse("There is no
value present!"));

        }

}
```

输出

```
netkiller
There is no value present!
```

指定默认值

```
        User user = new User();
        user.setId(1);
        user.setUsername("Neo");

        Optional<User> user =
Optional.ofNullable(user).orElse(new User(0, "Unknown"));

        System.out.println("Username is: " +
user.getUsername());
```

orElseGet与orElse方法类似，区别在于得到的默认值从 Supplier 返回。

orElseGet方法可以接受Supplier接口的实现用来生成默认值。示例如下:

```
package cn.netkiller.test;

import java.util.Optional;

public class OptionalTest {

        public OptionalTest() {
                // TODO Auto-generated constructor stub
        }


        public static void main(String[] args) {

                Optional<String> name =
Optional.ofNullable("netkiller");

                Optional<String> empty =
Optional.ofNullable(null);
```

```
                System.out.println(name.orElseGet(() ->
"There is some value!"));
                System.out.println(empty.orElseGet(() ->
"There is no value present!"));

        }

}
```

```
Optional<User> user = Optional.ofNullable(user).orElseGet(()
-> new User(0, "Unknown"));
```

orElseThrow 如果有值则将其返回，否则抛出supplier接口创建的异常

```
Optional<User> user = Optional
        .ofNullable(user)
        .orElseThrow(() -> new EntityNotFoundException("id="
+ id + " 的用户没有找到"));
```

使用场景举例

```
@RequestMapping("/user/{id}")
public User getUser(@PathVariable Integer id) {
    Optional<User> user = userService.getUserById(id);
    return user.orElseThrow(() -> new
EntityNotFoundException("id=" + id + " 的用户不存在"));
}
```

```
@ExceptionHandler(EntityNotFoundException.class)
public ResponseEntity<String>
handleException(EntityNotFoundException ex) {
    return new ResponseEntity<>(ex.getMessage(),
HttpStatus.NOT_FOUND);
}
```

```
package cn.netkiller.test;

import java.util.Optional;

public class OptionalTest {

        public OptionalTest() {
                // TODO Auto-generated constructor stub
        }

        public static class ValueAbsentException extends
Throwable {

                private static final long serialVersionUID =
-1758502952187236809L;

                public ValueAbsentException() {
                        super();
                }

                public ValueAbsentException(String msg) {
                        super(msg);
                }

                @Override
                public String getMessage() {
                        return "No value present in the
Optional instance";
                }
        }

        public static void main(String[] args) {
```

```
                Optional<String> empty =
Optional.ofNullable(null);

                try {
                        // orElseThrow会抛出lambda表达式或方法生
成的异常

empty.orElseThrow(ValueAbsentException::new);
                } catch (Throwable ex) {
                        // 输出 No value present in the
Optional instance
                        System.out.println(ex.getMessage());
                }

        }

}
```

map() 方法用来对Optional实例的值执行一系列操作

map方法用来对Optional实例的值执行一系列操作。通过一组实现了Function接口的lambda表达式传入操作。map方法示例如下:

```
        Optional<String> name =
Optional.ofNullable("netkiller");
        Optional<String> upperName = name.map((value) ->
value.toUpperCase());
        System.out.println(upperName.orElse("No value
found"));
```

```
                Optional<String> username =
Optional.ofNullable("netKiller-Neo")
```

```
                                    .map((value) ->
value.toLowerCase())
                                    .map((value) ->
value.replace("n", "N"))
                                    .map(value ->
value.replace('-', '_'));

                System.out.println("Username is: " +
username.orElse("Unknown"));
```

flatMap()

与 map() 区别在于flatMap中的mapper返回值必须是Optional

```
                Optional<String> username =
Optional.ofNullable("netKiller-Neo").flatMap((value) ->
Optional.of(value.toUpperCase()));

                System.out.println("Username is: " +
username.orElse("No value found"));
```

filter() 通过传入限定条件过滤Optional值

```
package cn.netkiller.test;

import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class OptionalTest {

        public OptionalTest() {
```

```
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {

                for (String item : List.of("Neo", "Jerry",
"Netkiller")) {
                        Optional<String> username =
Optional.of(item).filter((value) -> value.length() > 6);
                        System.out.println("name is: " +
username.orElse("The name is less than 6 characters"));
                }
        }

}
```

使用多个 filter 组合过滤数据

```
                List.of("Neo", "Jerry", "Netkiller", "Tom",
"Anni", "Lisa", "Leo").forEach(item -> {
                        Optional.of(item).filter((value) ->
value.length() > 2).filter((value) ->
value.contains("o")).ifPresent((n) -> {
                                System.out.println(n);
                        });
                });
```

stream()

```
Optional.of("javastack").stream().count();
```

or()

```
        String string = (String)
Optional.ofNullable(null).or(() ->
Optional.of("netkiller")).get();
        System.out.println(string);
```

example

Optional 与 Map

```
                Optional<Map<String, Object>> name =
Optional.of(new HashMap<String, Object>() {
                        {
                                put("id", 1);
                                put("name", "Neo");
                                put("age", 30);
                        }
                });

                System.out.println(name.toString());
                name.map((m) -> m.put("count", 1));
                System.out.println(name.get());
                name.map((m) -> m.put("nickname",
"netkiller"));
                name.map((m) -> m.remove("id"));
                System.out.println(name.get());
                Optional<Map<String, Object>> tmp =
name.filter((m) -> ((Integer) m.get("age")) == 30);
                System.out.println("filter: " + tmp.get());
```

判断 Object 是否为 null

```
package cn.netkiller.utils;

import lombok.Data;
import lombok.extern.slf4j.Slf4j;

import java.io.Serial;
import java.io.Serializable;
import java.util.Optional;

@Data
@Slf4j
public class ResponseJson implements Serializable {
    @Serial
    private static final long serialVersionUID = 1L;
    private final boolean status; // 状态代码
    private final Code code; // 业务响应码
    private final Object data; // 返回业务参数
    private String reason = ""; // 返回信息描述

    public ResponseJson(boolean status, Code code, String
reason, Object data) {
        this.status = status;
        this.code = code;
        this.reason = reason;
        this.data = data;
    }

    public ResponseJson(Object data) {
        Optional<Object> optional =
Optional.ofNullable(data);
//        log.info(String.valueOf(optional.isEmpty()));
//        log.info(String.valueOf(optional.isPresent()));
//        log.info(String.valueOf(optional.get()));
        if (optional.isEmpty()) {
            this.status = false;
            this.code = Code.FAIL;
            this.reason = "失败";
            this.data = null;
        } else {
            this.status = true;
            this.code = Code.SUCCESS;
            this.reason = "成功";
```

```
            this.data = data;
        }
    }

    public enum Code {
        SUCCESS, // 业务处理成功
        COMPLIANCE, NONCOMPLIANT, ILLEGAL, FAIL // 业务处理失败

    }
}
```

9. Network

Java 网络相关操作

9.1. URL

```
import java.net.*;
import java.io.*;

public class URLReader {
    public static void main(String[] args) throws Exception {

        URL url = new URL("http://www.netkiller.cn/");
        BufferedReader in = new BufferedReader(new
InputStreamReader(url.openStream()));

        String inputLine;
        while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);
        in.close();
    }
}
```

9.2. 获取IP地址何机器名

改变java.io.tmpdir的默认值

```
    @GetMapping("/host")
    public ResponseEntity<String> host() {
        try {
            InetAddress addr = InetAddress.getLocalHost();
            String hostAddress =
addr.getHostAddress().toString();
```

```
            String hostName = addr.getHostName().toString();
            String tmp = String.format("%s: %s", hostAddress,
hostName);
            return ResponseEntity.ok(tmp);
        } catch (Exception e) {
            return ResponseEntity.ok(e.toString());
        }
    }
```

10. JDBC

10.1. 安装 JDBC 包

将ojdbc6.jar包复制到jre/lib/ext目录下

```
# mv ojdbc6.jar /srv/jdk1.8.0_60/jre/lib/ext/
```

10.2. MySQL

10.3. Oracle

SID

jdbc:oracle:thin:@<host>:<port>:<SID> Example:

```
jdbc:oracle:thin:@192.168.2.1:1521:oral
```

SERVICE_NAME

jdbc:oracle:thin:@//<host>:<port>/<service_name>

Example:

```
jdbc:oracle:thin:@//192.168.2.1:1521/orcl.example.com
```

TNS

jdbc:oracle:thin:@<TNSName> Example:

jdbc:oracle:thin:@TNS

Oracle RAC Cluster

Oracle 11G 不能直接链接 RAC 的 VIP

```
jdbc:oracle:thin:@(DESCRIPTION =
        (ADDRESS_LIST =
                (ADDRESS = (PROTOCOL = TCP)(HOST =
172.16.10.10)(PORT = 1521))
                (ADDRESS = (PROTOCOL = TCP)(HOST =
172.16.10.20)(PORT = 1521))
                (LOAD_BALANCE = yes)
        )
        (CONNECT_DATA =
                (SERVER = DEDICATED)
                (SERVICE_NAME = orcl.example.com)
                (FAILOVER_MODE =(TYPE = SELECT )(METHOD =
BASIC)(RETRIES = 120)(DELAY = 5)))
        )
```

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=16.50.26.29)(PORT=1521))(LOAD_BALANCE=yes)(FAILOVER=ON)
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=orcl.example.com)
(FAILOVER_MODE=(TYPE=SESSION)(METHOD=BASIC))))
```

Oracle JDBC Demo

```
package cn.netkiller.zabbix;

import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
```

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

/**
 * Java JDBC Oracle Demo!
 *
 */
public class Oracle {

        String url = null; // 数据库链接地址
        String username = null;// 用户名,系统默认的账户名
        String password = null;// 你安装时选设置的密码

        public void openConfig() {
                String connectionfig = "jdbc.properties";
                Properties properties = new Properties();
                try {
                        properties.load(new
FileInputStream(connectionfig));
                        this.url =
properties.getProperty("jdbc.url");
                        this.username =
properties.getProperty("jdbc.username");
                        this.password =
properties.getProperty("jdbc.password");
                } catch (FileNotFoundException e) {
                        System.out.println(
                                        e.getMessage() + "
Working Directory = " + System.getProperty("user.dir") + "/" +
connectionfig);
                } catch (IOException e) {
                        System.out.println(e.getMessage());
                }
                if (this.url == null || this.username == null
|| this.password == null) {
                        System.out.println("This Propertie file
is invalid");
                        // throw new Exception("");
                }

        }

        public void testOracle() {
```

```
                Connection connection = null;// 创建一个数据库连接
                ResultSet result = null;// 创建一个结果集对象
                Statement statement = null;
                try {

Class.forName("oracle.jdbc.driver.OracleDriver");// 加载Oracle驱
动程序
                        connection =
DriverManager.getConnection(this.url, this.username,
this.password);
                        String sql = "select current_date from
dual";
                        statement =
connection.createStatement();
                        result = statement.executeQuery(sql);
                        result.next();
                        System.out.printf("%s %s",
result.getDate(1), result.getTime(1));
                } catch (ClassNotFoundException e) {
                        System.out.println(e.getMessage());
                } catch (SQLException e) {
                        System.out.println(e.getMessage());
                } finally {
                        try {
                                if (result != null)
                                        result.close();
                                if (connection != null)
                                        connection.close();
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        }

        public static void main(String[] args) {
                Oracle oracle = new Oracle();
                oracle.openConfig();
                oracle.testOracle();
        }
}
```

10.4. FAQ

java.sql.SQLRecoverableException: IO Error: The Network Adapter could not establish the connection

直接连接 RAC VIP 提示

```
java.sql.SQLRecoverableException: IO Error: The Network Adapter
could not establish the connection
```

解决方案

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=16.50.26.29)(PORT=1521))(LOAD_BALANCE=yes)(FAILOVER=ON)
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=orcl.example.com)
(FAILOVER_MODE=(TYPE=SESSION)(METHOD=BASIC))))
```

Exception in thread "main" java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver

检查 /srv/jdk1.8.0_60/jre/lib/ext/ 是否有jdbc包，然后运行下面程序检查是否已经载入

```
# java -verbose:class JdbcTest | grep jdbc

[Loaded oracle.jdbc.driver.OracleDriver from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleDriver from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleDriverExtension from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleDriver$1 from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.ClassRef from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.ClassRef$XMLTypeClassRef from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.ClassRef$Locale from
```

```
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.ClassRef$LocaleCategoryClassRef from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.DiagnosabilityMXBean from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleDiagnosabilityMBean from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.DatabaseError from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleSQLException from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.SQLStateMapping from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.SQLStateMapping$Tokenizer from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.Message from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.Message11 from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OracleConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.ClientDataSupport from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleConnectionWrapper from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.PhysicalConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.T4CDriverExtension from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OraclePreparedStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleCallableStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OracleStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OraclePreparedStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OracleCallableStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
```

```
[Loaded oracle.jdbc.driver.ScrollRsetStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OraclePreparedStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleCallableStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.T4CCallableStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.T4CStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.T4CPreparedStatement from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleBufferedStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.T4CInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.BfileDBAccess from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.BlobDBAccess from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.ClobDBAccess from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.T4CConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OracleDatumWithConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleClob from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OracleClob from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.Datum from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.DatumWithConnection from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.CLOB from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleNClob from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.OracleNClob from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.NCLOB from
```

```
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleSQLPermission from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.AdditionalDatabaseMetaData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleDatabaseMetaData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleDatabaseMetaData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleSavepoint from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.PhysicalConnection$2 from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.aq.AQMessage from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.NTFRegistration from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.NotificationRegistration from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.dcn.DatabaseChangeRegistration from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.NTFDCNRegistration from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleTypeMetaData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.ObjectData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.ORAData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleData from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.TypeDescriptor from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleTypeMetaData$Struct from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.StructDescriptor from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.OracleTypeMetaData$Array from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.sql.ArrayDescriptor from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleClobInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleBlobInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
```

```
[Loaded oracle.jdbc.driver.OracleConversionInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleConversionReader from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.NetException from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.aq.AQNotificationRegistration from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.NTFAQRegistration from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleBlobOutputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleClobOutputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleClobReader from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.OracleClobWriter from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.internal.XSEvent from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.NTFXSEvent from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.Communication from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.CRC64 from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.NTFManager from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.jdbc.driver.PhysicalConnection$1 from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.SQLnetDef from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.NSProtocol from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.NetOutputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.NLException from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.NetInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.SessionAtts from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.NVFactory from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.InvalidSyntaxException from
```

```
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.NVNavigator from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.AddrResolution from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.SOException from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.ClientProfile from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.nt.ConnStrategy from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.SchemaObjectFactoryInterface
from file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.NavSchemaObjectFactory from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.SchemaObject from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.NavSchemaObject from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.ServiceAlias from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.NavServiceAlias from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.NVTokens from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.UninitializedObjectException from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.nl.NVPair from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.Description from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.NavDescription from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.Address from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.NavAddress from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.jdbc.TNSAddress.DescriptionList from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.resolver.NavDescriptionList from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.nt.ConnOption from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.nt.NTAdapter from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
```

```
[Loaded oracle.net.nt.TcpNTAdapter from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.Packet from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.DataPacket from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.DataDescriptorPacket from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.BreakNetException from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.Ano from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.AnoNetInputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.AnoNetOutputStream from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.Service from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.AnoComm from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.SupervisorService from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.AuthenticationService from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.EncryptionService from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.aso.g from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ano.DataIntegrityService from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.aso.d from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.aso.i from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.aso.b from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.ConnectPacket from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
[Loaded oracle.net.ns.RedirectPacket from
file:/srv/jdk1.8.0_60/jre/lib/ext/ojdbc6.jar]
```

11. Util

11.1. Properties 处理 *.properties 文件

```
package netkiller.test;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.FileInputStream;
import java.util.Properties;

public class PropertiesTest {

        public static void main(String[] args) {

                System.out.println("Working Directory = " +
System.getProperty("user.dir"));

                Properties ps = new Properties();
                try {
                        ps.load(new
FileInputStream("netkiller.properties"));
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                System.out.println(ps.getProperty("key"));
        }

}
```

打开 properties 文件

文件方式打开

```
BufferedReader br = null;
Properties properties = new Properties();
br = new BufferedReader(new InputStreamReader(new
FileInputStream(new File("data.properties")), "UTF8"));
properties.load(br);
```

输入流

```
Properties properties = new Properties();
InputStream in =
getClass().getResourceAsStream("/IcisReport.properties");
properties.load(in);
Set keyValue = properties.keySet();
for (Iterator it = keyValue.iterator(); it.hasNext();)
{
        String key = (String) it.next();
}
```

propertyNames()

```
package cn.netkiller.properties;

import java.util.Enumeration;
import java.util.Map.Entry;
import java.util.Properties;

public class PropertiesTest {

        public PropertiesTest() {
                // TODO Auto-generated constructor stub
        }
```

```
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Properties properties = new Properties();

                properties.put("K1", "V1");
                properties.put("K2", "V2");

                for (Entry<Object, Object> x :
properties.entrySet()) {
                        System.out.println(x.getKey() + " " +
x.getValue());
                }

                Enumeration<?> e =
properties.propertyNames();
                while (e.hasMoreElements()) {
                        String key = (String)
e.nextElement();
                        String value =
properties.getProperty(key);
                        System.out.println(key + ": " +
value);
                }
        }

}
```

```
import java.io.FileInputStream;
import java.util.Enumeration;
import java.util.Properties;

public class MainClass {
  public static void main(String args[]) throws Exception {
    Properties p = new Properties();
    p.load(new FileInputStream("test.properties"));
    Enumeration e = p.propertyNames();

    for (; e.hasMoreElements();) {
      System.out.println(e.nextElement());
```

```
    }
  }
}
```

keySet()

```
package cn.netkiller.properties;

import java.util.Properties;
import java.util.Set;

public class PropertiesTest {

        public PropertiesTest() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Properties properties = new Properties();

                properties.put("K1", "V1");
                properties.put("K2", "V2");

                Set<Object> states = properties.keySet();

                for (Object name : states)
                        System.out.println("The value of " +
name + " is " + properties.getProperty((String) name) + ".");
        }

}
```

entrySet()

```
package cn.netkiller.properties;

import java.util.Map.Entry;
import java.util.Properties;

public class PropertiesTest {

        public PropertiesTest() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Properties properties = new Properties();

                properties.put("K1", "V1");
                properties.put("K2", "V2");

                for (Entry<Object, Object> x :
properties.entrySet()) {
                        System.out.println(x.getKey() + " " +
x.getValue());
                }
        }

}
```

方法中返回 **Properties**

```
        @RequestMapping("/host")
        public Enumeration<Object> host() throws IOException
{
                Properties properties =
```

```
PropertiesLoaderUtils.loadProperties(new
ClassPathResource(String.format("/%s.properties", "host")));
                return properties.keys();
        }

        @RequestMapping("/mail")
        public Collection<Object> mail() throws IOException {
                Properties properties =
PropertiesLoaderUtils.loadProperties(new
ClassPathResource(String.format("/%s.properties", "mail")));
                return properties.values();
        }
        @RequestMapping("/nameserver")
        public Set<Entry<Object, Object>> nameserver() throws
IOException {
                Properties properties =
PropertiesLoaderUtils.loadProperties(new
ClassPathResource(String.format("/%s.properties", "dns")));
                return properties.entrySet();
        }
        @RequestMapping("/dns")
        public Properties dns() throws IOException {
                Properties properties =
PropertiesLoaderUtils.loadProperties(new
ClassPathResource(String.format("/%s.properties", "dns")));
                return properties;
        }
```

```
```

getResourceAsStream()

```
Properties prop = new Properties();
prop.load(getServletContext().getResourceAsStream("/WEB-
```

```
INF/resource/sample.properties"));
```

```
prop.load(Thread.currentThread().getContextClassLoader().getR
esourceAsStream("netkiller.properties"));
```

store

```
package cn.netkiller.config;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Properties;

public class Config {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Properties prop = new Properties();
                OutputStream output = null;

                try {

                        output = new
FileOutputStream("config.properties");

                        // set the properties value
                        prop.setProperty("host",
"localhost");
                        prop.setProperty("port", "8000");
                        prop.setProperty("user", "neo");
                        prop.setProperty("pass", "password");

                        // save properties to project root
```

```
folder
                        prop.store(output, null);

                } catch (IOException io) {
                        io.printStackTrace();
                } finally {
                        if (output != null) {
                                try {
                                        output.close();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }

                }
        }

}
```

getProperty 取出key的值

```
package cn.netkiller.config;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class LoadConfig {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Properties prop = new Properties();
                InputStream input = null;

                try {

                        input = new
FileInputStream("config.properties");
```

```
                        // load a properties file
                        prop.load(input);

                        // get the property value and print
it out

System.out.println(prop.getProperty("host"));

System.out.println(prop.getProperty("port"));

System.out.println(prop.getProperty("user"));

System.out.println(prop.getProperty("pass"));

                } catch (IOException ex) {
                        ex.printStackTrace();
                } finally {
                        if (input != null) {
                                try {
                                        input.close();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                }

        }

}
```

循环打印所有 Properties 内容

```
package test;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Enumeration;
```

```
import java.util.Properties;

public class Application {

	public static void main(String[] args) {
		Application app = new Application();
		app.config();

	}

	private void config() {
		Properties prop = new Properties();
		InputStream input = null;
		try {

			String filename =
"config.properties";

			prop.load(new
FileInputStream(filename));

			Enumeration<?> e =
prop.propertyNames();
			while (e.hasMoreElements()) {
				String key = (String)
e.nextElement();
				String value =
prop.getProperty(key);
				System.out.println(key + ": "
+ value);
			}

		} catch (IOException ex) {
			ex.printStackTrace();
		} finally {
			if (input != null) {
				try {
					input.close();
				} catch (IOException e) {
					e.printStackTrace();
				}
			}
		}
	}
```

```
}
```

实现国际化

准备语言包文件 chinese.properties 内容如下

```
hello=你好世界
```

english.properties

```
hello=Helloworld
```

```
        @GetMapping("/lang")
        public String lang(@RequestHeader String lang) throws
IOException {
                System.out.println(lang);
                Properties properties =
PropertiesLoaderUtils.loadProperties(new
ClassPathResource(lang + ".properties"));
                String tmp = properties.getProperty("hello");
                return tmp;
        }
```

测试效果

```
```

```
curl -s -H lang:chinese http://localhost:8080/lang
你好世界

curl -s -H lang:english http://localhost:8080/lang
Helloworld
```

11.2. Logging

```
import java.util.logging.*;
public class Main {
        public static void main(String[] args) {
                Logger log = Logger.getLogger("test");
        log.setLevel(Level.INFO);
        log.info("-------------------------");
        log.info("Test");
        log.info("-------------------------");

        }
}
```

XML

```
import java.io.IOException;
import java.util.logging.*;

public class Main {
        public static void main(String[] args) {

                try {
                        Logger log =
Logger.getLogger("test");
                        FileHandler fileHandler = new
FileHandler("test.%g.log");
                        fileHandler.setLevel(Level.INFO);
```

```
                        log.addHandler(fileHandler);

                        log.setLevel(Level.INFO);
                        log.info("One");
                        log.info("Two");
                        log.info("Three");

                } catch (SecurityException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
}
```

XML 输出结果

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
  <date>2016-04-19T15:57:19</date>
  <millis>1461052639360</millis>
  <sequence>0</sequence>
  <logger>test</logger>
  <level>INFO</level>
  <class>Main</class>
  <method>main</method>
  <thread>1</thread>
  <message>One</message>
</record>
<record>
  <date>2016-04-19T15:57:19</date>
  <millis>1461052639394</millis>
  <sequence>1</sequence>
  <logger>test</logger>
```

```
  <level>INFO</level>
  <class>Main</class>
  <method>main</method>
  <thread>1</thread>
  <message>Two</message>
</record>
<record>
  <date>2016-04-19T15:57:19</date>
  <millis>1461052639395</millis>
  <sequence>2</sequence>
  <logger>test</logger>
  <level>INFO</level>
  <class>Main</class>
  <method>main</method>
  <thread>1</thread>
  <message>Three</message>
</record>
</log>
```

Formatter 日志格式化

```
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.*;

class LogFormatter extends Formatter {
        @Override
        public String format(LogRecord record) {
                return String.format("%s %s\t%s\n", new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS").format(new
Date()) , record.getLevel(), record.getMessage());
        }
}

public class Main {
        public static void main(String[] args) {

                try {
```

```
                        Logger log =
Logger.getLogger("test");
                        FileHandler fileHandler = new
FileHandler("test.%g.log");
                        fileHandler.setLevel(Level.INFO);
                        log.addHandler(fileHandler);
                        fileHandler.setFormatter(new
LogFormatter());
                        log.setLevel(Level.INFO);
                        log.info("One");
                        log.info("Two");
                        log.info("Three");

                } catch (SecurityException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }
}
```

输出样式

```
2016-04-19 16:05:53.324 INFO    One
2016-04-19 16:05:53.352 INFO    Two
2016-04-19 16:05:53.353 INFO    Three
```

console

控制台日志输入格式定义

```
                                    ConsoleHandler consoleHandler
= new ConsoleHandler();

consoleHandler.setLevel(Level.OFF);

logger.addHandler(consoleHandler);
```

禁止 Console 输出

```
logger.setUseParentHandlers(false);
```

11.3. BASE64

```
package cn.netkiller.test;

import java.nio.charset.StandardCharsets;
import java.util.Base64;

public class Base64Test {
        public static void main(String[] args) {
                final String text =
"http://www.netkiller.cn/index.html";

                final String encoded =
Base64.getEncoder().encodeToString(text.getBytes(StandardChar
sets.UTF_8));
                System.out.println(encoded);

                final String decoded = new
String(Base64.getDecoder().decode(encoded),
StandardCharsets.UTF_8);
                System.out.println(decoded);
        }
}
```

11.4. Locale 国际化

```
package cn.netkiller.i18n;

import java.util.Locale;

public class Lang {

        public Lang() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // create a new locale
                Locale locale = new Locale("ENGLISH", "US");

                // print locale
                System.out.println("Locale:" + locale);

                // print display name for locale - based on
inLocale
                System.out.println("Name:" +
locale.getDisplayName(new Locale("GERMAN", "GERMANY")));

        }

}
```

```
Locale locale = new Locale("zh", "CN");
Locale locale = Locale.US;
Locale locale = Locale.getDefault();    // 默认语言
```

11.5. ResourceBundle

```
java.util.ResourceBundle resourceBundle =
java.util.ResourceBundle.getBundle("message");

resourceBundle.getString("nickname");

// 指定语言

Locale  locale = new Locale("zh", "CN");
ResourceBundle res = ResourceBundle.getBundle("message",
locale);
```

11.6. Scanner

```
Scanner in = new Scanner(System.in);
System.out.println("Username:");
String username = in.next();
```

11.7. UUID

```
package cn.netkiller.example.uuid;

import java.util.UUID;

public class UuidTest {

        public UuidTest() {
                // TODO Auto-generated constructor stub
        }
```

```
        public static void main(String[] args) {
                UUID uuid = UUID.randomUUID();
                String randomUUIDString = uuid.toString();

                System.out.println("Random UUID String = " +
randomUUIDString);
                System.out.println("UUID version       = " +
uuid.version());
                System.out.println("UUID variant       = " +
uuid.variant());
        }

}
```

11.8. Arrays.equals 判断两个数组是否相等

static boolean equals(type[] a, type[] b)

```
Arrays.equals(array1, array2)
```

11.9. Random 随机字符串

```
package cn.netkiller.test;

import java.util.Random;

public class QueueTest {

        public static void main(String[] args) throws
InterruptedException {
                new Random().ints(10, 33,
```

```
38).forEach(System.out::println);
        }
}
```

```
36
34
34
36
37
35
34
35
34
33
```

取 0-n 范围内随机数

```
    private int salt() {
        Random random = new Random();
        return random.nextInt(99999999);
    }
```

指定随机数范围

```
package cn.netkiller.test;

import java.util.Random;
```

```
public class RandomTest {

        public static int random(int min, int max) {
                var value = new Random().ints(min, (max +
1)).limit(1).findFirst().getAsInt();
                return value;
        }

        public static void main(String[] args) throws
InterruptedException {
                System.out.println(random(10, 15));
        }
}
```

11.10. ArrayBlockingQueue

```
package cn.netkiller.test;

import java.util.Random;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class QueueTest {
        /**
         * 定义装苹果的篮子
         */
        public static class Basket {
                // 篮子，能够容纳10个苹果
                BlockingQueue<String> basket = new
ArrayBlockingQueue<String>(10);

                // 生产苹果，放入篮子
                public void produce() throws
InterruptedException {
                        // put方法放入一个苹果，若basket满了，等到
```

```
basket有位置
                        basket.put("An apple");
                }

                // 消费苹果，从篮子中取走
                public String consume() throws
InterruptedException {
                        // get方法取出一个苹果，若basket为空，等到
basket有苹果为止
                        return basket.take();
                }

                public int size() {
                        return basket.size();
                }

        }

        // 测试方法
        public static void testBasket() throws
InterruptedException {
                // 建立一个装苹果的篮子
                final Basket basket = new Basket();
                // 定义苹果生产者
                class Producer implements Runnable {
                        public void run() {
                                try {
                                        while (true) {
                                                int n =
random(1, 5);
                                                for (int i =
0; i < n; i++) {

basket.produce();
                                                }

System.out.println(System.currentTimeMillis() + " 放入" + n +
"个，当前总数：" + basket.size() + "个");

Thread.sleep(random(450, 1000));
                                        }
                                } catch (InterruptedException
ex) {
                                }
                        }
```

```
                }
                // 定义苹果消费者
                class Consumer implements Runnable {
                        public void run() {
                                try {
                                        while (true) {
                                                // 消费苹果
                                                int n =
random(1, 5);
                                                for (int i =
0; i < n; i++) {

basket.consume();
                                                }

System.out.println(System.currentTimeMillis() + " 取出" + n +
"个，剩余数量：" + basket.size() + "个");

Thread.sleep(random(400, 1000));
                                        }
                                } catch (InterruptedException
ex) {
                                }
                        }
                }

                ExecutorService service =
Executors.newCachedThreadPool();
                Producer producer = new Producer();
                Consumer consumer = new Consumer();
                service.submit(producer);
                // 延迟消费
                Thread.sleep(5000);
                service.submit(consumer);
                // 程序运行10s后，所有任务停止
                try {
                        Thread.sleep(20000);
                } catch (InterruptedException e) {
                }
                service.shutdownNow();
        }

        public static int random(int min, int max) {
                var value = new Random().ints(min, (max +
1)).limit(1).findFirst().getAsInt();
```

```
                return value;
        }

        public static void main(String[] args) throws
InterruptedException {
                QueueTest.testBasket();
        }
}
```

11.11. CRC32

```
package cn.netkiller.security;

import java.nio.ByteBuffer;
import java.util.zip.CRC32;

public class CRC {

        public static void main(String[] args) {

                final CRC32 crc32 = new CRC32();
                ByteBuffer data =
ByteBuffer.wrap("http://www.netkiller.cn".getBytes());
                crc32.update(data);
                System.out.println(crc32.getValue());

        }

}
```

11.12. 正则表达式

```
package cn.netkiller;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Test {


    public static void main(String[] args) {

        Pattern pattern = Pattern.compile("www|netkiller");
        Matcher matcher =
pattern.matcher("http://www.netkiller.cn/linux/index.html");
        if (matcher.find()) {
            System.out.println(matcher.group());
        }

    }
}
```

正则查找

```
        Matcher matcher = Pattern.compile("播放|暂停|停
止").matcher("当前暂停音乐");

        if(matcher.find()){
            System.out.println(matcher.group(0));
        }

        if(Pattern.compile("播放|暂停|停止").matcher("当前停音
乐").find()){
            System.out.println("查找到");
        }
```

正则替换

```
package cn.netkiller;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Test {

    public static void main(String[] args) {

        Pattern pattern = Pattern.compile("www|netkiller");
        Matcher matcher =
pattern.matcher("https://www.netkiller.cn/linux/index.html");
        if (matcher.find()) {
            String s = matcher.replaceFirst("api"); //替换后的
字符串
            System.out.println(s);
        }

    }
}
```

```
package cn.netkiller;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Test {


    public static void main(String[] args) {

        Pattern pattern = Pattern.compile("www|netkiller");
        Matcher matcher =
pattern.matcher("https://www.netkiller.cn/linux/index.html");
        if (matcher.find()) {
            String s = matcher.replaceAll("test"); //替换后的字
符串
```

```
            System.out.println(s);
            // 输出结果：https://test.test.cn/linux/index.html
        }

    }
}
```

```
"aab".replaceAll("a{1}", "x"); //xxb
"aba".replaceAll("a{1}", "x"); //xbx

"abaaabaaaba".replaceAll("a{2}", "x");  //abxabxaba
"abaabaaaaba".replaceAll("a{2}", "x"); //abxbxxba
```

字符串分割

```
        String input = "苹果!!香蕉!!鸭梨!!橘子";

System.out.println(Arrays.toString(Pattern.compile("!!").spli
t(input)));

System.out.println(Arrays.toString(Pattern.compile("!!").spli
t(input, 3)));
```

```
[苹果, 香蕉, 鸭梨, 橘子]
[苹果, 香蕉, 鸭梨!!橘子]
```

11.13. java.util.concurrent

TimeUnit

```
        try {
                TimeUnit.SECONDS.sleep(5);
        } catch (InterruptedException e) {
        }
```

FutureTask

```
package cn.netkiller.welcome;

import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.FutureTask;

public class Future {

        public static void main(String[] args) throws
InterruptedException, ExecutionException {
                // TODO Auto-generated method stub
                FutureTask<String> futureTask = new
FutureTask<>(new Callable<String>() {
                        @Override
                        public String call() throws Exception
{
                                String status = null;

System.out.println(Thread.currentThread().getName() + ":" +
"Send SMS ...");
                                Thread.sleep(2000);

System.out.println(Thread.currentThread().getName() + ":" +
"Sent");
                                status = "OK";
                                return status;
```

```
                        }
                });

//              开启了一个线程执行future的逻辑
                Thread thread = new Thread(futureTask);
                thread.start();

                // 主业务逻辑

System.out.println(Thread.currentThread().getName() + ":" +
"Begin");
                Thread.sleep(3000);

System.out.println(Thread.currentThread().getName() + ":" +
"End");

                String sent = futureTask.get();


System.out.println(Thread.currentThread().getName() + ":" +
"Status: " + sent + " done!");
        }

}
```

CompletableFuture

runAsync 创建没有返回值的异步任务

runAsync 创建没有返回值的异步任务。它有如下两个方法，一个是使用默认线程池（ForkJoinPool.commonPool()）的方法，一个是自定义线程池的重载方法

```
package cn.netkiller;

import cn.netkiller.thread.ThreadManager;
import lombok.SneakyThrows;
```

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Main {
    @SneakyThrows
    public static void main(String[] args) {

        CompletableFuture.runAsync(() -> {
            System.out.println("do something...");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        });

        ExecutorService executorService =
Executors.newSingleThreadExecutor();
        CompletableFuture.runAsync(() -> {
            System.out.println("do something...");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }, executorService);

        CompletableFuture<Void> completableFuture =
CompletableFuture.runAsync(() -> {
            System.out.println("do something....");
//            Thread.currentThread().setName("测试有返回值的异步
执行");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        });

        ThreadManager tm = new ThreadManager();
        System.out.println(tm.show());

        try {
```

```
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        System.out.println("Result ->" +
completableFuture.isDone());


    }
}
```

```
do something...
do something...
do something....

================================================================
========================
|   ID |                                Name | Group | Daemon |
State | Priority |
----------------------------------------------------------------
------------------------
|    1 |                                main |  main |  false |
RUNNABLE |        5 |
|   21 | ForkJoinPool.commonPool-worker-1 |  main |   true |
TIMED_WAITING |        5 |
|   22 |                 pool-1-thread-1 |  main |  false |
TIMED_WAITING |        5 |
|   23 | ForkJoinPool.commonPool-worker-2 |  main |   true |
TIMED_WAITING |        5 |
================================================================
========================

Result ->true
```

supplyAsync 创建带有返回值的异步任务。

supplyAsync 创建带有返回值的异步任务。它有如下两个方法，一个是使用默认线程池（ForkJoinPool.commonPool()）的方法，一个是带有自定义线程池的重载方法

```
// 带返回值异步请求，默认线程池
public static <U> CompletableFuture<U> supplyAsync(Supplier<U>
supplier)

// 带返回值的异步请求，可以自定义线程池
public static <U> CompletableFuture<U> supplyAsync(Supplier<U>
supplier, Executor executor)
```

```
package cn.netkiller;

import cn.netkiller.thread.ThreadManager;
import lombok.SneakyThrows;

import java.util.concurrent.CompletableFuture;

public class Main {
    @SneakyThrows
    public static void main(String[] args) {

        CompletableFuture<String> completableFuture =
CompletableFuture.supplyAsync(() -> {
            System.out.println("do something....");
            return "done";
        });

        System.out.println("Result ->" +
completableFuture.get());

        ThreadManager tm = new ThreadManager();
        System.out.println(tm.show());
    }
}
```

运行结果

```
do something....
Result ->done
================================================================
========================
|   ID |                                Name | Group | Daemon |
State | Priority |
----------------------------------------------------------------
------------------------
|    1 |                                main |  main |  false |
RUNNABLE |        5 |
|   21 | ForkJoinPool.commonPool-worker-1 |  main |   true |
TIMED_WAITING |        5 |
================================================================
========================
```

```
package cn.netkiller;

import cn.netkiller.thread.ThreadManager;
import lombok.SneakyThrows;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Main {
    @SneakyThrows
    public static void main(String[] args) {


        ExecutorService executorService =
Executors.newSingleThreadExecutor();
```

```
        CompletableFuture<String> completableFuture =
CompletableFuture.supplyAsync(() -> {
            System.out.println("do something....");
            try {
                Thread.sleep(30000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            return "done";
        }, executorService);

        System.out.println("Result ->" +
completableFuture.get());

        ThreadManager tm = new ThreadManager();
        System.out.println(tm.show());
    }
}
```

运行结果

```
do something....
Result ->done
=================================================================
=======
|   ID |            Name | Group | Daemon |         State |
Priority |
-----------------------------------------------------------------
-------
|    1 |            main |  main |  false |      RUNNABLE |
5 |
|   21 | pool-1-thread-1 |  main |  false |       WAITING |
5 |
=================================================================
=======
```

设置线程名称

```
package cn.netkiller;

import cn.netkiller.thread.ThreadManager;
import lombok.SneakyThrows;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Main {
    @SneakyThrows
    public static void main(String[] args) {

        CompletableFuture<String> completableFuture =
CompletableFuture.supplyAsync(() -> {
            System.out.println("do something....");
            Thread.currentThread().setName("测试有返回值的异步执
行");
            return "done";
        });

        System.out.println("Result ->" +
completableFuture.get());

        ThreadManager tm = new ThreadManager();
        System.out.println(tm.show());
    }
}
```

运行结果

```
do something....
Result ->done
================================================================
===
|   ID |        Name | Group | Daemon |         State |
Priority |
```

```
------------------------------------------------------------------
---
|    1 |        main |  main |  false |      RUNNABLE |
5 |
|   21 | 测试有返回值的异步执行 |  main |   true | TIMED_WAITING |
5 |
==================================================================
===
```

thenRun / thenRunAsync

thenRun/thenRunAsync 功能是什么？完成前置任务之后，自己在执行。

thenRun/thenRunAsync 区别是什么？thenRun 使用同一个线程执行任务，thenRunAsync 会再开一个新线程执行任务。

```
    @GetMapping("/completableFutureRun")
    public String completableFutureRun() {
        CompletableFuture<Void> completableFuture =
CompletableFuture.runAsync(() -> {

System.out.println(Thread.currentThread().getName() + " -
CompletableFuture 前置任务");
            try {
                Thread.sleep(5 * 1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        });
        CompletableFuture thenRun =
completableFuture.thenRun(() -> {

System.out.println(Thread.currentThread().getName() + " - 接着
执行第二个 thenRun 任务");
        });
        CompletableFuture thenRunAsync =
completableFuture.thenRunAsync(() -> {
```

```
System.out.println(Thread.currentThread().getName() + " - 接着
执行第二个 thenRunAsync 任务");
        });
        return "Done";
    }
```

运行结果

```
ForkJoinPool.commonPool-worker-1 - CompletableFuture 前置任务
ForkJoinPool.commonPool-worker-1 - 接着执行第二个 thenRun 任务
ForkJoinPool.commonPool-worker-2 - 接着执行第二个 thenRunAsync 任
务
```

这里可以看到 thenRunAsync 的线程变化，开启新线程 ForkJoinPool.commonPool-worker-2 处理任务

thenAccept / thenAcceptAsync

thenAccept/thenAcceptAsync 的功能是，前置任务执行完毕之后，将返回值给到 thenAccept/thenAcceptAsync，再执行接下来的任务。

```
    @GetMapping("/completableFutureAccept")
    public String completableFutureAccept() {
        CompletableFuture<String> supplyAsync =
CompletableFuture.supplyAsync(() -> {
            log.info(Thread.currentThread().getName() + " -
CompletableFuture 前置任务");
            try {
                Thread.sleep(5 * 1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            return "前置任务执行完成";
```

```
        });
        CompletableFuture<Void> thenAccept =
supplyAsync.thenAccept((rev) -> {
            log.info(Thread.currentThread().getName() + " - 接
着执行第二个 thenAccept 任务");
            log.info("前置任务返回值: " + rev);
        });
        CompletableFuture<Void> thenAcceptAsync =
supplyAsync.thenAcceptAsync((rev) -> {
            log.info(Thread.currentThread().getName() + " - 接
着执行第二个 thenAcceptAsync 任务");
            log.info("前置任务返回值: " + rev);
        });
        return "Done";
    }
```

输出结果

```
2023-05-10T10:38:48.008+08:00  INFO 96282 --- [onPool-worker-1]
c.n.c.test.TestThreadController          :
ForkJoinPool.commonPool-worker-1 - CompletableFuture 前置任务
2023-05-10T10:38:53.015+08:00  INFO 96282 --- [onPool-worker-2]
c.n.c.test.TestThreadController          :
ForkJoinPool.commonPool-worker-2 - 接着执行第二个 thenAcceptAsync
任务
2023-05-10T10:38:53.015+08:00  INFO 96282 --- [onPool-worker-1]
c.n.c.test.TestThreadController          :
ForkJoinPool.commonPool-worker-1 - 接着执行第二个 thenAccept 任务
2023-05-10T10:38:53.016+08:00  INFO 96282 --- [onPool-worker-2]
c.n.c.test.TestThreadController          : 前置任务返回值: 前置任务
执行完成
2023-05-10T10:38:53.016+08:00  INFO 96282 --- [onPool-worker-1]
c.n.c.test.TestThreadController          : 前置任务返回值: 前置任务
执行完成
```

thenApply / thenApplyAsync

thenApply/thenApplyAsync 前置任务执行完毕之后，结果作为入参，thenApply/thenApplyAsync 执行完毕之后再返回执行结果

```
    @GetMapping("/completableFutureApply")
    public String completableFutureApply() throws
ExecutionException, InterruptedException {
        CompletableFuture<String> supplyAsync =
CompletableFuture.supplyAsync(() -> {
            log.info(Thread.currentThread().getName() + " -
CompletableFuture 前置任务");
            try {
                Thread.sleep(5 * 1000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            return "第一步";
        });

        CompletableFuture<String> thenApply =
supplyAsync.thenApply((rev) -> {
            log.info(Thread.currentThread().getName() + " - 接
着执行第二个 thenApply 任务");
            log.info("前置任务返回值: " + rev);
            return "第二步";
        });

        CompletableFuture<String> thenApplyAsync =
supplyAsync.thenApplyAsync((rev) -> {
            log.info(Thread.currentThread().getName() + " - 接
着执行第二个 thenApplyAsync 任务");
            log.info("前置任务返回值: " + rev);
            return "第二步";
        });
        log.info("supplyAsync: {}", supplyAsync.get());
        log.info("thenApply: {}", thenApply.get());
        log.info("thenApplyAsync: {}", thenApplyAsync.get());
        return "Done";
    }
```

```
2023-05-10T10:39:57.913+08:00  INFO 96282 --- [onPool-worker-2]
c.n.c.test.TestThreadController          :
ForkJoinPool.commonPool-worker-2 - CompletableFuture 前置任务
2023-05-10T10:40:02.917+08:00  INFO 96282 --- [  XNIO-1 task-2]
c.n.c.test.TestThreadController          : XNIO-1 task-2 - 接着
执行第二个 thenApply 任务
2023-05-10T10:40:02.917+08:00  INFO 96282 --- [onPool-worker-2]
c.n.c.test.TestThreadController          :
ForkJoinPool.commonPool-worker-2 - 接着执行第二个 thenApplyAsync
任务
2023-05-10T10:40:02.918+08:00  INFO 96282 --- [  XNIO-1 task-2]
c.n.c.test.TestThreadController          : 前置任务返回值：第一步
2023-05-10T10:40:02.918+08:00  INFO 96282 --- [onPool-worker-2]
c.n.c.test.TestThreadController          : 前置任务返回值：第一步
2023-05-10T10:40:02.918+08:00  INFO 96282 --- [  XNIO-1 task-2]
c.n.c.test.TestThreadController          : supplyAsync：第一步
2023-05-10T10:40:02.918+08:00  INFO 96282 --- [  XNIO-1 task-2]
c.n.c.test.TestThreadController          : thenApply：第二步
2023-05-10T10:40:02.919+08:00  INFO 96282 --- [  XNIO-1 task-2]
c.n.c.test.TestThreadController          : thenApplyAsync：第二
步
```

runAsync / thenAccept / thenApply 区别

runAsync 配合 thenRun/thenRunAsync 使用

supplyAsync 配合 thenAccept/thenAcceptAsync 使用

```
supplyAsync -- 返回值 --> thenAccept/thenAcceptAsync --> 无返回值
```

supplyAsync 配合 thenApply/thenApplyAsync 使用

```
supplyAsync -- 返回值 --> thenApply/thenApplyAsync -- 返回值 -->
```

whenComplete

whenComplete 与 runAsync / thenAccept / thenApply 区别是能处理 Throwable

```
    @GetMapping("/completableFutureWhenComplete")
    public String completableFutureWhenComplete() throws
ExecutionException, InterruptedException {

        CompletableFuture<String> completableFuture =
CompletableFuture.supplyAsync(() -> {
            System.out.println("当前线程名称: " +
Thread.currentThread().getName());
            return "前置任务完成";
        }).whenComplete((result, throwable) -> {
            System.out.println("前置任务返回值: " + result);
        });
        System.out.println(completableFuture.get());
        return "Done";
    }
```

运行结果

```
当前线程名称: ForkJoinPool.commonPool-worker-1
前置任务返回值: 前置任务完成
前置任务完成
```

超时处理

```
                CompletableFuture<String> future =
CompletableFuture.supplyAsync(() -> {
            try {
                Thread.sleep(5 * 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            return "hello";
        });

        future.completeOnTimeout("default timeout result", 3
* 1000, TimeUnit.MILLISECONDS);
```

按顺序执行

```
@Service
public class MyService {

    @Async("threadPoolTaskExecutor")
    public CompletableFuture<String> asyncMethod1() {
        // 异步方法1逻辑...
        return CompletableFuture.completedFuture("Result1");
    }

    @Async("threadPoolTaskExecutor")
    public CompletableFuture<String> asyncMethod2() {
        // 异步方法2逻辑...
        return CompletableFuture.completedFuture("Result2");
    }
}

// 调用异步方法并处理结果顺序
CompletableFuture<String> future1 = myService.asyncMethod1();
CompletableFuture<String> future2 =
future1.thenCompose(result1 -> myService.asyncMethod2());

String finalResult = future2.get(); // 阻塞等待最终结果
```

```

```

获取结果

```
@Service
public class MyService {

    @Async
    public CompletableFuture<String> asyncMethod() {
        // 异步方法逻辑...
        return CompletableFuture.completedFuture("Result");
    }
}

// 调用异步方法并获取结果
CompletableFuture<String> future = myService.asyncMethod();
String result = future.get(); // 阻塞等待结果
```

异常处理

```
@Service
public class MyService {

    @Async
    public CompletableFuture<String> asyncMethod() {
        try {
            // 异步方法逻辑...
            return
CompletableFuture.completedFuture("Success");
        } catch (Exception e) {
            // 处理异常...
            return CompletableFuture.failedFuture(e);
        }
    }
}
```

```
// 调用异步方法并处理异常
CompletableFuture<String> future = myService.asyncMethod();
future.exceptionally(e -> {
    // 异常处理逻辑...
    return "Error";
});
```

```
    @GetMapping("/completableFutureExceptionally")
    public String completableFutureExceptionally() throws
ExecutionException, InterruptedException {

        CompletableFuture.supplyAsync(() -> {
            System.out.println("当前线程名称: " +
Thread.currentThread().getName());
            throw new RuntimeException();
        }).exceptionally((e) -> {
            System.out.println(e.getMessage());
            return "程序出现异常";
        });

//        CompletableFuture<String> completableFuture =
CompletableFuture.supplyAsync(() -> {
////            throw new RuntimeException();
//            return "程序出现异常";
//        }).exceptionally((e) -> {
//            System.out.println("程序出现异常");
//            return "程序出现异常";
//        });
//        System.out.println(completableFuture.get());

        return "Done";
    }
```

输出结果

```
当前线程名称：ForkJoinPool.commonPool-worker-1
java.lang.RuntimeException
```

11.14. java.util.stream

IntStream

```
                List<Picture> list = new ArrayList<Picture>
();

                IntStream.range(1, 10).forEach(i -> {
            Picture picture = new Picture();
            picture.setId(Long.valueOf(i));

picture.setImage("https://www.netkiller.cn/images/" + i +
".png");
            list.add(picture);
        });
```

12. IO

12.1. 取出文件名中的扩展名

```
        File file = new File("HelloWorld.jpeg");
        String fileName = file.getName();
        String suffix =
fileName.substring(fileName.lastIndexOf(".") + 1);
        System.out.println(suffix);
```

getAbsolutePath() 获取绝对路径

```
        File configFile = new
File(System.getProperty("user.dir") + configPath);
        System.out.printf("configFile : %s\n",
configFile.getAbsolutePath());
```

创建目录 mkdir()

```
//判断cache目录是否存在的代码，如果不存在则创建cache目录
String path="/tmp/cache";
File dirname = new File(path);

//目录不存在
if (!dirname.isDirectory())
{
```

```
     dirname.mkdir(); //创建目录
}
```

12.2. 临时文件

```
package cn.netkiller.file;

import java.io.File;
import java.io.IOException;

public class CreateTempFileExample
{
    public static void main(String[] args)
    {

        try{

           //create a temp file
           File temp = File.createTempFile("temp-file-name",
".tmp");

           System.out.println("Temp file : " +
temp.getAbsolutePath());

        }catch(IOException e){

           e.printStackTrace();

        }

    }
}
```

Temp file : C:\Users\mkyong\AppData\Local\Temp\temp-file-name623426.tmp

12.3. FileWriter 文本写入文件

```
import java.io.*;

public class Main {

    public static void main(String[] args) {

        try {
            String str = "SomeMoreTextIsHere";
            File newTextFile = new File("thetextfile.txt");

            FileWriter fw = new FileWriter(newTextFile);
            fw.write(str);
            fw.close();

        } catch (IOException iox) {
            //do stuff with exception
            iox.printStackTrace();
        }
    }
}
```

12.4. BufferedWriter

```
        File file = new File( fileName );

    // if file doesnt exists, then create it
    if ( ! file.exists( ) )
    {
        file.createNewFile( );
    }

    FileWriter fw = new FileWriter( file.getAbsoluteFile( )
);
```

```
    BufferedWriter bw = new BufferedWriter( fw );
    bw.write( text );
    bw.close( );
```

12.5. inputStream.transferTo()

```
var classLoader = ClassLoader.getSystemClassLoader();
var inputStream = classLoader.getResourceAsStream("hello.txt");
var tmp = File.createTempFile("tmp", "txt");
try (var outputStream = new FileOutputStream(tmp)) {
    inputStream.transferTo(outputStream);
}
```

12.6. InputStreamReader

```
InputStreamReader stream = new
InputStreamReader(this.getClass().getClassLoader().getResourc
eAsStream(filename));
```

12.7. 获得 Resource 下文件路径

```
String path =
Main.class.getClassLoader().getResource("netkiller-
config.yaml").getPath();
String path =
Main.class.getClassLoader().getResource("netkiller-
config.yaml").getPath();
```

12.8. PrintWriter

```
package cn.netkiller.io;

import java.io.FileNotFoundException;
import java.io.PrintWriter;

public class PrintWriterTest {

        public PrintWriterTest() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws
FileNotFoundException {
                // TODO Auto-generated method stub
                PrintWriter output = new
PrintWriter("temp.txt");
                output.print("Welcome to Java!");
                output.close();
        }

}
```

12.9. OutputStreamWriter

```
                File file = new File("/tmp" + File.separator
+ "netkiller.txt");
                OutputStreamWriter outputStreamWriter = new
OutputStreamWriter(new FileOutputStream(file), "utf-8");
                outputStreamWriter.write("Netkiller Java 手
札");
```

```
                outputStreamWriter.close();
```

12.10. FileOutputStream

```
byte[] data = ...

FileOutputStream fos = ...
fos.write(data, 0, data.length);
fos.flush();
fos.close();
```

12.11. FileInputStream

```
File inputFile = new File(filePath);
byte[] data = new byte[inputFile.length()];
FileInputStream fis = new FileInputStream(inputFile);
fis.read(data, 0, data.length);
fis.close();
```

12.12. Scanner

```
Scanner input = new Scanner(new File("temp.txt"));
System.out.print(input.nextLine());
```

```
package cn.netkiller.io;

import java.io.File;
import java.io.FileNotFoundException;
//import java.io.PrintWriter;
import java.util.Scanner;

public class PrintWriterTest {

        private static Scanner input;

        public PrintWriterTest() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws
FileNotFoundException {
                // TODO Auto-generated method stub
                // PrintWriter output = new
PrintWriter("temp.txt");
                // output.print("Welcome to Java!");
                // output.close();

                input = new Scanner(new File("temp.txt"));
                System.out.print(input.nextLine());
        }

}
```

12.13. 二进制文件

理解二进制文件

我们运行下面一段程序，向文件 netkiller.bin 中写入一个整形数值 1，然后观察文件变化

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeInt(1);
                out.close();
```

打开终端，使用 xxd 命令查看二进制文件

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -b netkiller.bin
00000000: 00000000 00000000 00000000 00000001
....
```

可以看到一串二进制 00000000 00000000 00000000 00000001，运行下面程序可以讲二进制转换为十进制，注意替换掉空格。

```
                int n = Integer.valueOf("00000000 00000000
00000000 00000001".replaceAll(" ", ""), 2);
                System.out.println(n);
```

运行结果是 1 ，为什前面那么多 0 呢？请运行下面一段程序

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeInt(Integer.MAX_VALUE);
                out.close();
```

现在观察结果

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -b netkiller.bin
00000000: 01111111 11111111 11111111 11111111
....
```

```
                int n = Integer.valueOf("01111111 11111111
11111111 11111111".replaceAll(" ", ""), 2);
                System.out.println(n);
```

输出结果是 2147483647, 这是 int 得最大值，2147483647 + 1 会怎么样呢?

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeInt(Integer.MAX_VALUE + 1);
                out.close();

                System.out.println(Integer.MAX_VALUE + 1);
```

输出结果是 -2147483648，正确应该是 2147483648 这就是整形溢出。整形变量得二进制表示方法是4个字节长度32位 00000000 00000000 00000000 00000000 到 01111111 11111111 11111111 11111111 ， 其中第一位0表示正数1表示负数。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -b netkiller.bin
00000000: 10000000 00000000 00000000 00000000
....
```

整形溢出演示，超出整形范围怎么办？ 使用 Long 型。

```
System.out.println(Integer.MAX_VALUE);
System.out.println(Integer.MAX_VALUE + 1);
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MIN_VALUE - 1);

输出结果如下：

2147483647
-2147483648
-2147483648
2147483647
```

负数演示

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
 DataOutputStream(new FileOutputStream(filename));
                out.writeInt(-1);
                out.writeInt(Integer.MAX_VALUE + 1);
                out.close();
```

-1 得结果是 11111111 11111111 11111111 11111111

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -b netkiller.bin
00000000: 11111111 11111111 11111111 11111111
....
```

现在我们存储两个整形数值

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeInt(1);
                out.writeInt(-1);
                out.close();
```

很清楚的看到里面有两个数值，1 和 -1

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 4 -b
netkiller.bin
00000000: 00000000 00000000 00000000 00000001  ....
00000004: 11111111 11111111 11111111 11111111  ....
```

读取二进制文件中的 int 数据

```
                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        int i = in.readInt();
                        System.out.println(i);
                } catch (EOFException e) {
```

```
                        e.printStackTrace();
                }
```

byte 类型

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
 DataOutputStream(new FileOutputStream(filename));
                out.writeByte(1);
                out.close();
```

byte 只占用一个字节8位

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 4 -b
netkiller.bin
00000000: 00000001
```

如果写入 -1 结果是，由此得出 第一位 0 是正数，1 是负数，可以得出他的取值范围 -128 ~ 127。超出范围也会溢出。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 4 -b
netkiller.bin
00000000: 11111111
```

常常写入最小值与最大值

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeByte(Byte.MIN_VALUE);
                out.writeByte(Byte.MAX_VALUE);
                out.close();
```

运行结果

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 1 -b
netkiller.bin
00000000: 10000000  .
00000001: 01111111  .
```

写入一个字符

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeBytes("a");
                out.close();
```

写入结果

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 1 -b
netkiller.bin
00000000: 01100001  a
```

从 ASCII 表中查出 01100001 十进制 97 十六进制 61 对应字母 a

写入一段字符串

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeBytes("http://www.netkiller.cn");
                out.close();
```

运行结果

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 8 -b
netkiller.bin
00000000: 01101000 01110100 01110100 01110000 00111010 00101111
00101111 01110111  http://w
00000008: 01110111 01110111 00101110 01101110 01100101 01110100
01101011 01101001  ww.netki
00000010: 01101100 01101100 01100101 01110010 00101110 01100011
01101110           ller.cn
```

读取二进制文件中的 byte 字符串，readAllBytes() 可以一次读取所有 byte 到 byte[] 中。

```
                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        System.out.println(new
```

```
String(in.readAllBytes()));
                } catch (EOFException e) {
                        e.printStackTrace();
                }
```

readByte() 逐字节读取

```
                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        char c = ' ';
                        while (true) {
                                try {
                                        c = (char)
in.readByte();
                                        System.out.print(c);

                                } catch (EOFException e) {
                                        System.out.println();
                                        break;
                                }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
```

现在我们已经掌握了 byte 的操作方法，现在我们来做一个例子，读取 int 数据，int 是由 4 个字节组成一组。所以我们每次取 4个字节。

```
                // 这个例子中，我们写入三个数值到 netkiller.bin 文
件，分别是 1024，-128，2147483647
                String filename = "netkiller.bin";
```

```
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeInt(1024);
                out.writeInt(-128);
                out.writeInt(Integer.MAX_VALUE);
                out.close();
```

二进制文件如下

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 4 -b
netkiller.bin
00000000: 00000000 00000000 00000100 00000000  ....
00000004: 11111111 11111111 11111111 10000000  ....
00000008: 01111111 11111111 11111111 11111111  ....
```

从二进制文件读出 int 数据。

```
                String filename = "netkiller.bin";
                FileInputStream stream = new
FileInputStream(filename);

                byte[] buffer = new byte[4];

                while (stream.read(buffer) != -1) {
                        ByteBuffer byteBuffer =
ByteBuffer.wrap(buffer);

System.out.println(byteBuffer.getInt());
                }
```

运行结果

```
1024
-128
2147483647
```

boolean 布尔型

我们想文件写入两个布尔类型，一个是 true, 另一个是 false

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
 DataOutputStream(new FileOutputStream(filename));
                out.writeBoolean(true);
                out.writeBoolean(false);
                out.close();
```

运行结果可以看出 boolean 使用了一个字节，最后一位 1 表示true, 0 表示 false。所以对于二进制文件最小单位就是 byte 字节，虽然 boolean型只需要一个 1 bit 位，但是存储的最小单位是字节，所以前面需要补7个零 0000000。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 1 -b
netkiller.bin
00000000: 00000001  .
00000001: 00000000  .
```

使用 ls 命令可以看这个文件占用了 2B（两个字节）

```
```

```
neo@MacBook-Pro ~/workspace/netkiller % ll netkiller.bin
-rw-r--r--  1 neo  staff     2B Oct 18 13:47 netkiller.bin
```

读取二进制文件中的 boolean 数据

```
                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        boolean bool = in.readBoolean();
                        System.out.println(bool);
                } catch (EOFException e) {
                        e.printStackTrace();
                }
```

Long 型

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeLong(1);
                out.close();
```

有了上面 int 型数据的经验，下面一看你就会明白。long 型采用 8 个字节保存数据，是 int 的一倍。取值范围这里就不多说了，也会存在溢出现象。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 8 -b
netkiller.bin
```

```
00000000: 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000001  ........
```

取值范围

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
 DataOutputStream(new FileOutputStream(filename));
                out.writeLong(Long.MIN_VALUE);
                out.writeLong(Long.MAX_VALUE);
                out.close();
```

输出文件

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 8 -b
netkiller.bin
00000000: 10000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000  ........
00000008: 01111111 11111111 11111111 11111111 11111111 11111111
11111111 11111111  ........
```

读取二进制文件中的 long 数据

```
                DataInputStream in = new DataInputStream(new
 BufferedInputStream(new FileInputStream(filename)));
                try {

                        long l = in.readLong();
                        System.out.println(l);
```

```
                } catch (EOFException e) {
                        e.printStackTrace();
                }
```

chat 类型

有符号 signed char 类型的范围为 -128~127

无符号 unsigned char 的范围为0~ 255

char 与 byte 操作类似，我们首先去 ASCII 表查找字符 A 对应 65，我们将 65 写入二进制文件。然后读取该字符，输出结果是 A。

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeChar(65);
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {

                        char c = in.readChar();
                        System.out.println(c);

                } catch (EOFException e) {
                        e.printStackTrace();
                }
```

从二进制文件中我们可以看到 char 类型占用2个字节16位

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 2 -b
```

```
netkiller.bin
00000000: 00000000 01000001  .A
```

使用 writeChars()写入字符串到二进制文件

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeChars("http://www.netkiller.cn");
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));

                char c = ' ';
                while (true) {
                        try {
                                c = in.readChar();
                                System.out.print(c);

                        } catch (EOFException e) {
                                System.out.println();
                                break;
                        }
                }
```

二进制文件如下，你会发现第一个字节没有用到，很多 00000000 所以如果存储英文 byte 更适合，char 是双倍 byte 开销。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 8 -b
netkiller.bin
00000000: 00000000 01101000 00000000 01110100 00000000 01110100
00000000 01110000  .h.t.t.p
00000008: 00000000 00111010 00000000 00101111 00000000 00101111
```

```
00000000 01110111  .:././.w
00000010: 00000000 01110111 00000000 01110111 00000000 00101110
00000000 01101110  .w.w...n
00000018: 00000000 01100101 00000000 01110100 00000000 01101011
00000000 01101001  .e.t.k.i
00000020: 00000000 01101100 00000000 01101100 00000000 01100101
00000000 01110010  .l.l.e.r
00000028: 00000000 00101110 00000000 01100011 00000000 01101110
...c.n
```

存储汉字

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                String s = "陈";
                char name = s.charAt(s.length() - 1);
                out.writeChar(name);
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));

                char c = ' ';
                while (true) {
                        try {
                                c = in.readChar();
                                System.out.print(c);

                        } catch (EOFException e) {
                                System.out.println();
                                break;
                        }
                }
```

二进制文件如下，使用两个字节表示一个汉字

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 2 -b
netkiller.bin
00000000: 10010110 01001000  .H
```

转成 Hex 十六进制，得到 96 48 两个数字。

```
neo@MacBook-Pro ~/workspace/netkiller % hexdump  netkiller.bin
0000000 96 48
0000002
```

现在去搜索引擎搜索“汉字内码”，然后查询“陈”这个汉字，可以看到 Unicode编码16进制就是 96 48

尝试写入汉字字符串

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeChars("陈景峰");
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        char c = ' ';
                        while (true) {
                                try {
                                        c = in.readChar();
                                        System.out.print(c);

                                } catch (EOFException e) {
                                        System.out.println();
                                        break;
```

```
                                }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
```

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -b netkiller.bin
00000000: 10010110 01001000 01100110 01101111 01011100 11110000
.Hfo\.
```

UTF 字符串

这次我们使用新的文件名 netkiller.txt

```
                String filename = "netkiller.txt";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));

                out.writeUTF("峰");

                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        System.out.println(in.readUTF());
                } catch (EOFException e) {
                        e.printStackTrace();
                }
```

查看二进制文件，一个汉字怎么这么多字节?

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -b netkiller.txt
00000000: 00000000 00000011 11100101 10110011 10110000
.....
```

转成 16 禁止看看。

```
neo@MacBook-Pro ~/workspace/netkiller % hexdump netkiller.txt
0000000 00 03 e5 b3 b0
0000005
```

我们在网上查询 “峰” 字的汉字内码，可以看到UTF-8 内码是 E5 B3 B0。这是因为UTF8使用三个字节存储汉字。 00000000 00000011 可能是 UTF 标志位，具体我也不太清楚，总之不是 BOM 信息。

我们现在写入一个字符串试试

```
out.writeUTF("陈景峰");
```

xxd -s 2 -c 3 表示跳过两个字节，三列显示

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -s 2 -c 3 -b
netkiller.txt
00000002: 11101001 10011001 10001000  ...
00000005: 11100110 10011001 10101111  ...
00000008: 11100101 10110011 10110000  ...
```

UTF字符是可以直接使用文本工具查看的。

```
neo@MacBook-Pro ~/workspace/netkiller % cat netkiller.txt
        陈景峰
```

Short 类型

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
 DataOutputStream(new FileOutputStream(filename));

                out.writeShort(1);

                out.flush();
                out.close();
```

输出结果，Short 使用两个字节16位表示。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 2 -b
netkiller.bin
00000000: 00000000 00000001  ..
```

Short 分为有符号和无符号类型

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
 DataOutputStream(new FileOutputStream(filename));
```

```
                out.writeShort(1);
                out.writeShort(1);
                out.writeShort(-1);
                out.writeShort(-1);

                out.flush();
                out.close();


                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));
                try {
                        System.out.println(in.readShort());

System.out.println(in.readUnsignedShort());
                        System.out.println(in.readShort());

System.out.println(in.readUnsignedShort());
                } catch (EOFException e) {
                        e.printStackTrace();
                }
```

运行结果

```
1
1
-1
65535
```

有符号的取值范围

```
最小值：Short.MIN_VALUE=-32768 （-2的15此方）
最大值：Short.MAX_VALUE=32767 （2的15次方-1）
```

无符号的取值范围是 0 ~ 65535

float 单精度浮点类型

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));

                out.writeFloat(0);
                out.writeFloat(1.0f);
                out.writeFloat(1.1f);
                out.flush();
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));

                float c = 0;
                while (true) {
                        try {
                                c = in.readFloat();
                                System.out.println(c);

                        } catch (EOFException e) {
                                System.out.println();
                                break;
                        }
                }
```

float 使用 4 字节 32 为表示浮点类型，float 不同于前面数据类型，无法直接读取浮点数，需要经过计算才能得出，有点复杂。

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 4 -b
```

```
netkiller.bin
00000000: 00000000 00000000 00000000 00000000  ....
00000004: 00111111 10000000 00000000 00000000  ?...
00000008: 00111111 10001100 11001100 11001101  ?...
```

浮点型示意图

```
 /-------------- 32 bit -----------------\
 | 1 |      8   |            23            |
 |-----------------------------------------|
 31 30         22                         0
  ^         ^                 ^
符号位    指数位                  尾数部分          32位

首先float二进制是从后向前读。与上面所有类型相反。
符号位(Sign) : 0代表正，1代表为负
指数位（Exponent）:用于存储科学计数法中的指数数据，并且采用移位存储
尾数部分（Mantissa）：尾数部分


将一个内存存储的float二进制格式转化为十进制的步骤：
（1）将第22位到第0位的二进制数写出来，在最左边补一位“1”，得到二十四位有效数字。将小数点点在最左边那个“1”的右边。
（2）取出第29到第23位所表示的值n。当30位是“0”时将n各位求反。当30位是“1”时将n增1。
（3）将小数点左移n位（当30位是“0”时）或右移n位（当30位是“1”时），得到一个二进制表示的实数。
（4）将这个二进制实数化为十进制，并根据第31位是“0”还是“1”加上正号或负号即可。


1.0f = 00111111 10000000 00000000 00000000

Sign 31 位是 0 表示正数
Exponent 23~30 位 0111111 1
Mantissa 0~22 位 0000000 00000000 00000000

得到

| 0 | 0111111 1 | 0000000 00000000 00000000 |
```

```
具体细节请参考 IEEE R32.24
```

double 数据类型

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeDouble(12.5d);
                out.flush();
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));

                double d = 0d;
                while (true) {
                        try {
                                d = in.readDouble();
                                System.out.println(d);

                        } catch (EOFException e) {
                                System.out.println();
                                break;
                        }
                }
```

二进制文件

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 8 -b
netkiller.bin
00000000: 01000000 00101001 00000000 00000000 00000000 00000000
00000000 00000000  @).....
```

```
 /-------------------------- 64 bit ----------------------------
--\
 | 1 |    11    |           52
|
 |-------------------------------------------------------------
---|
 63 62          51
0
  ^          ^                    ^
符号位      指数位               尾数部分
64位

首先float二进制是从后向前读。与上面所有类型相反。
符号位(Sign) : 0代表正，1代表为负
指数位（Exponent）:用于存储科学计数法中的指数数据，并且采用移位存储
尾数部分（Mantissa）：尾数部分

详细参加考 IEEE R64.53
```

二进制文件操作演示

所有类型演示一遍

```
                String filename = "netkiller.bin";
                DataOutputStream out = new
DataOutputStream(new FileOutputStream(filename));
                out.writeInt(1024);
                out.writeShort(255);
                out.writeLong(100000000000L);
                out.writeFloat(3.14f);
                out.writeDouble(3.141592653579d);
                out.writeBoolean(true);
                out.writeChar(165);
                out.writeChars("陈景峰");
```

```
                out.writeUTF("Netkiller Java 手札 -
http://www.netkiller.cn");
                out.writeChars("这是最后一行\r\n");
                out.flush();
                out.close();

                DataInputStream in = new DataInputStream(new
BufferedInputStream(new FileInputStream(filename)));

                System.out.println(in.readInt());
                System.out.println(in.readUnsignedShort());
                System.out.println(in.readLong());
                System.out.println(in.readFloat());
                System.out.println(in.readDouble());
                System.out.println(in.readBoolean());
                System.out.println(in.readChar());

                int i = 0;
                String name = "";
                while (i < 3) {
                        try {
                                char c = in.readChar();
                                name += c;
                        } catch (EOFException e) {
                                break;
                        }
                        i++;
                }

                System.out.println(name);
                System.out.println(in.readUTF());
                System.out.println(in.readUTF());
```

需要注意的一点是 out.writeChars("陈景峰"); 写入char字符串，在读取的时候你需要知道字符串的长度。然后循环取出char数据。

二进制文件内容

```
neo@MacBook-Pro ~/workspace/netkiller % xxd -c 8 -b
```

```
netkiller.bin
00000000: 00000000 00000000 00000100 00000000 00000000 11111111
00000000 00000000  ........
00000008: 00000000 00010111 01001000 01110110 11101000 00000000
01000000 01001000  ..Hv..@H
00000010: 11110101 11000011 01000000 00001001 00100001 11111011
01010100 01000011  ..@.!.TC
00000018: 11001110 00101000 00000001 00000000 10100101 10010110
01001000 01100110  .(....Hf
00000020: 01101111 01011100 11110000 00000000 00101111 01001110
01100101 01110100  o\../Net
00000028: 01101011 01101001 01101100 01101100 01100101 01110010
00100000 01001010  killer J
00000030: 01100001 01110110 01100001 00100000 11100110 10001001
10001011 11100110  ava ....
00000038: 10011100 10101101 00100000 00101101 00100000 01101000
01110100 01110100  .. - htt
00000040: 01110000 00111010 00101111 00101111 01110111 01110111
01110111 00101110  p://www.
00000048: 01101110 01100101 01110100 01101011 01101001 01101100
01101100 01100101  netkille
00000050: 01110010 00101110 01100011 01101110 10001111 11011001
01100110 00101111  r.cn..f/
00000058: 01100111 00000000 01010100 00001110 01001110 00000000
10001000 01001100  g.T.N..L
00000060: 00000000 00001101 00000000 00001010
....
```

16 进制编辑器更好阅读一些

```
neo@MacBook-Pro ~/workspace/netkiller % hexdump -C
netkiller.bin
00000000  00 00 04 00 00 ff 00 00  00 17 48 76 e8 00 40 48
|..........Hv..@H|
00000010  f5 c3 40 09 21 fb 54 43  ce 28 01 00 a5 96 48 66
|..@.!.TC.(....Hf|
00000020  6f 5c f0 00 2f 4e 65 74  6b 69 6c 6c 65 72 20 4a
|o\../Netkiller J|
00000030  61 76 61 20 e6 89 8b e6  9c ad 20 2d 20 68 74 74
|ava ...... - htt|
```

```
00000040  70 3a 2f 2f 77 77 77 2e  6e 65 74 6b 69 6c 6c 65
|p://www.netkille|
00000050  72 2e 63 6e 8f d9 66 2f  67 00 54 0e 4e 00 88 4c
|r.cn..f/g.T.N..L|
00000060  00 0d 00 0a
|....|
00000064
```

检查文件是否是 **png** 文件

```
package cn.netkiller.io;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Arrays;

public class PngFile {

        private static InputStream inputStream;

        public PngFile() {
        }

        public static void main(String[] args) throws
FileNotFoundException {

                int[] pngSignature = { 137, 80, 78, 71, 13,
10, 26, 10 };

                try {
                        inputStream = new
FileInputStream("/Users/neo/Downloads/NBRC.png");

                        byte[] headerBytes = new byte[8];
                        inputStream.read(headerBytes);

                        // for (byte b : headerBytes) {
```

```
                        // System.out.println(b);
                        // }

                        int[] pngHeader = new int[8];

                        for (int i = 0; i <
headerBytes.length; i++) {
                                pngHeader[i] = (int)
headerBytes[i] & 0xff;
                        }

                        if (Arrays.equals(pngHeader,
pngSignature)) {
                                System.out.println("This is a
PNG file! ");
                        }
                } catch (IOException e) {
                        e.printStackTrace();
                }

        }

}
```

13. Reflection 反射

```
this.getClass().getName() //当前Class名字
Thread.currentThread().getStackTrace()[1].getMethodName()); //
当前方法名
```

13.1. 获得所有变量

```
        Field[] fields = objClass.getFields();
        for (Field field : fields) {
                System.out.println(field.getName());
        }
```

注意：只能去除 public变量

13.2. 批量赋值

13.3. 方法操作

JAVA反射调用方法的步骤有三步

```
得到要调用类的class
得到要调用的类中的方法(Method)
方法调用(invoke)
```

获得所有方法

```
        Class<?> objClass = a.getClass();
        Method[] methods =  objClass.getDeclaredMethods();
        for (Method method : methods) {
                System.out.println(method);
        }
```

set/get 方法

```
package cn.netkiller.reflect;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class Member {
        public String name;
        private int age;

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public int getAge() {
                return age;
        }

        public void setAge(int age) {
                this.age = age;
        }

        @Override
        public String toString() {
                return "ClassA [name=" + name + ", age=" +
```

```
age + "]";
        }

        public Member() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws
ClassNotFoundException, InstantiationException,
IllegalAccessException, NoSuchMethodException,
SecurityException, IllegalArgumentException,
InvocationTargetException {
                Class<?> cls =
Class.forName("cn.netkiller.reflect.Member");
                Object member = cls.newInstance();
                Method setMethod =
cls.getDeclaredMethod("setAge", int.class);
                setMethod.invoke(member, 15);

                Method getMethod =
cls.getDeclaredMethod("getAge");
                System.out.println(getMethod.invoke(member));

        }

}
```

下面做一个稍微复杂点的例子，ClassB继承ClassA，取出ClassA的成员变量赋值到ClassA。

```
package cn.netkiller.reflect;

public class ClassA {
        public String name;
        private int age;

        public String getName() {
                return name;
        }
```

```
	public void setName(String name) {
		this.name = name;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public ClassA() {
		// TODO Auto-generated constructor stub
	}

	@Override
	public String toString() {
		return "ClassA [name=" + name + ", age=" + age
+ "]";
	}
}

package cn.netkiller.reflect;

public class ClassB extends ClassA{

	public ClassB() {
		// TODO Auto-generated constructor stub
	}
	private String address;

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	@Override
	public String toString() {
		return "ClassB [address=" + address + "]";
	}
```

```
}


package cn.netkiller.reflect;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class ReflectionTest {

        public ReflectionTest() {
                // TODO Auto-generated constructor stub
        }

        public void testSetMethod() throws
NoSuchMethodException, SecurityException,
IllegalAccessException, IllegalArgumentException,
InvocationTargetException, InstantiationException {

                // ClassA a = new ClassA();

                ClassB b = new ClassB();
                b.setAddress("Shenzhen");

                Class<ClassA> classA = ClassA.class;
                ClassA a = classA.newInstance();
                a.setName("Neo");
                a.setAge(30);


System.out.println(classA.getDeclaredMethod("getAge").invoke(a)
);

                Method m = classA.getDeclaredMethod("setAge",
int.class);
                m.setAccessible(true); // 因为写成private 所以这里
必须设置
                m.invoke(b, 26);

                System.out.println(a.toString());
                System.out.println(b.toString());

                System.out.println(b.getName());
```

```
                System.out.println(b.getAge());
        }

        public static void main(String[] args) throws
InvocationTargetException {

                ReflectionTest rt = new ReflectionTest();
                try {
                        rt.testSetMethod();

                } catch (NoSuchMethodException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (SecurityException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IllegalAccessException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IllegalArgumentException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (InstantiationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

}
```

set 方法

```
System.out.println(classA.getDeclaredMethod("getAge").invoke(
a));
```

```

```

get 方法

```
        Method m = classA.getDeclaredMethod("setAge",
int.class);
        m.setAccessible(true);  //因为写成private 所以这里必须设
置
        m.invoke(b, 26);
```

static 方法调用

```
Class cls = Class.forName("cn.netkiller.reflect.Student");
Method setMethod = cls.getDeclaredMethod("setAge",int.class);
setMethod.invoke(cls.newInstance(), 15);
```

static 方法调用时，不必得实例化对象

```
Class cls = Class.forName("cn.netkiller.reflect.Student");
Method staticMethod =
cls.getDeclaredMethod("setAge",int.class);
staticMethod.invoke(cls,20); //这里不需要newInstance
```

14. Java 线程

14.1. 多线程 Lambda 表达式

```
        new Thread(() -> {
                Thread.sleep(millis);
        });
```

```
new Thread(() -> System.out.println("多线程")).start();
```

14.2. 实现异步执行

```
        public void testThread() throws Exception {
                try {
                        Thread sendmail = new Thread(new
Runnable() {
                                @Override
                                public void run() {
                                        // Sendmail
                                        log.info("Sendmail
OK");
                                }
                        });
                        sendmail.setName("sendmail");
                        sendmail.start();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
```

14.3. 继承 Thread 类实现多线程

```
package cn.netkiller.ipo.test;

public class MyThread extends Thread {

        private String name;

        public MyThread(String name) {
                super();
                this.name = name;
        }

        public void run() {
                for (int i = 0; i < 10; i++) {
                        System.out.println("Thread:" +
this.name + ",i=" + i);
                }
        }

        public static void main(String[] args) {
                MyThread mt1 = new MyThread("A");
                MyThread mt2 = new MyThread("B");
                mt1.start();
                mt2.start();
        }
}
```

设置线程名称

```
    public static void setThreadName1() {
```

```
        new Thread("thread-name-1") {
            public void run() {
                try {
                    Thread.sleep(1000 * 15);
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
                System.out.println("threadName1: " +
this.getName());

            }
        }.start();
    }

    public static void setThreadName2() {
        new Thread() {
            @SneakyThrows
            public void run() {
                this.setName("thread-name-2");
                Thread.sleep(1000 * 15);
                System.out.println("threadName2: " +
this.getName());

            }
        }.start();
    }

    public static void setThreadName3() {
        Thread thread = new Thread() {
            public void run() {
                try {
                    Thread.sleep(1000 * 15);
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
                System.out.println("threadName3: " +
this.getName());

            }
        };

        thread.setName("thread-name-3");
        thread.start();
    }
```

```
    public static void setThreadName4() {
        new Thread("测试线程-1") {
            public void run() {
                try {
                    Thread.sleep(1000 * 15);
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
                System.out.println("threadName1: " +
this.getName());

            }
        }.start();
    }
```

```
public class MyThread extends Thread {
    public MyThread(){

    }
    public MyThread(String name){
        super(name);
    }
    @Override
    public void run(){
        System.out.println(Thread.currentThread().getName());
    }
}

public class DemoThreadName {

    public static void main(String[] args) {

        MyThread mt = new MyThread();
        mt.setName("景峰");
        mt.start();

        new MyThread("netkiller").start();

    }
}
```

14.4. 实现 Runnable 接口

```
package cn.netkiller.ipo.test;

public class MyRunnable implements Runnable {

        private String name;

        public MyRunnable(String name) {
                this.name = name;
        }

        @Override
        public void run() {
                for (int i = 0; i < 10; i++) {
                        System.out.println("Thread:" +
this.name + ",i=" + i);
                }

        }

        public static void main(String[] args) {

                MyRunnable mr1 = new MyRunnable("A");
                MyRunnable mr2 = new MyRunnable("B");

                new Thread(mr1).start();
                new Thread(mr2).start();
                new Thread(new MyRunnable("C")).start();
        }
}
```

14.5. 线程同步

```
package cn.netkiller.thread;

public class SynchronizedThread extends Thread {
        private int count = 0;

        @Override
        public /*synchronized*/ void run() {
                count++;

System.out.println(Thread.currentThread().getName() + "
count:" + count);
        }

        public static void main(String[] args) {
                SynchronizedThread myThread = new
SynchronizedThread();
                Thread thread1 = new Thread(myThread,
"thread1");
                Thread thread2 = new Thread(myThread,
"thread2");
                Thread thread3 = new Thread(myThread,
"thread3");
                Thread thread4 = new Thread(myThread,
"thread4");
                Thread thread5 = new Thread(myThread,
"thread5");
                thread1.start();
                thread2.start();
                thread3.start();
                thread4.start();
                thread5.start();
        }
}
```

线程运行不分先后

```
thread2 count:3
```

```
thread4 count:4
thread1 count:3
thread3 count:3
thread5 count:5
```

```
package cn.netkiller.thread;

public class SynchronizedThread extends Thread {
        private int count = 0;

        @Override
        public synchronized void run() {
                count++;

System.out.println(Thread.currentThread().getName() + "
count:" + count);
        }

        public static void main(String[] args) {
                SynchronizedThread myThread = new
SynchronizedThread();
                Thread thread1 = new Thread(myThread,
"thread1");
                Thread thread2 = new Thread(myThread,
"thread2");
                Thread thread3 = new Thread(myThread,
"thread3");
                Thread thread4 = new Thread(myThread,
"thread4");
                Thread thread5 = new Thread(myThread,
"thread5");
                thread1.start();
                thread2.start();
                thread3.start();
                thread4.start();
                thread5.start();
        }
}
```

```
thread1 count:1
thread5 count:2
thread4 count:3
thread2 count:4
thread3 count:5
```

```
package cn.netkiller.thread;

public class MultiThread {
        private static int count = 0;

        public synchronized void add() {
                count++;

System.out.println(Thread.currentThread().getName() + "
count:" + count);
        }

        public static void main(String[] args) throws
InterruptedException {

                final MultiThread multiThread1 = new
MultiThread();
                final MultiThread multiThread2 = new
MultiThread();
                final MultiThread multiThread3 = new
MultiThread();
                final MultiThread multiThread4 = new
MultiThread();
                final MultiThread multiThread5 = new
MultiThread();

                new Thread(new Runnable() {
                        public void run() {
                                multiThread1.add();
                        }
                }).start();
```

```
                new Thread(new Runnable() {
                        public void run() {
                                multiThread2.add();
                        }
                }).start();

                new Thread(new Runnable() {
                        public void run() {
                                multiThread3.add();
                        }
                }).start();

                new Thread(new Runnable() {
                        public void run() {
                                multiThread4.add();
                        }
                }).start();

                new Thread(new Runnable() {
                        public void run() {
                                multiThread5.add();
                        }
                }).start();
        }
}
```

14.6. java 线程池

newCachedThreadPool

```
        private void startTask(List<String> usersList){
        ExecutorService executor =
Executors.newCachedThreadPool();
        executor.submit(()->{
                        //do someting
        });
    }
```

固定线程池(newFixedThreadPool)

```
package cn.netkiller.test.grey;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class GreyTest {

        public GreyTest() {
                // TODO Auto-generated constructor stub
        }

        static class MyThread implements Runnable {
                public void run() {
                        System.out.println("Thread Name:" +
Thread.currentThread().getName());
                }
        }

        public static void main(String[] args) {
                // 创建五个线程池
                int nThreads = 5;
                ExecutorService pool =
Executors.newFixedThreadPool(nThreads);
                // 创建实现了Runnable接口对象
                MyThread t1 = new MyThread();
                MyThread t2 = new MyThread();
                MyThread t3 = new MyThread();
                MyThread t4 = new MyThread();
                MyThread t5 = new MyThread();
                // 将线程放入池中进行执行
                pool.execute(t1);
                pool.execute(t2);
                pool.execute(t3);
                pool.execute(t4);
                pool.execute(t5);
                // 关闭线程池
```

```
                pool.shutdown();
        }

}
```

Executors.newScheduledThreadPool

```
@Configuration
public class ScheduleConfig implements SchedulingConfigurer {

    @Override
    public void configureTasks(ScheduledTaskRegistrar
taskRegistrar) {
        //当然了，这里设置的线程池是corePoolSize也是很关键了，自己根
据业务需求设定

taskRegistrar.setScheduler(Executors.newScheduledThreadPool(1
0));
    }

}
```

14.7. ThreadLocal

```
    public static void threadLocal() {
        ThreadLocal<String> local = new ThreadLocal<>();

        IntStream.range(0, 10).forEach(i -> new Thread(() ->
{
            local.set(Thread.currentThread().getName() + ":"
+ i);
            System.out.println("线程: " +
```

```
Thread.currentThread().getName() + ",local:" + local.get());
        }).start());
    }
```

```
package cn.netkiller.thread;

public class ThreadLocaTest {

    private static ThreadLocal<String> local = new
ThreadLocal<String>();
    public static void main(String[] args) throws
InterruptedException {

        new Thread(new Runnable() {
            public void run() {

System.out.println(Thread.currentThread().getName() + ": " +
local.get());
                ThreadLocaTest.local.set("thread_A");

System.out.println(Thread.currentThread().getName() + ": " +
local.get());
            }
        }, "A").start();

        Thread.sleep(1000);

        new Thread(new Runnable() {
            public void run() {

System.out.println(Thread.currentThread().getName() + ": " +
local.get());
                ThreadLocaTest.local.set("thread_B");

System.out.println(Thread.currentThread().getName() + ": " +
local.get());
                local.remove();
                System.out.println("remove: " + local.get());
            }
        }, "B").start();
```

```
    }
}
```

14.8. InheritableThreadLocal

```
    public static void inheritableThreadLocal() {
        InheritableThreadLocal threadLocal = new
InheritableThreadLocal();
        IntStream.range(0, 10).forEach(i -> {
            //每个线程的序列号，希望在子线程中能够拿到
            threadLocal.set(i);
            //这里来了一个子线程，我们希望可以访问上面的threadLocal
            new Thread(() -> {

System.out.println(Thread.currentThread().getName() + ":" +
threadLocal.get());
            }).start();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });
    }
```

14.9. Java 协程

```
package cn.netkiller.thread;

import static java.lang.Thread.*;

public class VirtualThreadsTest {
    public static void main(String[] args) throws
```

```
InterruptedException {
        var virtualThread = startVirtualThread(() ->
System.out.println("Hello from the virtual thread"));
        virtualThread.join();
    }
}
```

运行演示

```
neo@MacBook-Pro-M2 thread % java --source 20 --enable-preview
VirtualThreadsTest.java
注: VirtualThreadsTest.java 使用 Java SE 20 的预览功能。
注: 有关详细信息, 请使用 -Xlint:preview 重新编译。
Hello from the virtual thread
```

```
import java.time.Duration;
import java.util.concurrent.Executors;
import java.util.stream.IntStream;

public class Main {
    public static void main(String[] args) {

        try (var executor =
Executors.newVirtualThreadPerTaskExecutor()) {
            IntStream.range(0, 10000).forEach(i -> {
                executor.submit(() -> {
                    Thread.sleep(Duration.ofSeconds(1));
                    return i;
                });
            });
        }
    }
}
```

15. java 脚本引擎

什么是脚本引擎，脚本引擎是指在程序运行期间嵌入另一种脚本语言，并与其交互，产生最终运行结果

脚本引擎存在的意义是什么？脚本引擎可以改变编译语言的内部运行逻辑，弥补编译语言的不足，使编译语言具备动态语言的一部分特性。

是否有成功案例？最成功的案例就是基于C++和Lua语言开发的端游（网游一种，需要按照客户端），编译语言最大的缺点就是客户端升级需要重新安装并且安装之后重启应用程序才能生效。脚本引擎弥补了这项致命的缺点，用户只需升级剧情脚本，而不需要退出整个游戏然后重新进入。

15.1. Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>cn.netkiller</groupId>
        <artifactId>script</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Java Script</name>
        <description>Java Script Engine</description>
        <dependencies>
                <!--
https://mvnrepository.com/artifact/org.mockito/mockito-all --
>
                <dependency>
                        <groupId>org.mockito</groupId>
                        <artifactId>mockito-all</artifactId>
                        <version>1.10.19</version>
                </dependency>
```

```
        </dependencies>
        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.5.1</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

15.2. Helloworld

```
package cn.netkiller.javascript;

import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.script.ScriptEngine;
import javax.script.ScriptEngineFactory;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class Helloworld {

        public Helloworld() {
                ScriptEngineManager manager = new
ScriptEngineManager();
                List<ScriptEngineFactory> factories =
manager.getEngineFactories();
                for (ScriptEngineFactory f : factories) {
                        System.out.println("egine name:" +
```

```
f.getEngineName());
                        System.out.println("engine version:"
+ f.getEngineVersion());
                        System.out.println("language name:" +
f.getLanguageName());
                        System.out.println("language
version:" + f.getLanguageVersion());
                        System.out.println("names:" +
f.getNames());
                        System.out.println("mime:" +
f.getMimeTypes());
                        System.out.println("extension:" +
f.getExtensions());
                        System.out.println("-----------------
------------------------------");
                }
        }

        public void hello() throws ScriptException {
                ScriptEngineManager manager = new
ScriptEngineManager();
                ScriptEngine engine =
manager.getEngineByName("JavaScript");
                // ScriptEngine engine =
manager.getEngineByExtension("js");
                // ScriptEngine engine =
manager.getEngineByMimeType("text/javascript");
                engine.eval("print('Hello, World')");
        }

        public static void main(String[] args) {
                try {
                        new Helloworld().hello();
                } catch (ScriptException ex) {

Logger.getLogger(Helloworld.class.getName()).log(Level.SEVERE
, null, ex);
                }
        }
}
```

15.3. 运行脚本文件

将脚本放入外部文件

```
package javascript;

import java.io.FileNotFoundException;
import java.net.URL;

import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class EvalFile {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ScriptEngineManager manager = new
ScriptEngineManager();
                ScriptEngine engine =
manager.getEngineByExtension("js");
                // ScriptEngine engine =
manager.getEngineByMimeType("text/javascript");

                try {

                        URL location =
EvalFile.class.getProtectionDomain().getCodeSource().getLocat
ion();
                        String file = location.getFile() +
"test.js";
                        System.out.println(file);

                        engine.eval(new
java.io.FileReader(file));

                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (ScriptException e) {
                        // TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                }
        }

}
```

test.js

```
print("This is hello from test.js");
```

15.4. 变量传递

```
package javascript;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.script.Bindings;
import javax.script.ScriptContext;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;
import javax.script.SimpleBindings;

public class ScriptVars {

        ScriptEngine engine = null;

        public ScriptVars() {
                ScriptEngineManager manager = new
ScriptEngineManager();
                engine =
manager.getEngineByMimeType("text/javascript");
```

```
        }

        public void variable() throws ScriptException {
                engine.put("name", "Neo");
                engine.eval("var message = 'Hello, ' +
name;");
                engine.eval("print(message);");
                Object obj = engine.get("message");
                System.out.println(obj);
        }

        public void simpleBindings() throws ScriptException {
                Bindings bindings = new SimpleBindings();
                bindings.put("name", "Neo");
                engine.eval("print('I am ' + name);",
bindings);
        }

        public void function() throws ScriptException {
                engine.put("a", 4);
                engine.put("b", 6);
                Object maxNum = engine.eval("function
max_num(a,b){return (a>b)?a:b;}max_num(a,b);");
                System.out.println("max_num:" + maxNum + ",
(class = " + maxNum.getClass() + ")");

                Map<String, Integer> m = new HashMap<String,
Integer>();
                m.put("c", 10);
                engine.put("m", m);
                engine.eval("var x= max_num(a,m.get('c'));");
                System.out.println("max_num:" +
engine.get("x"));
        }

        public void object() throws ScriptException {
                File f = new File("test.txt");
                // expose File object as variable to script
                engine.put("file", f);

                // evaluate a script string. The script
accesses "file"
                // variable and calls method on it
                engine.eval("print(file.getAbsolutePath())");
        }
```

```
        public void outputToFile() throws IOException,
ScriptException {
                ScriptContext context = engine.getContext();
                context.setWriter(new
FileWriter("script.log"));
                engine.eval("print('Hello World!');");
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                try {
                        ScriptVars sv = new ScriptVars();
                        sv.variable();
                        sv.simpleBindings();
                        sv.outputToFile();
                        sv.function();
                        sv.object();
                        sv.outputToFile();

                } catch (ScriptException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

}
```

15.5. 全局变量与局部变量定义

ScriptContext.GLOBAL_SCOPE 作用于 ScriptEngineManager，ScriptContext.ENGINE_SCOPE 作用于 ScriptEngine

```
```

```
package javascript;

import javax.script.ScriptContext;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class ContextVariable {

        public static void main(String[] args) throws
ScriptException {
                // TODO Auto-generated method stub
                ScriptEngineManager manager = new
ScriptEngineManager();
                ScriptEngine engine =
manager.getEngineByName("JavaScript");
                ScriptContext context = engine.getContext();
            context.setAttribute("name", "Netkiller",
ScriptContext.GLOBAL_SCOPE);
            context.setAttribute("name", "Neo",
ScriptContext.ENGINE_SCOPE);
            //context.getAttribute("name");
            engine.eval("print('I am ' + name);", context);

            // engine1,context1 并没有定义 name 但输出结果却显示
Netkiller, 所以ScriptContext.GLOBAL_SCOPE定义的变量是全局的。
            ScriptEngine engine1 =
manager.getEngineByName("JavaScript");
            ScriptContext context1 = engine1.getContext();
            engine.eval("print('I am ' + name);", context1);

        }

}
```

```
import javax.script.*;

public class MultiScopes {
    public static void main(String[] args) throws Exception {
        ScriptEngineManager manager = new
```

```
ScriptEngineManager();
        ScriptEngine engine =
manager.getEngineByName("JavaScript");

        engine.put("x", "hello");
        // print global variable "x"
        engine.eval("print(x);");
        // the above line prints "hello"

        // Now, pass a different script context
        ScriptContext newContext = new SimpleScriptContext();
        Bindings engineScope =
newContext.getBindings(ScriptContext.ENGINE_SCOPE);

        // add new variable "x" to the new engineScope
        engineScope.put("x", "world");

        // execute the same script - but this time pass a
different script context
        engine.eval("print(x);", newContext);
        // the above line prints "world"
    }
}
```

15.6. 调用脚本中的函数或方法

```
package javascript;

import javax.script.Invocable;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import javax.script.ScriptException;

public class InvokeScriptFunction {

        public static void main(String[] args) throws
ScriptException, NoSuchMethodException {
                // TODO Auto-generated method stub
                ScriptEngineManager manager = new
```

```
ScriptEngineManager();
                ScriptEngine engine =
manager.getEngineByName("JavaScript");

                // JavaScript code in a String
                String script = "function hello(name) {
print('Hello, ' + name); }";
                // evaluate script
                engine.eval(script);

                // javax.script.Invocable is an optional
interface.
                // Check whether your script engine
implements or not!
                // Note that the JavaScript engine implements
Invocable interface.
                Invocable inv = (Invocable) engine;

                // invoke the global function named "hello"
                inv.invokeFunction("hello", "Scripting!!");

                // JavaScript code in a String. This code
defines a script object 'obj'
                // with one method called 'hello'.
                script = "var obj = new Object(); obj.hello =
function(name) { print('Hello, ' + name); }";
                // evaluate script
                engine.eval(script);
                // get script object on which we want to call
the method
                Object obj = engine.get("obj");

                // invoke the method named "hello" on the
script object "obj"
                inv.invokeMethod(obj, "hello", "Script Method
!!");
        }
}
```

15.7. 脚本编译

只有重复执行脚本时才需要编译。只运行一次不建议编译运行。

```
package javascript;

import javax.script.*;

public class ScriptCompile {
        public CompiledScript compile(String script) throws
ScriptException {

                ScriptEngineManager manager = new
ScriptEngineManager();
                ScriptEngine engine =
manager.getEngineByName("JavaScript");

                return ((Compilable) engine).compile(script);

        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ScriptCompile sc = new ScriptCompile();
                try {

                        CompiledScript script =
sc.compile("print('Helloworld!!!');");

                        for (int i = 0; i < 100; i++) {
                                script.eval();
                        }

                } catch (ScriptException ex) {
                        ex.printStackTrace();
                }
        }

}
```

15.8. jjs - Invokes the Nashorn engine.

jjs 是一个在命令行运行脚本的命令

创建脚本文件

```
[neo@netkiller tmp]$ cat test.js
function f() {
     return 1;
};

print( f() + 1 );
```

运行结果

```
[neo@netkiller tmp]$ jjs test.js
2
```

16. Crypto

16.1. MD5

JDK 1.8

```
                        String hash =
DatatypeConverter.printHexBinary(MessageDigest.getInstance("M
D5").digest("SOMESTRING".getBytes("UTF-8")));
```

16.2. AES

```
package cn.netkiller.crypto;

import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.MessageDigest;
import java.security.SecureRandom;

public class TestAES {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                String key =
"fm6I1D2HTFVVOWUKny76TThagNq5Czrv";
                String clean = "Helloworld!!!";

                try {
                        byte[] encrypted = encrypt(clean,
key);
                        String decrypted = decrypt(encrypted,
key);
                        System.out.println(decrypted);
                } catch (Exception e) {
```

```
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

        public static byte[] encrypt(String plainText, String
key) throws Exception {
                byte[] clean = plainText.getBytes();

                // Generating IV.
                int ivSize = 16;
                byte[] iv = new byte[ivSize];
                SecureRandom random = new SecureRandom();
                random.nextBytes(iv);
                IvParameterSpec ivParameterSpec = new
IvParameterSpec(iv);

                // Hashing key.
                MessageDigest digest =
MessageDigest.getInstance("SHA-256");
                digest.update(key.getBytes("UTF-8"));
                byte[] keyBytes = new byte[16];
                System.arraycopy(digest.digest(), 0,
keyBytes, 0, keyBytes.length);
                SecretKeySpec secretKeySpec = new
SecretKeySpec(keyBytes, "AES");

                // Encrypt.
                Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5Padding");
                cipher.init(Cipher.ENCRYPT_MODE,
secretKeySpec, ivParameterSpec);
                byte[] encrypted = cipher.doFinal(clean);

                // Combine IV and encrypted part.
                byte[] encryptedIVAndText = new byte[ivSize +
encrypted.length];
                System.arraycopy(iv, 0, encryptedIVAndText,
0, ivSize);
                System.arraycopy(encrypted, 0,
encryptedIVAndText, ivSize, encrypted.length);

                return encryptedIVAndText;
        }
```

```
        public static String decrypt(byte[]
encryptedIvTextBytes, String key) throws Exception {
                int ivSize = 16;
                int keySize = 16;

                // Extract IV.
                byte[] iv = new byte[ivSize];
                System.arraycopy(encryptedIvTextBytes, 0, iv,
0, iv.length);
                IvParameterSpec ivParameterSpec = new
IvParameterSpec(iv);

                // Extract encrypted part.
                int encryptedSize =
encryptedIvTextBytes.length - ivSize;
                byte[] encryptedBytes = new
byte[encryptedSize];
                System.arraycopy(encryptedIvTextBytes,
ivSize, encryptedBytes, 0, encryptedSize);

                // Hash key.
                byte[] keyBytes = new byte[keySize];
                MessageDigest md =
MessageDigest.getInstance("SHA-256");
                md.update(key.getBytes());
                System.arraycopy(md.digest(), 0, keyBytes, 0,
keyBytes.length);
                SecretKeySpec secretKeySpec = new
SecretKeySpec(keyBytes, "AES");

                // Decrypt.
                Cipher cipherDecrypt =
Cipher.getInstance("AES/CBC/PKCS5Padding");
                cipherDecrypt.init(Cipher.DECRYPT_MODE,
secretKeySpec, ivParameterSpec);
                byte[] decrypted =
cipherDecrypt.doFinal(encryptedBytes);

                return new String(decrypted);
        }

}
```

16.3. AES/CBC/PKCS5PADDING

```
package cn.netkiller.security;

import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class AES {
        private static final String initVector =
"encryptionIntVec";
        private String key;

        public AES(String key) {
                // TODO Auto-generated constructor stub
                this.key = key;
        }

        public String encrypt(String value) {
                return this.encrypt(value, this.key);
        }

        public String encrypt(String value, String key) {
                try {
                        IvParameterSpec ivParameterSpec = new
IvParameterSpec(initVector.getBytes("UTF-8"));
                        SecretKeySpec secretKeySpec = new
SecretKeySpec(key.getBytes("UTF-8"), "AES");

                        Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5PADDING");
                        cipher.init(Cipher.ENCRYPT_MODE,
secretKeySpec, ivParameterSpec);

                        byte[] encrypted =
cipher.doFinal(value.getBytes());
```

```
                        return
Base64.getEncoder().encodeToString(encrypted);
                } catch (Exception ex) {
                        ex.printStackTrace();
                }
                return null;
        }

        public String decrypt(String encrypted) {
                return this.decrypt(encrypted, this.key);
        }

        public String decrypt(String encrypted, String key) {
                try {
                        IvParameterSpec ivParameterSpec = new
IvParameterSpec(initVector.getBytes("UTF-8"));
                        SecretKeySpec secretKeySpec = new
SecretKeySpec(key.getBytes("UTF-8"), "AES");

                        Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5PADDING");
                        cipher.init(Cipher.DECRYPT_MODE,
secretKeySpec, ivParameterSpec);
                        byte[] original =
cipher.doFinal(Base64.getDecoder().decode(encrypted));

                        return new String(original);
                } catch (Exception ex) {
                        ex.printStackTrace();
                }

                return null;
        }

        public static void main(String[] args) {
                // key 长度16个字节
                String key = "www.netkiller.cn";
                System.out.println(key.length());
                AES aes = new AES(key);
                String en = aes.encrypt("Helloworld!!!");
                String de = aes.decrypt(en);
                System.out.println(en);
                System.out.println(de);

        }
```

```
}
```

16.4. DES

```
package cn.netkiller.security;

import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;

public class DES {

        public DES() {
                // TODO Auto-generated constructor stub
        }

        public static String encrypt(String text, String
password) {
                try {
                        SecureRandom random = new
SecureRandom();
                        DESKeySpec desKey = new
DESKeySpec(password.getBytes());
                        // 创建一个密匙工厂，然后用它把DESKeySpec
转换成
                        SecretKeyFactory keyFactory =
SecretKeyFactory.getInstance("DES");
                        SecretKey securekey =
keyFactory.generateSecret(desKey);
                        // Cipher对象实际完成加密操作
                        Cipher cipher =
Cipher.getInstance("DES");
                        // 用密匙初始化Cipher对象
```

```
                        cipher.init(Cipher.ENCRYPT_MODE,
securekey, random);
                        // 现在，获取数据并加密
                        // 正式执行加密操作

                        return
Base64.getEncoder().encodeToString(cipher.doFinal(text.getByt
es(StandardCharsets.UTF_8)));
                } catch (Throwable e) {
                        e.printStackTrace();
                }
                return null;
        }

        private static String decrypt(String text, String
password) throws Exception {
                try {
                        // DES算法要求有一个可信任的随机数源
                        SecureRandom random = new
SecureRandom();
                        // 创建一个DESKeySpec对象
                        DESKeySpec desKey = new
DESKeySpec(password.getBytes());
                        // 创建一个密匙工厂
                        SecretKeyFactory keyFactory =
SecretKeyFactory.getInstance("DES");
                        // 将DESKeySpec对象转换成SecretKey对象
                        SecretKey securekey =
keyFactory.generateSecret(desKey);
                        // Cipher对象实际完成解密操作
                        Cipher cipher =
Cipher.getInstance("DES");
                        // 用密匙初始化Cipher对象
                        cipher.init(Cipher.DECRYPT_MODE,
securekey, random);
                        // 真正开始解密操作
                        return new
String(cipher.doFinal(Base64.getDecoder().decode(text)),
StandardCharsets.UTF_8);
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return null;
        }
```

```
        public static void main(String[] args) throws
Exception {
                // TODO Auto-generated method stub
                String en = DES.encrypt("Helloworld!!!",
"www.netkiller.cn");
                String de = DES.decrypt(en,
"www.netkiller.cn");
                System.out.println(en);
                System.out.println(de);

        }

}
```

17. java.security

17.1. 列出 Java 支持的数字摘要算法

```
package cn.netkiller.security;

import java.security.Security;
import java.util.Set;

public class ListMessageDigest {

        public static void main(String[] args) {

                // List of available MessageDigest Algorithms
                Set<String> messageDigest =
Security.getAlgorithms("MessageDigest");
                messageDigest.forEach(x ->
System.out.println(x));

        }

}
```

运行结果

```
SHA3-512
SHA-384
SHA
SHA3-384
SHA-224
SHA-512/256
SHA-256
MD2
SHA-512/224
```

```
SHA3-256
SHA-512
MD5
SHA3-224
```

17.2. 计算文件的 MD5，SHA 等 HASH 值

```
package cn.netkiller.security;

import java.io.FileInputStream;
import java.io.IOException;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class sha1sum {

        public sha1sum() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) throws
NoSuchAlgorithmException, IOException {

                String hex = checksum("/etc/hosts");
                System.out.println(hex);
        }

        private static String checksum(String filepath)
throws IOException, NoSuchAlgorithmException {

                MessageDigest md =
MessageDigest.getInstance("SHA"); // SHA, MD2, MD5, SHA-256,
SHA-384...
                // file hashing with DigestInputStream
                try (DigestInputStream dis = new
DigestInputStream(new FileInputStream(filepath), md)) {
                        while (dis.read() != -1)
                                ; // empty loop to clear the
```

```
data
                        md = dis.getMessageDigest();
                }

                // bytes to hex
                StringBuilder result = new StringBuilder();
                for (byte b : md.digest()) {
                        result.append(String.format("%02x",
b));
                }
                return result.toString();

        }

}
```

第 2 章 Build Tools

1. Apache Ant

http://ant.apache.org/

1.1. 安装 ant

1.8

```
cd /usr/local/src
wget http://mirror.bjtu.edu.cn/apache//ant/binaries/apache-ant-
1.8.1-bin.tar.gz
tar zxvf apache-ant-1.8.1-bin.tar.gz
mv apache-ant-1.8.1 /usr/local/
cd ..
ln -s apache-ant-1.8.1 apache-ant
```

```
ANT_HOME=/usr/local/apache-ant
export CLASSPATH=$CLASSPATH:$ANT_HOME/lib
```

1.10.1

Netkiller OSCM 一键安装

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/lang/java/a
nt/apache-ant-1.10.1.sh | bash
```

1.2. ANT

ant.project.name

ant.project.name 一般式定义在

```
<project name="www.netkiller.cn" default="compile" basedir="."
xmlns:if="ant:if">
<echo>${ant.project.name}</echo>
```

我们希望从命令行传递这个值

```
<project default="compile" basedir="." xmlns:if="ant:if">
<echo>${ant.project.name}</echo>
```

你需要将 project 中的定义去掉才能从命令行传递

```
ant -Dant.project.name=www.netkiller.cn -f build.xml
```

你也可以从 build.properties 文件定义 ant.project.name

```
MacBook-Pro:deployment neo$ cat build.properties
ant.project.name=www.netkiller.cn
```

```
ant -f build.xml -propertyfile build.properties
```

```

```

定义

```

```

1.3. Project

```
<description>project Name</description>
```

property

在 build.xml 中定义 property

```
<property name="src" value="src"/>
<property name="dest" value="classes"/>
<property name="hello" value="hello.jar"/>
```

引用 properties 文件

```
<property file="build.properties" />
<propety resource="build.properties"/>
```

设置系统的环境变量为前缀

```
<property environment="env"/>
<property name="tomcat.home" value="${env.CATALINA_HOME}"/>
```

命令行传值，使用-D参数是会覆盖build.xml中的先前定义的变量

```
$ ant --help | grep property
  -D<property>=<value>   use value for given property
  -propertyfile <name>   load all properties from file with -D
```

ant

Project name

```
${ant.project.name}
```

environment

```
<property environment="env"/>
<echo message="JAVA_HOME is set to = ${env.JAVA_HOME}" />
```

1.4. path

定义

```
        <path id="classpath">
        <fileset dir="${env.JAVA_HOME}/lib">
            <include name="*.jar" />
        </fileset>
        <fileset dir="${env.CATALINA_HOME}/lib">
            <include name="*.jar" />
        </fileset>
        <fileset dir="WebRoot/WEB-INF/lib" includes="*.jar"/>
    </path>
```

引用

```
        <javac srcdir="${src.dir}" destdir="${classes.dir}"
source="1.5" target="1.5">
                <classpath refid="classpath" />
        </javac>
```

```
    <classpath>
                <path refid="classpath"/>
    </classpath>
```

1.5. copy

复制目录

```
        <copy todir="${basedir}/WebContent">
```

```
                <fileset dir="${basedir}/WebRoot"
includes="**/*"/>
        </copy>
```

复制指定扩展名文件

```
        <copy todir="${dest}">
                <fileset dir="${src}">
                        <include name="**/*.xml" />
                        <include name="**/*.properties" />
                </fileset>
        </copy>
```

1.6. javac

```
        <path id="classpath">
                <fileset dir="${env.JAVA_HOME}/lib">
                        <include name="*.jar" />
                </fileset>
                <fileset dir="${env.CATALINA_HOME}/lib">
                        <include name="*.jar" />
                </fileset>
                <fileset dir="${project.dir}/WebRoot/WEB-
INF/lib" includes="*.jar" />
        </path>

        <javac srcdir="${project.src}"
destdir="${project.build}/WEB-INF/classes" debug="true"
listfiles="true">
                        <classpath refid="classpath" />
                        <include name="**/*.java"/>
                        <exclude name="**/*.xml"/>
                        <exclude name="**/*.properties"/>
        </javac>
```

```
```

listfiles 显示编译文件列表

debug 显示调试信息，编译错误信息

1.7. condition

```
<?xml version="1.0"?>
<project name="test" default="doFoo" basedir=".">
  <property name="directory" value="/www/directory"/>

  <target name="doFoo" depends="dir.check" if="dir.exists">
    <echo>${directory} exists</echo>
  </target>

  <target name="doBar" depends="dir.check" unless="dir.exists">
    <echo>${directory} missing"</echo>
  </target>

  <target name="dir.check">
    <condition property="dir.exists">
      <available file="${directory}" type="dir"/>
    </condition>
  </target>
</project>
```

1.8. exec

```
<project name="{{ name }}" default="help" basedir=".">

  <property name="username" value="{{ username }}"/>
  <property name="host" value="{{ host }}"/>
  <property name="dir" value="/srv/{{ path }}/"/>
```

```
  <tstamp>
    <format property="TODAY_UK" pattern="yyyyMMddhhmmss"
locale="en,UK"/>
  </tstamp>

  <target name="help" description="show available commands" >
    <exec executable="ant" dir="." failonerror="true">
      <arg value="-p"/>
    </exec>
  </target>

  <target name="deploy-to" description="show where we are
deploying to" >
    <echo>${username}@${host}:${dir}</echo>
  </target>

  <target name="deploy" description="deploy usng rsync" >
    <exec executable="rsync" dir="." failonerror="true">
      <arg value="-r"/>
      <arg value="."/>
      <arg value="${username}@${host}:${dir}"/>
      <arg value="--exclude-from=rsync.excludes"/>
      <arg value="-v"/>
    </exec>
  </target>

  <target name="deploy-test" description="test deploy usng
rsync with the dry run flag set" >
    <exec executable="rsync" dir="." failonerror="true">
      <arg value="-r"/>
      <arg value="."/>
      <arg value="${username}@${host}:${dir}"/>
      <arg value="--exclude-from=rsync.excludes"/>
      <arg value="--dry-run"/>
      <arg value="-v"/>
    </exec>
  </target>

  <target name="backup" description="backup site" >
    <exec executable="scp" dir="." failonerror="true">
      <arg value="-r"/>
      <arg value="${username}@${host}:${dir}"/>
      <arg value="backups/${TODAY_UK}"/>
    </exec>
  </target>
```

```
</project>
```

sshexec

```
<sshexec host="${remove}" keyfile="~/.ssh/id_rsa"
command="/srv/apache-tomcat/bin/catalina.sh stop -force" />
```

1.9. if

```
<if>
  <available file="my_directory" type="dir" />
  <then>
    <echo message="Directory exists" />
  </then>
  <else>
    <echo message="Directory does not exist" />
  </else>
</if>
```

Ant 1.9.x 新增 xmlns:if="ant:if"

```
<project name="tryit"
 xmlns:if="ant:if"
 xmlns:unless="ant:unless"
>
 <exec executable="java">
   <arg line="-X" if:true="${showextendedparams}"/>
```

```
   <arg line="-version" unless:true="${showextendedparams}"/>
 </exec>
 <condition property="onmac">
   <os family="mac"/>
 </condition>
 <echo if:set="onmac">running on MacOS</echo>
 <echo unless:set="onmac">not running on MacOS</echo>
</project>


<!DOCTYPE project>
<project xmlns:if="ant:if" xmlns:unless="ant:unless">
  <property unless:set="property" name="property.is.set"
value="false"/>
  <property if:set="property" name="property.is.set"
value="true"/>
  <echo>${property.is.set}</echo>
</project>
```

1.10. macrodef

arg value 与 arg line

arg line 可以处理参数的空格， 而arg value则不能. arg line 可以处理空参数， arg value传递空参数会报错.

```
<exec executable = "sh" dir = "@{dir}">
        <arg line = "ls -l /var/log" />
</exec>


<exec executable = "ls" dir = "@{dir}">
        <arg value = "-l" />
        <arg value = "/var/log" />
</exec>
```

```
        <macrodef name="mvn">
                <attribute name="options" default="" />
                <attribute name="goal" default="" />
                <attribute name="phase" default=" " />
                <attribute name="dir" default="" />
                <element name="args" optional="false" />
                <sequential>
                        <exec executable="mvn" dir="@{dir}" >
                                <arg value="@{options}" />
                                <arg value="@{goal}" />
                                <arg value="@{phase}" />
                        </exec>
                </sequential>
        </macrodef>

<!-- 执行下面宏将会出错，你必须传递options，phase参数 -->
<mvn goal="package" dir="${project.dir}"/>
<!-- 将vale改为line后正常 -->
                <exec executable="mvn" dir="@{dir}" >
                                <arg line="@{options}" />
                                <arg line="@{goal}" />
                                <arg line="@{phase}" />
        </exec>
```

运行方式sequential为顺序执行， parallel为并行执行。

Git

```
<macrodef name = "git">
    <attribute name = "command" />
    <attribute name = "dir" default = "" />
    <element name = "args" optional = "true" />
    <sequential>
        <echo message = "git @{command}" />
        <exec executable = "git" dir = "@{dir}">
            <arg value = "@{command}" />
```

```
            <args/>
        </exec>
    </sequential>
</macrodef>
<macrodef name = "git-clone-pull">
    <attribute name = "repository" />
    <attribute name = "dest" />
    <sequential>
        <git command = "clone">
            <args>
                <arg value = "@{repository}" />
                <arg value = "@{dest}" />
            </args>
        </git>
        <git command = "pull" dir = "@{dest}" />
    </sequential>
</macrodef>
```

```
<git command = "clone">
    <args>
        <arg value = "git://github.com/280north/ojunit.git" />
        <arg value = "ojunit" />
    </args>
</git>

<git command = "pull" dir = "repository_path" />

<git-clone-pull
repository="git://github.com/280north/ojunit.git" dest="ojunit"
/>
```

Rsync

```
        <macrodef name="rsync">
```

```
                <attribute name="option" default="auzv" />
                <attribute name="src" default="" />
                <attribute name="dest" default="" />
                <element name="args" optional="true" />
                <sequential>
                        <exec executable="rsync">
                                <arg value="@{option}" />
                                <arg value="@{src}" />
                                <arg value="@{dest}" />
                                <args />
                        </exec>
                </sequential>
        </macrodef>
```

```
        <target name="deploy" depends="compile">
                <rsync option="-auzv" src="${project.dest}"
dest="${remote}:${destination}">
                        <args>
                                <arg value="-P" />
                        </args>
                </rsync>
        </target>
```

SSH

```
        <macrodef name="ssh">
                <attribute name="host" />
                <attribute name="command" />
                <attribute name="keyfile"
default="~/.ssh/id_rsa" />
                <element name="args" optional="true" />
                <sequential>
                        <exec executable="ssh">
                                <arg value="@{host}" />
```

```
                                <!-- arg value="-i @{keyfile}"
/ -->
                                <args />
                                <arg value="@{command}" />
                        </exec>
                </sequential>
        </macrodef>
```

```
        <target name="stop" depends="">
                <!-- ssh host="${remote}" command="/srv/apache-
tomcat/bin/catalina.sh stop -force" keyfile="~/.ssh/id_rsa" / -
->
                <ssh host="${remote}" command="/srv/apache-
tomcat/bin/shutdown.sh" />
        </target>
        <target name="start" depends="">
                <ssh host="${remote}" command="/srv/apache-
tomcat/bin/startup.sh" keyfile="~/.ssh/id_rsa" />
        </target>
```

maven

```
        <macrodef name="mvn">
                <attribute name="options" default="" />
                <attribute name="goal" default="" />
                <attribute name="phase" default=" " />
                <attribute name="dir" default="" />
                <element name="args" optional="false" />
                <sequential>
                        <exec executable="mvn" dir="@{dir}" >
                                <arg line="@{options}" />
                                <arg value="@{goal}" />
                                <arg line="@{phase}" />
                        </exec>
                </sequential>
```

```
        </macrodef>
```

1.11. Javascript

```
$ cat build.xml
<project name="Attachments" default="print">
    <property name="numAttachments" value="20" />
    <target name="generate">
        <script language="javascript"><![CDATA[
            var list = '1';
            var limit = project.getProperty( "numAttachments"
);
            for (var i=2;i<limit;i++)
            {
                list = list + ',' + i;
            }
            project.setNewProperty("list", list);
                print(list);
        ]] >
        </script>
    </target>
</project>
```

```
$ ant generate
Buildfile: /www/testing/build.xml

generate:
   [script] 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19

BUILD SUCCESSFUL
Total time: 0 seconds
```

1.12. mail

https://ant.apache.org/manual/Tasks/mail.html

```
cp ~/.m2/repository/com/sun/mail/javax.mail/1.5.6/javax.mail-
1.5.6.jar /srv/apache-ant-1.9.6/lib
cp
/www/.m2/repository/com/sun/mail/javax.mail/1.5.6/javax.mail-
1.5.6.jar /srv/apache-ant-1.10.1/lib/
```

Examples

```
<mail from="me"
      tolist="you"
      subject="Results of nightly build"
      files="build.log"/>
Sends an email from me to you with a subject of Results of
nightly build and includes the contents of the file build.log
in the body of the message.

<mail mailhost="smtp.myisp.com" mailport="1025" subject="Test
build">
  <from address="config@myisp.com"/>
  <replyto address="me@myisp.com"/>
  <to address="all@xyz.com"/>
  <message>The ${buildname} nightly build has
completed</message>
  <attachments>
    <fileset dir="dist">
      <include name="**/*.zip"/>
    </fileset>
  </attachments>
</mail>
```

1.13. basename

```
<basename property="jar.filename" file="${lib.jarfile}"/>
will set jar.filename to myjar.jar, if lib.jarfile is defined
as either a full-path filename (eg., /usr/local/lib/myjar.jar),
a relative-path filename (eg., lib/myjar.jar), or a simple
filename (eg., myjar.jar).
<basename property="cmdname" file="D:/usr/local/foo.exe"
          suffix=".exe"/>
will set cmdname to foo.
<property environment="env"/>
<basename property="temp.dirname" file="${env.TEMP}"/>
```

1.14. 创建文件

```
<touch file="newfile.txt"/>
```

```
<echo file="myoutput.txt" append="true"/>
   The content to be written to myoutput.txt, the classpath is
${classpath}
<echo/>
```

1.15. FAQ

warning: 'includeantruntime' was not set, defaulting to build.sysclasspath=last; set to false for repeatable builds

includeantruntime="false"

```
<target name="compile" depends="init">
   <javac includeantruntime="false" srcdir="${src}"
destdir="${dest}"/>
</target>
```

or

```
<property name="build.sysclasspath" value="last"/>
```

调试 exec

将 executable="echo" 设置成 echo 是一种不错的调试手段

```
        <macrodef name="gulp">
                <attribute name="stage" default=""/>
                <attribute name="src" default=""/>
                <attribute name="dir" default="" />
                <sequential>
                        <exec vmlauncher="false"
executable="echo" dir="@{dir}" osfamily="unix">
                                <arg line="--stage @{stage} --
src @{src}"/>
                                <!-- arg value="stage @{stage}"
/ -->
                        </exec>
                </sequential>
        </macrodef>

        <target name="gulp">
                <gulp stage="${git.branch}" src="cn" dir="."/>
        </target>
```

2. Apache Ivy

http://ant.apache.org/ivy/index.html

2.1. Ivy Install

source code

```
cd /usr/local/src
wget http://labs.renren.com/apache-
mirror//ant/ivy/2.2.0/apache-ivy-2.2.0-bin.tar.gz
tar zxvf apache-ivy-2.2.0-bin.tar.gz
mv apache-ivy-2.2.0 /usr/local/
cd ..
ln -s apache-ivy-2.2.0 apache-ivy
```

```
IVY_HOME=/usr/local/apache-ivy
```

```
cp $IVY_HOME/ivy-2.2.0.jar $ANT_HOME/lib/
```

apt-get

```
$ sudo apt-get install ant
$ sudo apt-get install ivy
```

To know more about this package, you can use dpkg

```
$ dpkg -s ivy
```

2.2. Test example

ant

```
cd $IVY_HOME/src/example/hello-ivy
ant

Buildfile: /usr/local/apache-ivy-2.2.0/src/example/hello-
ivy/build.xml

resolve:
[ivy:retrieve] :: Ivy 2.2.0 - 20100923230623 ::
http://ant.apache.org/ivy/ ::
[ivy:retrieve] :: loading settings :: url =
jar:file:/usr/local/apache-ant/lib/ivy-
2.2.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
[ivy:retrieve] :: resolving dependencies :: org.apache#hello-
ivy;working@example.com
[ivy:retrieve]  confs: [default]
[ivy:retrieve]  found commons-lang#commons-lang;2.0 in public
[ivy:retrieve]  found commons-cli#commons-cli;1.0 in public
[ivy:retrieve]  found commons-logging#commons-logging;1.0 in
public
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-lang/commons-
lang/2.0/commons-lang-2.0.jar ...
[ivy:retrieve]
.....................................................................
.....................
[ivy:retrieve]
.....................................................................
.....................................................................
.................................
[ivy:retrieve]
.....................................................................
.......................... (165kB)
[ivy:retrieve] .. (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-lang#commons-
lang;2.0!commons-lang.jar (4790ms)
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-lang/commons-
lang/2.0/commons-lang-2.0-javadoc.jar ...
[ivy:retrieve]
```

```
.............................................................
...............................................
[ivy:retrieve] ........................................
[ivy:retrieve]
...............................................
[ivy:retrieve]
...................................................
[ivy:retrieve]
.............................................................
.............................................................
....
[ivy:retrieve]
.............................................................
.............................................................
....
[ivy:retrieve]
.............................................................
...............................................
[ivy:retrieve]
.............................................................
.............................................................
.............................................
[ivy:retrieve]
.............................................................
.............................................................
.............................................................
. (467kB)
[ivy:retrieve] .. (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-lang#commons-
lang;2.0!commons-lang.jar(javadoc) (14878ms)
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-lang/commons-
lang/2.0/commons-lang-2.0-sources.jar ...
[ivy:retrieve]
.............................................................
.............................................................
.............................................
[ivy:retrieve]
.............................................................
.............................................................
.............................................................
...................
[ivy:retrieve]
.............................................................
.............................................................
```

```
.............. (245kB)
[ivy:retrieve] .. (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-lang#commons-
lang;2.0!commons-lang.jar(source) (5046ms)
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-cli/commons-
cli/1.0/commons-cli-1.0-javadoc.jar ...
[ivy:retrieve]
.....................................................................
.....................................................................
.....................
[ivy:retrieve] ..................................... (92kB)
[ivy:retrieve] .. (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-cli#commons-
cli;1.0!commons-cli.jar(javadoc) (2838ms)
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-cli/commons-
cli/1.0/commons-cli-1.0.jar ...
[ivy:retrieve]
.....................................................
(29kB)
[ivy:retrieve] .. (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-cli#commons-
cli;1.0!commons-cli.jar (5147ms)
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-cli/commons-
cli/1.0/commons-cli-1.0-sources.jar ...
[ivy:retrieve]
.....................................................................
..................................... (48kB)
[ivy:retrieve] .. (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-cli#commons-
cli;1.0!commons-cli.jar(source) (2163ms)
[ivy:retrieve] downloading
http://repo1.maven.org/maven2/commons-logging/commons-
logging/1.0/commons-logging-1.0.jar ...
[ivy:retrieve] ........................................
(21kB)
[ivy:retrieve] ... (0kB)
[ivy:retrieve]  [SUCCESSFUL ] commons-logging#commons-
logging;1.0!commons-logging.jar (2638ms)
[ivy:retrieve] :: resolution report :: resolve 30806ms ::
artifacts dl 37511ms
[ivy:retrieve]  :: evicted modules:
[ivy:retrieve]  commons-lang#commons-lang;1.0 by [commons-
```

```
lang#commons-lang;2.0] in [default]
        ---------------------------------------------------------
--------------
        |                  |            modules            ||
artifacts   |
        |       conf       | number| search|dwnlded|evicted||
number|dwnlded|
        ---------------------------------------------------------
--------------
        |      default     |   4   |   3   |   3   |   1   ||
7   |   7   |
        ---------------------------------------------------------
--------------
[ivy:retrieve] :: retrieving :: org.apache#hello-ivy
[ivy:retrieve]  confs: [default]
[ivy:retrieve]  7 artifacts copied, 0 already retrieved
(1069kB/11ms)

run:
    [mkdir] Created dir: /usr/local/apache-ivy-
2.2.0/src/example/hello-ivy/build
    [javac] /usr/local/apache-ivy-2.2.0/src/example/hello-
ivy/build.xml:53: warning: 'includeantruntime' was not set,
defaulting to build.sysclasspath=last; set to false for
repeatable builds
    [javac] Compiling 1 source file to /usr/local/apache-ivy-
2.2.0/src/example/hello-ivy/build
     [java] standard message : hello ivy !
     [java] capitalized by org.apache.commons.lang.WordUtils :
Hello Ivy !

BUILD SUCCESSFUL
Total time: 1 second
```

run it

```
neo@debian:/usr/local/apache-ivy/src/example/hello-ivy/build$
export CLASSPATH=$CLASSPATH:/usr/local/apache-
ivy/src/example/hello-ivy/lib/*
neo@debian:/usr/local/apache-ivy/src/example/hello-ivy/build$
/usr/local/java/bin/java example.Hello
```

```
standard message : hello ivy !
capitalized by org.apache.commons.lang.WordUtils : Hello Ivy !
```

3. Apache Maven

http://maven.apache.org/

3.1. 安装 Maven

CentOS 8 安装 Maven

```
[root@localhost ~]# dnf install maven
```

Ubuntu

```
$ sudo apt-get install maven2

$ mvn -version
Apache Maven 2.2.1 (rdebian-4)
Java version: 1.6.0_22
Java home: /usr/lib/jvm/java-6-openjdk/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux" version: "2.6.38-8-generic" arch: "amd64"
Family: "unix"
```

```
JAVA_HOME="/usr/lib/jvm/java-6-openjdk/jre"
MAVEN_HOME="/usr/share/maven2/"
```

一键安装

```
curl -s
```

```
https://raw.githubusercontent.com/oscm/shell/master/lang/java/maven/maven.sh | bash
```

apache-maven-3.8.2

```
#!/bin/bash

cd /usr/local/src/
wget https://mirrors.bfsu.edu.cn/apache/maven/maven-3/3.8.2/binaries/apache-maven-3.8.2-bin.tar.gz
tar zxf apache-maven-3.8.2-bin.tar.gz
mv apache-maven-3.8.2 /srv/
rm -f /srv/apache-maven
ln -s /srv/apache-maven-3.8.2 /srv/apache-maven

alternatives --install /usr/local/bin/mvn apache-maven-3.8.2 /srv/apache-maven-3.8.2/bin/mvn 0

mvn -v
```

apache-maven-3.8.2 配置

```
[root@localhost ~]# vim /srv/apache-maven/conf/settings.xml
  <mirrors>
    <!-- mirror
     | Specifies a repository mirror site to use instead of a given repository. The repository that
     | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used
     | for inheritance and direct lookup purposes, and must be unique across the set of mirrors.
     |
    <mirror>
      <id>mirrorId</id>
      <mirrorOf>repositoryId</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
```

```
      <url>http://my.repository.com/repo/path</url>
    </mirror>
     -->
    <mirror>
      <id>maven-default-http-blocker</id>
      <mirrorOf>external:http:*</mirrorOf>
      <name>Pseudo repository to mirror external repositories
initially using HTTP.</name>
      <url>http://0.0.0.0/</url>
      <blocked>true</blocked>
    </mirror>
  </mirrors>
```

apache-maven-3.8.2 默认会阻止其他镜像，需要会去掉 maven-default-http-blocker 配置

mvnd

```
cd /usr/local/src
wget https://github.com/apache/maven-
mvnd/releases/download/0.7.1/mvnd-0.7.1-linux-amd64.zip
unzip mvnd-0.7.1-linux-amd64.zip
mv mvnd-0.7.1-linux-amd64 /srv/mvnd-0.7.1
ln -s /srv/mvnd-0.7.1 /srv/mvnd

alternatives --remove mvnd /usr/local/bin/mvnd
alternatives --install /usr/local/bin/mvnd mvnd-0.7.1
/srv/mvnd-0.7.1/bin/mvnd 0
```

修改配置文件 mvnd.properties 制定 JAVA_HOME

```
[root@localhost cloud.netkiller.cn]# grep java.home
/srv/mvnd/conf/mvnd.properties
java.home=/usr/lib/jvm/java
```

3.2. Maven 命令

切换 JAVA 版本

当前 Java 是 OpenJDK 17

```
Neo-iMac:~ neo$ java -version
openjdk version "17.0.1" 2021-10-19
OpenJDK Runtime Environment Homebrew (build 17.0.1+0)
OpenJDK 64-Bit Server VM Homebrew (build 17.0.1+0, mixed mode,
sharing)
```

代码中增加了

```
        <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <encoding>UTF-8</encoding>
        <java.version>1.8</java.version>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>
```

仍如不能通过编译，提示:

```
An exception has occurred in the compiler (17.0.1). Please file
a bug against the Java compiler via the Java bug reporting page
(http://bugreport.java.com) after checking the Bug Database
(http://bugs.java.com) for duplicates. Include your program,
```

```
the following diagnostic, and the parameters passed to the Java
compiler in your report. Thank you.
com.sun.tools.javac.code.Symbol$CompletionFailure: class file
for jakarta.validation.constraints.Pattern$Flag not found
```

这是因为有些第三方包依赖 OpenJDK 8 的一些包。安装 openjdk8 使用 1.8 编译问题得到解决。

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/openjdk-
8.jdk/Contents/Home
mvn package
```

参数

多线程下载 -Dmaven.artifact.threads=10 默认是 5

```
mvn -Dmaven.artifact.threads=10 test

您可以使用MAVEN_OPTS环境变量。例如：

export MAVEN_OPTS = -Dmaven.artifact.threads = 10

MAVEN_OPTS = -Dmaven.repo.local=.m2/repository
```

-s 指定 settings.xml 文件

```
mvn clean install -Dmaven.test.skip=true -s ~/.m2/settings.xml
```

多线程

使用4个线程构建

```
mvn -T 4 install
```

每个CPU核心1个线程

```
mvn -T 1C install
```

help

查看可用的pom定义

```
mvn help:effective-pom
```

archetype:create

创建 Maven 项目

```
mvn archetype:create   -
DarchetypeGroupId=org.apache.maven.archetypes   -
DgroupId=com.company -DartifactId=my-app
```

```

```

```
mvn archetype:create
    -DgroupId=packageName
    -DartifactId=webappName
    -DarchetypeArtifactId=maven-archetype-webapp
```

clean

清楚

```
$ mvn clean
```

compile

编译项目的源代码

```
$ mvn compile
```

多线程编译

增加编译-Dmaven.compile.fork=true 参数，用以指明多线程进行编译

增加 -T 1C 参数，表示每个CPU核心跑一个工程

```
mvn clean package -T 1C -Dmaven.test.skip=true  -
Dmaven.compile.fork=true
```

-T 4 是直接指定4线程

-T 1C 表示CPU线程的倍数,现在现在1个物理CPU，有4个核心，8个线程。那么此时-T 1C 就是8线程。

编译测试代码

```
mvn test-compile
```

test

编译测试的源代码

```
mvn test-compile
```

运行测试

```
$ mvn test
```

打包但不测试

```
mvn -D maven.test.skip=true package
```

忽略测试失败

```
mvn test -Dmaven.test.failure.ignore
```

```

```

package

将编译后的代码打包

```
$ mvn package
```

```
$ ls target/*.war
target/www.netkiller.cn-0.0.1-SNAPSHOT.war
```

防止出现脏数据，可以先 clean 然后再打包，同时为了加快打包速度，逃过测试

```
mvn clean package -Dmaven.test.skip=true
```

install

将项目打包后安装到本地仓库，可以作为其它项目的本地依赖。

```
$ mvn install
```

跳过测试

```
mvn install -Dmaven.test.skip=true
```

install-file

安装jar到本地仓库

```
mvn install:install-file -Dfile=<path-to-file> -DgroupId=
<group-id> -DartifactId=<artifact-id> -Dversion=<version> -
Dpackaging=<packaging>

<path-to-file>: 要安装的JAR的本地路径
<group-id>:             要安装的JAR的Group Id
<artifact-id>:  要安装的JAR的 Artificial Id
<version>:              JAR 版本
<packaging>:    打包类型，例如JAR
```

安装本地 path/to/your.jar 文件到Maven的.m2库中。

```
mvn install:install-file -Dfile=path/to/your.jar -
DgroupId=com.example -DartifactId=project -Dversion=1.2.0 -
Dpackaging=jar
```

war

```
mvn compile war:war
```

```
mvn compile war:exploded
```

```
mvn compile war:inplace
```

exec

```
$ mvn exec:java -Dexec.mainClass="cn.netkiller.Test"
```

dependency

build-classpath

```
$ mvn dependency:build-classpath
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
----------------
[INFO] Building api.netkiller.cn 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------
----------------
[INFO]
[INFO] --- maven-dependency-plugin:2.10:build-classpath
(default-cli) @ api.cf88.com ---
[INFO] Dependencies classpath:
/www/.m2/repository/org/springframework/boot/spring-boot-
starter-web/1.3.0.RELEASE/spring-boot-starter-web-
1.3.0.RELEASE.jar:......:/www/.m2/repository/junit/junit/4.12/j
unit-4.12.jar
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time: 2.692 s
[INFO] Finished at: 2016-08-10T17:32:49+08:00
[INFO] Final Memory: 25M/162M
[INFO] ------------------------------------------------------------
----------------
```

dependency:tree 显示包依赖树

```
[www@iZ62yln3rjjZ repository]$ mvn dependency:tree
```

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
---------------
[INFO] Building api.example.com 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------
---------------
[INFO]
[INFO] --- maven-dependency-plugin:2.10:tree (default-cli) @
api.cf88.com ---
[INFO] cf88.com:api.cf88.com:jar:0.0.1-SNAPSHOT
[INFO] +- org.springframework.boot:spring-boot-starter-
web:jar:1.3.0.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot-
starter:jar:1.3.0.RELEASE:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-starter-
logging:jar:1.3.0.RELEASE:compile
[INFO] |  |  |  +- ch.qos.logback:logback-
classic:jar:1.1.3:compile
[INFO] |  |  |  |  \- ch.qos.logback:logback-
core:jar:1.1.3:compile
[INFO] |  |  |  +- org.slf4j:jul-to-slf4j:jar:1.7.13:compile
[INFO] |  |  |  \- org.slf4j:log4j-over-
slf4j:jar:1.7.13:compile
[INFO] |  |  \- org.yaml:snakeyaml:jar:1.16:runtime
[INFO] |  +- org.springframework.boot:spring-boot-starter-
tomcat:jar:1.3.0.RELEASE:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-
core:jar:8.0.28:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-
el:jar:8.0.28:compile
[INFO] |  |  +- org.apache.tomcat.embed:tomcat-embed-logging-
juli:jar:8.0.28:compile
[INFO] |  |  \- org.apache.tomcat.embed:tomcat-embed-
websocket:jar:8.0.28:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter-
validation:jar:1.3.0.RELEASE:compile
[INFO] |  |  \- org.hibernate:hibernate-
validator:jar:5.2.2.Final:compile
[INFO] |  |     +- javax.validation:validation-
api:jar:1.1.0.Final:compile
[INFO] |  |     \- com.fasterxml:classmate:jar:1.1.0:compile
[INFO] |  +- com.fasterxml.jackson.core:jackson-
databind:jar:2.6.3:compile
[INFO] |  |  +- com.fasterxml.jackson.core:jackson-
```

```
annotations:jar:2.6.3:compile
[INFO] |  |  \- com.fasterxml.jackson.core:jackson-
core:jar:2.6.3:compile
[INFO] |  +- org.springframework:spring-
web:jar:4.2.3.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-
aop:jar:4.2.3.RELEASE:compile
[INFO] |  |     \- aopalliance:aopalliance:jar:1.0:compile
[INFO] |  \- org.springframework:spring-
webmvc:jar:4.2.3.RELEASE:compile
[INFO] +- org.springframework.boot:spring-boot-starter-data-
jpa:jar:1.3.0.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter-
aop:jar:1.3.0.RELEASE:compile
[INFO] |  |  \- org.aspectj:aspectjweaver:jar:1.8.7:compile
[INFO] |  +- org.hibernate:hibernate-
entitymanager:jar:4.3.11.Final:compile
[INFO] |  |  +- org.jboss.logging:jboss-
logging:jar:3.3.0.Final:compile
[INFO] |  |  +- org.jboss.logging:jboss-logging-
annotations:jar:1.2.0.Beta1:compile
[INFO] |  |  +- org.hibernate:hibernate-
core:jar:4.3.11.Final:compile
[INFO] |  |  |  +- antlr:antlr:jar:2.7.7:compile
[INFO] |  |  |  \- org.jboss:jandex:jar:1.1.0.Final:compile
[INFO] |  |  +- dom4j:dom4j:jar:1.6.1:compile
[INFO] |  |  |  \- xml-apis:xml-apis:jar:1.0.b2:compile
[INFO] |  |  +- org.hibernate.common:hibernate-commons-
annotations:jar:4.0.5.Final:compile
[INFO] |  |  +- org.hibernate.javax.persistence:hibernate-jpa-
2.1-api:jar:1.0.0.Final:compile
[INFO] |  |  \- org.javassist:javassist:jar:3.18.1-GA:compile
[INFO] |  +- javax.transaction:javax.transaction-
api:jar:1.2:compile
[INFO] |  +- org.springframework.data:spring-data-
jpa:jar:1.9.1.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-
orm:jar:4.2.3.RELEASE:compile
[INFO] |  \- org.springframework:spring-
aspects:jar:4.2.3.RELEASE:compile
[INFO] +- org.springframework.boot:spring-boot-starter-
jdbc:jar:1.3.0.RELEASE:compile
[INFO] |  +- org.apache.tomcat:tomcat-jdbc:jar:8.0.28:compile
[INFO] |  |  \- org.apache.tomcat:tomcat-
juli:jar:8.0.28:compile
```

```
[INFO] |  \- org.springframework:spring-
jdbc:jar:4.2.3.RELEASE:compile
[INFO] +- org.springframework.boot:spring-boot-starter-
redis:jar:1.3.0.RELEASE:compile
[INFO] |  +- org.springframework.data:spring-data-
redis:jar:1.6.1.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-context-
support:jar:4.2.3.RELEASE:compile
[INFO] |  \- redis.clients:jedis:jar:2.7.3:compile
[INFO] |     \- org.apache.commons:commons-
pool2:jar:2.4.2:compile
[INFO] +- org.springframework.boot:spring-boot-starter-data-
mongodb:jar:1.3.0.RELEASE:compile
[INFO] |  \- org.mongodb:mongo-java-driver:jar:2.13.3:compile
[INFO] +- org.springframework.boot:spring-boot-starter-
amqp:jar:1.3.0.RELEASE:compile
[INFO] |  +- org.springframework:spring-
messaging:jar:4.2.3.RELEASE:compile
[INFO] |  \- org.springframework.amqp:spring-
rabbit:jar:1.5.2.RELEASE:compile
[INFO] |     +- org.springframework.retry:spring-
retry:jar:1.1.2.RELEASE:compile
[INFO] |     +- org.springframework.amqp:spring-
amqp:jar:1.5.2.RELEASE:compile
[INFO] |     +- com.rabbitmq:http-
client:jar:1.0.0.RELEASE:compile
[INFO] |     |  \-
org.apache.httpcomponents:httpclient:jar:4.5.1:compile
[INFO] |     |     +-
org.apache.httpcomponents:httpcore:jar:4.4.4:compile
[INFO] |     |     \- commons-codec:commons-
codec:jar:1.9:compile
[INFO] |     \- com.rabbitmq:amqp-client:jar:3.5.6:compile
[INFO] +- org.springframework.boot:spring-boot-
devtools:jar:1.3.0.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-
boot:jar:1.3.0.RELEASE:compile
[INFO] |  \- org.springframework.boot:spring-boot-
autoconfigure:jar:1.3.0.RELEASE:compile
[INFO] +- org.springframework.boot:spring-boot-starter-
test:jar:1.3.0.RELEASE:test
[INFO] |  +- org.mockito:mockito-core:jar:1.10.19:test
[INFO] |  |  \- org.objenesis:objenesis:jar:2.1:test
[INFO] |  +- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO] |  +- org.hamcrest:hamcrest-library:jar:1.3:test
```

```
[INFO] |  +- org.springframework:spring-
core:jar:4.2.3.RELEASE:compile
[INFO] |  \- org.springframework:spring-
test:jar:4.2.3.RELEASE:test
[INFO] +- org.springframework.data:spring-data-
mongodb:jar:1.8.1.RELEASE:compile
[INFO] |  +- org.springframework:spring-
tx:jar:4.2.3.RELEASE:compile
[INFO] |  +- org.springframework:spring-
context:jar:4.2.3.RELEASE:compile
[INFO] |  +- org.springframework:spring-
beans:jar:4.2.3.RELEASE:compile
[INFO] |  +- org.springframework:spring-
expression:jar:4.2.3.RELEASE:compile
[INFO] |  +- org.springframework.data:spring-data-
commons:jar:1.11.1.RELEASE:compile
[INFO] |  +- org.slf4j:slf4j-api:jar:1.7.13:compile
[INFO] |  \- org.slf4j:jcl-over-slf4j:jar:1.7.13:compile
[INFO] +- mysql:mysql-connector-java:jar:5.1.37:compile
[INFO] +- com.google.code.gson:gson:jar:2.3.1:compile
[INFO] \- junit:junit:jar:4.12:test
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time: 4.400 s
[INFO] Finished at: 2016-08-09T10:25:07+08:00
[INFO] Final Memory: 23M/163M
[INFO] ------------------------------------------------------------
----------------
```

copy-dependencies 导出依赖包

```
$ mvn dependency:copy-dependencies
```

```
$ ls target/dependency/
antlr-2.7.7.jar                                  javassist-
3.18.1-GA.jar                    snakeyaml-1.16.jar
spring-boot-starter-web-1.3.0.RELEASE.jar        spring-webmvc-
```

```
4.2.3.RELEASE.jar
aopalliance-1.0.jar
javax.transaction-api-1.2.jar              spring-aop-
4.2.3.RELEASE.jar                       spring-boot-starter-
websocket-1.3.0.RELEASE.jar  spring-websocket-4.2.3.RELEASE.jar
aspectjweaver-1.8.7.jar                          jboss-logging-
3.3.0.Final.jar              spring-aspects-4.2.3.RELEASE.jar
spring-context-4.2.3.RELEASE.jar                   thymeleaf-
2.1.4.RELEASE.jar
classmate-1.1.0.jar                              jboss-logging-
annotations-1.2.0.Beta1.jar  spring-beans-4.2.3.RELEASE.jar
spring-core-4.2.3.RELEASE.jar                      thymeleaf-
layout-dialect-1.3.1.jar
dom4j-1.6.1.jar                                  jcl-over-slf4j-
1.7.13.jar                  spring-boot-1.3.0.RELEASE.jar
spring-data-commons-1.11.1.RELEASE.jar             thymeleaf-
spring4-2.1.4.RELEASE.jar
groovy-2.4.4.jar                                 jstl-1.2.jar
spring-boot-autoconfigure-1.3.0.RELEASE.jar       spring-data-
jpa-1.9.1.RELEASE.jar                tomcat-embed-core-
8.0.28.jar
hibernate-commons-annotations-4.0.5.Final.jar  jul-to-slf4j-
1.7.13.jar                     spring-boot-starter-
1.3.0.RELEASE.jar              spring-expression-
4.2.3.RELEASE.jar               tomcat-embed-el-8.0.28.jar
hibernate-core-4.3.11.Final.jar                  log4j-over-
slf4j-1.7.13.jar                 spring-boot-starter-aop-
1.3.0.RELEASE.jar          spring-jdbc-4.2.3.RELEASE.jar
tomcat-embed-logging-juli-8.0.28.jar
hibernate-entitymanager-4.3.11.Final.jar       logback-classic-
1.1.3.jar                   spring-boot-starter-data-jpa-
1.3.0.RELEASE.jar    spring-messaging-4.2.3.RELEASE.jar
tomcat-embed-websocket-8.0.28.jar
hibernate-jpa-2.1-api-1.0.0.Final.jar          logback-core-
1.1.3.jar                      spring-boot-starter-jdbc-
1.3.0.RELEASE.jar        spring-orm-4.2.3.RELEASE.jar
tomcat-jdbc-8.0.28.jar
hibernate-validator-5.2.2.Final.jar            mybatis-
3.3.0.jar                            spring-boot-starter-logging-
1.3.0.RELEASE.jar     spring-security-config-4.0.3.RELEASE.jar
tomcat-juli-8.0.28.jar
jackson-annotations-2.6.3.jar                    mybatis-spring-
1.2.3.jar                    spring-boot-starter-security-
1.3.0.RELEASE.jar    spring-security-core-4.0.3.RELEASE.jar
unbescape-1.1.0.RELEASE.jar
```

```
jackson-core-2.6.3.jar                          mysql-connector-
java-5.1.37.jar            spring-boot-starter-thymeleaf-
1.3.0.RELEASE.jar   spring-security-web-4.0.3.RELEASE.jar
validation-api-1.1.0.Final.jar
jackson-databind-2.6.3.jar                      ognl-3.0.8.jar
spring-boot-starter-tomcat-1.3.0.RELEASE.jar      spring-tx-
4.2.3.RELEASE.jar                       xml-apis-1.0.b2.jar
jandex-1.1.0.Final.jar                          slf4j-api-
1.7.13.jar                        spring-boot-starter-
validation-1.3.0.RELEASE.jar  spring-web-4.2.3.RELEASE.jar
```

analyze 查看未被使用的包

```
$ mvn dependency:analyze
```

sources 下载源码

```
mvn dependency:sources
```

jar

```
mvn -Djar.finalName=myCustomName package
```

```
mvn jar:jar -Djar.finalName=custom-jar-name
```

构建装配Maven Assembly

```
mvn install assembly:assembly
```

加密密码

--encrypt-master-password

```
neo@MacBook-Pro ~ % mvn --encrypt-master-password test
{LhCRW9y8CG4HfT7ExHI3msP56Cv+fgjdiNhTk883hZs=}
```

help:describe

显示插件的详细信息

```
MacBook-Pro:deployment neo$ mvn help:describe -
DgroupId=org.apache.maven.plugins                    -
DartifactId=maven-war-plugin                   -Ddetail=true
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
----------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------
----------------
[INFO]
[INFO] --- maven-help-plugin:2.2:describe (default-cli) @
standalone-pom ---
[INFO] org.apache.maven.plugins:maven-war-plugin:3.1.0

Name: Apache Maven WAR Plugin
Description: Builds a Web Application Archive (WAR) file from
the project
  output and its dependencies.
```

```
Group Id: org.apache.maven.plugins
Artifact Id: maven-war-plugin
Version: 3.1.0
Goal Prefix: war

This plugin has 4 goals:

war:exploded
  Description: Create an exploded webapp in a specified
directory.
  Implementation: org.apache.maven.plugins.war.WarExplodedMojo
  Language: java
  Bound to phase: package

  Available parameters:

    archive
      The archive configuration to use. See Maven Archiver
Reference.

    archiveClasses (Default: false)
      Whether a JAR file will be created for the classes in the
webapp. Using
      this optional configuration parameter will make the
compiled classes to
      be archived into a JAR file and the classes directory
will then be
      excluded from the webapp.

    cacheFile (Default:
${project.build.directory}/war/work/webapp-cache.xml)
      Required: true
      The file containing the webapp structure cache.

    containerConfigXML
      The path to a configuration file for the servlet
container. Note that the
      file name may be different for different servlet
containers. Apache
      Tomcat uses a configuration file named context.xml. The
file will be
      copied to the META-INF directory.

    delimiters
      Set of delimiters for expressions to filter within the
```

```
resources. These
      delimiters are specified in the form
'beginToken*endToken'. If no '*' is
      given, the delimiter is assumed to be the same for start
and end.

      So, the default filtering delimiters might be specified
as:

      <delimiters>
        <delimiter>${*}</delimiter>
        <delimiter>@</delimiter>
      </delimiters>

      Since the '@' delimiter is the same on both ends, we
don't need to
      specify '@*@' (though we can).

    dependentWarExcludes
      The comma separated list of tokens to exclude when doing
a WAR overlay.
      Default is Overlay.DEFAULT_EXCLUDES

    dependentWarIncludes
      The comma separated list of tokens to include when doing
a WAR overlay.
      Default is Overlay.DEFAULT_INCLUDES

    escapedBackslashesInFilePath (Default: false)
      To escape interpolated values with Windows path
c:\foo\bar will be
      replaced with c:\\foo\\bar.

    escapeString
      Expression preceded with this String won't be
interpolated. \${foo} will
      be replaced with ${foo}.

    filteringDeploymentDescriptors (Default: false)
      To filter deployment descriptors. Disabled by default.

    filters
      Filters (property files) to include during the
interpolation of the
      pom.xml.
```

```
    includeEmptyDirectories (Default: false)
      (no description available)

    nonFilteredFileExtensions
      A list of file extensions that should not be filtered.
Will be used when
      filtering webResources and overlays.

    outputFileNameMapping
      The file name mapping to use when copying libraries and
TLDs. If no file
      mapping is set (default) the files are copied with their
standard names.

    overlays
      The overlays to apply. Each <overlay> element may
contain:
      - id (defaults to currentBuild)
      - groupId (if this and artifactId are null, then the
current project is
        treated as its own overlay)
      - artifactId (see above)
      - classifier
      - type
      - includes (a list of string patterns)
      - excludes (a list of string patterns)
      - filtered (defaults to false)
      - skip (defaults to false)
      - targetPath (defaults to root of webapp structure)

    recompressZippedFiles (Default: true)
      Indicates if zip archives (jar,zip etc) being added to
the war should be
      compressed again. Compressing again can result in smaller
archive size,
      but gives noticeably longer execution time.

    resourceEncoding (Default: ${project.build.sourceEncoding})
      The encoding to use when copying filtered web resources.

    supportMultiLineFiltering (Default: false)
      Stop searching endToken at the end of line

    useCache (Default: false)
```

```
      Whether the cache should be used to save the status of
the webapp across
      multiple runs. Experimental feature so disabled by
default.

    useDefaultDelimiters (Default: true)
      Use default delimiters in addition to custom delimiters,
if any.

    useJvmChmod (Default: true)
      use jvmChmod rather that cli chmod and forking process

    warSourceDirectory (Default: ${basedir}/src/main/webapp)
      Required: true
      Single directory for extra files to include in the WAR.
This is where you
      place your JSP files.

    warSourceExcludes
      The comma separated list of tokens to exclude when
copying the content of
      the warSourceDirectory.

    warSourceIncludes (Default: **)
      The comma separated list of tokens to include when
copying the content of
      the warSourceDirectory.

    webappDirectory (Default:
    ${project.build.directory}/${project.build.finalName})
      Required: true
      The directory where the webapp is built.

    webResources
      The list of webResources we want to transfer.

    webXml
      The path to the web.xml file to use.

    workDirectory (Default:
${project.build.directory}/war/work)
      Required: true
      Directory to unpack dependent WARs into if needed.

war:help
```

```
  Description: Display help information on maven-war-plugin.
    Call mvn war:help -Ddetail=true -Dgoal=<goal-name> to
display parameter
    details.
  Implementation: org.apache.maven.plugins.war.HelpMojo
  Language: java

  Available parameters:

    detail (Default: false)
      User property: detail
      If true, display all settable properties for each goal.

    goal
      User property: goal
      The name of the goal for which to show help. If
unspecified, all goals
      will be displayed.

    indentSize (Default: 2)
      User property: indentSize
      The number of spaces per indentation level, should be
positive.

    lineLength (Default: 80)
      User property: lineLength
      The maximum length of a display line, should be positive.

war:inplace
  Description: Generate the webapp in the WAR source directory.
  Implementation: org.apache.maven.plugins.war.WarInPlaceMojo
  Language: java

  Available parameters:

    archive
      The archive configuration to use. See Maven Archiver
Reference.

    archiveClasses (Default: false)
      Whether a JAR file will be created for the classes in the
webapp. Using
      this optional configuration parameter will make the
compiled classes to
      be archived into a JAR file and the classes directory
```

```
will then be
      excluded from the webapp.

    cacheFile (Default:
${project.build.directory}/war/work/webapp-cache.xml)
      Required: true
      The file containing the webapp structure cache.

    containerConfigXML
      The path to a configuration file for the servlet
container. Note that the
      file name may be different for different servlet
containers. Apache
      Tomcat uses a configuration file named context.xml. The
file will be
      copied to the META-INF directory.

    delimiters
      Set of delimiters for expressions to filter within the
resources. These
      delimiters are specified in the form
'beginToken*endToken'. If no '*' is
      given, the delimiter is assumed to be the same for start
and end.

      So, the default filtering delimiters might be specified
as:

      <delimiters>
        <delimiter>${*}</delimiter>
        <delimiter>@</delimiter>
      </delimiters>

      Since the '@' delimiter is the same on both ends, we
don't need to
      specify '@*@' (though we can).

    dependentWarExcludes
      The comma separated list of tokens to exclude when doing
a WAR overlay.
      Default is Overlay.DEFAULT_EXCLUDES

    dependentWarIncludes
      The comma separated list of tokens to include when doing
a WAR overlay.
```

```
      Default is Overlay.DEFAULT_INCLUDES

    escapedBackslashesInFilePath (Default: false)
      To escape interpolated values with Windows path
c:\foo\bar will be
      replaced with c:\\foo\\bar.

    escapeString
      Expression preceded with this String won't be
interpolated. \${foo} will
      be replaced with ${foo}.

    filteringDeploymentDescriptors (Default: false)
      To filter deployment descriptors. Disabled by default.

    filters
      Filters (property files) to include during the
interpolation of the
      pom.xml.

    includeEmptyDirectories (Default: false)
      (no description available)

    nonFilteredFileExtensions
      A list of file extensions that should not be filtered.
Will be used when
      filtering webResources and overlays.

    outputFileNameMapping
      The file name mapping to use when copying libraries and
TLDs. If no file
      mapping is set (default) the files are copied with their
standard names.

    overlays
      The overlays to apply. Each <overlay> element may
contain:
      - id (defaults to currentBuild)
      - groupId (if this and artifactId are null, then the
current project is
        treated as its own overlay)
      - artifactId (see above)
      - classifier
      - type
      - includes (a list of string patterns)
```

```
      - excludes (a list of string patterns)
      - filtered (defaults to false)
      - skip (defaults to false)
      - targetPath (defaults to root of webapp structure)

    recompressZippedFiles (Default: true)
      Indicates if zip archives (jar,zip etc) being added to
the war should be
      compressed again. Compressing again can result in smaller
archive size,
      but gives noticeably longer execution time.

    resourceEncoding (Default: ${project.build.sourceEncoding})
      The encoding to use when copying filtered web resources.

    supportMultiLineFiltering (Default: false)
      Stop searching endToken at the end of line

    useCache (Default: false)
      Whether the cache should be used to save the status of
the webapp across
      multiple runs. Experimental feature so disabled by
default.

    useDefaultDelimiters (Default: true)
      Use default delimiters in addition to custom delimiters,
if any.

    useJvmChmod (Default: true)
      use jvmChmod rather that cli chmod and forking process

    warSourceDirectory (Default: ${basedir}/src/main/webapp)
      Required: true
      Single directory for extra files to include in the WAR.
This is where you
      place your JSP files.

    warSourceExcludes
      The comma separated list of tokens to exclude when
copying the content of
      the warSourceDirectory.

    warSourceIncludes (Default: **)
      The comma separated list of tokens to include when
copying the content of
```

```
      the warSourceDirectory.

    webappDirectory (Default:
    ${project.build.directory}/${project.build.finalName})
      Required: true
      The directory where the webapp is built.

    webResources
      The list of webResources we want to transfer.

    webXml
      The path to the web.xml file to use.

    workDirectory (Default:
${project.build.directory}/war/work)
      Required: true
      Directory to unpack dependent WARs into if needed.

war:war
  Description: Build a WAR file.
  Implementation: org.apache.maven.plugins.war.WarMojo
  Language: java
  Bound to phase: package

  Available parameters:

    archive
      The archive configuration to use. See Maven Archiver
Reference.

    archiveClasses (Default: false)
      Whether a JAR file will be created for the classes in the
webapp. Using
      this optional configuration parameter will make the
compiled classes to
      be archived into a JAR file and the classes directory
will then be
      excluded from the webapp.

    attachClasses (Default: false)
      Whether classes (that is the content of the WEB-
INF/classes directory)
      should be attached to the project as an additional
artifact.
      By default the classifier for the additional artifact is
```

```
'classes'. You
      can change it with the someclassifier parameter.

      If this parameter true, another project can depend on the
classes by
      writing something like:


        myGroup
        myArtifact
        myVersion
        classes

    cacheFile (Default:
${project.build.directory}/war/work/webapp-cache.xml)
      Required: true
      The file containing the webapp structure cache.

    classesClassifier (Default: classes)
      The classifier to use for the attached classes artifact.

    classifier
      Classifier to add to the generated WAR. If given, the
artifact will be an
      attachment instead. The classifier will not be applied to
the JAR file of
      the project - only to the WAR file.

    containerConfigXML
      The path to a configuration file for the servlet
container. Note that the
      file name may be different for different servlet
containers. Apache
      Tomcat uses a configuration file named context.xml. The
file will be
      copied to the META-INF directory.

    delimiters
      Set of delimiters for expressions to filter within the
resources. These
      delimiters are specified in the form
'beginToken*endToken'. If no '*' is
      given, the delimiter is assumed to be the same for start
and end.
```

```
      So, the default filtering delimiters might be specified
as:

      Since the '@' delimiter is the same on both ends, we
don't need to
      specify '@*@' (though we can).

    dependentWarExcludes
      The comma separated list of tokens to exclude when doing
a WAR overlay.
      Default is Overlay.DEFAULT_EXCLUDES

    dependentWarIncludes
      The comma separated list of tokens to include when doing
a WAR overlay.
      Default is Overlay.DEFAULT_INCLUDES

    escapedBackslashesInFilePath (Default: false)
      To escape interpolated values with Windows path
c:\foo\bar will be
      replaced with c:\\foo\\bar.

    escapeString
      Expression preceded with this String won't be
interpolated. \${foo} will
      be replaced with ${foo}.

    failOnMissingWebXml
      Whether or not to fail the build if the web.xml file is
missing. Set to
      false if you want your WAR built without a web.xml file.
This may be
      useful if you are building an overlay that has no web.xml
file.
      Starting with 3.1.0, this property defaults to false if
the project
      depends on the Servlet 3.0 API or newer.

    filteringDeploymentDescriptors (Default: false)
      To filter deployment descriptors. Disabled by default.

    filters
      Filters (property files) to include during the
interpolation of the
      pom.xml.
```

```
    includeEmptyDirectories (Default: false)
      (no description available)

    nonFilteredFileExtensions
      A list of file extensions that should not be filtered.
Will be used when
      filtering webResources and overlays.

    outputDirectory (Default: ${project.build.directory})
      Required: true
      The directory for the generated WAR.

    outputFileNameMapping
      The file name mapping to use when copying libraries and
TLDs. If no file
      mapping is set (default) the files are copied with their
standard names.

    overlays
      The overlays to apply. Each overlay element may contain:
      - id (defaults to currentBuild)
      - groupId (if this and artifactId are null, then the
current project is
        treated as its own overlay)
      - artifactId (see above)
      - classifier
      - type
      - includes (a list of string patterns)
      - excludes (a list of string patterns)
      - filtered (defaults to false)
      - skip (defaults to false)
      - targetPath (defaults to root of webapp structure)

    packagingExcludes
      The comma separated list of tokens to exclude from the
WAR before
      packaging. This option may be used to implement the
skinny WAR use case.
      Note that you can use the Java Regular Expressions engine
to include and
      exclude specific pattern using the expression %regex[].
Hint: read the
      about (?!Pattern).
```

```
    packagingIncludes
      The comma separated list of tokens to include in the WAR
before
      packaging. By default everything is included. This option
may be used to
      implement the skinny WAR use case. Note that you can use
the Java Regular
      Expressions engine to include and exclude specific
pattern using the
      expression %regex[].

    primaryArtifact (Default: true)
      Whether this is the main artifact being built. Set to
false if you don't
      want to install or deploy it to the local repository
instead of the
      default one in an execution.

    recompressZippedFiles (Default: true)
      Indicates if zip archives (jar,zip etc) being added to
the war should be
      compressed again. Compressing again can result in smaller
archive size,
      but gives noticeably longer execution time.

    resourceEncoding (Default: ${project.build.sourceEncoding})
      The encoding to use when copying filtered web resources.

    skip (Default: false)
      User property: maven.war.skip
      You can skip the execution of the plugin if you need to.
Its use is NOT
      RECOMMENDED, but quite convenient on occasion.

    supportMultiLineFiltering (Default: false)
      Stop searching endToken at the end of line

    useCache (Default: false)
      Whether the cache should be used to save the status of
the webapp across
      multiple runs. Experimental feature so disabled by
default.

    useDefaultDelimiters (Default: true)
      Use default delimiters in addition to custom delimiters,
```

```
if any.

    useJvmChmod (Default: true)
      use jvmChmod rather that cli chmod and forking process

    warSourceDirectory (Default: ${basedir}/src/main/webapp)
      Required: true
      Single directory for extra files to include in the WAR.
This is where you
      place your JSP files.

    warSourceExcludes
      The comma separated list of tokens to exclude when
copying the content of
      the warSourceDirectory.

    warSourceIncludes (Default: **)
      The comma separated list of tokens to include when
copying the content of
      the warSourceDirectory.

    webappDirectory (Default:
    ${project.build.directory}/${project.build.finalName})
      Required: true
      The directory where the webapp is built.

    webResources
      The list of webResources we want to transfer.

    webXml
      The path to the web.xml file to use.

    workDirectory (Default:
${project.build.directory}/war/work)
      Required: true
      Directory to unpack dependent WARs into if needed.


[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time: 0.960 s
[INFO] Finished at: 2017-08-03T14:33:18+08:00
```

```
[INFO] Final Memory: 10M/155M
[INFO] ------------------------------------------------------------
----------------
```

```
MacBook-Pro:api.netkiller.cn neo$ mvn help:describe \
        -DgroupId=org.springframework.boot \
        -DartifactId=spring-boot-maven-plugin \
        -Ddetail=true
```

3.3. settings.xml 配置

Maven 仓库

http://search.maven.org/

http://mvnrepository.com/

本地下载目录

```
$ ls ~/.m2
```

镜像配置

一般镜像

```
        <mirror>
      <id>aliyun</id>
      <name>aliyun maven</name>
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
```

```
        </mirror>
```

带有用户认证的镜像

```
<settings>
    <servers>
        <server>
            <id>netkiller.cn</id>
            <username>netkiller</username>
            <password>password</password>
        </server>
    </servers>
    <mirrors>
        <mirror>
            <id>netkiller.cn</id>
            <name>Netkiller Mirror</name>

<url>http://maven.netkiller.cn/repository/maven2</url>
            <mirrorOf>central</mirrorOf>
        </mirror>
    </mirrors>
</settings>
```

3.4. pom.xml

properties

定义 properties

```
<properties>
    <spring.version>4.0.1.RELEASE</spring.version>
</properties>
```

引用 properties

```
 <!-- Spring dependencies -->
  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>${spring.version}</version>
 </dependency>
```

例 2.1. Maven properties

pom.xml 文件

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

 <modelVersion>4.0.0</modelVersion>
 <groupId>com.javahash.web</groupId>
 <artifactId>Spring4MVCHelloWorld</artifactId>
 <packaging>war</packaging>
 <version>1.0-SNAPSHOT</version>
 <name>Spring4MVCHelloWorld Maven Webapp</name>
 <url>http://maven.apache.org</url>

 <properties>
    <spring.version>4.0.1.RELEASE</spring.version>
 </properties>

 <dependencies>

    <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>4.12</version>
     <scope>test</scope>
    </dependency>
```

```
  <!-- Spring dependencies -->
  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>${spring.version}</version>
 </dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${spring.version}</version>
</dependency>

<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-webmvc</artifactId>
   <version>${spring.version}</version>
</dependency>
</dependencies>

<build>
<finalName>Spring4MVCHelloWorld</finalName>
</build>
</project>
```

java.version

```
       <properties>
        <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        <java.version>1.8</java.version>
   </properties>
```

常用的POM属性

用户可以使用该属性引用POM文件中对应元素的值，常用的POM属性包括：

```
${project.build.sourceDirectory}：项目的主源码目录，默认为
src/main/java
${project.build.testSourceDirectory}：项目的测试源码目录，默认为
src/test/java
${project.build.directory}：项目构件输出目录，默认为 target/
${project.outputDirectory}：项目主代码编译输出目录，默认为
target/classes/
${project.testOutputDirectory}：项目测试代码编译输出目录，默认为
target/test-classes/
${project.groupId}：项目的 groupId
${project.artifactId}：项目的 artifactId
${project.version}：项目的 version，与 project.version：项目的
version，与{version}等价
${project.build.fianlName}：项目打包输出文件的名称。 默 认 为
{project.build.fianlName}：项目打包输出文件的名称。默认为
project.build.fianlName：项目打包输出文件的名称。默认为
{project.artifactId}-${project.version}
```

repositories 仓库配置

默认仓库

```
                <repository>
                        <id>mvnrepository</id>
                        <name>mvnrepository</name>

<url>http://www.mvnrepository.com/</url>
                        <layout>default</layout>
                        <releases>
                                <enabled>true</enabled>
                        </releases>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
```

```
                </repository>
```

阿里云仓库

```
        <repositories>
                <repository>
                        <id>alimaven</id>
                        <name>aliyun maven</name>
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>
                        <releases>
                                <enabled>true</enabled>
                        </releases>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
        </repositories>
```

dependencies

```
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.struts</groupId>
        <artifactId>struts2-core</artifactId>
        <version>2.3.24.1</version>
    </dependency>
  </dependencies>
```

dependencyManagement

声明父项目引用包的版本号。

```
        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
netflix</artifactId>

<version>1.3.5.RELEASE</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
config</artifactId>

<version>1.3.3.RELEASE</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
```

build

finalName

最终 jar 包得名字

```
<build>
                <finalName>crawler</finalName>
</build>
```

sourceDirectory

编译源码目录

```
<sourceDirectory>src</sourceDirectory>
```

resources 文件处理

resources 用来处理 src/main/resources 目录中得内容

```
                <resources>
                        <resource>

<directory>src/main/resources</directory>

<targetPath>${project.build.directory}/conf</targetPath>
                        </resource>
                        <resource>

<directory>src/main/resources</directory>
                                    <excludes>

<exclude>development.properties</exclude>
                                    </excludes>
                        </resource>
                        <resource>

<directory>src/main/resources</directory>
                                    <includes>
```

```
<include>development.properties</include>
                                </includes>

<targetPath>resources</targetPath>
                        </resource>

                        <resource>
                                <directory>lib</directory>
                                <includes>

<include>**/*.sh</include>

<include>**/*.bat</include>
                                </includes>

<targetPath>${project.build.directory}/lib</targetPath>
                        </resource>
                </resources>
```

例 **2.2.** 将本地 **lib/*.jar** 包添加到项目中

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>cn.netkiller</groupId>
        <artifactId>nacos</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>nacos</name>
        <description>Nacos Demo</description>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
```

```
parent</artifactId>
                <version>2.6.2</version>
                <relativePath />
        </parent>
        <dependencies>
                <dependency>
                        <groupId>cn.netkiller</groupId>
                        <artifactId>demo</artifactId>
                        <version>1.0.0</version>
                        <scope>system</scope>

<systemPath>${project.basedir}/lib/demo-1.0.0.jar</systemPath>
                        <exclusions>
                                <exclusion>

<groupId>org.slf4j</groupId>
                                                <artifactId>slf4j-
log4j12</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
        </dependencies>
        <build>
                <resources>
                        <resource>
                                <directory>lib</directory>
                                <targetPath>BOOT-
INF/lib/</targetPath>
                                <includes>

<include>**/*.jar</include>
                                </includes>
                        </resource>
                </resources>
        </build>

</project>
```

例 2.3. 将本地 src/resources 打包到项目

将资源文件编译后复制到 WEB-INF/classes 目录中

```
                <resources>
                        <resource>

<directory>src/resources</directory>
                        </resource>
                </resources>
```

include / exclude

```
<resources>
    <!-- Filter jdbc.properties & mail.properties. NOTE: We
don't filter applicationContext-infrastructure.xml,
        let it go with spring's resource process mechanism. -->
    <resource>
        <directory>src/main/resources</directory>
        <filtering>true</filtering>
        <includes>
            <include>jdbc.properties</include>
            <include>mail.properties</include>
        </includes>
    </resource>
    <!-- Include other files as resources files. -->
    <resource>
        <directory>src/main/resources</directory>
        <filtering>false</filtering>
        <excludes>
            <exclude>jdbc.properties</exclude>
            <exclude>mail.properties</exclude>
        </excludes>
    </resource>
</resources>
```

plugins

跳过Unit test

```
<project>
  [...]
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <configuration>
          <skip>true</skip>
        </configuration>
      </plugin>
    </plugins>
  </build>
  [...]
</project>
```

maven-shade-plugin

```
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-shade-
plugin</artifactId>
                                <version>1.4</version>
                                <executions>
                                        <execution>

<phase>package</phase>
                                                        <goals>

<goal>shade</goal>
                                                        </goals>
                                                        <configuration>

<transformers>

<transformer
implementation="org.apache.maven.plugins.shade.resource.Manifes
```

```
tResourceTransformer">

<mainClass>cn.netkiller.Oracle</mainClass>

</transformer>

</transformers>

</configuration>
                                        </execution>
                                </executions>
                        </plugin>
```

3.5. Maven Module

对于大型互联网项目，不可能把所有代码都放在一个项目目录下，常常会将一个项目拆分成多个子项目

例如一个电商系统

1. 网站
2. 资讯中心
3. 商品中心
4. 物流中心
5. 订单中心
6. 客服中心
7. 后台

在开发过程中常常会遇到这种需求，有一个部分代码是共用的，所有项目都会使用到。所以需要将这部分代码独立成一个项目。

测试目录深度

```
Project
    |
    |--- common -> https://example.com/xxxx/common.git
```

```
    |      | ---pom.xml
    |--- project1
    |      |--- pom.xml
    |--- project2
    |      |--- pom.xml
    |---pom.xml
```

common 是公共项目，独立仓库。通过git submodule 技术挂载到项目目录。project1，project2 构建依赖 common 项目产生的 jar 包。

Parent

例 2.4. Maven parent

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>demo</groupId>
        <artifactId>maven</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>maven</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>

<maven.compiler.source>11</maven.compiler.source>
                <maven.compiler.target>${maven.compiler.source}
</maven.compiler.target>

<junit.jupiter.version>5.4.0</junit.jupiter.version>
        </properties>
```

```
        <dependencies>


        </dependencies>
        <modules>
                <module>project1</module>
                <module>project2</module>
                <module>common</module>
        </modules>
        <build>

                <plugins>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <version>3.0.0-M3</version>
                        </plugin>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.8.0</version>
                        </plugin>

                </plugins>
        </build>
</project>
```

注意

```
<packaging>pom</packaging> 必须是 pom
```

项目下面的子模块

```
    <modules>
                <module>project1</module>
```

```
                <module>project2</module>
                <module>common</module>
    </modules>
```

公共项目 **common**

例 **2.5. watir-webdriver example**

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>demo</groupId>
                <artifactId>maven</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>demo</groupId>
        <artifactId>common</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>common</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>

        </dependencies>
</project>
```

添加 parent 标签, 声明项目的父子关系

```
```

```
    <parent>
                <groupId>demo</groupId>
                <artifactId>maven</artifactId>
                <version>0.0.1-SNAPSHOT</version>
    </parent>
```

常规项目

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>demo</groupId>
                <artifactId>maven</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>demo</groupId>
        <artifactId>project1</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>project1</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>demo</groupId>
                        <artifactId>common</artifactId>
                        <version>0.0.1-SNAPSHOT</version>
                </dependency>
        </dependencies>
</project>
```

声明项目的父子关系

```
    <parent>
                <groupId>demo</groupId>
                <artifactId>maven</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
```

由于 project1 依赖 common 加入下面依赖

```
                <dependency>
                        <groupId>demo</groupId>
                        <artifactId>common</artifactId>
                        <version>0.0.1-SNAPSHOT</version>
                </dependency>
```

project2 跟 project1 类似

现在测试效果

在父项目目录运行 mvn package

```
neo@MacBook-Pro ~/workspace/maven % mvn package
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------
----------------
[INFO] Reactor Build Order:
[INFO]
[INFO] maven
[pom]
[INFO] common
[jar]
```

```
[INFO] project1
[jar]
[INFO] project2
[jar]
[INFO]
[INFO] ----------------------------< demo:maven >------------
----------------
[INFO] Building maven 0.0.1-SNAPSHOT
[1/4]
[INFO] --------------------------------[ pom ]----------------
----------------
[INFO]
[INFO] ---------------------------< demo:common >------------
----------------
[INFO] Building common 0.0.1-SNAPSHOT
[2/4]
[INFO] --------------------------------[ jar ]----------------
----------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-
resources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/common/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-
compile) @ common ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-
testResources) @ common ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/common/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-
testCompile) @ common ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:3.0.0-M3:test (default-test) @
common ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
```

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ common ---
[INFO] Building jar:
/Users/neo/workspace/maven/common/target/common-0.0.1-
SNAPSHOT.jar
[INFO] META-INF/maven/demo/common/pom.xml already added,
skipping
[INFO] META-INF/maven/demo/common/pom.properties already added,
skipping
[INFO]
[INFO] ---------------------------< demo:project1 >------------
----------------
[INFO] Building project1 0.0.1-SNAPSHOT
[3/4]
[INFO] --------------------------------[ jar ]----------------
----------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-
resources) @ project1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/project1/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-
compile) @ project1 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-
testResources) @ project1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/project1/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-
testCompile) @ project1 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to
/Users/neo/workspace/maven/project1/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:3.0.0-M3:test (default-test) @
```

```
project1 ---
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ project1 --
-
[INFO] Building jar:
/Users/neo/workspace/maven/project1/target/project1-0.0.1-
SNAPSHOT.jar
[INFO] META-INF/maven/demo/project1/pom.xml already added,
skipping
[INFO] META-INF/maven/demo/project1/pom.properties already
added, skipping
[INFO]
[INFO] --------------------------< demo:project2 >------------
----------------
[INFO] Building project2 0.0.1-SNAPSHOT
[4/4]
[INFO] --------------------------------[ jar ]----------------
----------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-
resources) @ project2 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/project2/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-
compile) @ project2 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-
testResources) @ project2 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/project2/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-
testCompile) @ project2 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to
/Users/neo/workspace/maven/project2/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:3.0.0-M3:test (default-test) @
project2 ---
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ project2 --
```

```
-
[INFO] Building jar:
/Users/neo/workspace/maven/project2/target/project2-0.0.1-
SNAPSHOT.jar
[INFO] META-INF/maven/demo/project2/pom.xml already added,
skipping
[INFO] META-INF/maven/demo/project2/pom.properties already
added, skipping
[INFO] ------------------------------------------------------------
----------------
[INFO] Reactor Summary for maven 0.0.1-SNAPSHOT:
[INFO]
[INFO] maven ..............................................
SUCCESS [  0.006 s]
[INFO] common .............................................
SUCCESS [  2.317 s]
[INFO] project1 ...........................................
SUCCESS [  0.539 s]
[INFO] project2 ...........................................
SUCCESS [  0.101 s]
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time:  3.115 s
[INFO] Finished at: 2019-02-28T17:03:29+08:00
[INFO] ------------------------------------------------------------
----------------
neo@MacBook-Pro ~/workspace/maven %
```

我们可以看到有三个包产生

```
neo@MacBook-Pro ~/workspace/maven % find . -iname '*.jar'
./project1/target/project1-0.0.1-SNAPSHOT.jar
./common/target/common-0.0.1-SNAPSHOT.jar
./project2/target/project2-0.0.1-SNAPSHOT.jar
```

我们也可以单独编译子项目

```
neo@MacBook-Pro ~/workspace/maven/project1 % mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------------< demo:project1 >------------
---------------
[INFO] Building project1 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]----------------
---------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-
resources) @ project1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/project1/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-
compile) @ project1 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to
/Users/neo/workspace/maven/project1/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-
testResources) @ project1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/Users/neo/workspace/maven/project1/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-
testCompile) @ project1 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to
/Users/neo/workspace/maven/project1/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:3.0.0-M3:test (default-test) @
project1 ---
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ project1 --
-
[INFO] Building jar:
/Users/neo/workspace/maven/project1/target/project1-0.0.1-
```

```
SNAPSHOT.jar
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time:  2.015 s
[INFO] Finished at: 2019-02-28T17:09:18+08:00
[INFO] ------------------------------------------------------------
----------------
```

共享库 common-0.0.1-SNAPSHOT.jar 已经安装到 ~/.m2 目录下。

```
neo@MacBook-Pro ~/workspace/maven/project1 % find ~/.m2 -iname
'common-0.0.1-SNAPSHOT.jar'
/Users/neo/.m2/repository/demo/common/0.0.1-SNAPSHOT/common-
0.0.1-SNAPSHOT.jar
```

3.6. 依赖管理

创建依赖模块

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
```

```
        <artifactId>dependencies</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>dependencies</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.apache.httpcomponents</groupId>
                        <artifactId>httpclient</artifactId>
                        <version>4.5.7</version>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>

</project>
```

引用依赖管理

```
<?xml version="1.0" encoding="UTF-8"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <version>0.0.1-SNAPSHOT</version>
```

```
                <artifactId>parent</artifactId>
        </parent>
        <artifactId>example</artifactId>
        <name>example</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <dependencyManagement>
                <dependencies>
                        <dependency>
                                <groupId>cn.netkiller</groupId>

<artifactId>dependencies</artifactId>

<version>${project.parent.version}</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
oauth2</artifactId>
                </dependency>

        </dependencies>
</project>
```

3.7. plugins

maven-compiler-plugin

配置默认的jdk版本

```
        <properties>
      <maven.compiler.source>1.8</maven.compiler.source>
      <maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.compilerVersion>1.7</maven.compiler.compilerVer
sion>
    </properties>
```

在插件中配置

```
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.8.0</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
```

禁止编译警告 -Xlint:unchecked，-Xlint:deprecation

```
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-
```

```
plugin</artifactId>
                                <version>3.1</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                        <encoding>UTF-
8</encoding>
                                        <compilerArgs>
                                                <arg>-
verbose</arg>
                                                <arg>-
Xlint:unchecked</arg>
                                                <arg>-
Xlint:deprecation</arg>
                                        </compilerArgs>
                                </configuration>
                        </plugin>
```

```
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.1</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                        <encoding>UTF-
8</encoding>
                                        <compilerArgs>
                                                <arg>-
verbose</arg>
                                                <arg>-
Xlint:unchecked</arg>
                                                <arg>-
Xlint:deprecation</arg>
                                                <arg>-
bootclasspath</arg>

<arg>${env.JAVA_HOME}/jre/lib/rt.jar</arg>
                                                <arg>-
```

```
extdirs</arg>

<arg>${project.basedir}/libs</arg>
                                            </compilerArgs>
                                    </configuration>
                            </plugin>
```

compilerArgs可以实现编译参数的传递

```
<plugin>

    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.1</version>
    <configuration>
            <!-- 指定maven编译的jdk版本 -->
        <!-- 一般而言，target与source是保持一致的，但是，有时候为了让程序能在其他版本的jdk中运行(对于低版本目标jdk，源代码中不能使用低版本jdk中不支持的语法)，会存在target不同于source的情况 -->
        <source>1.8</source> <!-- 源代码使用的JDK版本 -->
        <target>1.8</target> <!-- 需要生成的目标class文件的编译版本 -->
        <encoding>UTF-8</encoding><!-- 字符集编码 -->
        <skipTests>true</skipTests><!-- 跳过测试 -->
        <verbose>true</verbose>
        <showWarnings>true</showWarnings>
        <fork>true</fork><!-- 要使compilerVersion标签生效，还需要将fork设为true，用于明确表示编译版本配置的可用 -->
        <executable><!-- path-to-javac --></executable><!-- 使用指定的javac命令，例如：
<executable>${JAVA_1_4_HOME}/bin/javac</executable> -->
        <compilerVersion>1.3</compilerVersion><!-- 指定插件将使用的编译器的版本 -->
        <meminitial>128m</meminitial><!-- 编译器使用的初始内存 -->
        <maxmem>512m</maxmem><!-- 编译器使用的最大内存 -->
        <compilerArgument>-verbose -bootclasspath ${java.home}\lib\rt.jar</compilerArgument><!-- 这个选项用来传递编译器自身不包含但是却支持的参数选项 -->
    </configuration>
</plugin>
```

maven-war-plugin

设置 war 文件名 warName

```
<project>
  ...
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.3</version>
        <configuration>
          <warName>bird.war</warName>
        </configuration>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

maven-antrun-plugin

查看可用的pom定义

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>my-test-app</artifactId>
  <groupId>my-test-group</groupId>
  <version>1.0-SNAPSHOT</version>

  <build>
    <plugins>
```

```
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-antrun-plugin</artifactId>
        <version>1.8</version>
        <executions>
          <execution>
            <id>compile</id>
            <phase>compile</phase>
            <configuration>
              <target>
                <property name="compile_classpath"
refid="maven.compile.classpath"/>
                <property name="runtime_classpath"
refid="maven.runtime.classpath"/>
                <property name="test_classpath"
refid="maven.test.classpath"/>
                <property name="plugin_classpath"
refid="maven.plugin.classpath"/>

                <echo message="compile classpath:
${compile_classpath}"/>
                <echo message="runtime classpath:
${runtime_classpath}"/>
                <echo message="test classpath:
${test_classpath}"/>
                <echo message="plugin classpath:
${plugin_classpath}"/>

                <ant antfile="${basedir}/build.xml">
                  <target name="test"/>
                </ant>
              </target>
            </configuration>
            <goals>
              <goal>run</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>

The build.xml:
```

```
<?xml version="1.0"?>
<project name="test6">

    <target name="test">

      <echo message="compile classpath: ${compile_classpath}"/>
      <echo message="runtime classpath: ${runtime_classpath}"/>
      <echo message="test classpath:    ${test_classpath}"/>
      <echo message="plugin classpath:  ${plugin_classpath}"/>

    </target>

</project>
```

echo 打印调试信息

```
        <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-antrun-plugin</artifactId>
                <version>3.0.0</version>
                <executions>
                        <execution>
                <phase>validate</phase>
                                <goals>
                                        <goal>run</goal>
                                </goals>
                                <configuration>
                                        <target>
                                                <echo>Current
branch: ${branch}</echo>
                                                <echo>
[project.artifactId] ${project.artifactId}</echo>
                                        </target>
                                </configuration>
                        </execution>
                </executions>
        </plugin>
```

```
root@netkiller ~/neo (master)# mvn package -Dbranch=master

[WARNING]      [echo] Current branch: master
[WARNING]      [echo] [project.artifactId] neo
```

maven-install-plugin

```
               <plugin>

<groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-install-
plugin</artifactId>
                        <version>2.5.2</version>
                        <inherited>false</inherited>
                        <executions>
                                <execution>

<id>install:com.oracle:ojdbc6:11g</id>
                                        <phase>validate</phase>
                                        <goals>
                                                <goal>install-
file</goal>
                                        </goals>
                                        <configuration>

<file>${project.basedir}/lib/ojdbc6.jar</file>

<groupId>com.oracle</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0.3</version>

<packaging>jar</packaging>

<createChecksum>true</createChecksum>
```

```
<generatePom>true</generatePom>
                                        </configuration>
                                </execution>
                        </executions>
                </plugin>
```

maven-surefire-plugin

```
        <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.20</version>
                <configuration>
                        <skip>true</skip>
                </configuration>
        </plugin>
```

maven-deploy-plugin

```
                        <plugin>
                      <!--skip deploy (this is just a test
module) -->
                      <artifactId>maven-deploy-
plugin</artifactId>
                      <configuration>
                        <skip>true</skip>
                      </configuration>
                </plugin>
```

deploy:deploy-file

```
mvn deploy:deploy-file
-Dfile=<path-to-file>
-DgroupId=<group-id>
-DartifactId=<artifact-id>
-Dversion=<version>
-Dpackaging=jar
-Durl=file:./maven-repository/
-DrepositoryId=maven-repository
-DupdateReleaseInfo=true
```

```
mvn deploy:deploy-file \
-DrepositoryId=gitlab-maven \
-
Durl=http://registry.netkiller.cn/api/v4/projects/14/packages/m
aven \
-Dpackaging=jar \
-Dfile=lib/cfca.jar \
-DgroupId=cn.netkiller \
-DartifactId=api \
-Dversion=1.0.0 \
-Dmaven.test.skip=true -s .ci_settings.xml
```

maven-jar-plugin

指定jar创建目录，下面配置运行 mvn package 后将在 target 目录下创建一个 project 目录。

```
        <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>2.3.1</version>
                <configuration>
```

```
<outputDirectory>${project.build.directory}/project</outputDire
ctory>
                </configuration>
        </plugin>
```

知道 jar 文件的默认 mainClass

```
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-jar-
plugin</artifactId>
                                <version>2.6</version>
                                <configuration>

<outputDirectory>${project.build.directory}/project/lib</output
Directory>
                                        <archive>
                                                <manifest>

<addClasspath>true</addClasspath>

<classpathPrefix>lib/</classpathPrefix>

<mainClass>cn.netkiller.web.App</mainClass>
                                                </manifest>
                                        </archive>
                                </configuration>
                        </plugin>
```

maven-dependency-plugin

把项目依赖的包复制出来

```
                        <plugin>
```

```
<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-dependency-
plugin</artifactId>
                                <!-- <version>2.10</version> --
>
                                <executions>
                                        <execution>
                                                <id>copy-
dependencies</id>

<phase>package</phase>
                                                <goals>

<goal>copy-dependencies</goal>
                                                </goals>
                                                <configuration>
                                                        <!-- 把
依赖的所有maven jar包拷贝到lib目录中（这样所有的jar包都在lib目录中） --
>

<outputDirectory>${project.build.directory}/project/lib</output
Directory>

<overWriteReleases>false</overWriteReleases>

<overWriteSnapshots>false</overWriteSnapshots>

<overWriteIfNewer>true</overWriteIfNewer>

</configuration>
                                        </execution>
                                </executions>
                        </plugin>
```

```
mvn install dependency:copy-dependencies
mvn install dependency:copy-dependencies -DincludeScope=runtime
-DoutputDirectory=target/lib
```

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <version>3.2.0</version>
            <executions>
                <execution>
                    <id>copy-dependencies</id>
                    <phase>prepare-package</phase>
                    <goals>
                        <goal>copy-dependencies</goal>
                    </goals>
                    <configuration>

<outputDirectory>${project.build.directory}/classes/lib</output
Directory>
                        <includeScope>runtime</includeScope>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

spring-boot-maven-plugin

```
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <configuration>

<mainClass>cn.netkiller.Application</mainClass>
                                </configuration>
                        </plugin>
```

```
MacBook-Pro:api.netkiller.cn neo$ mvn help:describe -
DgroupId=org.springframework.boot                    -
DartifactId=spring-boot-maven-plugin                    -
Ddetail=true
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
----------------
[INFO] Building api 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------
----------------
[INFO]
[INFO] --- maven-help-plugin:2.2:describe (default-cli) @ api -
--
[INFO] org.springframework.boot:spring-boot-maven-
plugin:1.5.6.RELEASE

Name: Spring Boot Maven Plugin
Description: Spring Boot Maven Plugin
Group Id: org.springframework.boot
Artifact Id: spring-boot-maven-plugin
Version: 1.5.6.RELEASE
Goal Prefix: spring-boot

This plugin has 6 goals:

spring-boot:build-info
  Description: Generate a build-info.properties file based the
content of the
    current MavenProject.
  Implementation: org.springframework.boot.maven.BuildInfoMojo
  Language: java
  Bound to phase: generate-resources

  Available parameters:

    additionalProperties
      Additional properties to store in the build-
info.properties. Each entry
      is prefixed by build. in the generated build-
```

```
info.properties.

    outputFile (Default:
    ${project.build.outputDirectory}/META-INF/build-
info.properties)
      The location of the generated build-info.properties.

spring-boot:help
  Description: Display help information on spring-boot-maven-
plugin.
    Call mvn spring-boot:help -Ddetail=true -Dgoal=<goal-name>
to display
    parameter details.
  Implementation: org.springframework.boot.maven.HelpMojo
  Language: java

  Available parameters:

    detail (Default: false)
      User property: detail
      If true, display all settable properties for each goal.

    goal
      User property: goal
      The name of the goal for which to show help. If
unspecified, all goals
      will be displayed.

    indentSize (Default: 2)
      User property: indentSize
      The number of spaces per indentation level, should be
positive.

    lineLength (Default: 80)
      User property: lineLength
      The maximum length of a display line, should be positive.

spring-boot:repackage
  Description: Repackages existing JAR and WAR archives so that
they can be
    executed from the command line using java -jar. With
layout=NONE can also
    be used simply to package a JAR with nested dependencies
(and no main
    class, so not executable).
```

```
  Implementation: org.springframework.boot.maven.RepackageMojo
  Language: java
  Bound to phase: package

  Available parameters:

    attach (Default: true)
      Attach the repackaged archive to be installed and
deployed.

    classifier
      Classifier to add to the artifact generated. If given,
the artifact will
      be attached with that classifier and the main artifact
will be deployed
      as the main artifact. If this is not given (default), it
will replace the
      main artifact and only the repackaged artifact will be
deployed.
      Attaching the artifact allows to deploy it alongside to
the original one,
      see the maven documentation for more details.

    embeddedLaunchScript
      The embedded launch script to prepend to the front of the
jar if it is
      fully executable. If not specified the 'Spring Boot'
default script will
      be used.

    embeddedLaunchScriptProperties
      Properties that should be expanded in the embedded launch
script.

    excludeArtifactIds
      User property: excludeArtifactIds
      Comma separated list of artifact names to exclude (exact
match).

    excludeDevtools (Default: true)
      Exclude Spring Boot devtools from the repackaged archive.

    excludeGroupIds
      User property: excludeGroupIds
      Comma separated list of groupId names to exclude (exact
```

```
match).

    excludes
      Collection of artifact definitions to exclude. The
Exclude element
      defines a groupId and artifactId mandatory properties and
an optional
      classifier property.

    executable (Default: false)
      Make a fully executable jar for *nix machines by
prepending a launch
      script to the jar.
      Currently, some tools do not accept this format so you
may not always be
      able to use this technique. For example, jar -xf may
silently fail to
      extract a jar or war that has been made fully-executable.
It is
      recommended that you only enable this option if you
intend to execute it
      directly, rather than running it with java -jar or
deploying it to a
      servlet container.

    finalName (Default: ${project.build.finalName})
      Required: true
      Name of the generated archive.

    includes
      Collection of artifact definitions to include. The
Include element
      defines a groupId and artifactId mandatory properties and
an optional
      classifier property.

    includeSystemScope (Default: false)
      Include system scoped dependencies.

    layout
      The type of archive (which corresponds to how the
dependencies are laid
      out inside it). Possible values are JAR, WAR, ZIP, DIR,
NONE. Defaults to
      a guess based on the archive type.
```

```
    layoutFactory
      The layout factory that will be used to create the
executable archive if
      no explicit layout is set. Alternative layouts
implementations can be
      provided by 3rd parties.

    mainClass
      The name of the main class. If not specified the first
compiled class
      found that contains a 'main' method will be used.

    outputDirectory (Default: ${project.build.directory})
      Required: true
      Directory containing the generated archive.

    requiresUnpack
      A list of the libraries that must be unpacked from fat
jars in order to
      run. Specify each library as a <dependency> with a
<groupId> and a
      <artifactId> and they will be unpacked at runtime.

    skip (Default: false)
      User property: skip
      Skip the execution.

spring-boot:run
  Description: Run an executable archive application.
  Implementation: org.springframework.boot.maven.RunMojo
  Language: java
  Bound to phase: validate
  Before this mojo executes, it will call:
    Phase: 'test-compile'

  Available parameters:

    addResources (Default: false)
      User property: run.addResources
      Add maven resources to the classpath directly, this
allows live in-place
      editing of resources. Duplicate resources are removed
from target/classes
      to prevent them to appear twice if
```

```
ClassLoader.getResources() is called.
      Please consider adding spring-boot-devtools to your
project instead as it
      provides this feature and many more.

    agent
      User property: run.agent
      Path to agent jar. NOTE: the use of agents means that
processes will be
      started by forking a new JVM.

    arguments
      User property: run.arguments
      Arguments that should be passed to the application. On
command line use
      commas to separate multiple arguments.

    classesDirectory (Default:
${project.build.outputDirectory})
      Required: true
      Directory containing the classes and resource files that
should be
      packaged into the archive.

    excludeArtifactIds
      User property: excludeArtifactIds
      Comma separated list of artifact names to exclude (exact
match).

    excludeGroupIds
      User property: excludeGroupIds
      Comma separated list of groupId names to exclude (exact
match).

    excludes
      Collection of artifact definitions to exclude. The
Exclude element
      defines a groupId and artifactId mandatory properties and
an optional
      classifier property.

    folders
      Additional folders besides the classes directory that
should be added to
      the classpath.
```

```
    fork
      User property: fork
      Flag to indicate if the run processes should be forked.
fork is
      automatically enabled if an agent, jvmArguments or
working directory are
      specified, or if devtools is present.

    includes
      Collection of artifact definitions to include. The
Include element
      defines a groupId and artifactId mandatory properties and
an optional
      classifier property.

    jvmArguments
      User property: run.jvmArguments
      JVM arguments that should be associated with the forked
process used to
      run the application. On command line, make sure to wrap
multiple values
      between quotes. NOTE: the use of JVM arguments means that
processes will
      be started by forking a new JVM.

    mainClass
      The name of the main class. If not specified the first
compiled class
      found that contains a 'main' method will be used.

    noverify
      User property: run.noverify
      Flag to say that the agent requires -noverify.

    profiles
      User property: run.profiles
      The spring profiles to activate. Convenience shortcut of
specifying the
      'spring.profiles.active' argument. On command line use
commas to separate
      multiple profiles.

    skip (Default: false)
      User property: skip
```

```
      Skip the execution.

    useTestClasspath (Default: false)
      User property: useTestClasspath
      Flag to include the test classpath when running.

    workingDirectory
      User property: run.workingDirectory
      Current working directory to use for the application. If
not specified,
      basedir will be used. NOTE: the use of working directory
means that
      processes will be started by forking a new JVM.

spring-boot:start
  Description: Start a spring application. Contrary to the run
goal, this
    does not block and allows other goal to operate on the
application. This
    goal is typically used in integration test scenario where
the application
    is started before a test suite and stopped after.
  Implementation: org.springframework.boot.maven.StartMojo
  Language: java
  Bound to phase: pre-integration-test

  Available parameters:

    addResources (Default: false)
      User property: run.addResources
      Add maven resources to the classpath directly, this
allows live in-place
      editing of resources. Duplicate resources are removed
from target/classes
      to prevent them to appear twice if
ClassLoader.getResources() is called.
      Please consider adding spring-boot-devtools to your
project instead as it
      provides this feature and many more.

    agent
      User property: run.agent
      Path to agent jar. NOTE: the use of agents means that
processes will be
      started by forking a new JVM.
```

```
    arguments
      User property: run.arguments
      Arguments that should be passed to the application. On
command line use
      commas to separate multiple arguments.

    classesDirectory (Default:
${project.build.outputDirectory})
      Required: true
      Directory containing the classes and resource files that
should be
      packaged into the archive.

    excludeArtifactIds
      User property: excludeArtifactIds
      Comma separated list of artifact names to exclude (exact
match).

    excludeGroupIds
      User property: excludeGroupIds
      Comma separated list of groupId names to exclude (exact
match).

    excludes
      Collection of artifact definitions to exclude. The
Exclude element
      defines a groupId and artifactId mandatory properties and
an optional
      classifier property.

    folders
      Additional folders besides the classes directory that
should be added to
      the classpath.

    fork
      User property: fork
      Flag to indicate if the run processes should be forked.
fork is
      automatically enabled if an agent, jvmArguments or
working directory are
      specified, or if devtools is present.

    includes
```

```
      Collection of artifact definitions to include. The
Include element
      defines a groupId and artifactId mandatory properties and
an optional
      classifier property.

    jmxName
      The JMX name of the automatically deployed MBean managing
the lifecycle
      of the spring application.

    jmxPort
      The port to use to expose the platform MBeanServer if the
application
      needs to be forked.

    jvmArguments
      User property: run.jvmArguments
      JVM arguments that should be associated with the forked
process used to
      run the application. On command line, make sure to wrap
multiple values
      between quotes. NOTE: the use of JVM arguments means that
processes will
      be started by forking a new JVM.

    mainClass
      The name of the main class. If not specified the first
compiled class
      found that contains a 'main' method will be used.

    maxAttempts
      The maximum number of attempts to check if the spring
application is
      ready. Combined with the 'wait' argument, this gives a
global timeout
      value (30 sec by default)

    noverify
      User property: run.noverify
      Flag to say that the agent requires -noverify.

    profiles
      User property: run.profiles
      The spring profiles to activate. Convenience shortcut of
```

```
specifying the
      'spring.profiles.active' argument. On command line use
commas to separate
      multiple profiles.

    skip (Default: false)
      User property: skip
      Skip the execution.

    useTestClasspath (Default: false)
      User property: useTestClasspath
      Flag to include the test classpath when running.

    wait
      The number of milli-seconds to wait between each attempt
to check if the
      spring application is ready.

    workingDirectory
      User property: run.workingDirectory
      Current working directory to use for the application. If
not specified,
      basedir will be used. NOTE: the use of working directory
means that
      processes will be started by forking a new JVM.

spring-boot:stop
  Description: Stop a spring application that has been started
by the 'start'
    goal. Typically invoked once a test suite has completed.
  Implementation: org.springframework.boot.maven.StopMojo
  Language: java
  Bound to phase: post-integration-test

  Available parameters:

    fork
      User property: fork
      Flag to indicate if process to stop was forked. By
default, the value is
      inherited from the MavenProject. If it is set, it must
match the value
      used to StartMojo start the process.

    jmxName
```

```
      The JMX name of the automatically deployed MBean managing
the lifecycle
      of the application.

    jmxPort
      The port to use to lookup the platform MBeanServer if the
application has
      been forked.

    skip (Default: false)
      User property: skip
      Skip the execution.


[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time: 0.976 s
[INFO] Finished at: 2017-08-03T15:05:53+08:00
[INFO] Final Memory: 12M/155M
[INFO] ------------------------------------------------------------
----------------
```

tomcat8-maven-plugin

```
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.5.1</version>
                        </plugin>
                        <plugin>

<groupId>org.apache.tomcat.maven</groupId>
                                <artifactId>tomcat8-maven-
plugin</artifactId>
```

```
                                <version>2.2</version>
                                <configuration>

<url>http://localhost:8082/manager/text</url>
                                        <server>tomcat</server>

<username>admin</username>

<password>admin@123456</password>

<path>/${project.build.finalName}</path>
                                        <!-- war文件路径缺省情况下
指向target -->
                                        <!--
<warFile>${basedir}/target/${project.build.finalName}.war</warF
ile> -->
                                </configuration>
                        </plugin>
```

```
mvn tomcat:deploy
```

docker-maven-plugin

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>alertmanager</artifactId>
        <version>0.0.1</version>
        <packaging>jar</packaging>

        <name>alertmanager</name>
```

```
        <url>http://www.netkiller.cn</url>
        <description>Send SMS for Alertmanager</description>

        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller 手札
</organization>

<organizationUrl>http://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
                </developer>
        </developers>

        <repositories>
                <repository>
                        <id>alimaven</id>
                        <name>Maven Aliyun Mirror</name>

<url>http://maven.aliyun.com/nexus/content/repositories/central
/</url>
                        <releases>
                                <enabled>true</enabled>
                        </releases>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
        </repositories>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.5.4</version>
                <relativePath />
        </parent>

        <properties>
                <java.version>17</java.version>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
```

```
        </properties>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
webflux</artifactId>
                </dependency>
                <dependency>
                        <groupId>com.aliyun</groupId>
                        <artifactId>aliyun-java-sdk-
core</artifactId>
                        <version>4.5.25</version>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <optional>true</optional>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <configuration>

<mainClass>cn.netkiller.alertmanager.Application</mainClass>
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.2</version>
```

```
                                <configuration>

<baseImage>openjdk</baseImage>

<imageName>netkiller/${project.artifactId}:${project.version}
</imageName>
                                        <imageTags>

<imageTag>${project.version}</imageTag>

<imageTag>latest</imageTag>
                                        </imageTags>
                                        <volumes>/tmp</volumes>

<workdir>/app/</workdir>
                                        <!-- <cmd>["java", "-
version"]</cmd> -->
                                        <entryPoint>["java", "-
Djava.security.egd=file:/dev/./urandom", "-jar",
"/app/${project.build.finalName}.jar"]</entryPoint>

<pushImage>false</pushImage>
                                        <resources>
                                                <resource>

<targetPath>/app</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>
                                </configuration>
                        </plugin>
                </plugins>
        </build>

</project>
```

3.8. 应用案例

并行开发解决不同环境包引用

Maven 父文件中配置

```
        <properties>
                <project.branch>-SNAPSHOT</project.branch>
        </properties>
```

模块中配置

```
        <version>1.0.0-${project.branch}</version>
```

针对不同环境打包，部署

```
root@netkiller ~/neo (maven)# mvn deploy -Dproject.branch=dev
root@netkiller ~/neo (maven)# mvn deploy -Dproject.branch=test
root@netkiller ~/neo (maven)# mvn deploy -Dproject.branch=main
```

依赖引用

```
        <dependency>
            <groupId>cn.netkiller</groupId>
            <artifactId>gantt</artifactId>
            <version>1.0.0-${project.branch}</version>
        </dependency>
```

4. Gradle 5

4.1. 安装 Gradle

CentOS

安装 Gradle

```
curl -s
https://raw.githubusercontent.com/oscm/shell/38c2d4d307ce7a0760
fc88d5d9703eef64d3b81c/lang/java/gradle/gradle-2.9.sh | bash
```

安装后运行 gradle --help 验证释放正确安装。

```
neo@MacBook-Pro ~ % gradle --help

Welcome to Gradle 5.1.1!

Here are the highlights of this release:
 - Control which dependencies can be retrieved from which
repositories
 - Production-ready configuration avoidance APIs

For more details see https://docs.gradle.org/5.1.1/release-
notes.html


USAGE: gradle [option...] [task...]

-?, -h, --help            Shows this help message.
-a, --no-rebuild          Do not rebuild project dependencies.
-b, --build-file          Specify the build file.
--build-cache             Enables the Gradle build cache.
Gradle will try to reuse outputs from previous builds.
```

```
-c, --settings-file       Specify the settings file.
--configure-on-demand     Configure necessary projects only.
Gradle will attempt to reduce configuration time for large
multi-project builds. [incubating]
--console                 Specifies which type of console
output to generate. Values are 'plain', 'auto' (default),
'rich' or 'verbose'.
--continue                Continue task execution after a task
failure.
-D, --system-prop         Set system property of the JVM (e.g.
-Dmyprop=myvalue).
-d, --debug               Log in debug mode (includes normal
stacktrace).
--daemon                  Uses the Gradle Daemon to run the
build. Starts the Daemon if not running.
--foreground              Starts the Gradle Daemon in the
foreground.
-g, --gradle-user-home    Specifies the gradle user home
directory.
-I, --init-script         Specify an initialization script.
-i, --info                Set log level to info.
--include-build           Include the specified build in the
composite.
-m, --dry-run             Run the builds with all task actions
disabled.
--max-workers             Configure the number of concurrent
workers Gradle is allowed to use.
--no-build-cache          Disables the Gradle build cache.
--no-configure-on-demand  Disables the use of configuration on
demand. [incubating]
--no-daemon               Do not use the Gradle daemon to run
the build. Useful occasionally if you have configured Gradle to
always run with the daemon by default.
--no-parallel             Disables parallel execution to build
projects.
--no-scan                 Disables the creation of a build
scan. For more information about build scans, please visit
https://gradle.com/build-scans.
--offline                 Execute the build without accessing
network resources.
-P, --project-prop        Set project property for the build
script (e.g. -Pmyprop=myvalue).
-p, --project-dir         Specifies the start directory for
Gradle. Defaults to current directory.
--parallel                Build projects in parallel. Gradle
```

```
will attempt to determine the optimal number of executor
threads to use.
--priority                Specifies the scheduling priority for
the Gradle daemon and all processes launched by it. Values are
'normal' (default) or 'low' [incubating]
--profile                 Profile build execution time and
generates a report in the <build_dir>/reports/profile
directory.
--project-cache-dir       Specify the project-specific cache
directory. Defaults to .gradle in the root project directory.
-q, --quiet               Log errors only.
--refresh-dependencies    Refresh the state of dependencies.
--rerun-tasks             Ignore previously cached task
results.
-S, --full-stacktrace     Print out the full (very verbose)
stacktrace for all exceptions.
-s, --stacktrace          Print out the stacktrace for all
exceptions.
--scan                    Creates a build scan. Gradle will
emit a warning if the build scan plugin has not been applied.
(https://gradle.com/build-scans)
--status                  Shows status of running and recently
stopped Gradle Daemon(s).
--stop                    Stops the Gradle Daemon if it is
running.
-t, --continuous          Enables continuous build. Gradle does
not exit and will re-execute tasks when task file inputs
change.
--update-locks            Perform a partial update of the
dependency lock, letting passed in module notations change
version. [incubating]
-v, --version             Print version info.
-w, --warn                Set log level to warn.
--warning-mode            Specifies which mode of warnings to
generate. Values are 'all', 'summary'(default) or 'none'
--write-locks             Persists dependency resolution for
locked configurations, ignoring existing locking information if
it exists [incubating]
-x, --exclude-task        Specify a task to be excluded from
execution.
```

Mac

```
neo@MacBook-Pro ~ %  brew install gradle
```

```
neo@MacBook-Pro-M2 test % gradle

ERROR: JAVA_HOME is set to an invalid directory:
@@HOMEBREW_JAVA@@

Please set the JAVA_HOME variable in your environment to match
the
location of your Java installation.

neo@MacBook-Pro-M2 test % export
JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk-
21.jdk/Contents/Home
```

4.2. Example

```
mkdir -p src/main/java/hello

vim src/main/java/hello/HelloWorld.java
```

```
package hello;

public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Helloworld!!!");
  }
}
```

```
$ gradle build
$ gradle run -q
Helloworld!!!
```

4.3. gradle 命令

tasks 列出任务

```
[neo@netkiller test]$ gradle tasks
:tasks

------------------------------------------------------------
All tasks runnable from root project
------------------------------------------------------------

Application tasks
-----------------
installApp - Installs the project as a JVM application along
with libs and OS specific scripts.
run - Runs this project as a JVM application

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all
projects that depend on it.
buildNeeded - Assembles and tests this project and all projects
it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles test classes.

Build Setup tasks
-----------------
init - Initializes a new Gradle build. [incubating]
```

```
Distribution tasks
------------------
assembleDist - Assembles the main distributions
distTar - Bundles the project as a distribution.
distZip - Bundles the project as a distribution.
installDist - Installs the project as a distribution as-is.

Documentation tasks
-------------------
javadoc - Generates Javadoc API documentation for the main
source code.

Help tasks
----------
components - Displays the components produced by root project
'test'. [incubating]
dependencies - Displays all dependencies declared in root
project 'test'.
dependencyInsight - Displays the insight into a specific
dependency in root project 'test'.
help - Displays a help message.
model - Displays the configuration model of root project
'test'. [incubating]
projects - Displays the sub-projects of root project 'test'.
properties - Displays the properties of root project 'test'.
tasks - Displays the tasks runnable from root project 'test'.

Verification tasks
------------------
check - Runs all checks.
test - Runs the unit tests.

Other tasks
-----------
wrapper

Rules
-----
Pattern: clean<TaskName>: Cleans the output files of a task.
Pattern: build<ConfigurationName>: Assembles the artifacts of a
configuration.
Pattern: upload<ConfigurationName>: Assembles and uploads the
artifacts belonging to a configuration.
```

```
To see all tasks and more detail, run gradle tasks --all

To see more detail about a task, run gradle help --task <task>

BUILD SUCCESSFUL

Total time: 5.157 secs
```

4.4. build.gradle

apply plugin

```
apply plugin: 'java'
```

repositories

```
repositories {
    mavenCentral()
}
```

配置阿里云仓库

```
allprojects {
    repositories {
        maven{ url
'http://maven.aliyun.com/nexus/content/groups/public/'}
    }
}
```

dependencies

```
        dependencies {
                compile 'org.springframework:spring-
context:4.2.2.RELEASE'
        }
```

jar

```
        jar {
                baseName = 'hello'
                version = '0.1.0'
        }
```

设置 Main-Class

```
jar {
    manifest {
        attributes 'Main-Class': 'demo.Test'
        attributes 'Class-Path': 'junit5.jar'
    }
}
```

Task

```
task Hello {
    doFirst {
```

```
        // 调用 Hello 任务时 , 先调用该闭包内容
        println 'doFirst1'
    }
    doFirst {
        // 调用 Hello 任务时 , 先调用该闭包内容
        println 'doFirst2'
    }

    // 任务主体内容
    println 'Hello World!'

    doLast {
        // 调用 Hello 任务结束时 , 最后调用该闭包内容
        println 'doLast1'
    }
    doLast {
        // 调用 Hello 任务结束时 , 最后调用该闭包内容
        println 'doLast2'
    }
}
```

```
neo@MacBook-Pro-M2 test % gradle Hello

> Configure project :
Hello World!

> Task :Hello
doFirst2
doFirst1
doLast1
doLast2

BUILD SUCCESSFUL in 444ms
1 actionable task: 1 executed
neo@MacBook-Pro-M2 test %
```

4.5. gradle.properties

列出 properties

```
[neo@netkiller gradle]$ gradle properties
:properties

------------------------------------------------------------
Root project
------------------------------------------------------------

allprojects: [root project 'gradle']
ant: org.gradle.api.internal.project.DefaultAntBuilder@12072edc
antBuilderFactory:
org.gradle.api.internal.project.DefaultAntBuilderFactory@159576
c3
artifacts:
org.gradle.api.internal.artifacts.dsl.DefaultArtifactHandler_De
corated@7a80747
asDynamicObject:
org.gradle.api.internal.ExtensibleDynamicObject@2875ca3e
baseClassLoaderScope:
org.gradle.api.internal.initialization.DefaultClassLoaderScope@
4d30c132
buildDir: /opt/www/gradle/build
buildFile: /opt/www/gradle/build.gradle
buildScriptSource:
org.gradle.groovy.scripts.UriScriptSource@3bdbe135
buildscript:
org.gradle.api.internal.initialization.DefaultScriptHandler@609
e7d46
childProjects: {}
class: class
org.gradle.api.internal.project.DefaultProject_Decorated
classLoaderScope:
org.gradle.api.internal.initialization.DefaultClassLoaderScope@
4532b038
clone: task ':clone'
compile: task ':compile'
compileTest: task ':compileTest'
components: []
configurationActions:
org.gradle.configuration.project.DefaultProjectConfigurationAct
ionContainer@2cf5006
```

```
configurations: []
convention:
org.gradle.api.internal.plugins.DefaultConvention@788ebb5a
defaultTasks: []
deferredProjectConfiguration:
org.gradle.api.internal.project.DeferredProjectConfiguration@62
ae4f8b
dependencies:
org.gradle.api.internal.artifacts.dsl.dependencies.DefaultDepen
dencyHandler_Decorated@21e8614a
depth: 0
description: null
dist: task ':dist'
ext:
org.gradle.api.internal.plugins.DefaultExtraPropertiesExtension
@1f4b52aa
extensions:
org.gradle.api.internal.plugins.DefaultConvention@788ebb5a
fileOperations:
org.gradle.api.internal.file.DefaultFileOperations@a2026f3
fileResolver:
org.gradle.api.internal.file.BaseDirFileResolver@44dd20b6
gradle: build 'gradle'
group:
inheritedScope:
org.gradle.api.internal.ExtensibleDynamicObject$InheritedDynami
cObject@118eb00c
logger:
org.gradle.logging.internal.slf4j.OutputEventListenerBackedLogg
er@2ec7ecd5
logging:
org.gradle.logging.internal.DefaultLoggingManager@478dabf1
modelRegistry:
org.gradle.model.internal.registry.DefaultModelRegistry@26137fe
a
modelSchemaStore:
org.gradle.model.internal.manage.schema.extract.DefaultModelSch
emaStore@4a32ef2d
module:
org.gradle.api.internal.artifacts.ProjectBackedModule@31bca1c3
name: gradle
parent: null
parentIdentifier: null
path: :
pluginManager:
```

```
org.gradle.api.internal.plugins.DefaultPluginManager_Decorated@
55f49969
plugins: [org.gradle.api.plugins.HelpTasksPlugin@4f88f506]
processOperations:
org.gradle.api.internal.file.DefaultFileOperations@a2026f3
project: root project 'gradle'
project.dir: repo
project.url: git@172.16.0.1:example.com/admin.example.com.git
projectDir: /opt/www/gradle
projectEvaluationBroadcaster: ProjectEvaluationListener
broadcast
projectEvaluator:
org.gradle.configuration.project.LifecycleProjectEvaluator@1903
5ff9
projectRegistry:
org.gradle.api.internal.project.DefaultProjectRegistry@2c91e143
properties: {...}
pull: task ':pull'
repositories: []
resources:
org.gradle.api.internal.resources.DefaultResourceHandler@1d5c0c
91
rootDir: /opt/www/gradle
rootProject: root project 'gradle'
scriptHandlerFactory:
org.gradle.api.internal.initialization.DefaultScriptHandlerFact
ory@63d12a6
scriptPluginFactory:
org.gradle.configuration.DefaultScriptPluginFactory@1393537d
serviceRegistryFactory:
org.gradle.internal.service.scopes.ProjectScopeServices$4@2d4e3
d95
services: ProjectScopeServices
standardOutputCapture:
org.gradle.logging.internal.DefaultLoggingManager@478dabf1
state: project state 'EXECUTED'
status: release
stop: task ':stop'
subprojects: []
tasks: [task ':clone', task ':compile', task ':compileTest',
task ':dist', task ':properties', task ':pull', task ':stop',
task ':test']
test: task ':test'
version:
unspecified
```

```
BUILD SUCCESSFUL

Total time: 4.672 secs
```

自定义 gradle.properties

```
[neo@netkiller gradle]$ cat gradle.properties
Name=Netkiller
Email=netkiller@msn.com
```

```
[neo@netkiller gradle]$ gradle properties | egrep "Name|Email"
Email: netkiller@msn.com
Name: Netkiller
```

```
[neo@netkiller gradle]$ cat build.gradle
task hello << {
     println "hello, $Name<$Email>"
}
```

```
[neo@netkiller gradle]$ gradle hello -q
hello, Netkiller<netkiller@msn.com>
```

通过 systemProp 前缀传递 System.properties 参数

```
[neo@netkiller gradle]$ cat gradle.properties
systemProp.Name = 'Neo Chen'

[neo@netkiller gradle]$ cat build.gradle

task hello << {
      println "hello, " + System.properties['Name']
}
```

```
ext {
     Name='Neo'
}

task hello << {
     println "hello, $Name"
}
```

System.properties

-D 参数传递

```
task hello << {
     println System.properties['Name']
}

$ gradle hello -DName='Neo' -q
Neo
```

-P 参数传递

```
task hello << {
      println "hello, $Name"
}

$ gradle hello -PName='Neo' -q
hello, Neo
```

5. JitPack - Easy to use package repository for Git

```
JitPack

   Easy to use package repository for Git

   Publish your JVM and Android libraries
```

6. Artifactory

6.1. Artifactory Web UI

通过访问 http://localhost:8081/artifactory/ ，用默认管理员账号密码 (用户名：admin，密码：password) 登录，进入管理界面体验 Artifactory。

6.2. build.gradle

```
buildscript {
    repositories {

        maven {
            url
"http://artifactory.netkiller.cn:8081/artifactory/libs-release-
local"
        }

        jcenter()
    }
    dependencies {
        ...
    }
}

allprojects {
    repositories {
        maven {
            url
"http://artifactory.netkiller.cn:8081/artifactory/libs-release-
local"
```

```
        }

        jcenter()
    }
}
```

Artifactory 默认将 jecnter 库缓存到私有 maven仓库中。我们可以在项目中配置 jcenter 仓库信息列表，后续编译所需要的包都直接从私服 maven仓库读取，加快项目编译速度。

```
buildscript {
    repositories {
        maven {
            url
"http://artifactory.netkiller.cn:8081/artifactory/jcenter" //
配置私服jcenter仓库信息
        }

        maven {
            url
"http://artifactory.netkiller.cn:8081/artifactory/libs-release-
local"
        }

        jcenter()
    }
    dependencies {
       ...
    }
}

allprojects {
    repositories {
        maven {
            url
"http://artifactory.netkiller.cn:8081/artifactory/jcenter" //
配置私服jcenter仓库信息
        }

        maven {
```

```
            url
"http://artifactory.netkiller.cn:8081/artifactory/libs-release-
local"
        }

        jcenter()
    }
}
```

第 3 章 Servlet

1. Example

```
package cn.netkiller.helloworld;

import java.io.IOException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebInitParam;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class HelloWorld
 */
@WebServlet("/HelloWorld")
public final class HelloWorld extends HttpServlet {
        private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloWorld() {
        super();
        // TODO Auto-generated constructor stub
    }

        /**
         * @see Servlet#init(ServletConfig)
         */
        public void init(ServletConfig config) throws
ServletException {
                // TODO Auto-generated method stub
        }

        /**
```

```
         * @see Servlet#destroy()
         */
        public void destroy() {
                // TODO Auto-generated method stub
        }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request,
HttpServletResponse response)
         */
        protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
                // TODO Auto-generated method stub
                response.getWriter().append("Served at:
").append(request.getContextPath()).append("<br></br>
Helloworld");
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest
request, HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
                // TODO Auto-generated method stub
                doGet(request, response);
        }

}
```

2. Session

web.xml 定义默认过期时间

```
<web-app ...>
        <session-config>
            <session-timeout>Minutes</session-timeout>
        </session-config>
</web-app>
```

修改Session时间

```
HttpSession session = request.getSession();
session.setMaxInactiveInterval(20*60);
```

3. HttpServletRequest

request.getParameter() 获取form参数

```
HttpServletRequest request =
ServletActionContext.getRequest();
String name = request.getParameter("name");
String age = request.getParameter("age");
```

4. Filter

4.1. web.xml

配置过滤器

```
<filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>cn.netkiller.Filter</filter-class>
    <init-param>
        <param-name>username</param-name>
        <param-value>netkiller</param-value>
    </init-param>
</filter>


<filter>
      <filter-name>LoginFilter</filter-name>
      <filter-class>cn.netkiller.Filter</filter-class>
</filter>
<filter-mapping>
      <filter-name>LoginFilter</filter-name>
      <url-pattern>/*</url-pattern>
</filter-mapping>
```

4.2. Filter 类

实现 Filter 接口

```
public class LoginFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws
```

```
ServletException {
        System.out.println("init LoginFilter");
        //获取Filter初始化参数
        String username =
filterConfig.getInitParameter("username");
    }

    @Override
    public void doFilter(ServletRequest request,
ServletResponse response,
            FilterChain chain) throws IOException,
ServletException {
        //把ServletRequest和ServletResponse转换成真正的类型
        HttpServletRequest req = (HttpServletRequest)request;
        HttpSession session = req.getSession();

        //由于web.xml中设置Filter过滤全部请求，可以排除不需要过滤的
url
        String requestURI = req.getRequestURI();
        if(requestURI.endsWith("login.jsp")){
            chain.doFilter(request, response);
            return;
        }

        //判断用户是否登录，进行页面的处理
        if(null == session.getAttribute("user")){
            //未登录用户，重定向到登录页面

((HttpServletResponse)response).sendRedirect("login.jsp");
            return;
        } else {
            //已登录用户，允许访问
            chain.doFilter(request, response);
        }
    }

    @Override
    public void destroy() {
        System.out.println("destroy!!!");
    }
}
```

5. Listener

5.1. web.xml

配置监听器

```
        <listener>
                <listener-
class>cn.netkiller.listener.NewsListener</listener-class>
        </listener>
```

5.2. NewsListener 类

实现 ServletContextListener 接口

```
package cn.netkiller.listener;

import java.util.Timer;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.log4j.Logger;

public class NewsListener implements ServletContextListener {

        private static final Logger log =
Logger.getLogger(NewsListener.class);

        private Timer timer = null;

        @Override
        public void contextInitialized(ServletContextEvent
```

```
event) {

                log.info("Listener start");

                timer = new Timer(true);
                timer.schedule(new
NewsTask(event.getServletContext()), 3*1000, 5*60*1000);
        }

        @Override
        public void contextDestroyed(ServletContextEvent
event) {
                if (timer != null) {
                        timer.cancel();
                }
                log.info("Listener end");
        }
}
```

5.3. NewsTask 类

继承 TimerTask

```
package cn.netkiller.listener;

import java.util.List;
import java.util.TimerTask;
import javax.servlet.ServletContext;

import org.apache.log4j.Logger;
import
org.springframework.web.context.support.WebApplicationContext
Utils;

import cn.netkiller.service.interface.NewsService;

public class NewsTask extends TimerTask{

        private ServletContext context;
```

```
        private static final Logger log =
Logger.getLogger(NewsTask.class);

        public NewsTask(ServletContext context) {
                this.context = context;
        }

        @Override
        public void run() {
                NewsService newsService = (NewsService)
WebApplicationContextUtils.getWebApplicationContext(context).
getBean("newsService");

                try {
                        List<cn.netkiller.listener.News>
newsList = newsService.getNews();
                        context.setAttribute("newsList",
newsList);

                        log.info("Getting News Finished");
                } catch (Exception e) { e.printStackTrace();
}
        }
}
```

5.4. JSP 中心显示

使用c:forEach显示列表

```
        <div class="news">
        <c:forEach items="${newsList}" var="news"
varStatus="index">
                <a href="/news/${news.Id}">${news.title}</a>
        </c:forEach>
        </div>
```

6. JSP

6.1. 注释

JSP页面中的注释有以下几种样式的注释方法:

```
JSP页面中的HTML注释
     <!-- 注释内容 -->

JSP页面中的普通注释,不会再浏览器中输出

    <% // 注释内容 %>
    <% /* 注释内容 */ %>

JSP页面中的隐藏注释,不会再浏览器中输出
<%-- 注释内容 --%>
```

6.2. pageContext

queryString

输出? 问号后面的变量

```
${pageContext.request.queryString}
<c:out value = "${pageContext.request.queryString}" />
```

6.3. request

```
new URL(
        request.getScheme(),
    request.getServerName(),
    request.getServerPort(),
    request.getContextPath()
    );
```

获取主机名

```
<%=request.getServerName() %>
```

Form

```
<%
   Enumeration paramNames = request.getParameterNames();

   while(paramNames.hasMoreElements()) {
      String paramName = (String)paramNames.nextElement();
      String paramValue = request.getParameter(paramName);
      out.println(paramName + ": " + paramValue + "<br/>\n");
   }
%>
```

```
<%= request.getParameter("first_name")%>

<c:set var="UA" value="${param.first_name}" />
<c:out value="${param.first_name}"/>
```

6.4.

sendRedirect

页面跳转

```
response.sendRedirect("/content/test.jsp");
```

6.5. cookie

获取 cookie 值

```
<html>
<head>
<title>Reading Cookies</title>
</head>
<body>
<center>
<h1>Reading Cookies</h1>
</center>
<%
   Cookie cookie = null;
   Cookie[] cookies = null;
   // Get an array of Cookies associated with this domain
   cookies = request.getCookies();
   if( cookies != null ){
      out.println("<h2> Found Cookies Name and Value</h2>");
      for (int i = 0; i < cookies.length; i++){
         cookie = cookies[i];
         out.print("Name : " + cookie.getName( ) + ",  ");
         out.print("Value: " + cookie.getValue( )+" <br/>");
      }
  }else{
      out.println("<h2>No cookies founds</h2>");
  }
%>
</body>
</html>
```

```

```

6.6. session

获取 Session 值

```
<% out.print(session.getAttribute("random")); %>
```

6.7. page

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
```

Session

禁用Session

```
<% @page session="false" %>
```

启用Session

```
<%@ page session="true" %>
```

6.8. trimDirectiveWhitespaces

删除JSP编译产生的空行

```
<%@ page trimDirectiveWhitespaces="true" %>
```

6.9. include

```
<%@ include file="analytics.jsp" %>
```

```
<jsp:include page="telephone.jsp" />
<jsp:include page="address.jsp" flush="true"/>
<jsp:include
page="${request.getContentType()}/module/html.body.begin.html"
flush="true" />
```

file 与 page 的区别是，file 将文件包含进当前文件，然后变成成 servlet。 page 是运行的时候才会包含文件输出结果包含进当前文件。

```
<pre>

<%@include file="inc.html" %>
=============
<%@include file="/inc.html" %>
=============
<jsp:include page="inc.html" />
=============
```

```
<jsp:include page="inc.html" flush="true" />
==============
<jsp:include page="/inc.html" flush="true" />
==============
<jsp:include page="${request.getContentType()}/inc.html"
flush="true" />

</pre>
```

注意上面包含结果相同，使用上略有差异。

6.10. jsp

jsp:forward

jsp:forward 只能跳转到存在的物理页面

```
<jsp:forward page="/test.jsp" />
```

6.11. error-page

在 web.xml 中配置 error-page

```
<error-page>
        <error-code>400</error-code>
        <location>/error.jsp</location>
</error-page>

<error-page>
        <error-code>404</error-code>
        <location>/error.jsp</location>
</error-page>
```

```
<error-page>
        <error-code>500</error-code>
        <location>/error.jsp</location>
</error-page>
<!-- java.lang.Exception异常错误,依据这个标记可定义多个类似错误提示
-->
<error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/error.jsp</location>
</error-page>
<!-- java.lang.NullPointerException异常错误,依据这个标记可定义多个
类似错误提示 -->
<error-page>
        <exception-type>java.lang.NullPointerException
</exception-type>
        <location>/error.jsp</location>
</error-page>

<error-page>
        <exception-
type>javax.servlet.ServletException</exception-type>
        <location>/error.jsp</location>
</error-page>
```

6.12. JSP 编程

随机数

```
<%= (int) (Math.random() * 1000) %>
```

6.13. FAQ

http://www.netkiller.cn/test.html;jsessionid=7D25CE666FF437F2094AA945E97CEB37

URL经常会出现 jsessionid，这是因为当你使用c:url的时候，它会判断当前域下是否已经创建jsessionid的cookie变量，如果没有并且你链接的是静态页面，那么c:url将传递jsessionid变量，迫使Tomcat为当前域创建jsessionid cookie。

当你使用 Apache Nginx 作为Tomcat前端是，由于Apache Nginx不识别;jsessionid=7D25CE666FF437F2094AA945E97CEB37这种URL，会跳出404页面。

解决方案一，放弃使用c:url，如果你需要保持context 的 path 设置，可以使用下面方法

```
<a
href="${pageContext.request.contextPath}/test.html">Netkiller
Test</a>
```

方案二，使用rewrite截取jsessionid做忽略处理

Apache：

```
ReWriteRule ^/(\w+);jsessionid=\w+$ /$1 [L,R=301]
ReWriteRule ^/(\w+\.do);jsessionid=\w+$ /$1 [L,R=301]
```

Nginx：

```
rewrite ^(.*)\;jsessionid=(.*)$ $1 break;
```

7. JSTL(JavaServer Pages Standard Tag Library)

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
```

7.1. c:set

设置变量

```
<c:set var="foo" scope="request" value="helloworld">

or

<%request.setAttribute("foo","helloworld") %>

<c:out value="${requestScope.foo}"/>
```

c:remove

```
<c:remove var="message" scope="session" />
```

7.2. c:out

输出变量variable的内容

```
<c:out value="${variable}"/>


<%=request.getParameter("UA")%>
相当于
<c:out value="${param.UA}"/>

默认值
<c:out value="${param.UA}" default="UA-69658002-1" />
```

7.3. c:url

生成URL

```
<c:url value="/news/china/"/>
```

7.4. c:redirect

```
<c:redirect url="/index.html"/>
<c:redirect url="http://www.netkiller.cn"/>
```

7.5. c:import

```
<c:import url="http://www.netkiller.cn" />

<c:import var="html" url="http://www.netkiller.cn"/>
<c:out value="${html}"/>
```

传递GET参数

```
<c:import url="http://www.netkiller.cn" >
        <c:param name="id" value="10" />
        <c:param name="name" value="neo" />
</c:import>
```

异常处理

```
<c:catch var="exception">

  <c:import url="ftp://ftp.example.com/package/README"/>

</c:catch>

<c:if test="${not empty exception}">

  Sorry, the remote content is not currently available.

</c:if>
```

在Context间切换

```
server.conf:
<Context  reloadable="true" crossContext="true" />

<c:import url="/path/to/file.jsp" context=/project1" />
<c:import url="/path/to/file.jsp" context=/project2" />
```

7.6. c:if

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%>
<html>
<head>
<title><c:if> Tag Example</title>
</head>
<body>
<c:set var="salary" scope="session" value="${2000*2}"/>
<c:if test="${salary > 2000}">
   <p>My salary is: <c:out value="${salary}"/><p>
</c:if>
</body>
</html>
```

boolean

```
<c:if test="${theBooleanVariable}">It's true!</c:if>
<c:if test="${not theBooleanVariable}">It's false!</c:if>
```

7.7. c:choose

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<c:choose>
        <c:when test="${session.auth eq 'true' }">

        </c:when>
        <c:otherwise>

        </c:otherwise>
</c:choose>
```

实现 if else/else if / else

```
<c:choose>
   <c:when test="${..}">...</c:when> <!-- if condition -->
   <c:when test="${..}">...</c:when> <!-- else if condition -
->
   <c:otherwise>...</c:otherwise>    <!-- else condition -->
</c:choose>
```

7.8. c:forEach

List 处理

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        ${bookList}
        <br>

        <c:forEach items="${bookList}" var="node">
                <c:out value="${node}"></c:out><br>
        </c:forEach>

</body>
</html>
```

Map 处理

```
<%@ page language="java" contentType="text/html; charset=utf-
8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <c:forEach items="${channel}" var="node">
                <a href="<c:out value="${node.value}">
</c:out>"><c:out value="${node.key}"></c:out></a>
        <br/>
```

```
        </c:forEach>
</body>
</html>
```

7.9. empty 判断是否为空

```
        <c:if test="${empty session.member }">

        </c:if>
```

7.10. JSTL fmt Tag setBundle Example

fmt:message

src/resources/config.properties

```
Name=Neo
Address=Shenzhen
Number=13322993040
```

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
        pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
prefix="fmt"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>JSTL fmt:setBundle Tag</title>
</head>
<body>
        <fmt:setBundle basename="resources.config"
var="config" />
        <fmt:message key="Name" bundle="${config}" />

        <fmt:message key="Address" bundle="${config}" />

        <fmt:message key="Number" bundle="${config}" />
</body>
</html>
```

```
        <fmt:bundle basename="lang">
                <fmt:message key="Name" />
                <fmt:message key="Address" />
        </fmt:bundle>
```

Language Package

```
<fmt:setLocale value="en" />
```

fmt:message 变量

```
<fmt:message key="js" bundle="${config}" var="val" />
```

```
<c:out value="${val}"/>
```

```
<fmt:setTimeZone value="Europe/London" scope="session"/>
```

```
<fmt:formatDate value="${isoDate}" type="both"/>
2004-5-31 23:59:59

<fmt:formatDate value="${date}" type="date"/>
2004-4-1

<fmt:formatDate value="${isoDate}" type="time"/>
23:59:59

<fmt:formatDate value="${isoDate}" type="date"
dateStyle="default"/>
2004-5-31

<fmt:formatDate value="${isoDate}" type="date"
dateStyle="short"/>
04-5-31

<fmt:formatDate value="${isoDate}" type="date"
dateStyle="medium"/>
2004-5-31

<fmt:formatDate value="${isoDate}" type="date"
dateStyle="long"/>
2004年5月31日

<fmt:formatDate value="${isoDate}" type="date"
dateStyle="full"/>
2004年5月31日 星期一

<fmt:formatDate value="${isoDate}" type="time"
timeStyle="default"/>
23:59:59
```

```
<fmt:formatDate value="${isoDate}" type="time"
timeStyle="short"/>
下午11:59

<fmt:formatDate value="${isoDate}" type="time"
timeStyle="medium"/>
23:59:59

<fmt:formatDate value="${isoDate}" type="time"
timeStyle="long"/>
下午11时59分59秒

<fmt:formatDate value="${isoDate}" type="time"
timeStyle="full"/>
下午11时59分59秒 CDT

<fmt:formatDate value="${date}" type="both" pattern="EEEE, MMMM
d, yyyy HH:mm:ss Z"/>
星期四, 四月 1, 2004 13:30:00 -0600

<fmt:formatDate value="${isoDate}" type="both" pattern="d MMM
yy, h:m:s a zzzz/>
31 五月 04, 11:59:59 下午 中央夏令时

格式模式:
  d    月中的某一天。一位数的日期没有前导零。
  dd    月中的某一天。一位数的日期有一个前导零。
  ddd    周中某天的缩写名称，在    AbbreviatedDayNames    中定义。
  dddd    周中某天的完整名称，在    DayNames    中定义。
  M    月份数字。一位数的月份没有前导零。
  MM    月份数字。一位数的月份有一个前导零。
  MMM    月份的缩写名称，在    AbbreviatedMonthNames    中定义。
  MMMM    月份的完整名称，在    MonthNames    中定义。
  y    不包含纪元的年份。如果不包含纪元的年份小于    10，则显示不具有前导零
的年份。
  yy    不包含纪元的年份。如果不包含纪元的年份小于    10，则显示具有前导零
的年份。
  yyyy    包括纪元的四位数的年份。
  gg    时期或纪元。如果要设置格式的日期不具有关联的时期或纪元字符串，则忽
略该模式。
  h    12    小时制的小时。一位数的小时数没有前导零。
  hh    12    小时制的小时。一位数的小时数有前导零。
  H    24    小时制的小时。一位数的小时数没有前导零。
  HH    24    小时制的小时。一位数的小时数有前导零。
```

```
  m    分钟。一位数的分钟数没有前导零。
  mm    分钟。一位数的分钟数有一个前导零。
  s    秒。一位数的秒数没有前导零。
  ss    秒。一位数的秒数有一个前导零。

<fmt:formatDate value="${xx}" pattern="dd/MM/yyyy HH:mm aa"/>和

<fmt:formatDate value="${xx}" pattern="dd/MM/yyyy hh:mm aa"/>
对于0点显示的结果不一样


<fmt:formatDate value="${dateValue}" pattern="yyyy-MM-dd
HH:mm:ss z" timeZone="GMT"/>

<fmt:formatDate value="${dateValue}" pattern="yyyy-MM-dd
HH:mm:ss z" timeZone="US/Eastern"/>
```

8. WebSocket

环境：Java8 + Tomcat8

8.1. Server

```
package websocket;

/**
 * Websocket Server
 *
 * @author netkiller<netkiller@msn.com>
 */

import java.util.Collections;
import java.util.HashSet;
import java.util.Set;

import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.ServerEndpoint;

@ServerEndpoint(value = "/echo")
public class PriceServer {

        private Set<Session> sessions =
Collections.synchronizedSet(new HashSet<Session>());

        /**
         * Callback hook for Connection open events. This
method will be invoked
         * when a client requests for a WebSocket connection.
         *
         * @param session
         *            the session which is opened.
         */
        @OnOpen
```

```
        public void onOpen(Session session) {
                sessions.add(session);
        }

        /**
         * Callback hook for Connection close events. This
method will be invoked
         * when a client closes a WebSocket connection.
         *
         * @param session
         *            the session which is opened.
         */
        @OnClose
        public void onClose(Session session) {
                sessions.remove(session);
        }

        /**
         * Callback hook for Message Events. This method will
be invoked when a
         * client send a message.
         *
         * @param message
         *            The text message
         * @param session
         *            The session of the client
         */
        @OnMessage
        public void onMessage(String message, Session
session) {
                System.out.println("Message Received: " +
message);
                for (Session remote : sessions) {
                        System.out.println("Sending to " +
remote.getId());

remote.getAsyncRemote().sendText(message);
                }
        }
}
```

8.2. Client

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

        <script language="JavaScript">
                var wsuri =
"ws://localhost:8080/m.example.com/echo";
                var ws = null;

                function connectEndpoint() {
                        ws = new WebSocket(wsuri);
                        ws.onmessage = function(evt) {
                                //alert(evt.data);

document.getElementById("echo").value = evt.data;
                        };

                        ws.onclose = function(evt) {
                                //alert("close");

document.getElementById("echo").value = "end";
                        };

                        ws.onopen = function(evt) {
                                //alert("open");

document.getElementById("echo").value = "open";
                        };
                }

                function sendmsg() {

ws.send(document.getElementById("send").value);
                }
        </script>
```

```
<body onload="connectEndpoint()">
        <input type="text" size="20" value="5" id="send">
        <input type="button" value="send"
onclick="sendmsg()">
        <br>
        <input type="text" id="echo">
</body>
</html>
```

部分 I. Spring Framework

第 4 章 Spring 开发环境

1. Java 开发环境

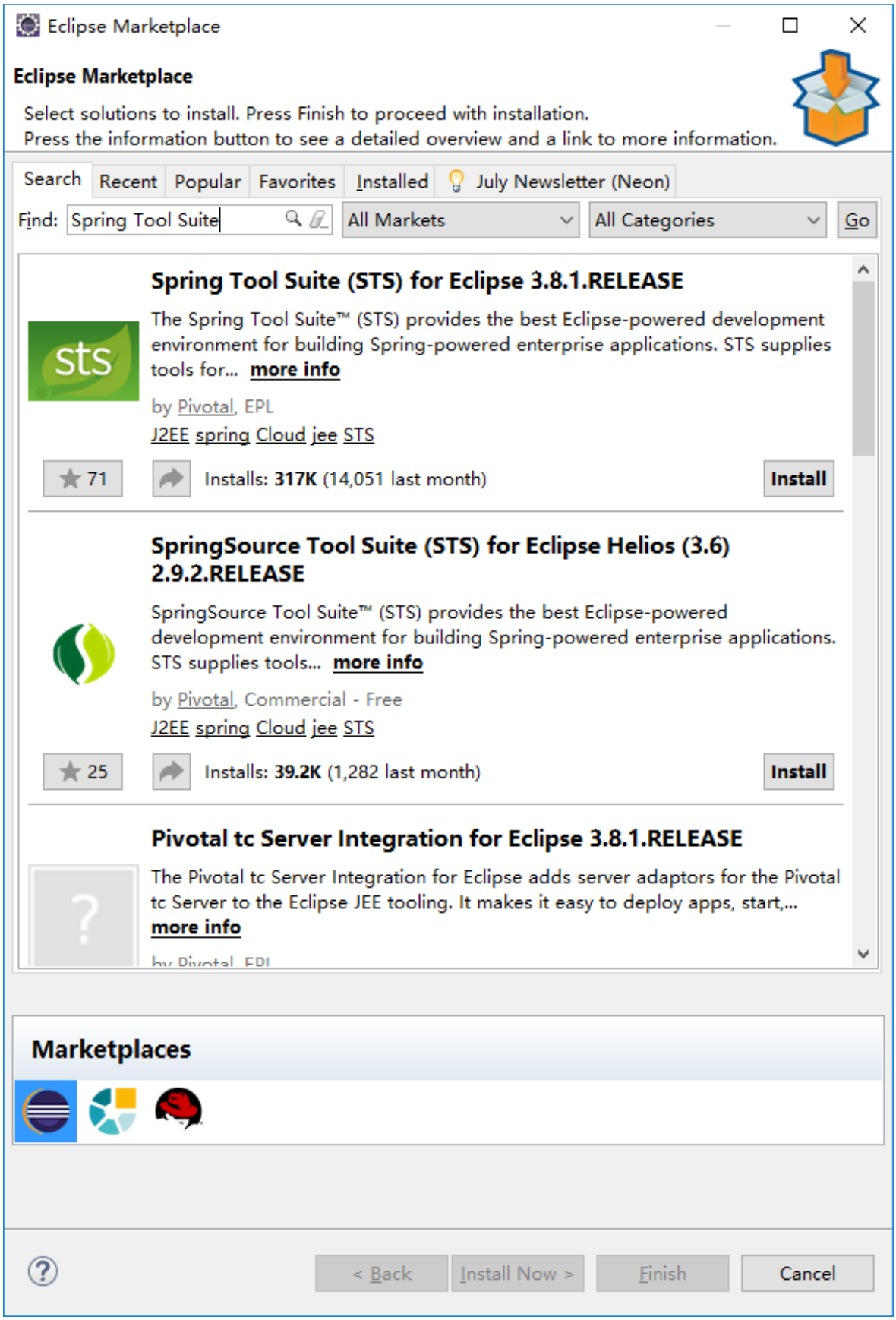
```
[root@localhost ~]# dnf install java-latest-openjdk-devel
[root@localhost ~]# dnf install maven
```

2. 安装 Spring Tool Suite

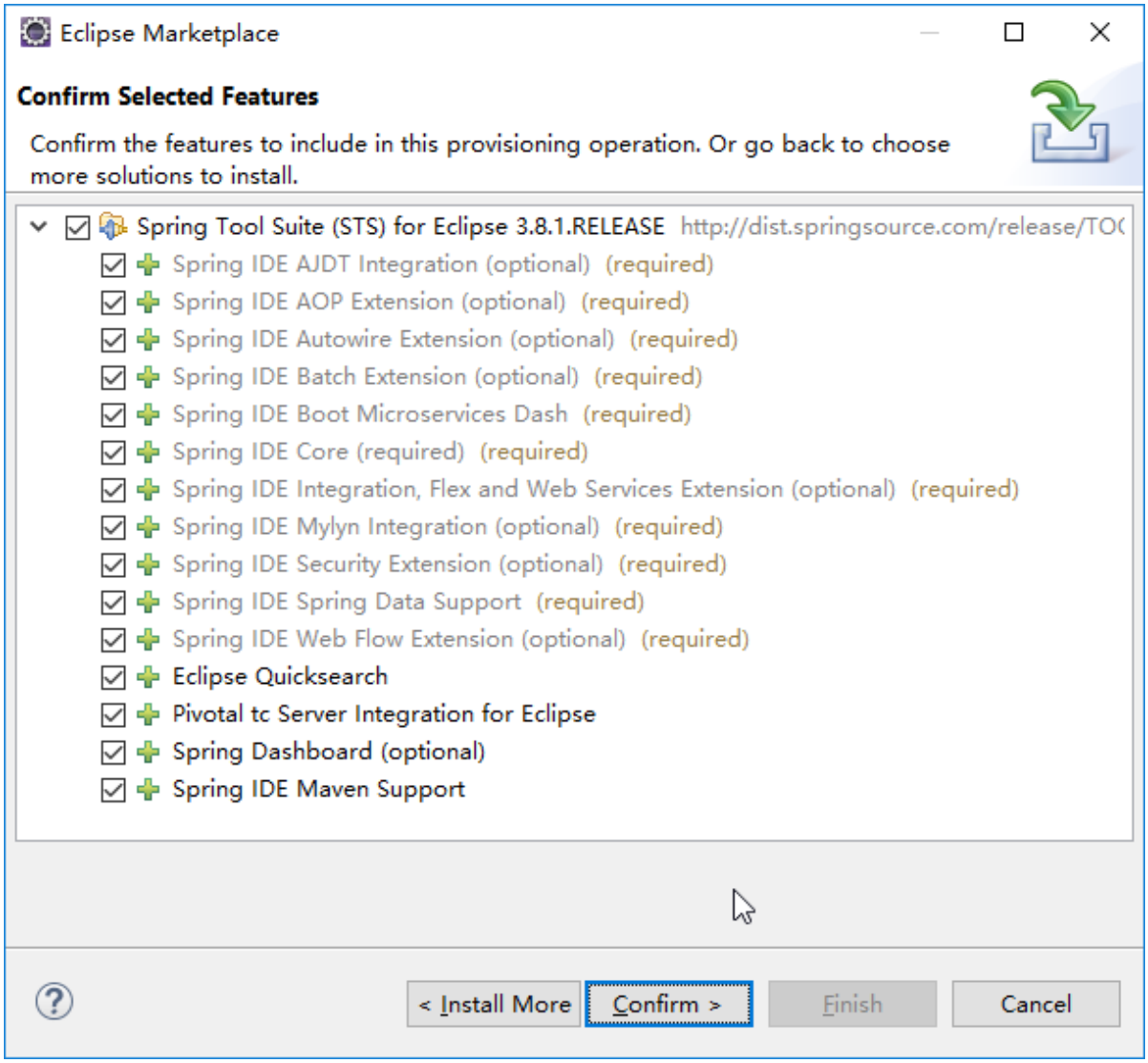
https://spring.io/tools/sts/

环境 Eclipse Jee Neon

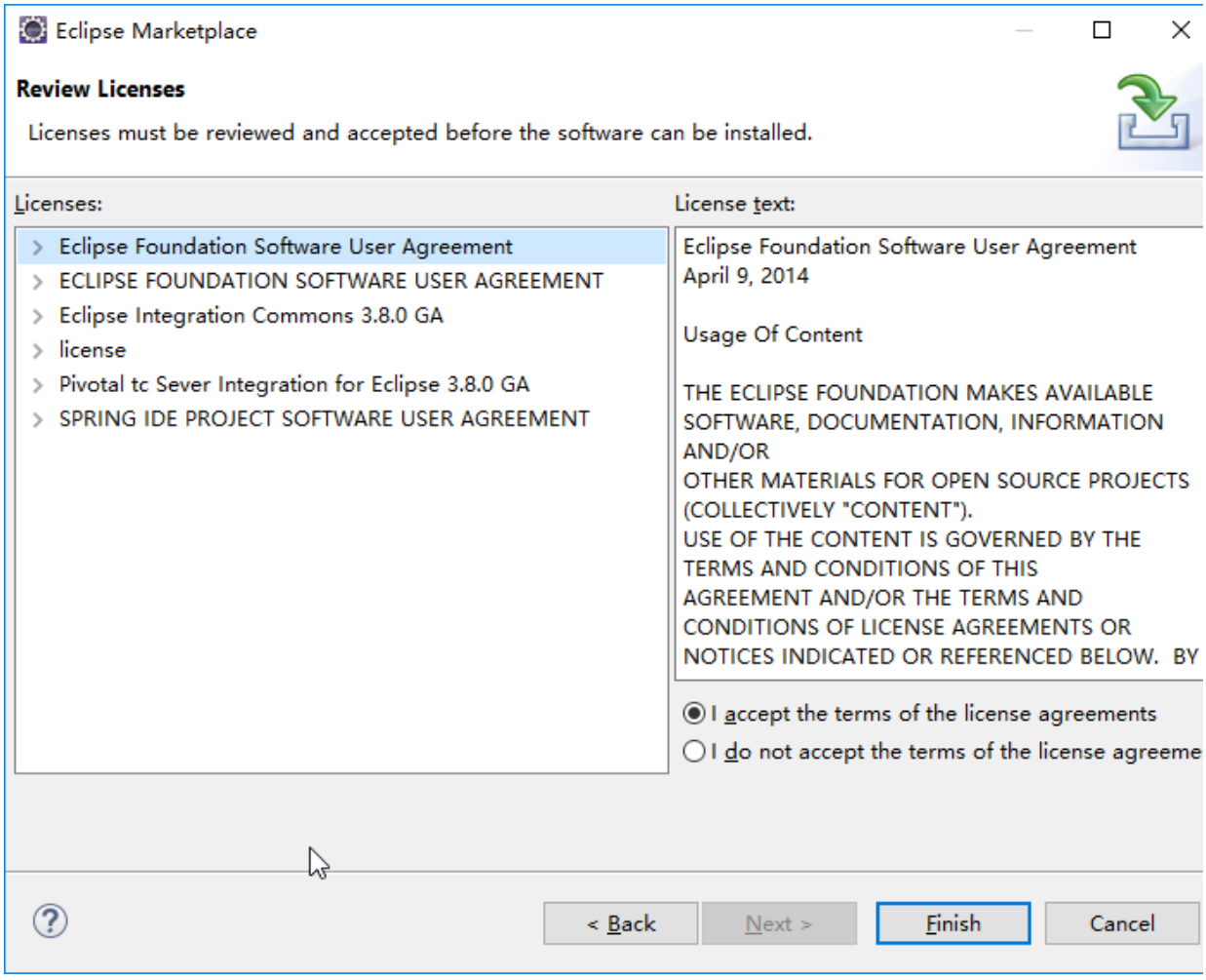
进入菜单 Help -> Marketpalce...


Eclipse Marketplace
Eclipse Marketplace
Select solutions to install. Press Finish to proceed with installation.
Press the information button to see a detailed overview and a link to more information.
Search
Recent
Popular
Favorites
Installed
July Newsletter (Neon)
Find:
Spring Tool Suite
All Markets
All Categories
Go
Spring Tool Suite (STS) for Eclipse 3.8.1.RELEASE
The Spring Tool Suite™ (STS) provides the best Eclipse-powered development environment for building Spring-powered enterprise applications. STS supplies tools for... more info
by Pivotal, EPL
J2EE spring Cloud jee STS
71
Installs: 317K (14,051 last month)
Install
SpringSource Tool Suite (STS) for Eclipse Helios (3.6) 2.9.2.RELEASE
SpringSource Tool Suite™ (STS) provides the best Eclipse-powered development environment for building Spring-powered enterprise applications. STS supplies tools... more info
by Pivotal, Commercial - Free
J2EE spring Cloud jee STS
25
Installs: 39.2K (1,282 last month)
Install
Pivotal tc Server Integration for Eclipse 3.8.1.RELEASE
The Pivotal tc Server Integration for Eclipse adds server adaptors for the Pivotal tc Server to the Eclipse JEE tooling. It makes it easy to deploy apps, start,... more info
Marketplaces
< Back
Install Now >
Finish
Cancel

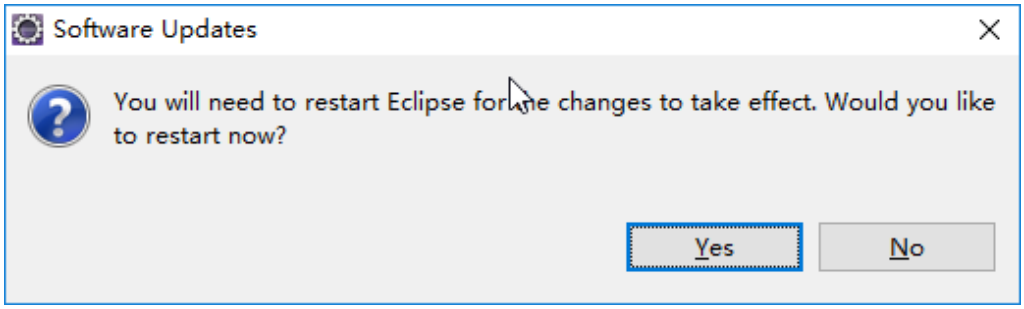
索搜 Spring Tool Suite 注意版本号



点击Confirm按钮



点击Finish按钮，等候漫长的下载，同时Progress窗口中显示Installing Software，安装成功会提示重新启动Eclipse.



点击 Yes 按钮重启 Eclipse

3. Dashboard

进入菜单 Help -> Dashboard

4. Spring Initializr - Bootstrap your application

https://start.spring.io

第 5 章 Spring Boot

注意以下使用 Spring boot 2

1. Spring Boot Quick start

1.1. 创建项目

```
curl https://start.spring.io/starter.tgz  \
  -d artifactId=creds-example-server \
  -d dependencies=security,web \
  -d language=java \
  -d type=maven-project \
  -d baseDir=example-server \
| tar -xzvf -
```

1.2. pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>api.netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Skyline</name>
        <description>skylinechencf@gmail.com</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
```

```
                <version>1.4.0.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

1.3. Controller

```
package hello;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
@EnableAutoConfiguration
public class SampleController {
```

```
    @RequestMapping("/")
    @ResponseBody
    String home() {
        return "Hello World!";
    }

    public static void main(String[] args) throws Exception {
        SpringApplication.run(SampleController.class, args);
    }
}
```

测试

```
curl http://127.0.0.1:8080/
```

2. Springboot with Maven

spring-boot-maven-plugin 插件

2.1. resource

将 resource 添加应用程序

```
<build>
    <resources>
        <resource>
            <directory>src/main/java/resources</directory>
            <filtering>true</filtering>
            <excludes>
                <exclude>*.jks</exclude>
            </excludes>
        </resource>
    </resources>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <addResources>true</addResources>
            </configuration>
        </plugin>
    </plugins>
</build>
```

2.2. Maven run

```
$ mvn spring-boot:run
$ mvn -P prod spring-boot:run
```

```
```

-P 指定 Maven 的 profile，如果指定 Springboot 的 profiles 请使用 -Drun.profiles=prod

```
$ mvn spring-boot:run -Drun.profiles=prod
```

打包后，使用jar包运行

```
$ mvn verify
$ mvn package
$ java -jar target/api.netkiller.cn-0.0.1-SNAPSHOT.jar
```

2.3. Spring Boot maven 插件 build-image

Spring Boot 构建 Docker 镜像，你不需要写 Dockerfile，plugin 帮你完成。

只需要简单的执行:

```
mvn spring-boot:build-image
```

执行完成后会看到成功提示信息:

```
[INFO] Successfully buit image 'docker.io/library/demo:0.0.1-
```

```
SNAPSHOT'
```

运行容器测试:

```
docker run -p 8000:8080 -t demo:0.0.1-SNAPSHOT
```

注意：这里映射的本机端口是8000。

```
curl http://localhost:8000/
```

2.4. 生成项目信息

mvn spring-boot:build-info

```
neo@MacBook-Pro-Neo ~/workspace/microservice/config % mvn
spring-boot:build-info
```

3. SpringApplication

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoCon
figuration;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@EnableAutoConfiguration(exclude=
{DataSourceAutoConfiguration.class})
@ComponentScan({"cn.netkiller.controller"})
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

3.1. 运行 Spring boot 项目

Linux systemd

/etc/systemd/system/spring.service

```
##################################################
# Homepage: http://netkiller.github.io
```

```
# Author: netkiller<netkiller@msn.com>
# Script: https://github.com/oscm/shell
# Date: 2015-11-03
#####################################################
[Unit]
Description=Spring Boot Application
After=network.target

[Service]
User=www
Group=www
Type=oneshot
WorkingDirectory=/www/netkiller.cn/api.netkiller.cn
ExecStart=/usr/bin/java -jar your_jar_file.jar --
spring.config.location=appliction-production.properties --
spring.profiles.active=profile
#ExecStop=pkill -9 -f
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

传统 **init.d** 脚本

```
#!/bin/bash
#################################################
# Author: netkiller<netkiller@msn.com>
# Homepage: http://www.netkiller.cn
# Date: 2017-02-08
# $Author$
# $Id$
#################################################
# chkconfig: 345 100 02
# description: Spring boot application
# processname: springbootd
# File : springbootd
#################################################
BASEDIR="/www/netkiller.cn/api.netkiller.cn"
JAVA_HOME=/srv/java
```

```
JAVA_OPTS="-server -Xms2048m -Xmx8192m -
Djava.security.egd=file:/dev/./urandom"
PACKAGE="api.netkiller.cn-0.0.2-release.jar"
CONFIG="--
spring.config.location=$BASEDIR/application.properties"
USER=www
################################################
NAME=springbootd
PROG="$JAVA_HOME/bin/java $JAVA_OPTS -jar $BASEDIR/$PACKAGE
$CONFIG"
LOGFILE=/var/tmp/$NAME.log
PIDFILE=/var/tmp/$NAME.pid
ACCESS_LOG=/var/tmp/$NAME.access.log
################################################

function log(){
        echo "$(date -d "today" +"%Y-%m-%d %H:%M:%S") $1
$2" >> $LOGFILE
}

function start(){
        if [ -f "$PIDFILE" ]; then
                echo $PIDFILE
                exit 2
        fi

        su - $USER -c "$PROG & echo \$! > $PIDFILE"
        log info start
}
function stop(){
        [ -f $PIDFILE ] && kill `cat $PIDFILE` && rm -rf
$PIDFILE
        log info stop
}
function status(){
        ps aux | grep $PACKAGE | grep -v grep | grep -v status
        log info status
}
function reset(){
        pkill -f $PACKAGE
        [ -f $PIDFILE ] && rm -rf $PIDFILE
        log info reset
}

case "$1" in
```

```
        start)
                start
                ;;
        stop)
                stop
                ;;
        status)
                status
                ;;
        restart)
                stop
                start
                ;;
        log)
                tail -f $LOGFILE
                ;;
        reset)
                reset
                ;;
        *)
                echo $"Usage: $0
{start|stop|status|restart|log|reset}"
esac
exit $?
```

编译用于Tomcat的 War

```
package demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.context.web.SpringBootServletInitial
izer;
```

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

@Configuration
@ComponentScan
@EnableAutoConfiguration
public class Application extends SpringBootServletInitializer
{

    private static Class<Application> applicationClass =
Application.class;

    public static void main(String[] args) {
        SpringApplication.run(applicationClass, args);
    }

    @Override
    protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
        return application.sources(applicationClass);
    }
}
```

3.2. @SpringBootApplication

@SpringBootApplication 是 @Configuration, @EnableAutoConfiguration 跟 @ComponentScan 的集合。

```
@SpringBootApplication
```

排除 @EnableAutoConfiguration 加载项

```
@SpringBootApplication(exclude =
DataSourceAutoConfiguration.class)
```

3.3. 获取 Resources 目录中的静态文件

```
        package cn.netkiller;

        import java.io.File;
        import java.io.IOException;
        import java.net.URL;

        import org.springframework.core.io.ClassPathResource;
        import
org.springframework.web.bind.annotation.GetMapping;
        import
org.springframework.web.bind.annotation.RestController;

        @RestController
        public class TestController {

                public TestController() {
                        // TODO Auto-generated constructor
stub
                }

                @GetMapping("/test")
                public String test() {
                        ClassPathResource resource = new
ClassPathResource("test.ttf");
                        File file = null;
                        try {
                                file = resource.getFile();
                        } catch (IOException e) {
                                e.printStackTrace();
```

```
                        }
                        // InputStream inStream = new
FileInputStream(file.getPath());
                        // BufferedReader br = new
BufferedReader(new
InputStreamReader(resource.getInputStream()));
                        return file.getPath();
                }
                @GetMapping("/test1")
                public String test1() {
                        URL url =
Thread.currentThread().getContextClassLoader().getResource("t
est.ttf");
                        return url.getPath();
                }

        }
```

3.4. @EnableAutoConfiguration

exclude 排除配置，下面例子是排除 DataSource配置

```
@EnableAutoConfiguration(exclude=
{DataSourceAutoConfiguration.class})
```

3.5. @ComponentScan

@ComponentScan 注入会扫描 @Controller 与 @RestController

```
@ComponentScan
@ComponentScan({"cn.netkiller.controller"})
@ComponentScan({"cn.netkiller.controller",
"cn.netkiller.rest"})
```

```

```

3.6. @EntityScan 实体扫描

```
@EntityScan("common.domain")
```

3.7. @EnableJpaRepositories

扫描 Jpa 仓库

```
@EnableJpaRepositories("common.domain")
```

3.8. 启动和销毁

```
package cn.netkiller;


import jakarta.annotation.PostConstruct;
import jakarta.annotation.PreDestroy;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import org.springframework.scheduling.annotation.EnableAsync;

@SpringBootApplication
@EnableJpaRepositories
@EnableAutoConfiguration
@EnableAsync
public class Application {
    private static final Logger logger =
LoggerFactory.getLogger(Application.class);

    @Value("${spring.application.name}")
    public String name;

    public static void main(String[] args) {
        System.out.println("Watch interface start...");
        SpringApplication.run(Application.class, args);
    }

    @PostConstruct
    public void init() {
        logger.info(String.format("==================== %s 系
统启动 ====================", name));
    }

    @PreDestroy
    public void destroy() {
        logger.info(String.format("==================== %s 系
统销毁 ====================", name));
    }
}
```

3.9. 打印环境变量

```
package cn.netkiller;
```

```
import jakarta.annotation.PostConstruct;
import jakarta.annotation.PreDestroy;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import org.springframework.scheduling.annotation.EnableAsync;

@SpringBootApplication
@EnableJpaRepositories
@EnableAutoConfiguration
@EnableAsync
public class Application {
    private static final Logger logger =
LoggerFactory.getLogger(Application.class);

    public static void main(String[] args) {
        System.out.println("Watch interface start...");
        SpringApplication.run(Application.class, args);
    }

    @Bean
    ApplicationRunner applicationRunner(Environment
environment) {
        return args -> {
            // log.info("our database URL connection will be
" +
            //
environment.getProperty("spring.datasource.url"));

System.out.println(environment.getProperty("spring.applicatio
n.name"));
        };
    }
}
```

3.10. CharacterEncodingFilter

```
        public @Bean Filter characterEncodingFilter() {
                CharacterEncodingFilter
characterEncodingFilter = new CharacterEncodingFilter();
                characterEncodingFilter.setEncoding("UTF-8");

characterEncodingFilter.setForceEncoding(true);
                return characterEncodingFilter;
        }
```

3.11. 隐藏 Banner

隐藏 Spring Boot Banner

```
.   ____          _            __ _ _
/\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::  (v2.3.1.RELEASE)
```

```
public static void main(String[] args) {
    SpringApplication app = new
SpringApplication(Application.class);
    app.setShowBanner(false);
    app.run(args);
```

```
}
```

3.12. 实体与仓库扫描

```
@EntityScan(basePackages = { "cn.netkiller.model" })
@EnableJpaRepositories(basePackages = {
"cn.netkiller.repository" })
```

3.13. 列出 Beans

```
package cn.netkiller;

import java.util.Arrays;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
```

```
@ComponentScan
@EnableMongoRepositories
@EnableJpaRepositories
@EnableScheduling
public class Application {

        public static void main(String[] args) {
                //SpringApplication.run(Application.class,
args);

                ApplicationContext ctx =
SpringApplication.run(Application.class, args);

                System.out.println("Let's inspect the beans
provided by Spring Boot:");

                String[] beanNames =
ctx.getBeanDefinitionNames();
                Arrays.sort(beanNames);
                for (String beanName : beanNames) {
                        System.out.println(beanName);
                }

        }
}
```

3.14. Tomcat 端口

```
@Configuration
public class TomcatConfiguration implements
EmbeddedServletContainerCustomizer {

        int ports[] = { 8080, 8081, 8082 };

        @Override
        public void
customize(ConfigurableEmbeddedServletContainer
configurableEmbeddedServletContainer) {
```

```
                if (ports != null) {
                        // 判断如果是Tomcat才进行如下配置
                        if
(configurableEmbeddedServletContainer instanceof
TomcatEmbeddedServletContainerFactory) {


TomcatEmbeddedServletContainerFactory tomcat =
(TomcatEmbeddedServletContainerFactory)
configurableEmbeddedServletContainer;

                                for (int port : ports) {
                                        // 一个Connecter监听一
个端口,指定协议为HTTP/1.1
                                        Connector
httpConnector = new Connector("HTTP/1.1");

httpConnector.setPort(port);

tomcat.addAdditionalTomcatConnectors(httpConnector);
                                }

                        }
                }
        }
}
```

3.15. 配置项设定

```
public static void main(String[] args) {
    SpringApplication.run(Backend.class,
        "--spring.application.name=backend",
        "--server.port=9000"
    );
  }
```

3.16. spring.profiles.active

```
在 Java 代码中激活 profile

直接指定环境变量来激活 profile：

System.setProperty("spring.profiles.active", "test");

在 Spring 容器中激活 profile：

AnnotationConfigApplicationContext ctx = new
AnnotationConfigApplicationContext();
ctx.getEnvironment().setActiveProfiles("development");
ctx.register(SomeConfig.class, StandaloneDataConfig.class,
JndiDataConfig.class);
ctx.refresh();
```

3.17. @Profile("dev") / @ActiveProfiles("dev")

不同环境运行不同的逻辑

```
@Configuration
public class DataSourceConfig {

    @Bean
    @Profile("dev")
    public DataSource devDataSource() {
        System.out.println(" dev DataSource !!");
        BasicDataSource basicDataSource = new
BasicDataSource();

basicDataSource.setDriverClassName("com.mysql.jdbc.Driver");

basicDataSource.setUrl("jdbc:mysql://localhost:3308/neo");
```

```
        basicDataSource.setUsername("root");
        basicDataSource.setPassword("123456");
        return basicDataSource;
    }

    @Bean
    @Profile("prod")
    public DataSource prodDataSource() {
        System.out.println(" prod DataSource !!");
        BasicDataSource basicDataSource = new
BasicDataSource();

basicDataSource.setDriverClassName("com.mysql.jdbc.Driver");

basicDataSource.setUrl("jdbc:mysql://localhost:3306/neo");
        basicDataSource.setUsername("root");
        basicDataSource.setPassword("123456");
        return basicDataSource;
    }
}
```

匹配多个环境

```
@Profile({"dev", "test", "grey", "prod"})
```

3.18. 设置默认时区

```
package cn.netkiller;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
```

```
org.springframework.cloud.client.discovery.EnableDiscoveryCli
ent;
import
org.springframework.cloud.openfeign.EnableFeignClients;
import
org.springframework.transaction.annotation.EnableTransactionM
anagement;

import javax.annotation.PostConstruct;
import java.util.TimeZone;

@SpringBootApplication
@EnableDiscoveryClient
@EnableTransactionManagement
@EnableFeignClients(basePackages =
{"cn.netkiller.feign.*","cn.netkiller.openfeign.*"})
@MapperScan("cn.netkiller.dao")
public class Application {
    @PostConstruct
    void setDefaultTimezone() {

TimeZone.setDefault(TimeZone.getTimeZone("Asia/Shanghai"));
    }

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

4. 如何优雅停止 Springboot 运行

4.1. 准备工作

@PreDestroy 会在系统关闭前执行

```
package cn.netkiller;

import javax.annotation.PreDestroy;

import org.springframework.context.annotation.Configuration;

@Configuration
public class ShutdownConfiguration {

        public ShutdownConfiguration() {
                // TODO Auto-generated constructor stub
        }

        @PreDestroy
        public void preDestroy() {

System.out.println("==============================");
                System.out.println("Destroying Spring");

System.out.println("==============================");
        }

}
```

4.2. kill 命令演示

kill 命令本质是给进程发送终止信号，进程接收到终止信号后退出运行。

可以看到 Springboot 启动后，进程 PID 44559，现在使用 kill 命令杀死这个进程

当执行 kill 44559 你会看到下面的输出

```
neo@MacBook-Pro-Neo ~/workspace/microservice/test % java -jar
target/test-0.0.1-SNAPSHOT.jar
Starting...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.3)

2021-07-29 11:05:09.862  INFO 44559 --- [           main]
cn.netkiller.Application                 : Starting Application
v0.0.1-SNAPSHOT using Java 16.0.1 on MacBook-Pro-Neo.local with
PID 44559 (/Users/neo/workspace/microservice/test/target/test-
0.0.1-SNAPSHOT.jar started by neo in
/Users/neo/workspace/microservice/test)
2021-07-29 11:05:09.865  INFO 44559 --- [           main]
cn.netkiller.Application                 : No active profile
set, falling back to default profiles: default
2021-07-29 11:05:11.363  WARN 44559 --- [           main]
io.undertow.websockets.jsr               : UT026010: Buffer
pool was not set on WebSocketDeploymentInfo, the default pool
will be used
2021-07-29 11:05:11.399  INFO 44559 --- [           main]
io.undertow.servlet                      : Initializing Spring
embedded WebApplicationContext
2021-07-29 11:05:11.400  INFO 44559 --- [           main]
w.s.c.ServletWebServerApplicationContext : Root
WebApplicationContext: initialization completed in 1451 ms
2021-07-29 11:05:12.041  INFO 44559 --- [           main]
o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1
endpoint(s) beneath base path '/actuator'
2021-07-29 11:05:12.073  INFO 44559 --- [           main]
io.undertow                              : starting server:
Undertow - 2.2.9.Final
```

```
2021-07-29 11:05:12.085  INFO 44559 --- [           main]
org.xnio                                 : XNIO version
3.8.4.Final
2021-07-29 11:05:12.099  INFO 44559 --- [           main]
org.xnio.nio                             : XNIO NIO
Implementation Version 3.8.4.Final
2021-07-29 11:05:12.197  INFO 44559 --- [           main]
org.jboss.threads                        : JBoss Threads
version 3.1.0.Final
2021-07-29 11:05:12.263  INFO 44559 --- [           main]
o.s.b.w.e.undertow.UndertowWebServer     : Undertow started on
port(s) 8080 (http)
2021-07-29 11:05:12.278  INFO 44559 --- [           main]
cn.netkiller.Application                 : Started Application
in 2.989 seconds (JVM running for 3.582)
2021-07-29 11:05:20.577  INFO 44559 --- [ionShutdownHook]
io.undertow                              : stopping server:
Undertow - 2.2.9.Final
==============================
Destroying Spring
==============================
```

而是用 kill -9 PID 就不会出现下面提示。

```
neo@MacBook-Pro-Neo ~/workspace/microservice/test % java -jar
target/test-0.0.1-SNAPSHOT.jar
Starting...

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.3)

2021-07-29 11:08:10.857  INFO 44613 --- [           main]
cn.netkiller.Application                 : Starting Application
v0.0.1-SNAPSHOT using Java 16.0.1 on MacBook-Pro-Neo.local with
PID 44613 (/Users/neo/workspace/microservice/test/target/test-
```

```
0.0.1-SNAPSHOT.jar started by neo in
/Users/neo/workspace/microservice/test)
2021-07-29 11:08:10.860  INFO 44613 --- [           main]
cn.netkiller.Application                 : No active profile
set, falling back to default profiles: default
2021-07-29 11:08:12.377  WARN 44613 --- [           main]
io.undertow.websockets.jsr               : UT026010: Buffer
pool was not set on WebSocketDeploymentInfo, the default pool
will be used
2021-07-29 11:08:12.411  INFO 44613 --- [           main]
io.undertow.servlet                      : Initializing Spring
embedded WebApplicationContext
2021-07-29 11:08:12.411  INFO 44613 --- [           main]
w.s.c.ServletWebServerApplicationContext : Root
WebApplicationContext: initialization completed in 1466 ms
2021-07-29 11:08:13.046  INFO 44613 --- [           main]
o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1
endpoint(s) beneath base path '/actuator'
2021-07-29 11:08:13.081  INFO 44613 --- [           main]
io.undertow                              : starting server:
Undertow - 2.2.9.Final
2021-07-29 11:08:13.100  INFO 44613 --- [           main]
org.xnio                                 : XNIO version
3.8.4.Final
2021-07-29 11:08:13.114  INFO 44613 --- [           main]
org.xnio.nio                             : XNIO NIO
Implementation Version 3.8.4.Final
2021-07-29 11:08:13.206  INFO 44613 --- [           main]
org.jboss.threads                        : JBoss Threads
version 3.1.0.Final
2021-07-29 11:08:13.275  INFO 44613 --- [           main]
o.s.b.w.e.undertow.UndertowWebServer     : Undertow started on
port(s) 8080 (http)
2021-07-29 11:08:13.290  INFO 44613 --- [           main]
cn.netkiller.Application                 : Started Application
in 3.195 seconds (JVM running for 3.808)
[1]    44613 killed     java -jar target/test-0.0.1-
SNAPSHOT.jar
```

这是因为 kill 命令会给进程发送终止信号，进程会正常退出.

什么是正常退出呢？例如:

- 完成为运行的逻辑
- 将为写入磁盘的文件后写入后退出
- 执行完SQL并关闭数据库
- 写入缓存，并关闭 redis
- 完成用户请求，并关闭链接

这就是为什么当我们正常关闭程序需要等待很长时间，如果我们此时没有运行状态显示，也没有通过日志反应执行状态，就会认为程序死了。其实此时程序可能尽职尽责的在工作，将未完成的工作完成，然后一步步正常退出。

尤其是多线程的程序，退出时需要等待每个线程完成请求，需要很长时间，我们常常因为升级时间紧迫而使用 kill -9 强行杀死进程，这会带来很多问题。

kill -9 的弊端：

1. 程序执行一半被强行退出，用户端会出现 Timeout 超时
2. 文件写入一半被终止，如果是文本文件只有一半内容；如果是二进制文件会造成损坏
3. 数据库操作一组SQL，只执行了一半，会产生脏数据；如果使用事务处理会引起回滚；
4.

Ctrl + C 与 kill 没有区别，也是给进程发送终止信号，现在我们来演示一下。

```
neo@MacBook-Pro-Neo ~/workspace/microservice/test % java -jar
target/test-0.0.1-SNAPSHOT.jar
Starting...


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
```

```
 :: Spring Boot ::                (v2.5.3)

2021-07-29 11:04:42.657  INFO 44546 --- [           main]
cn.netkiller.Application                 : Starting Application
v0.0.1-SNAPSHOT using Java 16.0.1 on MacBook-Pro-Neo.local with
PID 44546 (/Users/neo/workspace/microservice/test/target/test-
0.0.1-SNAPSHOT.jar started by neo in
/Users/neo/workspace/microservice/test)
2021-07-29 11:04:42.660  INFO 44546 --- [           main]
cn.netkiller.Application                 : No active profile
set, falling back to default profiles: default
2021-07-29 11:04:44.212  WARN 44546 --- [           main]
io.undertow.websockets.jsr               : UT026010: Buffer
pool was not set on WebSocketDeploymentInfo, the default pool
will be used
2021-07-29 11:04:44.246  INFO 44546 --- [           main]
io.undertow.servlet                      : Initializing Spring
embedded WebApplicationContext
2021-07-29 11:04:44.246  INFO 44546 --- [           main]
w.s.c.ServletWebServerApplicationContext : Root
WebApplicationContext: initialization completed in 1502 ms
2021-07-29 11:04:44.857  INFO 44546 --- [           main]
o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1
endpoint(s) beneath base path '/actuator'
2021-07-29 11:04:44.889  INFO 44546 --- [           main]
io.undertow                              : starting server:
Undertow - 2.2.9.Final
2021-07-29 11:04:44.902  INFO 44546 --- [           main]
org.xnio                                 : XNIO version
3.8.4.Final
2021-07-29 11:04:44.916  INFO 44546 --- [           main]
org.xnio.nio                             : XNIO NIO
Implementation Version 3.8.4.Final
2021-07-29 11:04:45.002  INFO 44546 --- [           main]
org.jboss.threads                        : JBoss Threads
version 3.1.0.Final
2021-07-29 11:04:45.068  INFO 44546 --- [           main]
o.s.b.w.e.undertow.UndertowWebServer     : Undertow started on
port(s) 8080 (http)
2021-07-29 11:04:45.084  INFO 44546 --- [           main]
cn.netkiller.Application                 : Started Application
in 3.149 seconds (JVM running for 3.748)
^C2021-07-29 11:04:47.082  INFO 44546 --- [ionShutdownHook]
io.undertow                              : stopping server:
Undertow - 2.2.9.Final
```

```
==============================
Destroying Spring
==============================
```

4.3. 容器中如何优雅关闭 Springboot

容器与进程模式并没有什么区别，我们给容器发送终止信号，容器会转发给 Springboot。

理论归理论，我们还是需要亲自实践，这样才能理解更深刻。

准备实验环境和素材，下面是 docker-compose.yaml 编排文件

```
version: '3.9'

services:
  spring:
    image: openjdk:latest
    container_name: spring
    restart: always
    hostname: www.netkiller.cn
    environment:
      TZ: Asia/Shanghai
      JAVA_OPTS: -Xms256m -Xmx512m -XX:MetaspaceSize=128m -
XX:MaxMetaspaceSize=512m
    ports:
      - 8099:8080
    volumes:
      - ./test-0.0.1-SNAPSHOT.jar:/app/test-0.0.1-SNAPSHOT.jar
    entrypoint: java -jar /app/test-0.0.1-SNAPSHOT.jar
    command:
      --spring.profiles.active=dev
      --server.port=8080
```

实验步骤

- 运行容器：docker-compose up
- 观察容器：docker-compose logs -f
- 停止容器：

运行容器

```
[root@localhost netkiller.cn]# docker-compose up -d
Starting spring ... done
```

观察容器日志

```
[root@localhost netkiller.cn]# docker-compose logs -f
spring    | Starting...
spring    |
spring    |   .   ____          _            __ _ _
spring    |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
spring    | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
spring    |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
spring    |   '  |____| .__|_| |_|_| |_\__, | / / / /
spring    |  =========|_|==============|___/=/_/_/_/
spring    |  :: Spring Boot ::                (v2.5.3)
spring    |
spring    | 2021-07-29 11:29:34.556  INFO 1 --- [
main] cn.netkiller.Application                : Starting
Application v0.0.1-SNAPSHOT using Java 16.0.2 on
www.netkiller.cn with PID 1 (/app/test-0.0.1-SNAPSHOT.jar
started by root in /)
spring    | 2021-07-29 11:29:34.559  INFO 1 --- [
main] cn.netkiller.Application                : The following
profiles are active: dev
spring    | 2021-07-29 11:29:35.903  WARN 1 --- [
main] io.undertow.websockets.jsr              : UT026010:
Buffer pool was not set on WebSocketDeploymentInfo, the default
pool will be used
spring    | 2021-07-29 11:29:35.921  INFO 1 --- [
main] io.undertow.servlet                     : Initializing
Spring embedded WebApplicationContext
```

```
spring    | 2021-07-29 11:29:35.921  INFO 1 --- [
main] w.s.c.ServletWebServerApplicationContext : Root
WebApplicationContext: initialization completed in 1274 ms
spring    | 2021-07-29 11:29:36.411  INFO 1 --- [
main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1
endpoint(s) beneath base path '/actuator'
spring    | 2021-07-29 11:29:36.437  INFO 1 --- [
main] io.undertow                              : starting
server: Undertow - 2.2.9.Final
spring    | 2021-07-29 11:29:36.444  INFO 1 --- [
main] org.xnio                                 : XNIO version
3.8.4.Final
spring    | 2021-07-29 11:29:36.451  INFO 1 --- [
main] org.xnio.nio                             : XNIO NIO
Implementation Version 3.8.4.Final
spring    | 2021-07-29 11:29:36.511  INFO 1 --- [
main] org.jboss.threads                        : JBoss Threads
version 3.1.0.Final
spring    | 2021-07-29 11:29:36.547  INFO 1 --- [
main] o.s.b.w.e.undertow.UndertowWebServer     : Undertow
started on port(s) 8080 (http)
spring    | 2021-07-29 11:29:36.560  INFO 1 --- [
main] cn.netkiller.Application                 : Started
Application in 2.48 seconds (JVM running for 2.923)
```

停止容器

```
[root@localhost netkiller.cn]# docker ps | grep spring
8901384d1973   openjdk:latest                "java -jar
/app/test…"   3 minutes ago   Up About a minute   0.0.0.0:8099-
>8080/tcp, :::8099->8080/tcp
spring
[root@localhost netkiller.cn]# docker stop spring
spring
[root@localhost netkiller.cn]# docker ps | grep spring
```

在观察日志

```
spring    | 2021-07-29 11:31:31.807  INFO 1 ---
[ionShutdownHook] io.undertow                                    :
stopping server: Undertow - 2.2.9.Final
spring    | ============================
spring    | Destroying Spring
spring    | ============================
spring exited with code 143
```

现在可以看到 Springboot 是正常退出的

下面我们再做一个实验 docker kill

```
[root@localhost netkiller.cn]# docker-compose start
Starting spring ... done

[root@localhost netkiller.cn]# docker-compose logs -f

[root@localhost netkiller.cn]# docker kill spring
spring
```

此时再观察日志，只输出了一行。

```
spring exited with code 137
```

结论，docker kill = kill -9

现在你应该明白什么时候该使用什么命令终止程序了吧，同时我们在写程序的时候，也应该将程序的运行状态反应出来，在我们停止程序运行的时候，可以去观察进程的状态，而不是半天没有反应，只

能怀疑进程死了，必须执行B计划（kill -9）这会造成很多数据丢失的问题。

4.4. 写入PID文件

我们明白了 kill 的原理后，常常需要与 pid 打交道，使用 ps 命令是可以查看 pid 的，但是当我们运行多个实例的时候会常常搞混，所以最好的方式是让 springboot 把PID写入到文件中。

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.ApplicationPidFileWriter;

@SpringBootApplication

public class Application {

        public static void main(String[] args) {

                System.out.println("Starting...");
                SpringApplication springApplication = new
SpringApplication(Application.class);
                springApplication.addListeners(new
ApplicationPidFileWriter());
                springApplication.run(args);
        }
}
```

程序运行后会在当前目录下产生一个 PID 文件

```
neo@MacBook-Pro-Neo ~/workspace/microservice/test % cat
```

```
application.pid
44027
```

修改 pid 文件位置可以配置 application.properties

```
server.port=8080
spring.pid.file=/tmp/spring.pid
```

在启动的时候指定 pid 文件位置

```
                SpringApplication application = new
 SpringApplication(Application.class);
         application.addListeners(new
 ApplicationPidFileWriter("/tmp/app.pid"));
         application.run();
```

最后说说容器，容器的进程ID永远是 1 所以配置与否自己斟酌。

```
[root@localhost netkiller.cn]# docker exec -it spring cat
/tmp/spring.pid
1
```

5. Properties 配置文件

5.1. application.properties 配置文件

https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html

application.properties 参考

http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

```
#
============================================================
====
# COMMON SPRING BOOT PROPERTIES
#
# This sample file is provided as a guideline. Do NOT copy it
in its
# entirety to your own application.                ^^^
#
============================================================
====


# ----------------------------------------
# CORE PROPERTIES
# ----------------------------------------
debug=false # Enable debug logs.
trace=false # Enable trace logs.

# LOGGING
logging.config= # Location of the logging configuration file.
For instance, `classpath:logback.xml` for Logback.
```

```
logging.exception-conversion-word=%wEx # Conversion word used
when logging exceptions.
logging.file= # Log file name (for instance, `myapp.log`).
Names can be an exact location or relative to the current
directory.
logging.file.max-history=0 # Maximum of archive log files to
keep. Only supported with the default logback setup.
logging.file.max-size=10MB # Maximum log file size. Only
supported with the default logback setup.
logging.level.*= # Log levels severity mapping. For instance,
`logging.level.org.springframework=DEBUG`.
logging.path= # Location of the log file. For instance,
`/var/log`.
logging.pattern.console= # Appender pattern for output to the
console. Supported only with the default Logback setup.
logging.pattern.dateformat=yyyy-MM-dd HH:mm:ss.SSS # Appender
pattern for log date format. Supported only with the default
Logback setup.
logging.pattern.file= # Appender pattern for output to a file.
Supported only with the default Logback setup.
logging.pattern.level=%5p # Appender pattern for log level.
Supported only with the default Logback setup.
logging.register-shutdown-hook=false # Register a shutdown hook
for the logging system when it is initialized.

# AOP
spring.aop.auto=true # Add @EnableAspectJAutoProxy.
spring.aop.proxy-target-class=true # Whether subclass-based
(CGLIB) proxies are to be created (true), as opposed to
standard Java interface-based proxies (false).

# IDENTITY (ContextIdApplicationContextInitializer)
spring.application.name= # Application name.

# ADMIN (SpringApplicationAdminJmxAutoConfiguration)
spring.application.admin.enabled=false # Whether to enable
admin features for the application.
spring.application.admin.jmx-
name=org.springframework.boot:type=Admin,name=SpringApplication
# JMX name of the application admin MBean.

# AUTO-CONFIGURATION
spring.autoconfigure.exclude= # Auto-configuration classes to
exclude.
```

```
# BANNER
spring.banner.charset=UTF-8 # Banner file encoding.
spring.banner.location=classpath:banner.txt # Banner text
resource location.
spring.banner.image.location=classpath:banner.gif # Banner
image file location (jpg or png can also be used).
spring.banner.image.width=76 # Width of the banner image in
chars.
spring.banner.image.height= # Height of the banner image in
chars (default based on image height).
spring.banner.image.margin=2 # Left hand image margin in chars.
spring.banner.image.invert=false # Whether images should be
inverted for dark terminal themes.

# SPRING CORE
spring.beaninfo.ignore=true # Whether to skip search of
BeanInfo classes.

# SPRING CACHE (CacheProperties)
spring.cache.cache-names= # Comma-separated list of cache names
to create if supported by the underlying cache manager.
spring.cache.caffeine.spec= # The spec to use to create caches.
See CaffeineSpec for more details on the spec format.
spring.cache.couchbase.expiration=0ms # Entry expiration. By
default the entries never expire. Note that this value is
ultimately converted to seconds.
spring.cache.ehcache.config= # The location of the
configuration file to use to initialize EhCache.
spring.cache.infinispan.config= # The location of the
configuration file to use to initialize Infinispan.
spring.cache.jcache.config= # The location of the configuration
file to use to initialize the cache manager.
spring.cache.jcache.provider= # Fully qualified name of the
CachingProvider implementation to use to retrieve the JSR-107
compliant cache manager. Needed only if more than one JSR-107
implementation is available on the classpath.
spring.cache.redis.cache-null-values=true # Allow caching null
values.
spring.cache.redis.key-prefix= # Key prefix.
spring.cache.redis.time-to-live=0ms # Entry expiration. By
default the entries never expire.
spring.cache.redis.use-key-prefix=true # Whether to use the key
prefix when writing to Redis.
spring.cache.type= # Cache type. By default, auto-detected
according to the environment.
```

```
# SPRING CONFIG - using environment property only
(ConfigFileApplicationListener)
spring.config.additional-location= # Config file locations used
in addition to the defaults.
spring.config.location= # Config file locations that replace
the defaults.
spring.config.name=application # Config file name.

# HAZELCAST (HazelcastProperties)
spring.hazelcast.config= # The location of the configuration
file to use to initialize Hazelcast.

# PROJECT INFORMATION (ProjectInfoProperties)
spring.info.build.location=classpath:META-INF/build-
info.properties # Location of the generated build-
info.properties file.
spring.info.git.location=classpath:git.properties # Location of
the generated git.properties file.

# JMX
spring.jmx.default-domain= # JMX domain name.
spring.jmx.enabled=true # Expose management beans to the JMX
domain.
spring.jmx.server=mbeanServer # MBeanServer bean name.

# Email (MailProperties)
spring.mail.default-encoding=UTF-8 # Default MimeMessage
encoding.
spring.mail.host= # SMTP server host. For instance,
`smtp.example.com`.
spring.mail.jndi-name= # Session JNDI name. When set, takes
precedence over other Session settings.
spring.mail.password= # Login password of the SMTP server.
spring.mail.port= # SMTP server port.
spring.mail.properties.*= # Additional JavaMail Session
properties.
spring.mail.protocol=smtp # Protocol used by the SMTP server.
spring.mail.-connection=false # Whether to  that the mail
server is available on startup.
spring.mail.username= # Login user of the SMTP server.

# APPLICATION SETTINGS (SpringApplication)
spring.main.banner-mode=console # Mode used to display the
banner when the application runs.
```

```
spring.main.sources= # Sources (class names, package names, or
XML resource locations) to include in the ApplicationContext.
spring.main.web-application-type= # Flag to explicitly request
a specific type of web application. If not set, auto-detected
based on the classpath.

# FILE ENCODING (FileEncodingApplicationListener)
spring.mandatory-file-encoding= # Expected character encoding
the application must use.

# INTERNATIONALIZATION (MessageSourceProperties)
spring.messages.always-use-message-format=false # Whether to
always apply the MessageFormat rules, parsing even messages
without arguments.
spring.messages.basename=messages # Comma-separated list of
basenames (essentially a fully-qualified classpath location),
each following the ResourceBundle convention with relaxed
support for slash based locations.
spring.messages.cache-duration= # Loaded resource bundle files
cache duration. When not set, bundles are cached forever. If a
duration suffix is not specified, seconds will be used.
spring.messages.encoding=UTF-8 # Message bundles encoding.
spring.messages.fallback-to-system-locale=true # Whether to
fall back to the system Locale if no files for a specific
Locale have been found.
spring.messages.use-code-as-default-message=false # Whether to
use the message code as the default message instead of throwing
a "NoSuchMessageException". Recommended during development
only.

# OUTPUT
spring.output.ansi.enabled=detect # Configures the ANSI output.

# PID FILE (ApplicationPidFileWriter)
spring.pid.fail-on-write-error= # Fails if
ApplicationPidFileWriter is used but it cannot write the PID
file.
spring.pid.file= # Location of the PID file to write (if
ApplicationPidFileWriter is used).

# PROFILES
spring.profiles.active= # Comma-separated list of active
profiles. Can be overridden by a command line switch.
spring.profiles.include= # Unconditionally activate the
specified comma-separated list of profiles (or list of profiles
```

```
if using YAML).

# QUARTZ SCHEDULER (QuartzProperties)
spring.quartz.jdbc.comment-prefix=-- # Prefix for single-line
comments in SQL initialization scripts.
spring.quartz.jdbc.initialize-schema=embedded # Database schema
initialization mode.
spring.quartz.jdbc.schema=classpath:org/quartz/impl/jdbcjobstor
e/tables_@@platform@@.sql # Path to the SQL file to use to
initialize the database schema.
spring.quartz.job-store-type=memory # Quartz job store type.
spring.quartz.properties.*= # Additional Quartz Scheduler
properties.

# REACTOR (ReactorCoreProperties)
spring.reactor.stacktrace-mode.enabled=false # Whether Reactor
should collect stacktrace information at runtime.

# SENDGRID (SendGridAutoConfiguration)
spring.sendgrid.api-key= # SendGrid API key.
spring.sendgrid.proxy.host= # SendGrid proxy host.
spring.sendgrid.proxy.port= # SendGrid proxy port.


# ----------------------------------------
# WEB PROPERTIES
# ----------------------------------------

# EMBEDDED SERVER CONFIGURATION (ServerProperties)
server.address= # Network address to which the server should
bind.
server.compression.enabled=false # Whether response compression
is enabled.
server.compression.excluded-user-agents= # List of user-agents
to exclude from compression.
server.compression.mime-
types=text/html,text/xml,text/plain,text/css,text/javascript,ap
plication/javascript # Comma-separated list of MIME types that
should be compressed.
server.compression.min-response-size=2048 # Minimum "Content-
Length" value that is required for compression to be performed.
server.connection-timeout= # Time that connectors wait for
another HTTP request before closing the connection. When not
set, the connector's container-specific default is used. Use a
value of -1 to indicate no (that is, an infinite) timeout.
```

```
server.error.include-exception=false # Include the "exception"
attribute.
server.error.include-stacktrace=never # When to include a
"stacktrace" attribute.
server.error.path=/error # Path of the error controller.
server.error.whitelabel.enabled=true # Whether to enable the
default error page displayed in browsers in case of a server
error.
server.http2.enabled=false # Whether to enable HTTP/2 support,
if the current environment supports it.
server.jetty.acceptors=-1 # Number of acceptor threads to use.
When the value is -1, the default, the number of acceptors is
derived from the operating environment.
server.jetty.accesslog.append=false # Append to log.
server.jetty.accesslog.date-format=dd/MMM/yyyy:HH:mm:ss Z #
Timestamp format of the request log.
server.jetty.accesslog.enabled=false # Enable access log.
server.jetty.accesslog.extended-format=false # Enable extended
NCSA format.
server.jetty.accesslog.file-date-format= # Date format to place
in log file name.
server.jetty.accesslog.filename= # Log filename. If not
specified, logs redirect to "System.err".
server.jetty.accesslog.locale= # Locale of the request log.
server.jetty.accesslog.log-cookies=false # Enable logging of
the request cookies.
server.jetty.accesslog.log-latency=false # Enable logging of
request processing time.
server.jetty.accesslog.log-server=false # Enable logging of the
request hostname.
server.jetty.accesslog.retention-period=31 # Number of days
before rotated log files are deleted.
server.jetty.accesslog.time-zone=GMT # Timezone of the request
log.
server.jetty.max-http-post-size=200000 # Maximum size in bytes
of the HTTP post or put content.
server.jetty.selectors=-1 # Number of selector threads to use.
When the value is -1, the default, the number of selectors is
derived from the operating environment.
server.max-http-header-size=0 # Maximum size, in bytes, of the
HTTP message header.
server.port=8080 # Server HTTP port.
server.server-header= # Value to use for the Server response
header (if empty, no header is sent).
server.use-forward-headers= # Whether X-Forwarded-* headers
```

```
should be applied to the HttpRequest.
server.servlet.context-parameters.*= # Servlet context init
parameters.
server.servlet.context-path= # Context path of the application.
server.servlet.application-display-name=application # Display
name of the application.
server.servlet.jsp.class-
name=org.apache.jasper.servlet.JspServlet # The class name of
the JSP servlet.
server.servlet.jsp.init-parameters.*= # Init parameters used to
configure the JSP servlet.
server.servlet.jsp.registered=true # Whether the JSP servlet is
registered.
server.servlet.path=/ # Path of the main dispatcher servlet.
server.servlet.session.cookie.comment= # Comment for the
session cookie.
server.servlet.session.cookie.domain= # Domain for the session
cookie.
server.servlet.session.cookie.http-only= # "HttpOnly" flag for
the session cookie.
server.servlet.session.cookie.max-age= # Maximum age of the
session cookie. If a duration suffix is not specified, seconds
will be used.
server.servlet.session.cookie.name= # Session cookie name.
server.servlet.session.cookie.path= # Path of the session
cookie.
server.servlet.session.cookie.secure= # "Secure" flag for the
session cookie.
server.servlet.session.persistent=false # Whether to persist
session data between restarts.
server.servlet.session.store-dir= # Directory used to store
session data.
server.servlet.session.timeout= # Session timeout. If a
duration suffix is not specified, seconds will be used.
server.servlet.session.tracking-modes= # Session tracking modes
(one or more of the following: "cookie", "url", "ssl").
server.ssl.ciphers= # Supported SSL ciphers.
server.ssl.client-auth= # Whether client authentication is
wanted ("want") or needed ("need"). Requires a trust store.
server.ssl.enabled= # Enable SSL support.
server.ssl.enabled-protocols= # Enabled SSL protocols.
server.ssl.key-alias= # Alias that identifies the key in the
key store.
server.ssl.key-password= # Password used to access the key in
the key store.
```

```
server.ssl.key-store= # Path to the key store that holds the
SSL certificate (typically a jks file).
server.ssl.key-store-password= # Password used to access the
key store.
server.ssl.key-store-provider= # Provider for the key store.
server.ssl.key-store-type= # Type of the key store.
server.ssl.protocol=TLS # SSL protocol to use.
server.ssl.trust-store= # Trust store that holds SSL
certificates.
server.ssl.trust-store-password= # Password used to access the
trust store.
server.ssl.trust-store-provider= # Provider for the trust
store.
server.ssl.trust-store-type= # Type of the trust store.
server.tomcat.accept-count=100 # Maximum queue length for
incoming connection requests when all possible request
processing threads are in use.
server.tomcat.accesslog.buffered=true # Whether to buffer
output such that it is flushed only periodically.
server.tomcat.accesslog.directory=logs # Directory in which log
files are created. Can be absolute or relative to the Tomcat
base dir.
server.tomcat.accesslog.enabled=false # Enable access log.
server.tomcat.accesslog.file-date-format=.yyyy-MM-dd # Date
format to place in the log file name.
server.tomcat.accesslog.pattern=common # Format pattern for
access logs.
server.tomcat.accesslog.prefix=access_log # Log file name
prefix.
server.tomcat.accesslog.rename-on-rotate=false # Whether to
defer inclusion of the date stamp in the file name until rotate
time.
server.tomcat.accesslog.request-attributes-enabled=false # Set
request attributes for the IP address, Hostname, protocol, and
port used for the request.
server.tomcat.accesslog.rotate=true # Whether to enable access
log rotation.
server.tomcat.accesslog.suffix=.log # Log file name suffix.
server.tomcat.additional-tld-skip-patterns= # Comma-separated
list of additional patterns that match jars to ignore for TLD
scanning.
server.tomcat.background-processor-delay=10 # Delay in seconds
between the invocation of backgroundProcess methods.
server.tomcat.basedir= # Tomcat base directory. If not
specified, a temporary directory is used.
```

```
server.tomcat.internal-
proxies=10\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|\\
                192\\.168\\.\\d{1,3}\\.\\d{1,3}|\\
                169\\.254\\.\\d{1,3}\\.\\d{1,3}|\\
                127\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}|\\
                172\\.1[6-9]{1}\\.\\d{1,3}\\.\\d{1,3}|\\
                172\\.2[0-9]{1}\\.\\d{1,3}\\.\\d{1,3}|\\
                172\\.3[0-1]{1}\\.\\d{1,3}\\.\\d{1,3} # Regular
expression matching trusted IP addresses.
server.tomcat.max-connections=10000 # Maximum number of
connections that the server will accept and process at any
given time.
server.tomcat.max-http-header-size=0 # Maximum size in bytes of
the HTTP message header.
server.tomcat.max-http-post-size=2097152 # Maximum size in
bytes of the HTTP post content.
server.tomcat.max-threads=200 # Maximum amount of worker
threads.
server.tomcat.min-spare-threads=10 # Minimum amount of worker
threads.
server.tomcat.port-header=X-Forwarded-Port # Name of the HTTP
header used to override the original port value.
server.tomcat.protocol-header= # Header that holds the incoming
protocol, usually named "X-Forwarded-Proto".
server.tomcat.protocol-header-https-value=https # Value of the
protocol header indicating whether the incoming request uses
SSL.
server.tomcat.redirect-context-root=true # Whether requests to
the context root should be redirected by appending a / to the
path.
server.tomcat.remote-ip-header= # Name of the HTTP header from
which the remote IP is extracted. For instance, `X-FORWARDED-
FOR`.
server.tomcat.resource.cache-ttl= # Time-to-live of the static
resource cache.
server.tomcat.uri-encoding=UTF-8 # Character encoding to use to
decode the URI.
server.tomcat.use-relative-redirects= # Whether HTTP 1.1 and
later location headers generated by a call to sendRedirect will
use relative or absolute redirects.
server.undertow.accesslog.dir= # Undertow access log directory.
server.undertow.accesslog.enabled=false # Whether to enable the
access log.
server.undertow.accesslog.pattern=common # Format pattern for
access logs.
```

```
server.undertow.accesslog.prefix=access_log. # Log file name
prefix.
server.undertow.accesslog.rotate=true # Whether to enable
access log rotation.
server.undertow.accesslog.suffix=log # Log file name suffix.
server.undertow.buffer-size= # Size of each buffer, in bytes.
server.undertow.direct-buffers= # Allocate buffers outside the
Java heap. The default is derived from the maximum amount of
memory that is available to the JVM.
server.undertow.eager-filter-init=true # Whether servlet
filters should be initialized on startup.
server.undertow.io-threads= # Number of I/O threads to create
for the worker. The default is derived from the number of
available processors.
server.undertow.max-http-post-size=-1 # Maximum size in bytes
of the HTTP post content. When the value is -1, the default,
the size is unlimited.
server.undertow.worker-threads= # Number of worker threads. The
default is 8 times the number of I/O threads.

# FREEMARKER (FreeMarkerProperties)
spring.freemarker.allow-request-override=false # Whether
HttpServletRequest attributes are allowed to override (hide)
controller generated model attributes of the same name.
spring.freemarker.allow-session-override=false # Whether
HttpSession attributes are allowed to override (hide)
controller generated model attributes of the same name.
spring.freemarker.cache=false # Whether to enable template
caching.
spring.freemarker.charset=UTF-8 # Template encoding.
spring.freemarker.check-template-location=true # Whether to
check that the templates location exists.
spring.freemarker.content-type=text/html # Content-Type value.
spring.freemarker.enabled=true # Whether to enable MVC view
resolution for this technology.
spring.freemarker.expose-request-attributes=false # Whether all
request attributes should be added to the model prior to
merging with the template.
spring.freemarker.expose-session-attributes=false # Whether all
HttpSession attributes should be added to the model prior to
merging with the template.
spring.freemarker.expose-spring-macro-helpers=true # Whether to
expose a RequestContext for use by Spring's macro library,
under the name "springMacroRequestContext".
spring.freemarker.prefer-file-system-access=true # Whether to
```

```
prefer file system access for template loading. File system
access enables hot detection of template changes.
spring.freemarker.prefix= # Prefix that gets prepended to view
names when building a URL.
spring.freemarker.request-context-attribute= # Name of the
RequestContext attribute for all views.
spring.freemarker.settings.*= # Well-known FreeMarker keys
which are passed to FreeMarker's Configuration.
spring.freemarker.suffix=.ftl # Suffix that gets appended to
view names when building a URL.
spring.freemarker.template-loader-path=classpath:/templates/ #
Comma-separated list of template paths.
spring.freemarker.view-names= # White list of view names that
can be resolved.

# GROOVY TEMPLATES (GroovyTemplateProperties)
spring.groovy.template.allow-request-override=false # Whether
HttpServletRequest attributes are allowed to override (hide)
controller generated model attributes of the same name.
spring.groovy.template.allow-session-override=false # Whether
HttpSession attributes are allowed to override (hide)
controller generated model attributes of the same name.
spring.groovy.template.cache=false # Whether to enable template
caching.
spring.groovy.template.charset=UTF-8 # Template encoding.
spring.groovy.template.check-template-location=true # Whether
to check that the templates location exists.
spring.groovy.template.configuration.*= # See
GroovyMarkupConfigurer
spring.groovy.template.content-type=text/html # Content-Type
value.
spring.groovy.template.enabled=true # Whether to enable MVC
view resolution for this technology.
spring.groovy.template.expose-request-attributes=false #
Whether all request attributes should be added to the model
prior to merging with the template.
spring.groovy.template.expose-session-attributes=false #
Whether all HttpSession attributes should be added to the model
prior to merging with the template.
spring.groovy.template.expose-spring-macro-helpers=true #
Whether to expose a RequestContext for use by Spring's macro
library, under the name "springMacroRequestContext".
spring.groovy.template.prefix= # Prefix that gets prepended to
view names when building a URL.
spring.groovy.template.request-context-attribute= # Name of the
```

```
RequestContext attribute for all views.
spring.groovy.template.resource-loader-
path=classpath:/templates/ # Template path.
spring.groovy.template.suffix=.tpl # Suffix that gets appended
to view names when building a URL.
spring.groovy.template.view-names= # White list of view names
that can be resolved.

# SPRING HATEOAS (HateoasProperties)
spring.hateoas.use-hal-as-default-json-media-type=true #
Whether application/hal+json responses should be sent to
requests that accept application/json.

# HTTP message conversion
spring.http.converters.preferred-json-mapper= # Preferred JSON
mapper to use for HTTP message conversion. By default, auto-
detected according to the environment.

# HTTP encoding (HttpEncodingProperties)
spring.http.encoding.charset=UTF-8 # Charset of HTTP requests
and responses. Added to the "Content-Type" header if not set
explicitly.
spring.http.encoding.enabled=true # Whether to enable http
encoding support.
spring.http.encoding.force= # Whether to force the encoding to
the configured charset on HTTP requests and responses.
spring.http.encoding.force-request= # Whether to force the
encoding to the configured charset on HTTP requests. Defaults
to true when "force" has not been specified.
spring.http.encoding.force-response= # Whether to force the
encoding to the configured charset on HTTP responses.
spring.http.encoding.mapping= # Locale in which to encode
mapping.

# MULTIPART (MultipartProperties)
spring.servlet.multipart.enabled=true # Whether to enable
support of multipart uploads.
spring.servlet.multipart.file-size-threshold=0 # Threshold
after which files are written to disk. Values can use the
suffixes "MB" or "KB" to indicate megabytes or kilobytes,
respectively.
spring.servlet.multipart.location= # Intermediate location of
uploaded files.
spring.servlet.multipart.max-file-size=1MB # Max file size.
Values can use the suffixes "MB" or "KB" to indicate megabytes
```

```
or kilobytes, respectively.
spring.servlet.multipart.max-request-size=10MB # Max request
size. Values can use the suffixes "MB" or "KB" to indicate
megabytes or kilobytes, respectively.
spring.servlet.multipart.resolve-lazily=false # Whether to
resolve the multipart request lazily at the time of file or
parameter access.

# JACKSON (JacksonProperties)
spring.jackson.date-format= # Date format string or a fully-
qualified date format class name. For instance, `yyyy-MM-dd
HH:mm:ss`.
spring.jackson.default-property-inclusion= # Controls the
inclusion of properties during serialization. Configured with
one of the values in Jackson's JsonInclude.Include enumeration.
spring.jackson.deserialization.*= # Jackson on/off features
that affect the way Java objects are deserialized.
spring.jackson.generator.*= # Jackson on/off features for
generators.
spring.jackson.joda-date-time-format= # Joda date time format
string. If not configured, "date-format" is used as a fallback
if it is configured with a format string.
spring.jackson.locale= # Locale used for formatting.
spring.jackson.mapper.*= # Jackson general purpose on/off
features.
spring.jackson.parser.*= # Jackson on/off features for parsers.
spring.jackson.property-naming-strategy= # One of the constants
on Jackson's PropertyNamingStrategy. Can also be a fully-
qualified class name of a PropertyNamingStrategy subclass.
spring.jackson.serialization.*= # Jackson on/off features that
affect the way Java objects are serialized.
spring.jackson.time-zone= #  Time zone used when formatting
dates. For instance, "America/Los_Angeles" or "GMT+10".

# GSON (GsonProperties)
spring.gson.date-format= # Format to use when serializing Date
objects.
spring.gson.disable-html-escaping= # Whether to disable the
escaping of HTML characters such as '<', '>', etc.
spring.gson.disable-inner-class-serialization= # Whether to
exclude inner classes during serialization.
spring.gson.enable-complex-map-key-serialization= # Whether to
enable serialization of complex map keys (i.e. non-primitives).
spring.gson.exclude-fields-without-expose-annotation= # Whether
to exclude all fields from consideration for serialization or
```

```
deserialization that do not have the "Expose" annotation.
spring.gson.field-naming-policy= # Naming policy that should be
applied to an object's field during serialization and
deserialization.
spring.gson.generate-non-executable-json= # Whether to generate
non executable JSON by prefixing the output with some special
text.
spring.gson.lenient= # Whether to be lenient about parsing JSON
that doesn't conform to RFC 4627.
spring.gson.long-serialization-policy= # Serialization policy
for Long and long types.
spring.gson.pretty-printing= # Whether to output serialized
JSON that fits in a page for pretty printing.
spring.gson.serialize-nulls= # Whether to serialize null
fields.

# JERSEY (JerseyProperties)
spring.jersey.application-path= # Path that serves as the base
URI for the application. If specified, overrides the value of
"@ApplicationPath".
spring.jersey.filter.order=0 # Jersey filter chain order.
spring.jersey.init.*= # Init parameters to pass to Jersey
through the servlet or filter.
spring.jersey.servlet.load-on-startup=-1 # Load on startup
priority of the Jersey servlet.
spring.jersey.type=servlet # Jersey integration type.

# SPRING LDAP (LdapProperties)
spring.ldap.anonymous-read-only=false # Whether read-only
operations should use an anonymous environment.
spring.ldap.base= # Base suffix from which all operations
should originate.
spring.ldap.base-environment.*= # LDAP specification settings.
spring.ldap.password= # Login password of the server.
spring.ldap.urls= # LDAP URLs of the server.
spring.ldap.username= # Login username of the server.

# EMBEDDED LDAP (EmbeddedLdapProperties)
spring.ldap.embedded.base-dn= # List of base DNs.
spring.ldap.embedded.credential.username= # Embedded LDAP
username.
spring.ldap.embedded.credential.password= # Embedded LDAP
password.
spring.ldap.embedded.ldif=classpath:schema.ldif # Schema (LDIF)
script resource reference.
```

```
spring.ldap.embedded.port=0 # Embedded LDAP port.
spring.ldap.embedded.validation.enabled=true # Whether to
enable LDAP schema validation.
spring.ldap.embedded.validation.schema= # Path to the custom
schema.

# MUSTACHE TEMPLATES (MustacheAutoConfiguration)
spring.mustache.allow-request-override=false # Whether
HttpServletRequest attributes are allowed to override (hide)
controller generated model attributes of the same name.
spring.mustache.allow-session-override=false # Whether
HttpSession attributes are allowed to override (hide)
controller generated model attributes of the same name.
spring.mustache.cache=false # Whether to enable template
caching.
spring.mustache.charset=UTF-8 # Template encoding.
spring.mustache.check-template-location=true # Whether to check
that the templates location exists.
spring.mustache.content-type=text/html # Content-Type value.
spring.mustache.enabled=true # Whether to enable MVC view
resolution for this technology.
spring.mustache.expose-request-attributes=false # Whether all
request attributes should be added to the model prior to
merging with the template.
spring.mustache.expose-session-attributes=false # Whether all
HttpSession attributes should be added to the model prior to
merging with the template.
spring.mustache.expose-spring-macro-helpers=true # Whether to
expose a RequestContext for use by Spring's macro library,
under the name "springMacroRequestContext".
spring.mustache.prefix=classpath:/templates/ # Prefix to apply
to template names.
spring.mustache.request-context-attribute= # Name of the
RequestContext attribute for all views.
spring.mustache.suffix=.mustache # Suffix to apply to template
names.
spring.mustache.view-names= # White list of view names that can
be resolved.

# SPRING MVC (WebMvcProperties)
spring.mvc.async.request-timeout= # Amount of time before
asynchronous request handling times out.
spring.mvc.contentnegotiation.favor-parameter=false # Whether a
request parameter ("format" by default) should be used to
determine the requested media type.
```

```
spring.mvc.contentnegotiation.favor-path-extension=false # Whether the path extension in the URL path should be used to determine the requested media type.
spring.mvc.contentnegotiation.media-types.*= # Map file extensions to media types for content negotiation. For instance, yml to text/yaml.
spring.mvc.contentnegotiation.parameter-name= # Query parameter name to use when "favor-parameter" is enabled.
spring.mvc.date-format= # Date format to use. For instance, `dd/MM/yyyy`.
spring.mvc.dispatch-trace-request=false # Whether to dispatch TRACE requests to the FrameworkServlet doService method.
spring.mvc.dispatch-options-request=true # Whether to dispatch OPTIONS requests to the FrameworkServlet doService method.
spring.mvc.favicon.enabled=true # Whether to enable resolution of favicon.ico.
spring.mvc.formcontent.putfilter.enabled=true # Whether to enable Spring's HttpPutFormContentFilter.
spring.mvc.ignore-default-model-on-redirect=true # Whether the content of the "default" model should be ignored during redirect scenarios.
spring.mvc.locale= # Locale to use. By default, this locale is overridden by the "Accept-Language" header.
spring.mvc.locale-resolver=accept-header # Define how the locale should be resolved.
spring.mvc.log-resolved-exception=false # Whether to enable warn logging of exceptions resolved by a "HandlerExceptionResolver".
spring.mvc.message-codes-resolver-format= # Formatting strategy for message codes. For instance, `PREFIX_ERROR_CODE`.
spring.mvc.pathmatch.use-registered-suffix-pattern=false # Whether suffix pattern matching should work only against extensions registered with "spring.mvc.contentnegotiation.media-types.*".
spring.mvc.pathmatch.use-suffix-pattern=false # Whether to use suffix pattern match (".*") when matching patterns to requests.
spring.mvc.servlet.load-on-startup=-1 # Load on startup priority of the dispatcher servlet.
spring.mvc.static-path-pattern=/** # Path pattern used for static resources.
spring.mvc.throw-exception-if-no-handler-found=false # Whether a "NoHandlerFoundException" should be thrown if no Handler was found to process a request.
spring.mvc.view.prefix= # Spring MVC view prefix.
spring.mvc.view.suffix= # Spring MVC view suffix.
```

```
# SPRING RESOURCES HANDLING (ResourceProperties)
spring.resources.add-mappings=true # Whether to enable default
resource handling.
spring.resources.cache.cachecontrol.cache-private= # Indicate
that the response message is intended for a single user and
must not be stored by a shared cache.
spring.resources.cache.cachecontrol.cache-public= # Indicate
that any cache may store the response.
spring.resources.cache.cachecontrol.max-age= # Maximum time the
response should be cached, in seconds if no duration suffix is
not specified.
spring.resources.cache.cachecontrol.must-revalidate= # Indicate
that once it has become stale, a cache must not use the
response without re-validating it with the server.
spring.resources.cache.cachecontrol.no-cache= # Indicate that
the cached response can be reused only if re-validated with the
server.
spring.resources.cache.cachecontrol.no-store= # Indicate to not
cache the response in any case.
spring.resources.cache.cachecontrol.no-transform= # Indicate
intermediaries (caches and others) that they should not
transform the response content.
spring.resources.cache.cachecontrol.proxy-revalidate= # Same
meaning as the "must-revalidate" directive, except that it does
not apply to private caches.
spring.resources.cache.cachecontrol.s-max-age= # Maximum time
the response should be cached by shared caches, in seconds if
no duration suffix is not specified.
spring.resources.cache.cachecontrol.stale-if-error= # Maximum
time the response may be used when errors are encountered, in
seconds if no duration suffix is not specified.
spring.resources.cache.cachecontrol.stale-while-revalidate= #
Maximum time the response can be served after it becomes stale,
in seconds if no duration suffix is not specified.
spring.resources.cache.period= # Cache period for the resources
served by the resource handler. If a duration suffix is not
specified, seconds will be used.
spring.resources.chain.cache=true # Whether to enable caching
in the Resource chain.
spring.resources.chain.enabled= # Whether to enable the Spring
Resource Handling chain. By default, disabled unless at least
one strategy has been enabled.
spring.resources.chain.gzipped=false # Whether to enable
resolution of already gzipped resources.
```

```
spring.resources.chain.html-application-cache=false # Whether
to enable HTML5 application cache manifest rewriting.
spring.resources.chain.strategy.content.enabled=false # Whether
to enable the content Version Strategy.
spring.resources.chain.strategy.content.paths=/** # Comma-
separated list of patterns to apply to the content Version
Strategy.
spring.resources.chain.strategy.fixed.enabled=false # Whether
to enable the fixed Version Strategy.
spring.resources.chain.strategy.fixed.paths=/** # Comma-
separated list of patterns to apply to the fixed Version
Strategy.
spring.resources.chain.strategy.fixed.version= # Version string
to use for the fixed Version Strategy.
spring.resources.static-locations=classpath:/META-
INF/resources/,classpath:/resources/,classpath:/static/,classpa
th:/public/ # Locations of static resources.

# SPRING SESSION (SessionProperties)
spring.session.store-type= # Session store type.
spring.session.timeout= # Session timeout. If a duration suffix
is not specified, seconds will be used.
spring.session.servlet.filter-order=-2147483598 # Session
repository filter order.
spring.session.servlet.filter-dispatcher-
types=async,error,request # Session repository filter
dispatcher types.

# SPRING SESSION HAZELCAST (HazelcastSessionProperties)
spring.session.hazelcast.flush-mode=on-save # Sessions flush
mode.
spring.session.hazelcast.map-name=spring:session:sessions #
Name of the map used to store sessions.

# SPRING SESSION JDBC (JdbcSessionProperties)
spring.session.jdbc.cleanup-cron=0 * * * * * # Cron expression
for expired session cleanup job.
spring.session.jdbc.initialize-schema=embedded # Database
schema initialization mode.
spring.session.jdbc.schema=classpath:org/springframework/sessio
n/jdbc/schema-@@platform@@.sql # Path to the SQL file to use to
initialize the database schema.
spring.session.jdbc.table-name=SPRING_SESSION # Name of the
database table used to store sessions.
```

```
# SPRING SESSION MONGODB (MongoSessionProperties)
spring.session.mongodb.collection-name=sessions # Collection
name used to store sessions.

# SPRING SESSION REDIS (RedisSessionProperties)
spring.session.redis.cleanup-cron=0 * * * * * # Cron expression
for expired session cleanup job.
spring.session.redis.flush-mode=on-save # Sessions flush mode.
spring.session.redis.namespace=spring:session # Namespace for
keys used to store sessions.

# THYMELEAF (ThymeleafAutoConfiguration)
spring.thymeleaf.cache=true # Whether to enable template
caching.
spring.thymeleaf.check-template=true # Whether to check that
the template exists before rendering it.
spring.thymeleaf.check-template-location=true # Whether to
check that the templates location exists.
spring.thymeleaf.enabled=true # Whether to enable Thymeleaf
view resolution for Web frameworks.
spring.thymeleaf.enable-spring-el-compiler=false # Enable the
SpringEL compiler in SpringEL expressions.
spring.thymeleaf.encoding=UTF-8 # Template files encoding.
spring.thymeleaf.excluded-view-names= # Comma-separated list of
view names (patterns allowed) that should be excluded from
resolution.
spring.thymeleaf.mode=HTML # Template mode to be applied to
templates. See also Thymeleaf's TemplateMode enum.
spring.thymeleaf.prefix=classpath:/templates/ # Prefix that
gets prepended to view names when building a URL.
spring.thymeleaf.reactive.chunked-mode-view-names= # Comma-
separated list of view names (patterns allowed) that should be
the only ones executed in CHUNKED mode when a max chunk size is
set.
spring.thymeleaf.reactive.full-mode-view-names= # Comma-
separated list of view names (patterns allowed) that should be
executed in FULL mode even if a max chunk size is set.
spring.thymeleaf.reactive.max-chunk-size=0 # Maximum size of
data buffers used for writing to the response, in bytes.
spring.thymeleaf.reactive.media-types= # Media types supported
by the view technology.
spring.thymeleaf.servlet.content-type=text/html # Content-Type
value written to HTTP responses.
spring.thymeleaf.suffix=.html # Suffix that gets appended to
view names when building a URL.
```

```
spring.thymeleaf.template-resolver-order= # Order of the
template resolver in the chain.
spring.thymeleaf.view-names= # Comma-separated list of view
names (patterns allowed) that can be resolved.

# SPRING WEBFLUX (WebFluxProperties)
spring.webflux.date-format= # Date format to use. For instance,
`dd/MM/yyyy`.
spring.webflux.static-path-pattern=/** # Path pattern used for
static resources.

# SPRING WEB SERVICES (WebServicesProperties)
spring.webservices.path=/services # Path that serves as the
base URI for the services.
spring.webservices.servlet.init= # Servlet init parameters to
pass to Spring Web Services.
spring.webservices.servlet.load-on-startup=-1 # Load on startup
priority of the Spring Web Services servlet.
spring.webservices.wsdl-locations= # Comma-separated list of
locations of WSDLs and accompanying XSDs to be exposed as
beans.



# ----------------------------------------
# SECURITY PROPERTIES
# ----------------------------------------
# SECURITY (SecurityProperties)
spring.security.filter.order=-100 # Security filter chain
order.
spring.security.filter.dispatcher-types=async,error,request #
Security filter chain dispatcher types.
spring.security.user.name=user # Default user name.
spring.security.user.password= # Password for the default user
name.
spring.security.user.roles= # Granted roles for the default
user name.

# SECURITY OAUTH2 CLIENT (OAuth2ClientProperties)
spring.security.oauth2.client.provider.*= # OAuth provider
details.
spring.security.oauth2.client.registration.*= # OAuth client
registrations.

# ----------------------------------------
```

```
# DATA PROPERTIES
# ----------------------------------------

# FLYWAY (FlywayProperties)
spring.flyway.baseline-description= #
spring.flyway.baseline-on-migrate= #
spring.flyway.baseline-version=1 # Version to start migration
spring.flyway.check-location=true # Whether to check that
migration scripts location exists.
spring.flyway.clean-disabled= #
spring.flyway.clean-on-validation-error= #
spring.flyway.dry-run-output= #
spring.flyway.enabled=true # Whether to enable flyway.
spring.flyway.encoding= #
spring.flyway.error-handlers= #
spring.flyway.group= #
spring.flyway.ignore-future-migrations= #
spring.flyway.ignore-missing-migrations= #
spring.flyway.init-sqls= # SQL statements to execute to
initialize a connection immediately after obtaining it.
spring.flyway.installed-by= #
spring.flyway.locations=classpath:db/migration # The locations
of migrations scripts.
spring.flyway.mixed= #
spring.flyway.out-of-order= #
spring.flyway.password= # JDBC password to use if you want
Flyway to create its own DataSource.
spring.flyway.placeholder-prefix= #
spring.flyway.placeholder-replacement= #
spring.flyway.placeholder-suffix= #
spring.flyway.placeholders.*= #
spring.flyway.repeatable-sql-migration-prefix= #
spring.flyway.schemas= # schemas to update
spring.flyway.skip-default-callbacks= #
spring.flyway.skip-default-resolvers= #
spring.flyway.sql-migration-prefix=V #
spring.flyway.sql-migration-separator= #
spring.flyway.sql-migration-suffix=.sql #
spring.flyway.sql-migration-suffixes= #
spring.flyway.table= #
spring.flyway.target= #
spring.flyway.undo-sql-migration-prefix= #
spring.flyway.url= # JDBC url of the database to migrate. If
not set, the primary configured data source is used.
spring.flyway.user= # Login user of the database to migrate.
```

```
spring.flyway.validate-on-migrate= #

# LIQUIBASE (LiquibaseProperties)
spring.liquibase.change-
log=classpath:/db/changelog/db.changelog-master.yaml # Change
log configuration path.
spring.liquibase.check-change-log-location=true # Whether to
check that the change log location exists.
spring.liquibase.contexts= # Comma-separated list of runtime
contexts to use.
spring.liquibase.default-schema= # Default database schema.
spring.liquibase.drop-first=false # Whether to first drop the
database schema.
spring.liquibase.enabled=true # Whether to enable Liquibase
support.
spring.liquibase.labels= # Comma-separated list of runtime
labels to use.
spring.liquibase.parameters.*= # Change log parameters.
spring.liquibase.password= # Login password of the database to
migrate.
spring.liquibase.rollback-file= # File to which rollback SQL is
written when an update is performed.
spring.liquibase.url= # JDBC URL of the database to migrate. If
not set, the primary configured data source is used.
spring.liquibase.user= # Login user of the database to migrate.

# COUCHBASE (CouchbaseProperties)
spring.couchbase.bootstrap-hosts= # Couchbase nodes (host or IP
address) to bootstrap from.
spring.couchbase.bucket.name=default # Name of the bucket to
connect to.
spring.couchbase.bucket.password=  # Password of the bucket.
spring.couchbase.env.endpoints.key-value=1 # Number of sockets
per node against the key/value service.
spring.couchbase.env.endpoints.queryservice.min-endpoints=1 #
Minimum number of sockets per node.
spring.couchbase.env.endpoints.queryservice.max-endpoints=1 #
Maximum number of sockets per node.
spring.couchbase.env.endpoints.viewservice.min-endpoints=1 #
Minimum number of sockets per node.
spring.couchbase.env.endpoints.viewservice.max-endpoints=1 #
Maximum number of sockets per node.
spring.couchbase.env.ssl.enabled= # Whether to enable SSL
support. Enabled automatically if a "keyStore" is provided
unless specified otherwise.
```

```
spring.couchbase.env.ssl.key-store= # Path to the JVM key store
that holds the certificates.
spring.couchbase.env.ssl.key-store-password= # Password used to
access the key store.
spring.couchbase.env.timeouts.connect=5000ms # Bucket
connections timeouts.
spring.couchbase.env.timeouts.key-value=2500ms # Blocking
operations performed on a specific key timeout.
spring.couchbase.env.timeouts.query=7500ms # N1QL query
operations timeout.
spring.couchbase.env.timeouts.socket-connect=1000ms # Socket
connect connections timeout.
spring.couchbase.env.timeouts.view=7500ms # Regular and
geospatial view operations timeout.

# DAO (PersistenceExceptionTranslationAutoConfiguration)
spring.dao.exceptiontranslation.enabled=true # Whether to
enable the PersistenceExceptionTranslationPostProcessor.

# CASSANDRA (CassandraProperties)
spring.data.cassandra.cluster-name= # Name of the Cassandra
cluster.
spring.data.cassandra.compression=none # Compression supported
by the Cassandra binary protocol.
spring.data.cassandra.connect-timeout= # Socket option:
connection time out.
spring.data.cassandra.consistency-level= # Queries consistency
level.
spring.data.cassandra.contact-points=localhost # Cluster node
addresses.
spring.data.cassandra.fetch-size= # Queries default fetch size.
spring.data.cassandra.keyspace-name= # Keyspace name to use.
spring.data.cassandra.load-balancing-policy= # Class name of
the load balancing policy.
spring.data.cassandra.port= # Port of the Cassandra server.
spring.data.cassandra.password= # Login password of the server.
spring.data.cassandra.pool.heartbeat-interval=30s # Heartbeat
interval after which a message is sent on an idle connection to
make sure it's still alive. If a duration suffix is not
specified, seconds will be used.
spring.data.cassandra.pool.idle-timeout=120s # Idle timeout
before an idle connection is removed. If a duration suffix is
not specified, seconds will be used.
spring.data.cassandra.pool.max-queue-size=256 # Maximum number
of requests that get queued if no connection is available.
```

```
spring.data.cassandra.pool.pool-timeout=5000ms # Pool timeout
when trying to acquire a connection from a host's pool.
spring.data.cassandra.read-timeout= # Socket option: read time
out.
spring.data.cassandra.reconnection-policy= # Reconnection
policy class.
spring.data.cassandra.repositories.type=auto # Type of
Cassandra repositories to enable.
spring.data.cassandra.retry-policy= # Class name of the retry
policy.
spring.data.cassandra.serial-consistency-level= # Queries
serial consistency level.
spring.data.cassandra.schema-action=none # Schema action to
take at startup.
spring.data.cassandra.ssl=false # Enable SSL support.
spring.data.cassandra.username= # Login user of the server.

# DATA COUCHBASE (CouchbaseDataProperties)
spring.data.couchbase.auto-index=false # Automatically create
views and indexes.
spring.data.couchbase.consistency=read-your-own-writes #
Consistency to apply by default on generated queries.
spring.data.couchbase.repositories.type=auto # Type of
Couchbase repositories to enable.

# ELASTICSEARCH (ElasticsearchProperties)
spring.data.elasticsearch.cluster-name=elasticsearch #
Elasticsearch cluster name.
spring.data.elasticsearch.cluster-nodes= # Comma-separated list
of cluster node addresses.
spring.data.elasticsearch.properties.*= # Additional properties
used to configure the client.
spring.data.elasticsearch.repositories.enabled=true # Whether
to enable Elasticsearch repositories.

# DATA LDAP
spring.data.ldap.repositories.enabled=true # Whether to enable
LDAP repositories.

# MONGODB (MongoProperties)
spring.data.mongodb.authentication-database= # Authentication
database name.
spring.data.mongodb.database= # Database name.
spring.data.mongodb.field-naming-strategy= # Fully qualified
name of the FieldNamingStrategy to use.
```

```
spring.data.mongodb.grid-fs-database= # GridFS database name.
spring.data.mongodb.host= # Mongo server host. Cannot be set
with URI.
spring.data.mongodb.password= # Login password of the mongo
server. Cannot be set with URI.
spring.data.mongodb.port= # Mongo server port. Cannot be set
with URI.
spring.data.mongodb.repositories.type=auto # Type of Mongo
repositories to enable.
spring.data.mongodb.uri=mongodb://localhost/ # Mongo database
URI. Cannot be set with host, port and credentials.
spring.data.mongodb.username= # Login user of the mongo server.
Cannot be set with URI.

# DATA REDIS
spring.data.redis.repositories.enabled=true # Whether to enable
Redis repositories.

# NEO4J (Neo4jProperties)
spring.data.neo4j.auto-index=none # Auto index mode.
spring.data.neo4j.embedded.enabled=true # Whether to enable
embedded mode if the embedded driver is available.
spring.data.neo4j.open-in-view=true # Register
OpenSessionInViewInterceptor. Binds a Neo4j Session to the
thread for the entire processing of the request.
spring.data.neo4j.password= # Login password of the server.
spring.data.neo4j.repositories.enabled=true # Whether to enable
Neo4j repositories.
spring.data.neo4j.uri= # URI used by the driver. Auto-detected
by default.
spring.data.neo4j.username= # Login user of the server.

# DATA REST (RepositoryRestProperties)
spring.data.rest.base-path= # Base path to be used by Spring
Data REST to expose repository resources.
spring.data.rest.default-media-type= # Content type to use as a
default when none is specified.
spring.data.rest.default-page-size= # Default size of pages.
spring.data.rest.detection-strategy=default # Strategy to use
to determine which repositories get exposed.
spring.data.rest.enable-enum-translation= # Whether to enable
enum value translation through the Spring Data REST default
resource bundle.
spring.data.rest.limit-param-name= # Name of the URL query
string parameter that indicates how many results to return at
```

```
once.
spring.data.rest.max-page-size= # Maximum size of pages.
spring.data.rest.page-param-name= # Name of the URL query
string parameter that indicates what page to return.
spring.data.rest.return-body-on-create= # Whether to return a
response body after creating an entity.
spring.data.rest.return-body-on-update= # Whether to return a
response body after updating an entity.
spring.data.rest.sort-param-name= # Name of the URL query
string parameter that indicates what direction to sort results.

# SOLR (SolrProperties)
spring.data.solr.host=http://127.0.0.1:8983/solr # Solr host.
Ignored if "zk-host" is set.
spring.data.solr.repositories.enabled=true # Whether to enable
Solr repositories.
spring.data.solr.zk-host= # ZooKeeper host address in the form
HOST:PORT.

# DATA WEB (SpringDataWebProperties)
spring.data.web.pageable.default-page-size=20 # Default page
size.
spring.data.web.pageable.max-page-size=2000 # Maximum page size
to be accepted.
spring.data.web.pageable.one-indexed-parameters=false # Whether
to expose and assume 1-based page number indexes.
spring.data.web.pageable.page-parameter=page # Page index
parameter name.
spring.data.web.pageable.prefix= # General prefix to be
prepended to the page number and page size parameters.
spring.data.web.pageable.qualifier-delimiter=_ # Delimiter to
be used between the qualifier and the actual page number and
size properties.
spring.data.web.pageable.size-parameter=size # Page size
parameter name.
spring.data.web.sort.sort-parameter=sort # Sort parameter name.

# DATASOURCE (DataSourceAutoConfiguration &
DataSourceProperties)
spring.datasource.continue-on-error=false # Whether to stop if
an error occurs while initializing the database.
spring.datasource.data= # Data (DML) script resource
references.
spring.datasource.data-username= # Username of the database to
execute DML scripts (if different).
```

```
spring.datasource.data-password= # Password of the database to
execute DML scripts (if different).
spring.datasource.dbcp2.*= # Commons DBCP2 specific settings
spring.datasource.driver-class-name= # Fully qualified name of
the JDBC driver. Auto-detected based on the URL by default.
spring.datasource.generate-unique-name=false # Whether to
generate a random datasource name.
spring.datasource.hikari.*= # Hikari specific settings
spring.datasource.initialization-mode=embedded # Initialize the
datasource with available DDL and DML scripts.
spring.datasource.jmx-enabled=false # Whether to enable JMX
support (if provided by the underlying pool).
spring.datasource.jndi-name= # JNDI location of the datasource.
Class, url, username & password are ignored when set.
spring.datasource.name= # Name of the datasource. Default to
"db" when using an embedded database.
spring.datasource.password= # Login password of the database.
spring.datasource.platform=all # Platform to use in the DDL or
DML scripts (such as schema-${platform}.sql or
data-${platform}.sql).
spring.datasource.schema= # Schema (DDL) script resource
references.
spring.datasource.schema-username= # Username of the database
to execute DDL scripts (if different).
spring.datasource.schema-password= # Password of the database
to execute DDL scripts (if different).
spring.datasource.separator=; # Statement separator in SQL
initialization scripts.
spring.datasource.sql-script-encoding= # SQL scripts encoding.
spring.datasource.tomcat.*= # Tomcat datasource specific
settings
spring.datasource.type= # Fully qualified name of the
connection pool implementation to use. By default, it is auto-
detected from the classpath.
spring.datasource.url= # JDBC URL of the database.
spring.datasource.username= # Login username of the database.
spring.datasource.xa.data-source-class-name= # XA datasource
fully qualified name.
spring.datasource.xa.properties= # Properties to pass to the XA
data source.

# JEST (Elasticsearch HTTP client) (JestProperties)
spring.elasticsearch.jest.connection-timeout=3s # Connection
timeout.
spring.elasticsearch.jest.multi-threaded=true # Whether to
```

```
enable connection requests from multiple execution threads.
spring.elasticsearch.jest.password= # Login password.
spring.elasticsearch.jest.proxy.host= # Proxy host the HTTP
client should use.
spring.elasticsearch.jest.proxy.port= # Proxy port the HTTP
client should use.
spring.elasticsearch.jest.read-timeout=3s # Read timeout.
spring.elasticsearch.jest.uris=http://localhost:9200 # Comma-
separated list of the Elasticsearch instances to use.
spring.elasticsearch.jest.username= # Login username.

# H2 Web Console (H2ConsoleProperties)
spring.h2.console.enabled=false # Whether to enable the
console.
spring.h2.console.path=/h2-console # Path at which the console
is available.
spring.h2.console.settings.trace=false # Whether to enable
trace output.
spring.h2.console.settings.web-allow-others=false # Whether to
enable remote access.

# InfluxDB (InfluxDbProperties)
spring.influx.password= # Login password.
spring.influx.url= # URL of the InfluxDB instance to which to
connect.
spring.influx.user= # Login user.

# JOOQ (JooqProperties)
spring.jooq.sql-dialect= # SQL dialect to use. Auto-detected by
default.

# JDBC (JdbcProperties)
spring.jdbc.template.fetch-size=-1 # Number of rows that should
be fetched from the database when more rows are needed.
spring.jdbc.template.max-rows=-1 # Maximum number of rows.
spring.jdbc.template.query-timeout= # Query timeout. Default is
to use the JDBC driver's default configuration. If a duration
suffix is not specified, seconds will be used.

# JPA (JpaBaseConfiguration, HibernateJpaAutoConfiguration)
spring.data.jpa.repositories.enabled=true # Whether to enable
JPA repositories.
spring.jpa.database= # Target database to operate on, auto-
detected by default. Can be alternatively set using the
"databasePlatform" property.
```

```
spring.jpa.database-platform= # Name of the target database to
operate on, auto-detected by default. Can be alternatively set
using the "Database" enum.
spring.jpa.generate-ddl=false # Whether to initialize the
schema on startup.
spring.jpa.hibernate.ddl-auto= # DDL mode. This is actually a
shortcut for the "hibernate.hbm2ddl.auto" property. Defaults to
"create-drop" when using an embedded database and no schema
manager was detected. Otherwise, defaults to "none".
spring.jpa.hibernate.naming.implicit-strategy= # Fully
qualified name of the implicit naming strategy.
spring.jpa.hibernate.naming.physical-strategy= # Fully
qualified name of the physical naming strategy.
spring.jpa.hibernate.use-new-id-generator-mappings= # Whether
to use Hibernate's newer IdentifierGenerator for AUTO, TABLE
and SEQUENCE.
spring.jpa.mapping-resources= # Mapping resources (equivalent
to "mapping-file" entries in persistence.xml).
spring.jpa.open-in-view=true # Register
OpenEntityManagerInViewInterceptor. Binds a JPA EntityManager
to the thread for the entire processing of the request.
spring.jpa.properties.*= # Additional native properties to set
on the JPA provider.
spring.jpa.show-sql=false # Whether to enable logging of SQL
statements.

# JTA (JtaAutoConfiguration)
spring.jta.enabled=true # Whether to enable JTA support.
spring.jta.log-dir= # Transaction logs directory.
spring.jta.transaction-manager-id= # Transaction manager unique
identifier.

# ATOMIKOS (AtomikosProperties)
spring.jta.atomikos.connectionfactory.borrow-connection-
timeout=30 # Timeout, in seconds, for borrowing connections
from the pool.
spring.jta.atomikos.connectionfactory.ignore-session-
transacted-flag=true # Whether to ignore the transacted flag
when creating session.
spring.jta.atomikos.connectionfactory.local-transaction-
mode=false # Whether local transactions are desired.
spring.jta.atomikos.connectionfactory.maintenance-interval=60 #
The time, in seconds, between runs of the pool's maintenance
thread.
spring.jta.atomikos.connectionfactory.max-idle-time=60 # The
```

```
time, in seconds, after which connections are cleaned up from
the pool.
spring.jta.atomikos.connectionfactory.max-lifetime=0 # The
time, in seconds, that a connection can be pooled for before
being destroyed. 0 denotes no limit.
spring.jta.atomikos.connectionfactory.max-pool-size=1 # The
maximum size of the pool.
spring.jta.atomikos.connectionfactory.min-pool-size=1 # The
minimum size of the pool.
spring.jta.atomikos.connectionfactory.reap-timeout=0 # The reap
timeout, in seconds, for borrowed connections. 0 denotes no
limit.
spring.jta.atomikos.connectionfactory.unique-resource-
name=jmsConnectionFactory # The unique name used to identify
the resource during recovery.
spring.jta.atomikos.connectionfactory.xa-connection-factory-
class-name= # Vendor-specific implementation of
XAConnectionFactory.
spring.jta.atomikos.connectionfactory.xa-properties= # Vendor-
specific XA properties.
spring.jta.atomikos.datasource.borrow-connection-timeout=30 #
Timeout, in seconds, for borrowing connections from the pool.
spring.jta.atomikos.datasource.concurrent-connection-
validation= # Whether to use concurrent connection validation.
spring.jta.atomikos.datasource.default-isolation-level= #
Default isolation level of connections provided by the pool.
spring.jta.atomikos.datasource.login-timeout= # Timeout, in
seconds, for establishing a database connection.
spring.jta.atomikos.datasource.maintenance-interval=60 # The
time, in seconds, between runs of the pool's maintenance
thread.
spring.jta.atomikos.datasource.max-idle-time=60 # The time, in
seconds, after which connections are cleaned up from the pool.
spring.jta.atomikos.datasource.max-lifetime=0 # The time, in
seconds, that a connection can be pooled for before being
destroyed. 0 denotes no limit.
spring.jta.atomikos.datasource.max-pool-size=1 # The maximum
size of the pool.
spring.jta.atomikos.datasource.min-pool-size=1 # The minimum
size of the pool.
spring.jta.atomikos.datasource.reap-timeout=0 # The reap
timeout, in seconds, for borrowed connections. 0 denotes no
limit.
spring.jta.atomikos.datasource.-query= # SQL query or statement
used to validate a connection before returning it.
```

```
spring.jta.atomikos.datasource.unique-resource-name=dataSource
# The unique name used to identify the resource during
recovery.
spring.jta.atomikos.datasource.xa-data-source-class-name= #
Vendor-specific implementation of XAConnectionFactory.
spring.jta.atomikos.datasource.xa-properties= # Vendor-specific
XA properties.
spring.jta.atomikos.properties.allow-sub-transactions=true #
Specify whether sub-transactions are allowed.
spring.jta.atomikos.properties.checkpoint-interval=500 #
Interval between checkpoints, expressed as the number of log
writes between two checkpoints.
spring.jta.atomikos.properties.default-jta-timeout=10000ms #
Default timeout for JTA transactions.
spring.jta.atomikos.properties.default-max-wait-time-on-
shutdown=9223372036854775807 # How long should normal shutdown
(no-force) wait for transactions to complete.
spring.jta.atomikos.properties.enable-logging=true # Whether to
enable disk logging.
spring.jta.atomikos.properties.force-shutdown-on-vm-exit=false
# Whether a VM shutdown should trigger forced shutdown of the
transaction core.
spring.jta.atomikos.properties.log-base-dir= # Directory in
which the log files should be stored.
spring.jta.atomikos.properties.log-base-name=tmlog #
Transactions log file base name.
spring.jta.atomikos.properties.max-actives=50 # Maximum number
of active transactions.
spring.jta.atomikos.properties.max-timeout=300000ms # Maximum
timeout that can be allowed for transactions.
spring.jta.atomikos.properties.recovery.delay=10000ms # Delay
between two recovery scans.
spring.jta.atomikos.properties.recovery.forget-orphaned-log-
entries-delay=86400000ms # Delay after which recovery can
cleanup pending ('orphaned') log entries.
spring.jta.atomikos.properties.recovery.max-retries=5 # Number
of retry attempts to commit the transaction before throwing an
exception.
spring.jta.atomikos.properties.recovery.retry-interval=10000ms
# Delay between retry attempts.
spring.jta.atomikos.properties.serial-jta-transactions=true #
Whether sub-transactions should be joined when possible.
spring.jta.atomikos.properties.service= # Transaction manager
implementation that should be started.
spring.jta.atomikos.properties.threaded-two-phase-commit=false
```

```
# Whether to use different (and concurrent) threads for two-
phase commit on the participating resources.
spring.jta.atomikos.properties.transaction-manager-unique-name=
# The transaction manager's unique name.

# BITRONIX
spring.jta.bitronix.connectionfactory.acquire-increment=1 #
Number of connections to create when growing the pool.
spring.jta.bitronix.connectionfactory.acquisition-interval=1 #
Time, in seconds, to wait before trying to acquire a connection
again after an invalid connection was acquired.
spring.jta.bitronix.connectionfactory.acquisition-timeout=30 #
Timeout, in seconds, for acquiring connections from the pool.
spring.jta.bitronix.connectionfactory.allow-local-
transactions=true # Whether the transaction manager should
allow mixing XA and non-XA transactions.
spring.jta.bitronix.connectionfactory.apply-transaction-
timeout=false # Whether the transaction timeout should be set
on the XAResource when it is enlisted.
spring.jta.bitronix.connectionfactory.automatic-enlisting-
enabled=true # Whether resources should be enlisted and
delisted automatically.
spring.jta.bitronix.connectionfactory.cache-producers-
consumers=true # Whether producers and consumers should be
cached.
spring.jta.bitronix.connectionfactory.class-name= # Underlying
implementation class name of the XA resource.
spring.jta.bitronix.connectionfactory.defer-connection-
release=true # Whether the provider can run many transactions
on the same connection and supports transaction interleaving.
spring.jta.bitronix.connectionfactory.disabled= # Whether this
resource is disabled, meaning it's temporarily forbidden to
acquire a connection from its pool.
spring.jta.bitronix.connectionfactory.driver-properties= #
Properties that should be set on the underlying implementation.
spring.jta.bitronix.connectionfactory.failed= # Mark this
resource producer as failed.
spring.jta.bitronix.connectionfactory.ignore-recovery-
failures=false # Whether recovery failures should be ignored.
spring.jta.bitronix.connectionfactory.max-idle-time=60 # The
time, in seconds, after which connections are cleaned up from
the pool.
spring.jta.bitronix.connectionfactory.max-pool-size=10 # The
maximum size of the pool. 0 denotes no limit.
spring.jta.bitronix.connectionfactory.min-pool-size=0 # The
```

```
minimum size of the pool.
spring.jta.bitronix.connectionfactory.password= # The password
to use to connect to the JMS provider.
spring.jta.bitronix.connectionfactory.share-transaction-
connections=false #  Whether connections in the ACCESSIBLE
state can be shared within the context of a transaction.
spring.jta.bitronix.connectionfactory.-connections=true #
Whether connections should be ed when acquired from the pool.
spring.jta.bitronix.connectionfactory.two-pc-ordering-
position=1 # The position that this resource should take during
two-phase commit (always first is Integer.MIN_VALUE, always
last is Integer.MAX_VALUE).
spring.jta.bitronix.connectionfactory.unique-
name=jmsConnectionFactory # The unique name used to identify
the resource during recovery.
spring.jta.bitronix.connectionfactory.use-tm-join=true #
Whether TMJOIN should be used when starting XAResources.
spring.jta.bitronix.connectionfactory.user= # The user to use
to connect to the JMS provider.
spring.jta.bitronix.datasource.acquire-increment=1 # Number of
connections to create when growing the pool.
spring.jta.bitronix.datasource.acquisition-interval=1 # Time,
in seconds, to wait before trying to acquire a connection again
after an invalid connection was acquired.
spring.jta.bitronix.datasource.acquisition-timeout=30 #
Timeout, in seconds, for acquiring connections from the pool.
spring.jta.bitronix.datasource.allow-local-transactions=true #
Whether the transaction manager should allow mixing XA and non-
XA transactions.
spring.jta.bitronix.datasource.apply-transaction-timeout=false
# Whether the transaction timeout should be set on the
XAResource when it is enlisted.
spring.jta.bitronix.datasource.automatic-enlisting-enabled=true
# Whether resources should be enlisted and delisted
automatically.
spring.jta.bitronix.datasource.class-name= # Underlying
implementation class name of the XA resource.
spring.jta.bitronix.datasource.cursor-holdability= # The
default cursor holdability for connections.
spring.jta.bitronix.datasource.defer-connection-release=true #
Whether the database can run many transactions on the same
connection and supports transaction interleaving.
spring.jta.bitronix.datasource.disabled= # Whether this
resource is disabled, meaning it's temporarily forbidden to
acquire a connection from its pool.
```

```
spring.jta.bitronix.datasource.driver-properties= # Properties
that should be set on the underlying implementation.
spring.jta.bitronix.datasource.enable-jdbc4-connection-= #
Whether Connection.isValid() is called when acquiring a
connection from the pool.
spring.jta.bitronix.datasource.failed= # Mark this resource
producer as failed.
spring.jta.bitronix.datasource.ignore-recovery-failures=false #
Whether recovery failures should be ignored.
spring.jta.bitronix.datasource.isolation-level= # The default
isolation level for connections.
spring.jta.bitronix.datasource.local-auto-commit= # The default
auto-commit mode for local transactions.
spring.jta.bitronix.datasource.login-timeout= # Timeout, in
seconds, for establishing a database connection.
spring.jta.bitronix.datasource.max-idle-time=60 # The time, in
seconds, after which connections are cleaned up from the pool.
spring.jta.bitronix.datasource.max-pool-size=10 # The maximum
size of the pool. 0 denotes no limit.
spring.jta.bitronix.datasource.min-pool-size=0 # The minimum
size of the pool.
spring.jta.bitronix.datasource.prepared-statement-cache-size=0
# The target size of the prepared statement cache. 0 disables
the cache.
spring.jta.bitronix.datasource.share-transaction-
connections=false #  Whether connections in the ACCESSIBLE
state can be shared within the context of a transaction.
spring.jta.bitronix.datasource.-query= # SQL query or statement
used to validate a connection before returning it.
spring.jta.bitronix.datasource.two-pc-ordering-position=1 # The
position that this resource should take during two-phase commit
(always first is Integer.MIN_VALUE, and always last is
Integer.MAX_VALUE).
spring.jta.bitronix.datasource.unique-name=dataSource # The
unique name used to identify the resource during recovery.
spring.jta.bitronix.datasource.use-tm-join=true # Whether
TMJOIN should be used when starting XAResources.
spring.jta.bitronix.properties.allow-multiple-lrc=false #
Whether to allow multiple LRC resources to be enlisted into the
same transaction.
spring.jta.bitronix.properties.asynchronous2-pc=false # Whether
to enable asynchronously execution of two phase commit.
spring.jta.bitronix.properties.background-recovery-interval-
seconds=60 # Interval in seconds at which to run the recovery
process in the background.
```

```
spring.jta.bitronix.properties.current-node-only-recovery=true
# Whether to recover only the current node.
spring.jta.bitronix.properties.debug-zero-resource-
transaction=false # Whether to log the creation and commit call
stacks of transactions executed without a single enlisted
resource.
spring.jta.bitronix.properties.default-transaction-timeout=60 #
Default transaction timeout, in seconds.
spring.jta.bitronix.properties.disable-jmx=false # Whether to
enable JMX support.
spring.jta.bitronix.properties.exception-analyzer= # Set the
fully qualified name of the exception analyzer implementation
to use.
spring.jta.bitronix.properties.filter-log-status=false #
Whether to enable filtering of logs so that only mandatory logs
are written.
spring.jta.bitronix.properties.force-batching-enabled=true #
Whether disk forces are batched.
spring.jta.bitronix.properties.forced-write-enabled=true #
Whether logs are forced to disk.
spring.jta.bitronix.properties.graceful-shutdown-interval=60 #
Maximum amount of seconds the TM waits for transactions to get
done before aborting them at shutdown time.
spring.jta.bitronix.properties.jndi-transaction-
synchronization-registry-name= # JNDI name of the
TransactionSynchronizationRegistry.
spring.jta.bitronix.properties.jndi-user-transaction-name= #
JNDI name of the UserTransaction.
spring.jta.bitronix.properties.journal=disk # Name of the
journal. Can be 'disk', 'null', or a class name.
spring.jta.bitronix.properties.log-part1-filename=btm1.tlog #
Name of the first fragment of the journal.
spring.jta.bitronix.properties.log-part2-filename=btm2.tlog #
Name of the second fragment of the journal.
spring.jta.bitronix.properties.max-log-size-in-mb=2 # Maximum
size in megabytes of the journal fragments.
spring.jta.bitronix.properties.resource-configuration-filename=
# ResourceLoader configuration file name.
spring.jta.bitronix.properties.server-id= # ASCII ID that must
uniquely identify this TM instance. Defaults to the machine's
IP address.
spring.jta.bitronix.properties.skip-corrupted-logs=false # Skip
corrupted transactions log entries.
spring.jta.bitronix.properties.warn-about-zero-resource-
transaction=true # Whether to log a warning for transactions
```

```
executed without a single enlisted resource.

# NARAYANA (NarayanaProperties)
spring.jta.narayana.default-timeout=60s # Transaction timeout.
If a duration suffix is not specified, seconds will be used.
spring.jta.narayana.expiry-
scanners=com.arjuna.ats.internal.arjuna.recovery.ExpiredTransac
tionStatusManagerScanner # Comma-separated list of expiry
scanners.
spring.jta.narayana.log-dir= # Transaction object store
directory.
spring.jta.narayana.one-phase-commit=true # Whether to enable
one phase commit optimization.
spring.jta.narayana.periodic-recovery-period=120s # Interval in
which periodic recovery scans are performed. If a duration
suffix is not specified, seconds will be used.
spring.jta.narayana.recovery-backoff-period=10s # Back off
period between first and second phases of the recovery scan. If
a duration suffix is not specified, seconds will be used.
spring.jta.narayana.recovery-db-pass= # Database password to be
used by the recovery manager.
spring.jta.narayana.recovery-db-user= # Database username to be
used by the recovery manager.
spring.jta.narayana.recovery-jms-pass= # JMS password to be
used by the recovery manager.
spring.jta.narayana.recovery-jms-user= # JMS username to be
used by the recovery manager.
spring.jta.narayana.recovery-modules= # Comma-separated list of
recovery modules.
spring.jta.narayana.transaction-manager-id=1 # Unique
transaction manager id.
spring.jta.narayana.xa-resource-orphan-filters= # Comma-
separated list of orphan filters.

# EMBEDDED MONGODB (EmbeddedMongoProperties)
spring.mongodb.embedded.features=sync_delay # Comma-separated
list of features to enable.
spring.mongodb.embedded.storage.database-dir= # Directory used
for data storage.
spring.mongodb.embedded.storage.oplog-size= # Maximum size of
the oplog, in megabytes.
spring.mongodb.embedded.storage.repl-set-name= # Name of the
replica set.
spring.mongodb.embedded.version=3.2.2 # Version of Mongo to
use.
```

```
# REDIS (RedisProperties)
spring.redis.cluster.max-redirects= # Maximum number of
redirects to follow when executing commands across the cluster.
spring.redis.cluster.nodes= # Comma-separated list of
"host:port" pairs to bootstrap from.
spring.redis.database=0 # Database index used by the connection
factory.
spring.redis.url= # Connection URL. Overrides host, port, and
password. User is ignored. Example:
redis://user:password@example.com:6379
spring.redis.host=localhost # Redis server host.
spring.redis.jedis.pool.max-active=8 # Maximum number of
connections that can be allocated by the pool at a given time.
Use a negative value for no limit.
spring.redis.jedis.pool.max-idle=8 # Maximum number of "idle"
connections in the pool. Use a negative value to indicate an
unlimited number of idle connections.
spring.redis.jedis.pool.max-wait=-1ms # Maximum amount of time
a connection allocation should block before throwing an
exception when the pool is exhausted. Use a negative value to
block indefinitely.
spring.redis.jedis.pool.min-idle=0 # Target for the minimum
number of idle connections to maintain in the pool. This
setting only has an effect if it is positive.
spring.redis.lettuce.pool.max-active=8 # Maximum number of
connections that can be allocated by the pool at a given time.
Use a negative value for no limit.
spring.redis.lettuce.pool.max-idle=8 # Maximum number of "idle"
connections in the pool. Use a negative value to indicate an
unlimited number of idle connections.
spring.redis.lettuce.pool.max-wait=-1ms # Maximum amount of
time a connection allocation should block before throwing an
exception when the pool is exhausted. Use a negative value to
block indefinitely.
spring.redis.lettuce.pool.min-idle=0 # Target for the minimum
number of idle connections to maintain in the pool. This
setting only has an effect if it is positive.
spring.redis.lettuce.shutdown-timeout=100ms # Shutdown timeout.
spring.redis.password= # Login password of the redis server.
spring.redis.port=6379 # Redis server port.
spring.redis.sentinel.master= # Name of the Redis server.
spring.redis.sentinel.nodes= # Comma-separated list of
"host:port" pairs.
spring.redis.ssl=false # Whether to enable SSL support.
```

```
spring.redis.timeout= # Connection timeout.

# TRANSACTION (TransactionProperties)
spring.transaction.default-timeout= # Default transaction
timeout. If a duration suffix is not specified, seconds will be
used.
spring.transaction.rollback-on-commit-failure= # Whether to
roll back on commit failures.



# ----------------------------------------
# INTEGRATION PROPERTIES
# ----------------------------------------

# ACTIVEMQ (ActiveMQProperties)
spring.activemq.broker-url= # URL of the ActiveMQ broker. Auto-
generated by default.
spring.activemq.close-timeout=15s # Time to wait before
considering a close complete.
spring.activemq.in-memory=true # Whether the default broker URL
should be in memory. Ignored if an explicit broker has been
specified.
spring.activemq.non-blocking-redelivery=false # Whether to stop
message delivery before re-delivering messages from a rolled
back transaction. This implies that message order is not
preserved when this is enabled.
spring.activemq.password= # Login password of the broker.
spring.activemq.send-timeout=0ms # Time to wait on message
sends for a response. Set it to 0 to wait forever.
spring.activemq.user= # Login user of the broker.
spring.activemq.packages.trust-all= # Whether to trust all
packages.
spring.activemq.packages.trusted= # Comma-separated list of
specific packages to trust (when not trusting all packages).
spring.activemq.pool.block-if-full=true # Whether to block when
a connection is requested and the pool is full. Set it to false
to throw a "JMSException" instead.
spring.activemq.pool.block-if-full-timeout=-1ms # Blocking
period before throwing an exception if the pool is still full.
spring.activemq.pool.enabled=false # Whether a
PooledConnectionFactory should be created, instead of a regular
ConnectionFactory.
spring.activemq.pool.idle-timeout=30s # Connection idle
timeout.
```

```
spring.activemq.pool.max-connections=1 # Maximum number of
pooled connections.
spring.activemq.pool.maximum-active-session-per-connection=500
# Maximum number of active sessions per connection.
spring.activemq.pool.time-between-expiration-check=-1ms # Time
to sleep between runs of the idle connection eviction thread.
When negative, no idle connection eviction thread runs.
spring.activemq.pool.use-anonymous-producers=true # Whether to
use only one anonymous "MessageProducer" instance. Set it to
false to create one "MessageProducer" every time one is
required.

# ARTEMIS (ArtemisProperties)
spring.artemis.embedded.cluster-password= # Cluster password.
Randomly generated on startup by default.
spring.artemis.embedded.data-directory= # Journal file
directory. Not necessary if persistence is turned off.
spring.artemis.embedded.enabled=true # Whether to enable
embedded mode if the Artemis server APIs are available.
spring.artemis.embedded.persistent=false # Whether to enable
persistent store.
spring.artemis.embedded.queues= # Comma-separated list of
queues to create on startup.
spring.artemis.embedded.server-id= # Server ID. By default, an
auto-incremented counter is used.
spring.artemis.embedded.topics= # Comma-separated list of
topics to create on startup.
spring.artemis.host=localhost # Artemis broker host.
spring.artemis.mode= # Artemis deployment mode, auto-detected
by default.
spring.artemis.password= # Login password of the broker.
spring.artemis.port=61616 # Artemis broker port.
spring.artemis.user= # Login user of the broker.

# SPRING BATCH (BatchProperties)
spring.batch.initialize-schema=embedded # Database schema
initialization mode.
spring.batch.job.enabled=true # Execute all Spring Batch jobs
in the context on startup.
spring.batch.job.names= # Comma-separated list of job names to
execute on startup (for instance, `job1,job2`). By default, all
Jobs found in the context are executed.
spring.batch.schema=classpath:org/springframework/batch/core/sc
hema-@@platform@@.sql # Path to the SQL file to use to
initialize the database schema.
```

```
spring.batch.table-prefix= # Table prefix for all the batch
meta-data tables.

# SPRING INTEGRATION (IntegrationProperties)
spring.integration.jdbc.initialize-schema=embedded # Database
schema initialization mode.
spring.integration.jdbc.schema=classpath:org/springframework/in
tegration/jdbc/schema-@@platform@@.sql # Path to the SQL file
to use to initialize the database schema.

# JMS (JmsProperties)
spring.jms.jndi-name= # Connection factory JNDI name. When set,
takes precedence to others connection factory auto-
configurations.
spring.jms.listener.acknowledge-mode= # Acknowledge mode of the
container. By default, the listener is transacted with
automatic acknowledgment.
spring.jms.listener.auto-startup=true # Start the container
automatically on startup.
spring.jms.listener.concurrency= # Minimum number of concurrent
consumers.
spring.jms.listener.max-concurrency= # Maximum number of
concurrent consumers.
spring.jms.pub-sub-domain=false # Whether the default
destination type is topic.
spring.jms.template.default-destination= # Default destination
to use on send and receive operations that do not have a
destination parameter.
spring.jms.template.delivery-delay= # Delivery delay to use for
send calls.
spring.jms.template.delivery-mode= # Delivery mode. Enables QoS
(Quality of Service) when set.
spring.jms.template.priority= # Priority of a message when
sending. Enables QoS (Quality of Service) when set.
spring.jms.template.qos-enabled= # Whether to enable explicit
QoS (Quality of Service) when sending a message.
spring.jms.template.receive-timeout= # Timeout to use for
receive calls.
spring.jms.template.time-to-live= # Time-to-live of a message
when sending. Enables QoS (Quality of Service) when set.

# APACHE KAFKA (KafkaProperties)
spring.kafka.admin.client-id= # ID to pass to the server when
making requests. Used for server-side logging.
spring.kafka.admin.fail-fast=false # Whether to fail fast if
```

```
the broker is not available on startup.
spring.kafka.admin.properties.*= # Additional admin-specific
properties used to configure the client.
spring.kafka.admin.ssl.key-password= # Password of the private
key in the key store file.
spring.kafka.admin.ssl.keystore-location= # Location of the key
store file.
spring.kafka.admin.ssl.keystore-password= # Store password for
the key store file.
spring.kafka.admin.ssl.keystore-type= # Type of the key store.
spring.kafka.admin.ssl.protocol= # SSL protocol to use.
spring.kafka.admin.ssl.truststore-location= # Location of the
trust store file.
spring.kafka.admin.ssl.truststore-password= # Store password
for the trust store file.
spring.kafka.admin.ssl.truststore-type= # Type of the trust
store.
spring.kafka.bootstrap-servers= # Comma-delimited list of
host:port pairs to use for establishing the initial connections
to the Kafka cluster. Applies to all components unless
overridden.
spring.kafka.client-id= # ID to pass to the server when making
requests. Used for server-side logging.
spring.kafka.consumer.auto-commit-interval= # Frequency with
which the consumer offsets are auto-committed to Kafka if
'enable.auto.commit' is set to true.
spring.kafka.consumer.auto-offset-reset= # What to do when
there is no initial offset in Kafka or if the current offset no
longer exists on the server.
spring.kafka.consumer.bootstrap-servers= # Comma-delimited list
of host:port pairs to use for establishing the initial
connections to the Kafka cluster. Overrides the global
property, for consumers.
spring.kafka.consumer.client-id= # ID to pass to the server
when making requests. Used for server-side logging.
spring.kafka.consumer.enable-auto-commit= # Whether the
consumer's offset is periodically committed in the background.
spring.kafka.consumer.fetch-max-wait= # Maximum amount of time
the server blocks before answering the fetch request if there
isn't sufficient data to immediately satisfy the requirement
given by "fetch.min.bytes".
spring.kafka.consumer.fetch-min-size= # Minimum amount of data,
in bytes, the server should return for a fetch request.
spring.kafka.consumer.group-id= # Unique string that identifies
the consumer group to which this consumer belongs.
```

```
spring.kafka.consumer.heartbeat-interval= # Expected time
between heartbeats to the consumer coordinator.
spring.kafka.consumer.key-deserializer= # Deserializer class
for keys.
spring.kafka.consumer.max-poll-records= # Maximum number of
records returned in a single call to poll().
spring.kafka.consumer.properties.*= # Additional consumer-
specific properties used to configure the client.
spring.kafka.consumer.ssl.key-password= # Password of the
private key in the key store file.
spring.kafka.consumer.ssl.keystore-location= # Location of the
key store file.
spring.kafka.consumer.ssl.keystore-password= # Store password
for the key store file.
spring.kafka.consumer.ssl.keystore-type= # Type of the key
store.
spring.kafka.consumer.ssl.protocol= # SSL protocol to use.
spring.kafka.consumer.ssl.truststore-location= # Location of
the trust store file.
spring.kafka.consumer.ssl.truststore-password= # Store password
for the trust store file.
spring.kafka.consumer.ssl.truststore-type= # Type of the trust
store.
spring.kafka.consumer.value-deserializer= # Deserializer class
for values.
spring.kafka.jaas.control-flag=required # Control flag for
login configuration.
spring.kafka.jaas.enabled=false # Whether to enable JAAS
configuration.
spring.kafka.jaas.login-
module=com.sun.security.auth.module.Krb5LoginModule # Login
module.
spring.kafka.jaas.options= # Additional JAAS options.
spring.kafka.listener.ack-count= # Number of records between
offset commits when ackMode is "COUNT" or "COUNT_TIME".
spring.kafka.listener.ack-mode= # Listener AckMode. See the
spring-kafka documentation.
spring.kafka.listener.ack-time= # Time between offset commits
when ackMode is "TIME" or "COUNT_TIME".
spring.kafka.listener.client-id= # Prefix for the listener's
consumer client.id property.
spring.kafka.listener.concurrency= # Number of threads to run
in the listener containers.
spring.kafka.listener.idle-event-interval= # Time between
publishing idle consumer events (no data received).
```

```
spring.kafka.listener.log-container-config= # Whether to log
the container configuration during initialization (INFO level).
spring.kafka.listener.monitor-interval= # Time between checks
for non-responsive consumers. If a duration suffix is not
specified, seconds will be used.
spring.kafka.listener.no-poll-threshold= # Multiplier applied
to "pollTimeout" to determine if a consumer is non-responsive.
spring.kafka.listener.poll-timeout= # Timeout to use when
polling the consumer.
spring.kafka.listener.type=single # Listener type.
spring.kafka.producer.acks= # Number of acknowledgments the
producer requires the leader to have received before
considering a request complete.
spring.kafka.producer.batch-size= # Default batch size in
bytes.
spring.kafka.producer.bootstrap-servers= # Comma-delimited list
of host:port pairs to use for establishing the initial
connections to the Kafka cluster. Overrides the global
property, for producers.
spring.kafka.producer.buffer-memory= # Total bytes of memory
the producer can use to buffer records waiting to be sent to
the server.
spring.kafka.producer.client-id= # ID to pass to the server
when making requests. Used for server-side logging.
spring.kafka.producer.compression-type= # Compression type for
all data generated by the producer.
spring.kafka.producer.key-serializer= # Serializer class for
keys.
spring.kafka.producer.properties.*= # Additional producer-
specific properties used to configure the client.
spring.kafka.producer.retries= # When greater than zero,
enables retrying of failed sends.
spring.kafka.producer.ssl.key-password= # Password of the
private key in the key store file.
spring.kafka.producer.ssl.keystore-location= # Location of the
key store file.
spring.kafka.producer.ssl.keystore-password= # Store password
for the key store file.
spring.kafka.producer.ssl.keystore-type= # Type of the key
store.
spring.kafka.producer.ssl.protocol= # SSL protocol to use.
spring.kafka.producer.ssl.truststore-location= # Location of
the trust store file.
spring.kafka.producer.ssl.truststore-password= # Store password
for the trust store file.
```

```
spring.kafka.producer.ssl.truststore-type= # Type of the trust
store.
spring.kafka.producer.transaction-id-prefix= # When non empty,
enables transaction support for producer.
spring.kafka.producer.value-serializer= # Serializer class for
values.
spring.kafka.properties.*= # Additional properties, common to
producers and consumers, used to configure the client.
spring.kafka.ssl.key-password= # Password of the private key in
the key store file.
spring.kafka.ssl.keystore-location= # Location of the key store
file.
spring.kafka.ssl.keystore-password= # Store password for the
key store file.
spring.kafka.ssl.keystore-type= # Type of the key store.
spring.kafka.ssl.protocol= # SSL protocol to use.
spring.kafka.ssl.truststore-location= # Location of the trust
store file.
spring.kafka.ssl.truststore-password= # Store password for the
trust store file.
spring.kafka.ssl.truststore-type= # Type of the trust store.
spring.kafka.template.default-topic= # Default topic to which
messages are sent.

# RABBIT (RabbitProperties)
spring.rabbitmq.addresses= # Comma-separated list of addresses
to which the client should connect.
spring.rabbitmq.cache.channel.checkout-timeout= # Duration to
wait to obtain a channel if the cache size has been reached.
spring.rabbitmq.cache.channel.size= # Number of channels to
retain in the cache.
spring.rabbitmq.cache.connection.mode=channel # Connection
factory cache mode.
spring.rabbitmq.cache.connection.size= # Number of connections
to cache.
spring.rabbitmq.connection-timeout= # Connection timeout. Set
it to zero to wait forever.
spring.rabbitmq.dynamic=true # Whether to create an AmqpAdmin
bean.
spring.rabbitmq.host=localhost # RabbitMQ host.
spring.rabbitmq.listener.direct.acknowledge-mode= # Acknowledge
mode of container.
spring.rabbitmq.listener.direct.auto-startup=true # Whether to
start the container automatically on startup.
spring.rabbitmq.listener.direct.consumers-per-queue= # Number
```

```
of consumers per queue.
spring.rabbitmq.listener.direct.default-requeue-rejected= #
Whether rejected deliveries are re-queued by default.
spring.rabbitmq.listener.direct.idle-event-interval= # How
often idle container events should be published.
spring.rabbitmq.listener.direct.prefetch= # Number of messages
to be handled in a single request. It should be greater than or
equal to the transaction size (if used).
spring.rabbitmq.listener.direct.retry.enabled=false # Whether
publishing retries are enabled.
spring.rabbitmq.listener.direct.retry.initial-interval=1000ms #
Duration between the first and second attempt to deliver a
message.
spring.rabbitmq.listener.direct.retry.max-attempts=3 # Maximum
number of attempts to deliver a message.
spring.rabbitmq.listener.direct.retry.max-interval=10000ms #
Maximum duration between attempts.
spring.rabbitmq.listener.direct.retry.multiplier=1 # Multiplier
to apply to the previous retry interval.
spring.rabbitmq.listener.direct.retry.stateless=true # Whether
retries are stateless or stateful.
spring.rabbitmq.listener.simple.acknowledge-mode= # Acknowledge
mode of container.
spring.rabbitmq.listener.simple.auto-startup=true # Whether to
start the container automatically on startup.
spring.rabbitmq.listener.simple.concurrency= # Minimum number
of listener invoker threads.
spring.rabbitmq.listener.simple.default-requeue-rejected= #
Whether rejected deliveries are re-queued by default.
spring.rabbitmq.listener.simple.idle-event-interval= # How
often idle container events should be published.
spring.rabbitmq.listener.simple.max-concurrency= # Maximum
number of listener invoker threads.
spring.rabbitmq.listener.simple.prefetch= # Number of messages
to be handled in a single request. It should be greater than or
equal to the transaction size (if used).
spring.rabbitmq.listener.simple.retry.enabled=false # Whether
publishing retries are enabled.
spring.rabbitmq.listener.simple.retry.initial-interval=1000ms #
Duration between the first and second attempt to deliver a
message.
spring.rabbitmq.listener.simple.retry.max-attempts=3 # Maximum
number of attempts to deliver a message.
spring.rabbitmq.listener.simple.retry.max-interval=10000ms #
Maximum duration between attempts.
```

```
spring.rabbitmq.listener.simple.retry.multiplier=1 # Multiplier
to apply to the previous retry interval.
spring.rabbitmq.listener.simple.retry.stateless=true # Whether
retries are stateless or stateful.
spring.rabbitmq.listener.simple.transaction-size= # Number of
messages to be processed in a transaction. That is, the number
of messages between acks. For best results, it should be less
than or equal to the prefetch count.
spring.rabbitmq.listener.type=simple # Listener container type.
spring.rabbitmq.password=guest # Login to authenticate against
the broker.
spring.rabbitmq.port=5672 # RabbitMQ port.
spring.rabbitmq.publisher-confirms=false # Whether to enable
publisher confirms.
spring.rabbitmq.publisher-returns=false # Whether to enable
publisher returns.
spring.rabbitmq.requested-heartbeat= # Requested heartbeat
timeout; zero for none. If a duration suffix is not specified,
seconds will be used.
spring.rabbitmq.ssl.algorithm= # SSL algorithm to use. By
default, configured by the Rabbit client library.
spring.rabbitmq.ssl.enabled=false # Whether to enable SSL
support.
spring.rabbitmq.ssl.key-store= # Path to the key store that
holds the SSL certificate.
spring.rabbitmq.ssl.key-store-password= # Password used to
access the key store.
spring.rabbitmq.ssl.key-store-type=PKCS12 # Key store type.
spring.rabbitmq.ssl.trust-store= # Trust store that holds SSL
certificates.
spring.rabbitmq.ssl.trust-store-password= # Password used to
access the trust store.
spring.rabbitmq.ssl.trust-store-type=JKS # Trust store type.
spring.rabbitmq.ssl.validate-server-certificate=true # Whether
to enable server side certificate validation.
spring.rabbitmq.ssl.verify-hostname= # Whether to enable
hostname verification. Requires AMQP client 4.8 or above and
defaults to true when a suitable client version is used.
spring.rabbitmq.template.exchange= # Name of the default
exchange to use for send operations.
spring.rabbitmq.template.mandatory= # Whether to enable
mandatory messages.
spring.rabbitmq.template.receive-timeout= # Timeout for
`receive()` operations.
spring.rabbitmq.template.reply-timeout= # Timeout for
```

```
`sendAndReceive()` operations.
spring.rabbitmq.template.retry.enabled=false # Whether
publishing retries are enabled.
spring.rabbitmq.template.retry.initial-interval=1000ms #
Duration between the first and second attempt to deliver a
message.
spring.rabbitmq.template.retry.max-attempts=3 # Maximum number
of attempts to deliver a message.
spring.rabbitmq.template.retry.max-interval=10000ms # Maximum
duration between attempts.
spring.rabbitmq.template.retry.multiplier=1 # Multiplier to
apply to the previous retry interval.
spring.rabbitmq.template.routing-key= # Value of a default
routing key to use for send operations.
spring.rabbitmq.username=guest # Login user to authenticate to
the broker.
spring.rabbitmq.virtual-host= # Virtual host to use when
connecting to the broker.


# ----------------------------------------
# ACTUATOR PROPERTIES
# ----------------------------------------

# MANAGEMENT HTTP SERVER (ManagementServerProperties)
management.server.add-application-context-header=false # Add
the "X-Application-Context" HTTP header in each response.
management.server.address= # Network address to which the
management endpoints should bind. Requires a custom
management.server.port.
management.server.port= # Management endpoint HTTP port (uses
the same port as the application by default). Configure a
different port to use management-specific SSL.
management.server.servlet.context-path= # Management endpoint
context-path (for instance, `/management`). Requires a custom
management.server.port.
management.server.ssl.ciphers= # Supported SSL ciphers.
Requires a custom management.port.
management.server.ssl.client-auth= # Whether client
authentication is wanted ("want") or needed ("need"). Requires
a trust store. Requires a custom management.server.port.
management.server.ssl.enabled= # Whether to enable SSL support.
Requires a custom management.server.port.
management.server.ssl.enabled-protocols= # Enabled SSL
protocols. Requires a custom management.server.port.
```

```
management.server.ssl.key-alias= # Alias that identifies the
key in the key store. Requires a custom management.server.port.
management.server.ssl.key-password= # Password used to access
the key in the key store. Requires a custom
management.server.port.
management.server.ssl.key-store= # Path to the key store that
holds the SSL certificate (typically a jks file). Requires a
custom management.server.port.
management.server.ssl.key-store-password= # Password used to
access the key store. Requires a custom management.server.port.
management.server.ssl.key-store-provider= # Provider for the
key store. Requires a custom management.server.port.
management.server.ssl.key-store-type= # Type of the key store.
Requires a custom management.server.port.
management.server.ssl.protocol=TLS # SSL protocol to use.
Requires a custom management.server.port.
management.server.ssl.trust-store= # Trust store that holds SSL
certificates. Requires a custom management.server.port.
management.server.ssl.trust-store-password= # Password used to
access the trust store. Requires a custom
management.server.port.
management.server.ssl.trust-store-provider= # Provider for the
trust store. Requires a custom management.server.port.
management.server.ssl.trust-store-type= # Type of the trust
store. Requires a custom management.server.port.

# CLOUDFOUNDRY
management.cloudfoundry.enabled=true # Whether to enable
extended Cloud Foundry actuator endpoints.
management.cloudfoundry.skip-ssl-validation=false # Whether to
skip SSL verification for Cloud Foundry actuator endpoint
security calls.

# ENDPOINTS GENERAL CONFIGURATION
management.endpoints.enabled-by-default= # Whether to enable or
disable all endpoints by default.

# ENDPOINTS JMX CONFIGURATION (JmxEndpointProperties)
management.endpoints.jmx.domain=org.springframework.boot #
Endpoints JMX domain name. Fallback to 'spring.jmx.default-
domain' if set.
management.endpoints.jmx.exposure.include=* # Endpoint IDs that
should be included or '*' for all.
management.endpoints.jmx.exposure.exclude= # Endpoint IDs that
should be excluded or '*' for all.
```

```
management.endpoints.jmx.static-names= # Additional static
properties to append to all ObjectNames of MBeans representing
Endpoints.
management.endpoints.jmx.unique-names=false # Whether to ensure
that ObjectNames are modified in case of conflict.

# ENDPOINTS WEB CONFIGURATION (WebEndpointProperties)
management.endpoints.web.exposure.include=health,info #
Endpoint IDs that should be included or '*' for all.
management.endpoints.web.exposure.exclude= # Endpoint IDs that
should be excluded or '*' for all.
management.endpoints.web.base-path=/actuator # Base path for
Web endpoints. Relative to server.servlet.context-path or
management.server.servlet.context-path if
management.server.port is configured.
management.endpoints.web.path-mapping= # Mapping between
endpoint IDs and the path that should expose them.

# ENDPOINTS CORS CONFIGURATION (CorsEndpointProperties)
management.endpoints.web.cors.allow-credentials= # Whether
credentials are supported. When not set, credentials are not
supported.
management.endpoints.web.cors.allowed-headers= # Comma-
separated list of headers to allow in a request. '*' allows all
headers.
management.endpoints.web.cors.allowed-methods= # Comma-
separated list of methods to allow. '*' allows all methods.
When not set, defaults to GET.
management.endpoints.web.cors.allowed-origins= # Comma-
separated list of origins to allow. '*' allows all origins.
When not set, CORS support is disabled.
management.endpoints.web.cors.exposed-headers= # Comma-
separated list of headers to include in a response.
management.endpoints.web.cors.max-age=1800s # How long the
response from a pre-flight request can be cached by clients. If
a duration suffix is not specified, seconds will be used.

# AUDIT EVENTS ENDPOINT (AuditEventsEndpoint)
management.endpoint.auditevents.cache.time-to-live=0ms #
Maximum time that a response can be cached.
management.endpoint.auditevents.enabled=true # Whether to
enable the auditevents endpoint.

# BEANS ENDPOINT (BeansEndpoint)
management.endpoint.beans.cache.time-to-live=0ms # Maximum time
```

```
that a response can be cached.
management.endpoint.beans.enabled=true # Whether to enable the
beans endpoint.

# CONDITIONS REPORT ENDPOINT (ConditionsReportEndpoint)
management.endpoint.conditions.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.conditions.enabled=true # Whether to enable
the conditions endpoint.

# CONFIGURATION PROPERTIES REPORT ENDPOINT
(ConfigurationPropertiesReportEndpoint,
ConfigurationPropertiesReportEndpointProperties)
management.endpoint.configprops.cache.time-to-live=0ms #
Maximum time that a response can be cached.
management.endpoint.configprops.enabled=true # Whether to
enable the configprops endpoint.
management.endpoint.configprops.keys-to-
sanitize=password,secret,key,token,.*credentials.*,vcap_service
s,sun.java.command # Keys that should be sanitized. Keys can be
simple strings that the property ends with or regular
expressions.

# ENVIRONMENT ENDPOINT (EnvironmentEndpoint,
EnvironmentEndpointProperties)
management.endpoint.env.cache.time-to-live=0ms # Maximum time
that a response can be cached.
management.endpoint.env.enabled=true # Whether to enable the
env endpoint.
management.endpoint.env.keys-to-
sanitize=password,secret,key,token,.*credentials.*,vcap_service
s,sun.java.command # Keys that should be sanitized. Keys can be
simple strings that the property ends with or regular
expressions.

# FLYWAY ENDPOINT (FlywayEndpoint)
management.endpoint.flyway.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.flyway.enabled=true # Whether to enable the
flyway endpoint.

# HEALTH ENDPOINT (HealthEndpoint, HealthEndpointProperties)
management.endpoint.health.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.health.enabled=true # Whether to enable the
```

```
health endpoint.
management.endpoint.health.roles= # Roles used to determine
whether or not a user is authorized to be shown details. When
empty, all authenticated users are authorized.
management.endpoint.health.show-details=never # When to show
full health details.

# HEAP DUMP ENDPOINT (HeapDumpWebEndpoint)
management.endpoint.heapdump.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.heapdump.enabled=true # Whether to enable
the heapdump endpoint.

# HTTP TRACE ENDPOINT (HttpTraceEndpoint)
management.endpoint.httptrace.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.httptrace.enabled=true # Whether to enable
the httptrace endpoint.

# INFO ENDPOINT (InfoEndpoint)
info= # Arbitrary properties to add to the info endpoint.
management.endpoint.info.cache.time-to-live=0ms # Maximum time
that a response can be cached.
management.endpoint.info.enabled=true # Whether to enable the
info endpoint.

# JOLOKIA ENDPOINT (JolokiaProperties)
management.endpoint.jolokia.config.*= # Jolokia settings. Refer
to the documentation of Jolokia for more details.
management.endpoint.jolokia.enabled=true # Whether to enable
the jolokia endpoint.

# LIQUIBASE ENDPOINT (LiquibaseEndpoint)
management.endpoint.liquibase.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.liquibase.enabled=true # Whether to enable
the liquibase endpoint.

# LOG FILE ENDPOINT (LogFileWebEndpoint,
LogFileWebEndpointProperties)
management.endpoint.logfile.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.logfile.enabled=true # Whether to enable
the logfile endpoint.
management.endpoint.logfile.external-file= # External Logfile
```

```
to be accessed. Can be used if the logfile is written by output
redirect and not by the logging system itself.

# LOGGERS ENDPOINT (LoggersEndpoint)
management.endpoint.loggers.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.loggers.enabled=true # Whether to enable
the loggers endpoint.

# REQUEST MAPPING ENDPOINT  (MappingsEndpoint)
management.endpoint.mappings.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.mappings.enabled=true # Whether to enable
the mappings endpoint.

# METRICS ENDPOINT (MetricsEndpoint)
management.endpoint.metrics.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.metrics.enabled=true # Whether to enable
the metrics endpoint.

# PROMETHEUS ENDPOINT (PrometheusScrapeEndpoint)
management.endpoint.prometheus.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.prometheus.enabled=true # Whether to enable
the prometheus endpoint.

# SCHEDULED TASKS ENDPOINT (ScheduledTasksEndpoint)
management.endpoint.scheduledtasks.cache.time-to-live=0ms #
Maximum time that a response can be cached.
management.endpoint.scheduledtasks.enabled=true # Whether to
enable the scheduledtasks endpoint.

# SESSIONS ENDPOINT (SessionsEndpoint)
management.endpoint.sessions.enabled=true # Whether to enable
the sessions endpoint.

# SHUTDOWN ENDPOINT (ShutdownEndpoint)
management.endpoint.shutdown.enabled=false # Whether to enable
the shutdown endpoint.

# THREAD DUMP ENDPOINT (ThreadDumpEndpoint)
management.endpoint.threaddump.cache.time-to-live=0ms # Maximum
time that a response can be cached.
management.endpoint.threaddump.enabled=true # Whether to enable
```

```
the threaddump endpoint.

# HEALTH INDICATORS
management.health.db.enabled=true # Whether to enable database
health check.
management.health.cassandra.enabled=true # Whether to enable
Cassandra health check.
management.health.couchbase.enabled=true # Whether to enable
Couchbase health check.
management.health.couchbase.timeout=1000ms # Timeout for
getting the Bucket information from the server.
management.health.defaults.enabled=true # Whether to enable
default health indicators.
management.health.diskspace.enabled=true # Whether to enable
disk space health check.
management.health.diskspace.path= # Path used to compute the
available disk space.
management.health.diskspace.threshold=0 # Minimum disk space,
in bytes, that should be available.
management.health.elasticsearch.enabled=true # Whether to
enable Elasticsearch health check.
management.health.elasticsearch.indices= # Comma-separated
index names.
management.health.elasticsearch.response-timeout=100ms # Time
to wait for a response from the cluster.
management.health.influxdb.enabled=true # Whether to enable
InfluxDB health check.
management.health.jms.enabled=true # Whether to enable JMS
health check.
management.health.ldap.enabled=true # Whether to enable LDAP
health check.
management.health.mail.enabled=true # Whether to enable Mail
health check.
management.health.mongo.enabled=true # Whether to enable
MongoDB health check.
management.health.neo4j.enabled=true # Whether to enable Neo4j
health check.
management.health.rabbit.enabled=true # Whether to enable
RabbitMQ health check.
management.health.redis.enabled=true # Whether to enable Redis
health check.
management.health.solr.enabled=true # Whether to enable Solr
health check.
management.health.status.http-mapping= # Mapping of health
statuses to HTTP status codes. By default, registered health
```

```
statuses map to sensible defaults (for example, UP maps to
200).
management.health.status.order=DOWN,OUT_OF_SERVICE,UP,UNKNOWN #
Comma-separated list of health statuses in order of severity.

# HTTP TRACING (HttpTraceProperties)
management.trace.http.enabled=true # Whether to enable HTTP
request-response tracing.
management.trace.http.include=request-headers,response-
headers,cookies,errors # Items to be included in the trace.

# INFO CONTRIBUTORS (InfoContributorProperties)
management.info.build.enabled=true # Whether to enable build
info.
management.info.defaults.enabled=true # Whether to enable
default info contributors.
management.info.env.enabled=true # Whether to enable
environment info.
management.info.git.enabled=true # Whether to enable git info.
management.info.git.mode=simple # Mode to use to expose git
information.

# METRICS
management.metrics.binders.files.enabled=true # Whether to
enable files metrics.
management.metrics.binders.jvm.enabled=true # Whether to enable
JVM metrics.
management.metrics.binders.logback.enabled=true # Whether to
enable Logback metrics.
management.metrics.binders.processor.enabled=true # Whether to
enable processor metrics.
management.metrics.binders.uptime.enabled=true # Whether to
enable uptime metrics.
management.metrics.distribution.percentiles-histogram.*= #
Whether meter IDs starting-with the specified name should be
publish percentile histograms.
management.metrics.distribution.percentiles.*= # Specific
computed non-aggregable percentiles to ship to the backend for
meter IDs starting-with the specified name.
management.metrics.distribution.sla.*= # Specific SLA
boundaries for meter IDs starting-with the specified name. The
longest match wins, the key `all` can also be used to configure
all meters.
management.metrics.enable.*= # Whether meter IDs starting-with
the specified name should be enabled. The longest match wins,
```

```
the key `all` can also be used to configure all meters.
management.metrics.export.atlas.batch-size=10000 # Number of
measurements per request to use for this backend. If more
measurements are found, then multiple requests will be made.
management.metrics.export.atlas.config-refresh-frequency=10s #
Frequency for refreshing config settings from the LWC service.
management.metrics.export.atlas.config-time-to-live=150s # Time
to live for subscriptions from the LWC service.
management.metrics.export.atlas.config-
uri=http://localhost:7101/lwc/api/v1/expressions/local-dev #
URI for the Atlas LWC endpoint to retrieve current
subscriptions.
management.metrics.export.atlas.connect-timeout=1s # Connection
timeout for requests to this backend.
management.metrics.export.atlas.enabled=true # Whether
exporting of metrics to this backend is enabled.
management.metrics.export.atlas.eval-
uri=http://localhost:7101/lwc/api/v1/evaluate # URI for the
Atlas LWC endpoint to evaluate the data for a subscription.
management.metrics.export.atlas.lwc-enabled=false # Whether to
enable streaming to Atlas LWC.
management.metrics.export.atlas.meter-time-to-live=15m # Time
to live for meters that do not have any activity. After this
period the meter will be considered expired and will not get
reported.
management.metrics.export.atlas.num-threads=2 # Number of
threads to use with the metrics publishing scheduler.
management.metrics.export.atlas.read-timeout=10s # Read timeout
for requests to this backend.
management.metrics.export.atlas.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.atlas.uri=http://localhost:7101/api/v
1/publish # URI of the Atlas server.
management.metrics.export.datadog.api-key= # Datadog API key.
management.metrics.export.datadog.application-key= # Datadog
application key. Not strictly required, but improves the
Datadog experience by sending meter descriptions, types, and
base units to Datadog.
management.metrics.export.datadog.batch-size=10000 # Number of
measurements per request to use for this backend. If more
measurements are found, then multiple requests will be made.
management.metrics.export.datadog.connect-timeout=1s #
Connection timeout for requests to this backend.
management.metrics.export.datadog.descriptions=true # Whether
to publish descriptions metadata to Datadog. Turn this off to
```

```
minimize the amount of metadata sent.
management.metrics.export.datadog.enabled=true # Whether
exporting of metrics to this backend is enabled.
management.metrics.export.datadog.host-tag=instance # Tag that
will be mapped to "host" when shipping metrics to Datadog.
management.metrics.export.datadog.num-threads=2 # Number of
threads to use with the metrics publishing scheduler.
management.metrics.export.datadog.read-timeout=10s # Read
timeout for requests to this backend.
management.metrics.export.datadog.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.datadog.uri=https://app.datadoghq.com
# URI to ship metrics to. If you need to publish metrics to an
internal proxy en-route to Datadog, you can define the location
of the proxy with this.
management.metrics.export.ganglia.addressing-mode=multicast #
UDP addressing mode, either unicast or multicast.
management.metrics.export.ganglia.duration-units=milliseconds #
Base time unit used to report durations.
management.metrics.export.ganglia.enabled=true # Whether
exporting of metrics to Ganglia is enabled.
management.metrics.export.ganglia.host=localhost # Host of the
Ganglia server to receive exported metrics.
management.metrics.export.ganglia.port=8649 # Port of the
Ganglia server to receive exported metrics.
management.metrics.export.ganglia.protocol-version=3.1 #
Ganglia protocol version. Must be either 3.1 or 3.0.
management.metrics.export.ganglia.rate-units=seconds # Base
time unit used to report rates.
management.metrics.export.ganglia.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.ganglia.time-to-live=1 # Time to live
for metrics on Ganglia. Set the multi-cast Time-To-Live to be
one greater than the number of hops (routers) between the
hosts.
management.metrics.export.graphite.duration-units=milliseconds
# Base time unit used to report durations.
management.metrics.export.graphite.enabled=true # Whether
exporting of metrics to Graphite is enabled.
management.metrics.export.graphite.host=localhost # Host of the
Graphite server to receive exported metrics.
management.metrics.export.graphite.port=2004 # Port of the
Graphite server to receive exported metrics.
management.metrics.export.graphite.protocol=pickled # Protocol
to use while shipping data to Graphite.
```

```
management.metrics.export.graphite.rate-units=seconds # Base
time unit used to report rates.
management.metrics.export.graphite.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.graphite.tags-as-prefix= # For the
default naming convention, turn the specified tag keys into
part of the metric prefix.
management.metrics.export.influx.auto-create-db=true # Whether
to create the Influx database if it does not exist before
attempting to publish metrics to it.
management.metrics.export.influx.batch-size=10000 # Number of
measurements per request to use for this backend. If more
measurements are found, then multiple requests will be made.
management.metrics.export.influx.compressed=true # Whether to
enable GZIP compression of metrics batches published to Influx.
management.metrics.export.influx.connect-timeout=1s #
Connection timeout for requests to this backend.
management.metrics.export.influx.consistency=one # Write
consistency for each point.
management.metrics.export.influx.db=mydb # Tag that will be
mapped to "host" when shipping metrics to Influx.
management.metrics.export.influx.enabled=true # Whether
exporting of metrics to this backend is enabled.
management.metrics.export.influx.num-threads=2 # Number of
threads to use with the metrics publishing scheduler.
management.metrics.export.influx.password= # Login password of
the Influx server.
management.metrics.export.influx.read-timeout=10s # Read
timeout for requests to this backend.
management.metrics.export.influx.retention-duration= # Time
period for which Influx should retain data in the current
database.
management.metrics.export.influx.retention-shard-duration= #
Time range covered by a shard group.
management.metrics.export.influx.retention-policy= # Retention
policy to use (Influx writes to the DEFAULT retention policy if
one is not specified).
management.metrics.export.influx.retention-replication-factor=
# How many copies of the data are stored in the cluster.
management.metrics.export.influx.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.influx.uri=http://localhost:8086 #
URI of the Influx server.
management.metrics.export.influx.user-name= # Login user of the
Influx server.
```

```
management.metrics.export.jmx.domain=metrics # Metrics JMX
domain name.
management.metrics.export.jmx.enabled=true # Whether exporting
of metrics to JMX is enabled.
management.metrics.export.jmx.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.newrelic.account-id= # New Relic
account ID.
management.metrics.export.newrelic.api-key= # New Relic API
key.
management.metrics.export.newrelic.batch-size=10000 # Number of
measurements per request to use for this backend. If more
measurements are found, then multiple requests will be made.
management.metrics.export.newrelic.connect-timeout=1s #
Connection timeout for requests to this backend.
management.metrics.export.newrelic.enabled=true # Whether
exporting of metrics to this backend is enabled.
management.metrics.export.newrelic.num-threads=2 # Number of
threads to use with the metrics publishing scheduler.
management.metrics.export.newrelic.read-timeout=10s # Read
timeout for requests to this backend.
management.metrics.export.newrelic.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.newrelic.uri=https://insights-
collector.newrelic.com # URI to ship metrics to.
management.metrics.export.prometheus.descriptions=true #
Whether to enable publishing descriptions as part of the scrape
payload to Prometheus. Turn this off to minimize the amount of
data sent on each scrape.
management.metrics.export.prometheus.enabled=true # Whether
exporting of metrics to Prometheus is enabled.
management.metrics.export.prometheus.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.signalfx.access-token= # SignalFX
access token.
management.metrics.export.signalfx.batch-size=10000 # Number of
measurements per request to use for this backend. If more
measurements are found, then multiple requests will be made.
management.metrics.export.signalfx.connect-timeout=1s #
Connection timeout for requests to this backend.
management.metrics.export.signalfx.enabled=true # Whether
exporting of metrics to this backend is enabled.
management.metrics.export.signalfx.num-threads=2 # Number of
threads to use with the metrics publishing scheduler.
management.metrics.export.signalfx.read-timeout=10s # Read
```

```
timeout for requests to this backend.
management.metrics.export.signalfx.source= # Uniquely
identifies the app instance that is publishing metrics to
SignalFx. Defaults to the local host name.
management.metrics.export.signalfx.step=10s # Step size (i.e.
reporting frequency) to use.
management.metrics.export.signalfx.uri=https://ingest.signalfx.
com # URI to ship metrics to.
management.metrics.export.simple.enabled=true # Whether, in the
absence of any other exporter, exporting of metrics to an in-
memory backend is enabled.
management.metrics.export.simple.mode=cumulative # Counting
mode.
management.metrics.export.simple.step=1m # Step size (i.e.
reporting frequency) to use.
management.metrics.export.statsd.enabled=true # Whether
exporting of metrics to StatsD is enabled.
management.metrics.export.statsd.flavor=datadog # StatsD line
protocol to use.
management.metrics.export.statsd.host=localhost # Host of the
StatsD server to receive exported metrics.
management.metrics.export.statsd.max-packet-length=1400 # Total
length of a single payload should be kept within your network's
MTU.
management.metrics.export.statsd.polling-frequency=10s # How
often gauges will be polled. When a gauge is polled, its value
is recalculated and if the value has changed (or
publishUnchangedMeters is true), it is sent to the StatsD
server.
management.metrics.export.statsd.port=8125 # Port of the StatsD
server to receive exported metrics.
management.metrics.export.statsd.publish-unchanged-meters=true
# Whether to send unchanged meters to the StatsD server.
management.metrics.export.wavefront.api-token= # API token used
when publishing metrics directly to the Wavefront API host.
management.metrics.export.wavefront.batch-size=10000 # Number
of measurements per request to use for this backend. If more
measurements are found, then multiple requests will be made.
management.metrics.export.wavefront.connect-timeout=1s #
Connection timeout for requests to this backend.
management.metrics.export.wavefront.enabled=true # Whether
exporting of metrics to this backend is enabled.
management.metrics.export.wavefront.global-prefix= # Global
prefix to separate metrics originating from this app's white
box instrumentation from those originating from other Wavefront
```

```
integrations when viewed in the Wavefront UI.
management.metrics.export.wavefront.num-threads=2 # Number of
threads to use with the metrics publishing scheduler.
management.metrics.export.wavefront.read-timeout=10s # Read
timeout for requests to this backend.
management.metrics.export.wavefront.source= # Unique identifier
for the app instance that is the source of metrics being
published to Wavefront. Defaults to the local host name.
management.metrics.export.wavefront.step=10s # Step size (i.e.
reporting frequency) to use.
management.metrics.export.wavefront.uri=https://longboard.wavef
ront.com # URI to ship metrics to.
management.metrics.use-global-registry=true # Whether auto-
configured MeterRegistry implementations should be bound to the
global static registry on Metrics.
management.metrics.web.client.max-uri-tags=100 # Maximum number
of unique URI tag values allowed. After the max number of tag
values is reached, metrics with additional tag values are
denied by filter.
management.metrics.web.client.requests-metric-
name=http.client.requests # Name of the metric for sent
requests.
management.metrics.web.server.auto-time-requests=true # Whether
requests handled by Spring MVC or WebFlux should be
automatically timed.
management.metrics.web.server.max-uri-tags=100 # Maximum number
of unique URI tag values allowed. After the max number of tag
values is reached, metrics with additional tag values are
denied by filter.
management.metrics.web.server.requests-metric-
name=http.server.requests # Name of the metric for received
requests.


# ----------------------------------------
# DEVTOOLS PROPERTIES
# ----------------------------------------

# DEVTOOLS (DevToolsProperties)
spring.devtools.livereload.enabled=true # Whether to enable a
livereload.com-compatible server.
spring.devtools.livereload.port=35729 # Server port.
spring.devtools.restart.additional-exclude= # Additional
patterns that should be excluded from triggering a full
restart.
```

```
spring.devtools.restart.additional-paths= # Additional paths to
watch for changes.
spring.devtools.restart.enabled=true # Whether to enable
automatic restart.
spring.devtools.restart.exclude=META-INF/maven/**,META-
INF/resources/**,resources/**,static/**,public/**,templates/**,
**/*Test.class,**/*Tests.class,git.properties,META-INF/build-
info.properties # Patterns that should be excluded from
triggering a full restart.
spring.devtools.restart.log-condition-evaluation-delta=true #
Whether to log the condition evaluation delta upon restart.
spring.devtools.restart.poll-interval=1s # Amount of time to
wait between polling for classpath changes.
spring.devtools.restart.quiet-period=400ms # Amount of quiet
time required without any classpath changes before a restart is
triggered.
spring.devtools.restart.trigger-file= # Name of a specific file
that, when changed, triggers the restart check. If not
specified, any classpath file change triggers the restart.

# REMOTE DEVTOOLS (RemoteDevToolsProperties)
spring.devtools.remote.context-path=/.~~spring-boot!~ # Context
path used to handle the remote connection.
spring.devtools.remote.proxy.host= # The host of the proxy to
use to connect to the remote application.
spring.devtools.remote.proxy.port= # The port of the proxy to
use to connect to the remote application.
spring.devtools.remote.restart.enabled=true # Whether to enable
remote restart.
spring.devtools.remote.secret= # A shared secret required to
establish a connection (required to enable remote support).
spring.devtools.remote.secret-header-name=X-AUTH-TOKEN # HTTP
header used to transfer the shared secret.


# ----------------------------------------
# TESTING PROPERTIES
# ----------------------------------------

spring..database.replace=any # Type of existing DataSource to
replace.
spring..mockmvc.print=default # MVC Print option.
```

启动指定参数

配置资源文件位置，默认application.properties是放在jar包中的，通过spring.config.location可以制定外部配置文件，这样更便于运维。

--spring.config.location 指定配置文件

```
java -jar demo.jar --
spring.config.location=/opt/config/application.properties
```

--spring.profiles.active 切换配置文件

```
java -jar demo.jar --spring.profiles.active=dev
```

src/main/resources 准备三个环境的配置文件

```
application-dev.properties
application-pro.properties
application-tes.properties
```

加载排除

```
spring.autoconfigure.exclude=org.springframework.boot.autocon
figure.jdbc.DataSourceAutoConfiguration
```

PID FILE

```
spring.pid.fail-on-write-error= # Fail if
ApplicationPidFileWriter is used but it cannot write the PID
file.
spring.pid.file= # Location of the PID file to write (if
ApplicationPidFileWriter is used).
```

banner 关闭

```
spring.main.banner-mode=off
```

server

优雅关闭

```
server.shutdown=graceful
# timeout-per-shutdown-phase 默认缓冲 30秒,超时会强行关闭
spring.lifecycle.timeout-per-shutdown-phase=60s
```

启动之后使用 kill / kill -2 杀掉进城，注意不能使用 kill -9

```
neo@MacBook-Pro-M2 engineering % java -jar target/engineering-
0.0.1-SNAPSHOT.jar
neo@MacBook-Pro-M2 engineering % ps ax | grep engineering
57854 s005  S+     0:12.45 /usr/bin/java -jar
target/engineering-0.0.1-SNAPSHOT.jar
```

```
53810 s006  Ss+    0:00.12 /opt/homebrew/bin/fish --init-
command=source '/Applications/IntelliJ IDEA
CE.app/Contents/plugins/terminal/fish/init.fish' --login -i
56253 s007  Ss+    0:00.10 /opt/homebrew/bin/fish
neo@MacBook-Pro-M2 engineering % kill -2 57854
```

日志输出 Graceful shutdown complete 字样

```
2023-04-24T17:34:30.535+08:00  INFO 57854 --- [ionShutdownHook]
o.s.b.w.e.undertow.UndertowWebServer     : Commencing graceful
shutdown. Waiting for active requests to complete
2023-04-24T17:34:30.554+08:00  INFO 57854 --- [ionShutdownHook]
o.s.b.w.e.undertow.UndertowWebServer     : Graceful shutdown
complete
```

端口配置

```
server.port=8080 # 监听端口
server.address= # 绑定的地址
```

随机产生端口

```
# 方法一
server.port=0

# 方法二
server.port=${random.int[1000,1999]}
```

Session 配置

```
server.session.persistent 重启时是否持久化session, 默认false
server.session.timeout session的超时时间
server.session.tracking-modes 设定Session的追踪模式(cookie, url,
ssl).
server.session.timeout=1800 #session有效时长
```

cookie

```
server.session.cookie.comment 指定session cookie的comment
server.session.cookie.domain 指定session cookie的domain
server.session.cookie.http-only 否开启HttpOnly.
server.session.cookie.max-age 设定session cookie的最大age.
server.session.cookie.name 设定Session cookie 的名称.
server.session.cookie.path 设定session cookie的路径.
server.session.cookie.secure 设定session cookie的“Secure” flag.
```

案例

```
server.session.cookie.name=PHPSESSID
server.session.cookie.domain=.example.com
server.session.cookie.http-only=true
server.session.cookie.path=/
```

error 路径

```
server.error.path=/error
```

压缩传输

```
server.compression.enabled=true #是否开启压缩，默认为false.
server.compression.excluded-user-agents #指定不压缩的user-agent，多个以逗号分隔，默认值为:text/html,text/xml,text/plain,text/css
server.compression.mime-types #指定要压缩的MIME type，多个以逗号分隔.
server.compression.min-response-size #执行压缩的阈值，默认为2048

server.compression.enabled=true
server.compression.mime-types=application/json,application/xml,text/html,text/xml,text/plain,text/css,application/javascript
server.compression.min-response-size=1024
```

ssl

```
server.ssl.ciphers 是否支持SSL ciphers.
server.ssl.client-auth 设定client authentication是wanted 还是needed.
server.ssl.enabled 是否开启ssl，默认：true
server.ssl.key-alias 设定key store中key的别名.
server.ssl.key-password 访问key store中key的密码.
server.ssl.key-store 设定持有SSL certificate的key store的路径，通常是一个.jks文件.
server.ssl.key-store-password设定访问key store的密码.
server.ssl.key-store-provider设定key store的提供者.
server.ssl.key-store-type 设定key store的类型.
server.ssl.protocol 使用的SSL协议，默认：TLS
server.ssl.trust-store 持有SSL certificates的Trust store.
server.ssl.trust-store-password访问trust store的密码.
server.ssl.trust-store-provider设定trust store的提供者.
server.ssl.trust-store-type 指定trust store的类型.
```

生成证书

server.ssl.* #ssl相关配置

```
keytool -genkey -alias springboot -keyalg RSA -keystore
/www/ssl/spring
设置密码 123456
```

配置 application.properties

```
server.ssl.enabled 启动tomcat ssl配置
server.ssl.keyAlias 别名
server.ssl.keyPassword 密码
server.ssl.keyStore 位置
```

```
server.port=8443
server.ssl.enabled=true
server.ssl.keyAlias=springboot
server.ssl.keyPassword=123456
server.ssl.keyStore=/www/ssl/spring
```

logging

```
logging.file.path=/tmp # 日志目录默认为 /tmp
logging.file.name=your.log # 日志文件名称，默认为spring.log
```

```
java -jar spring-boot-app.jar --
logging.file.name=/tmp/springboot.log
```

内嵌 tomcat server

连接数配置

```
server.tomcat.max-threads=2048 # 最大线程数
```

server.tomcat.basedir

设置 Tomcat 工作目录，默认 /tmp/tomcat-docbase.7057591687859485145.7000 通过下面配置修改

```
server.tomcat.basedir=/tmp/your_project
```

access.log

如果前端有 nginx 代理这个配置可以禁用

```
server.tomcat.accesslog.enabled=true
server.tomcat.accesslog.directory=/tmp/logs
server.tomcat.accesslog.pattern=common
server.tomcat.accesslog.prefix=www.netkiller.cn.access
server.tomcat.accesslog.suffix=.log
```

charset

Spring boot 默认并非UTF-8 所以下面配置必设，否则将会出现

```
spring.messages.encoding=UTF-8
server.tomcat.uri-encoding=UTF-8
spring.http.encoding.charset=UTF-8
spring.http.encoding.enabled=true
spring.http.encoding.force=true
```

servlet

上传限制

```
spring.servlet.multipart.max-file-size=2MB
spring.servlet.multipart.max-request-size=10MB
spring.http.multipart.enabled=false
```

server.servlet.context-path

```
server.servlet.context-path=/your
```

JSON 输出与日期格式化

```
# Pretty-print JSON responses
spring.jackson.serialization.indent_output=true
```

```
#序列化时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.mvc.date-format=yyyy-MM-dd HH:mm:ss
#mvc序列化时候时区选择
spring.jackson.time-zone=GMT+8
```

SMTP 相关配置

```
spring.mail.host=smtp.netkiller.cn
#spring.mail.username=
#spring.mail.password=
#spring.mail.properties.mail.smtp.auth=true
#spring.mail.properties.mail.smtp.starttls.enable=true
#spring.mail.properties.mail.smtp.starttls.required=true
mail.smtp.debug=true
```

Redis

Springboot 1.5

```
# REDIS (RedisProperties)
# Redis数据库索引（默认为0）
spring.redis.database=0
# Redis服务器地址
spring.redis.host=localhost
# Redis服务器连接端口
spring.redis.port=6379
# Redis服务器连接密码（默认为空）
spring.redis.password=
# 连接池最大连接数（使用负值表示没有限制）
spring.redis.pool.max-active=8
# 连接池最大阻塞等待时间（使用负值表示没有限制）
```

```
spring.redis.pool.max-wait=-1
# 连接池中的最大空闲连接
spring.redis.pool.max-idle=8
# 连接池中的最小空闲连接
spring.redis.pool.min-idle=0
# 连接超时时间（毫秒）
spring.redis.timeout=0
```

Springboot 2.0

```
# REDIS (RedisProperties)
# Redis数据库索引（默认为0）
spring.redis.database=0
# Redis服务器地址
spring.redis.host=192.168.30.103
# Redis服务器连接端口
spring.redis.port=6379
# Redis服务器连接密码（默认为空）
spring.redis.password=
# 连接超时时间（毫秒）
spring.redis.timeout=0
# 连接池最大连接数（使用负值表示没有限制）
spring.redis.jedis.pool.max-active=8
# 连接池最大阻塞等待时间（使用负值表示没有限制）
spring.redis.jedis.pool.max-wait=-1
# 连接池中的最大空闲连接
spring.redis.jedis.pool.max-idle=8
# 连接池中的最小空闲连接
spring.redis.jedis.pool.min-idle=0
```

MongoDB

格式：mongodb://用户名:密码@主机地址/数据库

```
```

```
spring.data.mongodb.uri=mongodb://user:passw0rd@mdb.netkiller.c
n/

spring.data.mongodb.repositories.enabled=true
```

MySQL

```
                                spring.datasource.driver-class-
name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://主机地址:端口号/数据库
                                spring.datasource.username=用户
名
                                spring.datasource.password=密码
                                spring.jpa.database=MYSQL # 启用
JPA支持
```

Oracle

```
                                spring.datasource.driver-class-
name=oracle.jdbc.OracleDriver

spring.datasource.url=jdbc:oracle:thin:@//odb.netkiller.cn:1521
/orcl
                                spring.datasource.username=orcl

spring.datasource.password=passw0rd
                                spring.datasource.connection--
query="SELECT 1 FROM DUAL"
                                spring.jpa.database-
platform=org.hibernate.dialect.Oracle10gDialect

                                spring.jpa.show-sql=true
                                #spring.jpa.hibernate.ddl-
auto=none
                                #spring.jpa.hibernate.ddl-
auto=create-drop
                                spring.datasource.max-active=50
```

```
                                spring.datasource.initial-
size=5
                                spring.datasource.max-idle=10
                                spring.datasource.min-idle=5
                                spring.datasource.-while-
idle=true
                                spring.datasource.-on-
borrow=false
                                spring.datasource.validation-
query=SELECT 1 FROM DUAL
                                spring.datasource.time-between-
eviction-runs-millis=5000
                                spring.datasource.min-
evictable-idle-time-millis=60000
```

default_schema

```
spring.jpa.properties.hibernate.default_schema=schema
```

datasource

启用/禁用 导入 schema.sql 和 data.sql / data-${platform}.sql 其中 platform 是 spring.datasource.platform 所定义的平台

```
spring.datasource.initialize=false

spring.datasource.platform=MYSQL
```

velocity

```
spring.velocity.resourceLoaderPath=classpath:/templates/
                                spring.velocity.prefix=
```

```
                                    spring.velocity.suffix=.vm
                                    spring.velocity.cache=false
                                    spring.velocity.check-template-
location=true
                                    spring.velocity.content-
type=text/html
                                    spring.velocity.charset=UTF-8

spring.velocity.properties.input.encoding=UTF-8

spring.velocity.properties.output.encoding=UTF-8
```

禁用 velocity 模板引擎

```
                                    spring.velocity.enabled=false
                                    spring.velocity.check-template-
location=false
```

Security 相关配置

```
                                    security.user.name=user
                                    security.user.password=password
                                    security.user.role=USER
```

Web 安全

```
                                    # X-Frame-Options: DENY
                                    security.headers.frame=false

                                    security.headers.cache
                                    security.headers.content-type
                                    security.headers.hsts
                                    security.headers.xss
```

参考 https://github.com/spring-projects/spring-boot/blob/master/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/security/SecurityProperties.java#L171

MVC 配置

是否支持favicon.ico，默认为: true

```
spring.mvc.favicon.enabled=false
```

Kafka 相关配置

```
# APACHE KAFKA (KafkaProperties)
spring.kafka.admin.client-id= # ID to pass to the server when
making requests. Used for server-side logging.
spring.kafka.admin.fail-fast=false # Whether to fail fast if
the broker is not available on startup.
spring.kafka.admin.properties.*= # Additional admin-specific
properties used to configure the client.
spring.kafka.admin.ssl.key-password= # Password of the
private key in the key store file.
spring.kafka.admin.ssl.keystore-location= # Location of the
key store file.
spring.kafka.admin.ssl.keystore-password= # Store password
for the key store file.
spring.kafka.admin.ssl.keystore-type= # Type of the key
store.
spring.kafka.admin.ssl.protocol= # SSL protocol to use.
spring.kafka.admin.ssl.truststore-location= # Location of the
trust store file.
spring.kafka.admin.ssl.truststore-password= # Store password
for the trust store file.
spring.kafka.admin.ssl.truststore-type= # Type of the trust
store.
```

```
spring.kafka.bootstrap-servers= # Comma-delimited list of
host:port pairs to use for establishing the initial
connection to the Kafka cluster.
spring.kafka.client-id= # ID to pass to the server when
making requests. Used for server-side logging.
spring.kafka.consumer.auto-commit-interval= # Frequency with
which the consumer offsets are auto-committed to Kafka if
'enable.auto.commit' is set to true.
spring.kafka.consumer.auto-offset-reset= # What to do when
there is no initial offset in Kafka or if the current offset
no longer exists on the server.
spring.kafka.consumer.bootstrap-servers= # Comma-delimited
list of host:port pairs to use for establishing the initial
connection to the Kafka cluster.
spring.kafka.consumer.client-id= # ID to pass to the server
when making requests. Used for server-side logging.
spring.kafka.consumer.enable-auto-commit= # Whether the
consumer's offset is periodically committed in the
background.
spring.kafka.consumer.fetch-max-wait= # Maximum amount of
time the server blocks before answering the fetch request if
there isn't sufficient data to immediately satisfy the
requirement given by "fetch.min.bytes".
spring.kafka.consumer.fetch-min-size= # Minimum amount of
data, in bytes, the server should return for a fetch request.
spring.kafka.consumer.group-id= # Unique string that
identifies the consumer group to which this consumer belongs.
spring.kafka.consumer.heartbeat-interval= # Expected time
between heartbeats to the consumer coordinator.
spring.kafka.consumer.key-deserializer= # Deserializer class
for keys.
spring.kafka.consumer.max-poll-records= # Maximum number of
records returned in a single call to poll().
spring.kafka.consumer.properties.*= # Additional consumer-
specific properties used to configure the client.
spring.kafka.consumer.ssl.key-password= # Password of the
private key in the key store file.
spring.kafka.consumer.ssl.keystore-location= # Location of
the key store file.
spring.kafka.consumer.ssl.keystore-password= # Store password
for the key store file.
spring.kafka.consumer.ssl.keystore-type= # Type of the key
store.
spring.kafka.consumer.ssl.protocol= # SSL protocol to use.
spring.kafka.consumer.ssl.truststore-location= # Location of
```

```
the trust store file.
spring.kafka.consumer.ssl.truststore-password= # Store
password for the trust store file.
spring.kafka.consumer.ssl.truststore-type= # Type of the
trust store.
spring.kafka.consumer.value-deserializer= # Deserializer
class for values.
spring.kafka.jaas.control-flag=required # Control flag for
login configuration.
spring.kafka.jaas.enabled=false # Whether to enable JAAS
configuration.
spring.kafka.jaas.login-
module=com.sun.security.auth.module.Krb5LoginModule # Login
module.
spring.kafka.jaas.options= # Additional JAAS options.
spring.kafka.listener.ack-count= # Number of records between
offset commits when ackMode is "COUNT" or "COUNT_TIME".
spring.kafka.listener.ack-mode= # Listener AckMode. See the
spring-kafka documentation.
spring.kafka.listener.ack-time= # Time between offset commits
when ackMode is "TIME" or "COUNT_TIME".
spring.kafka.listener.client-id= # Prefix for the listener's
consumer client.id property.
spring.kafka.listener.concurrency= # Number of threads to run
in the listener containers.
spring.kafka.listener.idle-event-interval= # Time between
publishing idle consumer events (no data received).
spring.kafka.listener.log-container-config= # Whether to log
the container configuration during initialization (INFO
level).
spring.kafka.listener.monitor-interval= # Time between checks
for non-responsive consumers. If a duration suffix is not
specified, seconds will be used.
spring.kafka.listener.no-poll-threshold= # Multiplier applied
to "pollTimeout" to determine if a consumer is non-
responsive.
spring.kafka.listener.poll-timeout= # Timeout to use when
polling the consumer.
spring.kafka.listener.type=single # Listener type.
spring.kafka.producer.acks= # Number of acknowledgments the
producer requires the leader to have received before
considering a request complete.
spring.kafka.producer.batch-size= # Default batch size in
bytes.
spring.kafka.producer.bootstrap-servers= # Comma-delimited
```

```
list of host:port pairs to use for establishing the initial
connection to the Kafka cluster.
spring.kafka.producer.buffer-memory= # Total bytes of memory
the producer can use to buffer records waiting to be sent to
the server.
spring.kafka.producer.client-id= # ID to pass to the server
when making requests. Used for server-side logging.
spring.kafka.producer.compression-type= # Compression type
for all data generated by the producer.
spring.kafka.producer.key-serializer= # Serializer class for
keys.
spring.kafka.producer.properties.*= # Additional producer-
specific properties used to configure the client.
spring.kafka.producer.retries= # When greater than zero,
enables retrying of failed sends.
spring.kafka.producer.ssl.key-password= # Password of the
private key in the key store file.
spring.kafka.producer.ssl.keystore-location= # Location of
the key store file.
spring.kafka.producer.ssl.keystore-password= # Store password
for the key store file.
spring.kafka.producer.ssl.keystore-type= # Type of the key
store.
spring.kafka.producer.ssl.protocol= # SSL protocol to use.
spring.kafka.producer.ssl.truststore-location= # Location of
the trust store file.
spring.kafka.producer.ssl.truststore-password= # Store
password for the trust store file.
spring.kafka.producer.ssl.truststore-type= # Type of the
trust store.
spring.kafka.producer.transaction-id-prefix= # When non
empty, enables transaction support for producer.
spring.kafka.producer.value-serializer= # Serializer class
for values.
spring.kafka.properties.*= # Additional properties, common to
producers and consumers, used to configure the client.
spring.kafka.ssl.key-password= # Password of the private key
in the key store file.
spring.kafka.ssl.keystore-location= # Location of the key
store file.
spring.kafka.ssl.keystore-password= # Store password for the
key store file.
spring.kafka.ssl.keystore-type= # Type of the key store.
spring.kafka.ssl.protocol= # SSL protocol to use.
spring.kafka.ssl.truststore-location= # Location of the trust
```

```
store file.
spring.kafka.ssl.truststore-password= # Store password for
the trust store file.
spring.kafka.ssl.truststore-type= # Type of the trust store.
spring.kafka.template.default-topic= # Default topic to which
messages are sent.
```

5.2. Properties 文件

禁用命令行注入环境变量

```
SpringApplication.setAddCommandLineProperties(false)
```

@Value 注解

application.properties

```
server.name=Linux
server.host=192.168.0.1,172.16.0.1
```

```
        @Value("${server.name}")
        private String name;
```

@EnableConfigurationProperties 引用自定义 *.properties 配置文件

Application.java 配置NetkillerProperties.java是 @ComponentScan 扫描范围，可以不用声明下面注解。

```
@EnableConfigurationProperties(NetkillerProperties.class)
```

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoCon
figuration;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.authentication.UserCredentials;
import org.springframework.data.mongodb.MongoDbFactory;
import org.springframework.data.mongodb.core.MongoTemplate;
import
org.springframework.data.mongodb.core.SimpleMongoDbFactory;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;

import com.mongodb.Mongo;

import pojo.NetkillerProperties;

@Configuration
@SpringBootApplication
@EnableConfigurationProperties(NetkillerProperties.class)
@EnableAutoConfiguration(exclude = {
```

```
DataSourceAutoConfiguration.class })
@ComponentScan({ "web", "rest" })
@EnableMongoRepositories
public class Application {

        @SuppressWarnings("deprecation")
        public @Bean MongoDbFactory mongoDbFactory() throws
Exception {
                UserCredentials userCredentials = new
UserCredentials("finance", "your_password");
                return new SimpleMongoDbFactory(new
Mongo("mdb.netkiller.cn"), "finance", userCredentials);
        }

        public @Bean MongoTemplate mongoTemplate() throws
Exception {
                return new MongoTemplate(mongoDbFactory());
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

NetkillerProperties.java

```
package pojo;

import
org.springframework.boot.context.properties.ConfigurationProp
erties;
import org.springframework.context.annotation.Configuration;

@ConfigurationProperties(prefix="netkiller")
public class NetkillerProperties {
        private String name;
        private String email;
```

```
        private String home;

        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public String getHome() {
                return home;
        }
        public void setHome(String home) {
                this.home = home;
        }
        @Override
        public String toString() {
                return "NetkillerProperties [name=" + name +
", email=" + email + ", home=" + home + "]";
        }
}
```

IndexController.java

```
package web;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import domain.City;
```

```
import pojo.NetkillerProperties;
import repository.CityRepository;

@Controller
public class IndexController {

        @Autowired
        private CityRepository repository;

        @Autowired
        private NetkillerProperties propertie;

        @RequestMapping("/index")
        @ResponseBody
        public String index() {
        //public ModelAndView index() {

                String message = "Hello";
                //return new ModelAndView("home/welcome",
"variable", message);
                return message;
        }

        @RequestMapping("/config")
        @ResponseBody
        public String config() {
                return propertie.toString();
        }
}
```

src/main/resource/application.properties

```
netkiller.name=Neo
netkiller.email=netkiller@msn.com
netkiller.home=http://www.netkiller.cn
```

@ConfigurationProperties 默认配置是 application.properties

你可以通过 locations 指向特定配置文件

```
                                @ConfigurationProperties(prefix
= "message.api",locations =
"classpath:config/message.properties")
```

@EnableConfigurationProperties 可以导入多个配置文件

```
@EnableConfigurationProperties({NetkillerProperties.class,
NeoProperties.class})
```

手工载入 ***.properties** 文件

```
        @RequestMapping("/config")
        @ResponseBody
        public void config() {
                try {
                        Properties properties =
PropertiesLoaderUtils.loadProperties(new
ClassPathResource("/config.properties"));
                        for(String key :
properties.stringPropertyNames()) {
                                String value =
properties.getProperty(key);
                                System.out.println(key + "
=> " + value);
                        }
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
```

```

```

spring.profiles.active 参数切换配置文件

首先我们准备三个配置文件

```
src/main/resource/application-development.properties
src/main/resource/application-testing.properties
src/main/resource/application-production.properties
```

使用下面--spring.profiles.active参数切换运行环境配置文件

```
java -jar application.jar --spring.profiles.active=development
java -jar application.jar --spring.profiles.active=testing
java -jar application.jar --spring.profiles.active=production
```

分别为三个环境打包

```
mvn clean package -Pdevelopment
mvn clean package -Ptesting
mvn clean package -Pproduction
```

SpringApplicationBuilder.properties() 方法添加配置项

```

```

```
  public static void main(String[] args) {
    new
SpringApplicationBuilder(Application.class).properties("sprin
g.config.name=client").run(args);
  }
```

5.3. 参数引用

```
book.name=SpringCloud
book.author=netkiller
book.title=《${book.name}》作者 ${book.author}
```

5.4. 默认值

默认值 ${变量:默认值}

```
@Value("${server.name:Windows}") 如果application.properties没有配
置server.name那么默认值将是 Windows
private String name;

@Value("${some.key:my default value}")
private String stringWithDefaultValue;

@Value("${some.key:true}")
private boolean booleanWithDefaultValue;

@Value("${some.key:42}")
private int intWithDefaultValue;

@Value("${some.key:one,two,three}")
private String[] stringArrayWithDefaults;
```

```
@Value("${some.key:1,2,3}")
private int[] intArrayWithDefaults;

// Using SpEL
@Value("#{systemProperties['some.key'] ?: 'my default system
property value'}")
private String spelWithDefaultValue;
```

Null 默认值

```
@Value("${app.name:@null}") // app.name = null
private String name;
```

5.5. 产生随机数

```
my.secret=${random.value}
my.number=${random.int}
my.bignumber=${random.long}
my.uuid=${random.uuid}
my.number.less.than.ten=${random.int(10)}
my.number.in.range=${random.int[1024,65536]}

# 随机字符串
cn.netkiller.blog.value=${random.value}
# 随机整数
cn.netkiller.blog.number=${random.int}
# 随机长整数
cn.netkiller.blog.bignumber=${random.long}
# 随机10以内的数
cn.netkiller.blog.1=${random.int(10)}
# 随机10-20之间的数值
cn.netkiller.blog.2=${random.int[10,20]}
```

随机数

```
import lombok.extern.slf4j.Slf4j;
import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.env.RandomValuePropertySource;
import org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.test.context.junit4.SpringJUnit4ClassRunn
er;

import java.util.List;


@Slf4j
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest(webEnvironment =
SpringBootTest.WebEnvironment.NONE)
public class RandomValuePropertySourceTest {
    @Test
    public void testRandomValuePropertySource() {
        // 自定义的一个随机值属性源,起名叫做 myRandom
        RandomValuePropertySource random = new
RandomValuePropertySource("myRandom");
        // 随机生成一个整数
        log.info("random int:{}",
random.getProperty("random.int"));

        // 随机生成一个整数，指定上边界，不大于等于1
        log.info("random int(1):{}",
random.getProperty("random.int(1)"));
```

```
        // 随机生成一个整数，指定上边界，不大于等于5
        log.info("random int(5):{}",
random.getProperty("random.int(5)"));

        // 随机生成一个整数，使用区间[0,1),前闭后开=>只能是1
        // 注意区间的表示法：使用()包围,2个字符
        log.info("random int(0,1):{}",
random.getProperty("random.int(0,1)"));
        // 随机生成一个整数，使用区间[1,3),前闭后开=>只能是1或者2
        // 注意区间的表示法：使用空格包围,2个字符，前后各一个空格
        log.info("random int(1,3):{}",
random.getProperty("random.int 1,3 "));
        // 随机生成一个整数，使用区间[3,4),前闭后开=>只能是3
        // 注意区间的表示法：使用汉字包围,2个字符，前后各一个汉字自负
        log.info("random int(3,4):{}",
random.getProperty("random.int底3,4顶"));
        // 随机生成一个整数，使用区间[5,6),前闭后开=>只能是5
        // 注意区间的表示法：使用英文字母包围,2个字符，前后各一个英文字
母
        log.info("random int(5,6):{}",
random.getProperty("random.intL5,6U"));
        // 随机生成一个整数，使用区间[5,6),前闭后开=>只能是5
        // 注意区间的表示法：使用数字包围,2个字符，前一个数字5，后一个
数字6
        log.info("random int(5,6):{}",
random.getProperty("random.int55,66"));

        // 随机生成一个长整数
        log.info("random long:{}",
random.getProperty("random.long"));
        // 随机生成一个整数，使用区间[100,101),前闭后开=>只能是100
        log.info("random long(100,101):{}",
random.getProperty("random.long(100,101)"));

        // 随机生成一个 uuid
        log.info("random uuid:{}",
random.getProperty("random.uuid"));
    }
}
```

5.6. 多行字符串

```
text.content=Bright moonlight in front of bed\
 ground frost\
 up at the bright moon\
 down at home
```

5.7. 注入多值属性 arrays, list, set

处理逗号分割得值

```
        @Value("#{'${server.host}'.split(',')}")
        private List<String> host;
```

```
my.numbers=1,2,3,4,5,6,1,2
```

自定义列表属性分隔符

```
my.lists=aa;bb;cc;dd
```

```
    public test(@Value("#{'${my.lists}'.split(';')}")
List<String> list) {
        this.list = list;
        log.debug("list", list);
    }
```

5.8. containsProperty 读取配置文件

```
this.environment.containsProperty("spring.jpa.database-
platform")
```

5.9. @PropertySource 注解载入 properties 文件

```
@PropertySource("classpath:/config.properties}")

忽略FileNotFoundException，当配置文件不存在系统抛出
FileNotFoundException并终止程序运行，ignoreResourceNotFound=true
会跳过使程序能够正常运行
@PropertySource(value="classpath:config.properties",
ignoreResourceNotFound=true)
```

载入多个配置文件

```
@PropertySources({
        @PropertySource("classpath:config.properties"),
        @PropertySource("classpath:db.properties")
})
```

test.properties

```
name=Neo
```

```
age=30
```

```
package cn.netkiller.web;

import java.util.Date;

import javax.servlet.http.HttpSession;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@PropertySource("classpath:test.properties")
public class TestController {

        @Autowired
        Environment environment;

        @Value("${age}")
        private String age;

        public TestController() {
                // TODO Auto-generated constructor stub
        }

        // 环境变量方式
        @RequestMapping("/test/env")
        @ResponseBody
        public String env() {
                String message =
environment.getProperty("name");
                return message;
        }
```

```
        @RequestMapping("/test/age")
        @ResponseBody
        public String age() {
                String message = age;
                return message;
        }

}
```

5.10. List 列表类型

List类型在properties文件中使用[]来定义列表类型，比如：

```
sms.url[0]=http://api1.example.com
sms.url[1]=http://api2.example.com

netkiller.book[0].title=Netkiller Linux 手札
netkiller.book[0].author=netkiller

netkiller.book[1].title=Netkiller Spring 手札
netkiller.book[1].author=netkiller
```

注意：在Spring Boot 2.0中对于List类型数组下标的配置必须是连续的，否则会抛出UnboundConfigurationPropertiesException异常，所以如下配置是不允许的：

```
foo[0]=a
foo[2]=b
```

使用逗号分割的配置方式，上面与下面的配置是等价的：

```
sms.url[0]=http://api1.example.com,http://api2.example.com
```

在yaml文件中使用可以使用如下配置:

```
email:
  to:
    address:
      - neo@netkiller.cn
      - jam@netkiller.cn
```

逗号分割的方式:

```
email:
  to:
    address: neo@netkiller.cn, jam@netkiller.cn
```

命令行传递 List 数据

```
java -jar -D"api.url[0]=http://api1.example.com" \
        -D"api.url[1]=http://api2.example.com" \
        api.netkiller.cn-v1.0.jar

逗号分割的方式，比如:

java -jar -
Dapi.url=http://api1.example.com,http://api2.example.com
demo.jar
```

5.11. Map类型

Map类型在properties和yaml中的标准配置方式如下：

```
properties格式：
netkiller.key=value

yaml格式：
netkiller:
  key: value

举例：
user:
  name: neo
  gender: male
  age: 30
```

注意：如果Map类型的key包含字母数字和-以外的字符，需要用[]括起来，比如：

```
user:
  name:
    '[first.name]': neo
    '[last#name]': chen
```

5.12. Binder

```
cn.netkiller.author=bar
cn.netkiller.journal[0]=Spring Boot
cn.netkiller.journal[1]=Spring Cloud

cn.netkiller.books[0].title=Netkiller Spring Boot 手札
```

```
cn.netkiller.books[0].url=http://www.netkiller.cn/spring/
cn.netkiller.books[1].title=Netkiller Java 手札
cn.netkiller.books[1].url=http://www.netkiller.cn/linux/
```

```
@Data
@ConfigurationProperties(prefix = "cn.netkiller")
public class NetkillerProperties {

    public String author;

}
```

```
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(Application.class, args);

        Binder binder = Binder.get(context.getEnvironment());
        NetkillerProperties prop =
binder.bind("cn.netkiller",
Bindable.of(NetkillerProperties.class)).get();
        System.out.println(prop.author);

                List<String> journal =
binder.bind("cn.netkiller.journal",
Bindable.listOf(String.class)).get();
                System.out.println(journal);

                List<Book> books =
binder.bind("cn.netkiller.book",
Bindable.listOf(Book.class)).get();
                System.out.println(books);

    }
```

```
}
```

5.13. 加密 application.properties 中的敏感内容

http://www.jasypt.org

Maven 配置

```
                <!--
https://mvnrepository.com/artifact/com.github.ulisesbocchio/jas
ypt-spring-boot-starter -->
                <dependency>

<groupId>com.github.ulisesbocchio</groupId>
                        <artifactId>jasypt-spring-boot-
starter</artifactId>
                        <version>3.0.4</version>
                </dependency>
```

生成加密信息

```
package cn.netkiller.controller;

import org.jasypt.encryption.StringEncryptor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PasswordController {
        @Autowired
        private StringEncryptor encryptor;
```

```
        @Value("${test.password}")
        private String cleartext;

        public PasswordController() {
                // TODO Auto-generated constructor stub
        }

        @GetMapping("/password")
        public String password(@RequestParam("text") String
text) {
                return encryptor.encrypt(text);
        }

        @GetMapping("/cleartext")
        public String getPassword() {
                return this.cleartext;
        }
}
```

启动 Springboot 应用

```
java -jar your_springboot_application.jar --
jasypt.encryptor.password=123456
```

将文本 neo 加密

```
neo@MacBook-Pro-Neo ~ % curl http://localhost:8080/password\?
text\=neo
YrEdNoIyJlRoO+QhHGGwhxorlrc1e0B6Sk2iWwWMeUFd5AeCh3uAuxFr0FhEi3d
i
```

修改 application.properties 配置文件

```
test.password=ENC(YrEdNoIyJlRoO+QhHGGwhxorlrc1e0B6Sk2iWwWMeUFd5
AeCh3uAuxFr0FhEi3di)
```

重启 Springboot 项目，检验加密是否生效

```
neo@MacBook-Pro-Neo ~ % curl http://localhost:8080/cleartext
neo
```

测试环境可以将 jasypt.encryptor.password 放入配置文件，无需每次启动加入该参数。

```
jasypt.encryptor.password=123456

test.password=ENC(cH0s45ZDOHtbCNgVGgs0etnigdfZgvrnhdFokG9ysnvy4
DK0jZFPGOqe7Myow64y)
```

BasicTextEncryptor 加密文本内容

```
package cn.netkiller;

import org.jasypt.util.text.BasicTextEncryptor;

public class Password {

        public Password() {
                // TODO Auto-generated constructor stub
```

```
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                BasicTextEncryptor textEncryptor = new
BasicTextEncryptor();
                textEncryptor.setPassword("123456");
                String username =
textEncryptor.encrypt("root");
                String password =
textEncryptor.encrypt("123456");
                System.out.println("username:" + username);
                System.out.println("password:" + password);
        }
}
```

6. Spring boot with Logging

通过命令行改变日志的输出级别

```
java -jar app.jar --debug

在application.properties中配置
debug=true

application.yml
debug=true

相同的方式使能TRACE级别的日志
java -jar app.jar --trace

application.properties
trace=true

application.yml
trace=true
```

6.1. 配置日志文件

一般的日志需求可以通过配置 application.properties实现。

Spring Boot中 日志默认是输出到控制台的，这样是为了方便开发人员，但是在生产环境中应该输出到日志文件中。

配置如下

- logging.file.path: 指定日志文件的路径
- logging.file.name: 日志的文件名(默认为spring.log)
- logging.pattern.console: 控制台的输出格式
- logging.pattern.file: 日志文件的输出格式
- logging.pattern.level: 定义渲染不同级别日志的格式。默认是%5p.

提示

注意：这两个属性不能同时配置，只需要配置一个即可。

提示

旧版本

logging.file，设置文件，可以是绝对路径，也可以是相对路径。如：logging.file=my.log

logging.path，设置目录，如果 logging.file 没有设置，会在该目录下创建spring.log文件作为默认日志文件。

```
logging.file=target/spring.log
#logging.path=
```

如果仍不能满足可以使用 logback.xml 配置日志。

```
logging.path=/tmp
logging.config=classpath:logback.xml
```

日志输出级别

```
几种常见的日志级别由低到高分为：TRACE < DEBUG < INFO < WARN < ERROR < FATAL
```

显示所有DEBUG信息

```
logging.level.root=DEBUG
```

```

```

仅仅显示 springframework 调试信息

```
logging.level.org.springframework.web=DEBUG
```

仅仅显示 cn.netkiller.web.TestController 调试信息

```
private static final Logger log =
LoggerFactory.getLogger(TestController.class);

log.debug(message);

logging.level.cn.netkiller.web.TestController=DEBUG
```

YAML 配置文件写法

```
logging:
  level:
    root: info
    cn.netkiller.test: debug
    cn.netkiller.sharding.MonthShardingAlgorithm: DEBUG
```

Spring boot 2.1 以后的版本不打印 Mapped 日志问题

```
logging.level.org.springframework.web=trace
```

```

```

禁止控制台输出日志

禁止控制台日志输出，同时将日志写入日志文件.

src/main/resources/application.properties

```
logging.file.path=/tmp
logging.file.name=/tmp/spring.log
logging.level.root=INFO
logging.level.org.springframework.web=DEBUG
logging.level.org.hibernate=ERROR
```

src/main/resources/logback.xml

```
$ cat src/main/resources/logback.xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include
resource="org/springframework/boot/logging/logback/defaults.xml
" />
  <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />

  <root level="INFO">
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

使用 java -jar project-version-xxx.jar 启动后控制不会再输出日志

定制日志格式

定制日志格式有两个配置

- logging.pattern.console：控制台的输出格式
- logging.pattern.file：日志文件的输出格式

  分别是控制台的输出格式和文件中的日志输出格式

  举例

```
logging.pattern.console=%d{yyyy/MM/dd-HH:mm:ss} [%thread]
%-5level %logger- %msg%n
logging.pattern.file=%d{yyyy/MM/dd-HH:mm} [%thread] %-5level
%logger- %msg%n
```

  格式说明

```
%d{HH:mm:ss.SSS} 日志输出时间
%thread          输出日志的进程名字，这在Web应用以及异步任务处理中很有用
%-5level         日志级别，并且使用5个字符靠左对齐
%logger          日志输出者的名字
%msg             日志消息
%n                       平台的换行符
```

彩色输出

```
spring.main.banner-mode=off
spring.output.ansi.enabled=ALWAYS
logging.pattern.console=%clr(%d{yy-MM-dd E HH:mm:ss.SSS})
```

```
{blue} %clr(%-5p) %clr(${PID}){faint} %clr(---){faint}
%clr([%8.15t]){cyan} %clr(%-40.40logger{0}){blue} %clr(:)
{red} %clr(%m){faint}%n
```

6.2. 打印日志

日志的用法，首先开发中我们根据实际的需要打印不同级别的日志。

```
package cn.netkiller.web;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class TestController {

        private static final Logger log =
LoggerFactory.getLogger(TestController.class);

        @RequestMapping("/test/log")
        @ResponseBody
        public String log() {
                String message = "Test";
                log.debug(message);
                log.info(message);
                log.warn(message);
                log.error(message);
                log.trace(message);
                return message;
        }
}
```

然后通过application.properties配置那些需要显示，那些不需要，以及显示的级别是什么。

lombok

```
            <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
```

```
@Slf4j
class DemoApplicationTests {
  @Test
  public void test(){
    log.debug("输出DEBUG日志.......");
  }
}
```

6.3. logback 配置详解

配置文件名默认是：logback-spring.xml，使用其他文件名通过下面配置项指定即可。

```
logging.config=classpath:logback.xml
```

标准输出

基本配置

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appender name="stdout"
class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{yy-MMMM-dd HH:mm:ss:SSS} %5p %t
%c{2}:%L - %m%n</pattern>
        </encoder>
    </appender>
    <root level="INFO">
        <appender-ref ref="stdout"/>
    </root>
</configuration>
```

禁止 logback 日志输出

```
        <statusListener
class="ch.qos.logback.core.status.NopStatusListener" />
```

指定Class过滤日志

```
    <logger name="cn.netkiller.controller"/>

    <logger name="cn.netkiller.controller.HomeController"
 level="WARN" additivity="false">
         <appender-ref ref="console"/>
    </logger>
```

configuration 属性配置

scan:当此属性设置为true时，配置文件如果发生改变，将会被重新加载，默认值为true。
scanPeriod:设置监测配置文件是否有修改的时间间隔，如果没有给出时间单位，默认单位是毫秒。当scan为true时，此属性生效。默认的时间间隔为1分钟。
debug:当此属性设置为true时，将打印出logback内部日志信息，实时查看logback运行状态。默认值为false。

contextName 设置上下文名称

每个logger都关联到logger上下文，默认上下文名称为“default”。但可以使用设置成其他名字，用于区分不同应用程序的记录。设置后可以通过%contextName来打印日志上下文名称。

```
<contextName>logback</contextName>
```

property 设置变量

用来定义变量值的标签， 有两个属性，name和value；其中name的值是变量的名称，value的值时变量定义的值。通过定义的值会被插入到logger上下文中。定义变量后，可以使“${}”来使用变量。

```
<property name="log.path" value="/tmp" />
```

encoder 日志格式设置

<encoder>表示对日志进行编码：

%d{HH: mm:ss.SSS}——日志输出时间
%thread——输出日志的进程名字，这在Web应用以及异步任务处理中很有用
%-5level——日志级别，并且使用5个字符靠左对齐
%logger{36}——日志输出者的名字
%msg——日志消息
%n——平台的换行符

RollingFileAppender

上例中<fileNamePattern>${log.path}/logback.%d{yyyy-MM-dd}.log</fileNamePattern>定义了日志的切分方式——把每一天的日志归档到一个文件中，同理，可以使用%d{yyyy-MM-dd_HH-mm}来定义精确到分的日志切分方式。
<maxHistory>30</maxHistory>表示只保留最近30天的日志，以防止日志填满整个磁盘空间。
<totalSizeCap>1GB</totalSizeCap>用来指定日志文件的上限大小，例如设置为1GB的话，那么到了这个值，就会删除旧的日志。

按日期分割日志

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration  scan="true" scanPeriod="60 seconds"
debug="false">
    <contextName>logback</contextName>
    <property name="log.path" value="target" />
    <!--输出到控制台-->
    <appender name="console"
class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} %contextName [%thread]
%-5level %logger{36} - %msg%n</pattern>
```

```
        </encoder>
    </appender>

    <!--输出到文件-->
    <appender name="file"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${log.path}/spring.%d{yyyy-MM-
dd}.log</fileNamePattern>
        </rollingPolicy>
        <encoder>
            <pattern>%d{HH:mm:ss.SSS} %contextName [%thread]
%-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>

    <root level="info">
        <appender-ref ref="console" />
        <appender-ref ref="file" />
    </root>
</configuration>
```

按照文件尺寸分割日志

按日期分割文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <include
resource="org/springframework/boot/logging/logback/defaults.xml
" />
  <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />

<appender name="dailyRollingFileAppender"
class="ch.qos.logback.core.rolling.RollingFileAppender">
  <File>logs/spring.log</File>
  <rollingPolicy
```

```
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- daily rollover -->
    <FileNamePattern>spring.%d{yyyy-MM-
dd}.log</FileNamePattern>
    <!-- keep 30 days' worth of history -->
    <maxHistory>60</maxHistory>
  </rollingPolicy>
  <encoder>
    <Pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{35} -
%msg %n</Pattern>
  </encoder>
</appender>

  <root level="INFO">
    <appender-ref ref="FILE" />
        <appender-ref ref="dailyRollingFileAppender" />
  </root>
</configuration>
```

通过级别分割日志将 info, error, debug 分割到指定文件中。

```
<configuration scan="true" scanPeriod="10 seconds">
     <!-- 控制台日志输出-->
    <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d %p (%file:%line\)- %m%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
    </appender>
    <!-- info日志输出-->
    <appender name="INFO_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <encoder>
            <pattern>%d %p (%file:%line\)- %m%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <filter
class="ch.qos.logback.classic.filter.ThresholdFilter">
```

```
            <level>INFO</level>
        </filter>
        <File>${LOG_PATH}/www.netkiller.cn.info.log</File>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG_PATH}/www.netkiller.cn.info-
%d{yyyyMMdd}.log.%i
            </fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>10MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <layout class="ch.qos.logback.classic.PatternLayout">
            <Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread]
%-5level %logger{36} -%msg%n
            </Pattern>
        </layout>
    </appender>
    <!-- debug 日志输出-->
    <appender name="DEBUG_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <encoder>
            <pattern>%d %p (%file:%line\)- %m%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <filter
class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>DEBUG</level>
        </filter>
        <File>${LOG_PATH}/www.netkiller.cn.debug.log</File>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${LOG_PATH}/www.netkiller.cn.debug-
%d{yyyyMMdd}.log.%i
            </fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>10MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <layout class="ch.qos.logback.classic.PatternLayout">
```

```
            <Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread]
%-5level %logger{36} -%msg%n
            </Pattern>
        </layout>
    </appender>

      <!--error 日志输出配置 -->
    <appender name="ERROR_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <encoder>
            <pattern>%d %p (%file:%line\)- %m%n</pattern>
            <charset>UTF-8</charset>
        </encoder>
        <filter
class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>ERROR</level>
        </filter>
        <File>${LOG_PATH}/www.netkiller.cn.error.log</File>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${LOG_PATH}/www.netkiller.cn.error-
%d{yyyyMMdd}.log.%i</fileNamePattern>
            <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
                <maxFileSize>10MB</maxFileSize>
            </timeBasedFileNamingAndTriggeringPolicy>
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <layout class="ch.qos.logback.classic.PatternLayout">
            <Pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread]
%-5level %logger{36} -%msg%n</Pattern>
        </layout>
    </appender>

    <root level="DEBUG">
        <!--
        <appender-ref ref="STDOUT" />
        <appender-ref ref="INFO_FILE" />
        <appender-ref ref="ERROR_FILE" />
        <appender-ref ref="DEBUG_FILE" />
        -->
        <appender-ref ref="ERROR_FILE" />
        <appender-ref ref="INFO_FILE" />
        <appender-ref ref="DEBUG_FILE" />
```

```
    </root>

</configuration>
```

日志过滤

将特定日志输出保存到指定位置

```
package cn.netkiller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.Marker;
import org.slf4j.MarkerFactory;

public class Application {
        public static void main(String[] args) {
                final Logger logger =
LoggerFactory.getLogger(Application.class);
                Marker notifyAdmin =
MarkerFactory.getMarker("netkiller");
                logger.info("AAAAAAAAA");
                logger.info(notifyAdmin, "BBBBBBBBB");
                logger.error(notifyAdmin, "This is a serious
an error requiring the admin's attention", new
Exception("Just testing"));
        }
}
```

匹配到 marker 的日志才输出，通过 RollingFileAppender 可以保存到指定文件。

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<configuration>
        <include
resource="org/springframework/boot/logging/logback/defaults.x
ml" />
        <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />
        <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
                <filter
class="ch.qos.logback.core.filter.EvaluatorFilter">
                        <evaluator
class="ch.qos.logback.classic.boolex.OnMarkerEvaluator">
                                <marker>netkiller</marker>
                        </evaluator>
                        <onMatch>ACCEPT</onMatch>
                        <onMismatch>DENY</onMismatch>
                </filter>
                <encoder>
                        <pattern>%date{yyyy-MM-dd HH:mm:ss}
%-4relative [%thread] %-5level %logger{35} : %msg
%n</pattern>
                </encoder>
        </appender>
        <root level="INFO">
                <appender-ref ref="STDOUT" />
                <appender-ref ref="FILE" />
        </root>
</configuration>
```

标准输出

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
        <include
resource="org/springframework/boot/logging/logback/defaults.xml
" />
        <include
resource="org/springframework/boot/logging/logback/file-
```

```
appender.xml" />
        <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
                <encoder>
                        <pattern>%date{yyyy-MM-dd HH:mm:ss}
%-4relative [%thread] %-5level %logger{35} : %msg %n</pattern>
                </encoder>
        </appender>
        <root level="INFO">
                <appender-ref ref="STDOUT" />
                <appender-ref ref="FILE" />
        </root>
</configuration>
```

MDC

每个 userId 生成一个日志文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
        <include
resource="org/springframework/boot/logging/logback/defaults.x
ml" />
        <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />

        <property name="log.pattern" value="%d{yyyy-MM-dd
HH:mm:ss} - [%25.25(%thread)] - [%-5level] -
%-30.30(%logger{30}) : %msg%n" />

        <appender name="siftingAppender"
class="ch.qos.logback.classic.sift.SiftingAppender">
                <discriminator>
                        <key>userId</key>
                        <defaultValue>unknown</defaultValue>
                </discriminator>
                <sift>
                        <appender name="${userId}"
```

```
class="ch.qos.logback.core.rolling.RollingFileAppender">
                                <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${log.path}/${userId}.%d{yyyy-MM-
dd}.log</fileNamePattern>
                                </rollingPolicy>
                                <encoder>

<pattern>${log.pattern}</pattern>
                                </encoder>
                        </appender>
                </sift>
        </appender>

        <root level="INFO">
                <appender-ref ref="siftingAppender" />
        </root>
</configuration>
```

```
package cn.netkiller.log;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.MDC;

public class LogTest {

        private static final Logger logger =
LoggerFactory.getLogger(LogTest.class);

        public LogTest() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {

                MDC.put("userId","0001");
                logger.info("0001用户");
        MDC.clear();
```

```
        MDC.put("userId","0002");
        logger.info("0002用户");
        MDC.clear();

        }

}
```

日志写入 MongoDB

```
```

日志发送给 logstash

https://github.com/logfellow/logstash-logback-encoder/

logstash 配置

配置 logstash 增加 tcp 接收输入端。

```
input {
    tcp {
            mode => "server"
            host => "127.0.0.1"
            port => 4567
            codec => json_lines
        }
}
```

Java 项目

Maven 配置文件 pom.xml 中添加

```
<dependency>
    <groupId>net.logstash.logback</groupId>
    <artifactId>logstash-logback-encoder</artifactId>
    <version>7.2</version>
</dependency>
```

然后再resources添加logback.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true" scanPeriod="60 seconds"
debug="true">
        <include
resource="org/springframework/boot/logging/logback/defaults.xml
" />
        <include
resource="org/springframework/boot/logging/logback/console-
appender.xml" />
        <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />

        <appender name="logstash"
class="net.logstash.logback.appender.LogstashTcpSocketAppender"
>
                <destination>127.0.0.1:4567</destination>
                <encoder
class="net.logstash.logback.encoder.LogstashEncoder">
                        <providers>
                                <timestamp />
                                <logLevel />
                                <threadName />
                                <loggerName />
                                <message />
                        </providers>
```

```
                </encoder>
        </appender>
        <root level="info">
                <appender-ref ref="CONSOLE" />
                <appender-ref ref="logstash" />
        </root>
</configuration>
```

通过 **tags** 区分日志文件

logstash pipeline 配置

```
[root@netkiller ~]# cat /etc/logstash/conf.d/file.conf
input {
  tcp {
    port => 4567
    codec => json_lines
  }
}

filter {
    ruby {
        code => "event.set('datetime',
event.get('@timestamp').time.localtime.strftime('%Y-%m-%d
%H:%M:%S'))"
    }
}

output {
        if "finance" in [tags] {
            file {
                path => "/opt/log/%{app}.finance.%{+yyyy}-%
{+MM}-%{+dd}.log"
                        codec => line { format => "[%
{datetime}] %{level} %{message} %{tags}"}
                }
        } else if "market" in [tags] {
                file {
                        path => "/opt/log/%{app}.market.%
{+yyyy}-%{+MM}-%{+dd}.log"
```

```
                        codec => line { format => "[%
{datetime}] %{level} %{message} %{tags}"}
                }
        } else {
            file {
                        path => "/opt/log/%{app}.unknow.%
{+yyyy}-%{+MM}-%{+dd}.log"
                        codec => line { format => "[%
{datetime}] %{level} %{message} %{tags}"}
                }
        }
}
```

logback-spring.xml 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="false" scanPeriod="60 seconds"
debug="false">
        <include
resource="org/springframework/boot/logging/logback/defaults.xml
" />
        <include
resource="org/springframework/boot/logging/logback/console-
appender.xml" />
        <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />

        <logger name="org.springframework.web" level="INFO" />
        <logger name="org.springboot.sample" level="TRACE" />
        <property name="log.pattern" value="%date{yyyy-MM-dd
HH:mm:ss} [%thread] %-5level %logger{35}.%method: %msg%n" />
        <springProperty scope="context" name="app"
source="spring.application.name" defaultValue="spring-boot-
fusion" />
        <property name="log.path" value="/tmp" />

        <appender name="siftingAppender"
class="ch.qos.logback.classic.sift.SiftingAppender">
                <discriminator>
```

```
                        <key>userId</key>
                        <defaultValue>unknown</defaultValue>
                </discriminator>
                <sift>
                        <appender name="${userId}"
class="ch.qos.logback.core.rolling.RollingFileAppender">
                                <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${log.path}/${userId}.%d{yyyy-MM-
dd}.log</fileNamePattern>
                                </rollingPolicy>
                                <encoder>
                                        <pattern>${log.pattern}
</pattern>
                                </encoder>
                        </appender>
                </sift>
        </appender>
        <springProfile name="prod">
                <appender name="logstash"
class="net.logstash.logback.appender.LogstashTcpSocketAppender"
>

<destination>172.18.200.10:4567</destination>
                        <keepAliveDuration>5
minutes</keepAliveDuration>
                        <reconnectionDelay>3
second</reconnectionDelay>
                        <writeBufferSize>8192</writeBufferSize>

<includeCallerData>true</includeCallerData>
                        <encoder
class="net.logstash.logback.encoder.LogstashEncoder">

<shortenedLoggerNameLength>36</shortenedLoggerNameLength>
                                <timestampPattern>yyyy-MM-dd
HH:mm:ss.Asia/Shanghai</timestampPattern>

<timeZone>Asia/Shanghai</timeZone>

                                <fieldNames>

<timestamp>@timestamp</timestamp>
```

```
<version>@version</version>

<message>message</message>

<logger>logger_name</logger>
                                        <!--
<thread>thread_name</thread> -->
                                        <level>level</level>
                                        <thread>[ignore]
</thread>
                                        <levelValue>[ignore]
</levelValue>
                                </fieldNames>
                        </encoder>

                        <filter
class="ch.qos.logback.core.filter.EvaluatorFilter">
                                <evaluator
class="ch.qos.logback.classic.boolex.OnMarkerEvaluator">

<marker>finance</marker>
                                        <marker>market</marker>

<marker>customer</marker>
                                </evaluator>
                                <onMatch>ACCEPT</onMatch>
                                <onMismatch>DENY</onMismatch>
                        </filter>
                </appender>

        </springProfile>

        <root level="info">
                <springProfile name="dev">
                        <appender-ref ref="CONSOLE" />
                </springProfile>
                <springProfile name="test">
                        <appender-ref ref="CONSOLE" />
                        <appender-ref ref="FILE" />
                </springProfile>
                <springProfile name="prod">
                        <appender-ref ref="CONSOLE" />
                        <appender-ref ref="logstash" />
                </springProfile>
        </root>
```

```
</configuration>
```

打印日志

```
package cn.netkiller.controller;

import java.util.concurrent.TimeUnit;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.MDC;
import org.slf4j.Marker;
import org.slf4j.MarkerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import cn.netkiller.LogMarker;
import lombok.extern.slf4j.Slf4j;

@RestController
@Slf4j
public class HomeController {
        private static final Logger logger =
LoggerFactory.getLogger(HomeController.class);

        public HomeController() {
                // TODO Auto-generated constructor stub
        }

        @GetMapping("/")
        public String index() {

                Marker finance =
```

```
MarkerFactory.getMarker(LogMarker.finance.toString());
                Marker customer =
MarkerFactory.getMarker(LogMarker.customer.toString());
                Marker market =
MarkerFactory.getMarker(LogMarker.market.toString());
                logger.info("AAAAAAAAA");
                logger.info(finance, "test");
                logger.info(finance, "finance");
                logger.info(customer, "customer");
                logger.info(market, "market");

                MDC.put("userId", "0001");
                logger.info("0001用户");
                MDC.clear();

                MDC.put("userId", "0002");
                logger.info("0002用户");
                MDC.clear();
                return "Hello world!!!";
        }
}
```

fluentd

Maven 依赖

```
                <dependency>
                        <groupId>org.fluentd</groupId>
                        <artifactId>fluent-logger</artifactId>
                        <version>0.3.4</version>
                </dependency>
                <dependency>
                        <groupId>com.sndyuk</groupId>
                        <artifactId>logback-more-
appenders</artifactId>
                        <version>1.8.7</version>
                </dependency>
```

```
```

安装 fluent-bit

```
dnf install -y fluent-bit
```

启动 fluent-bit

```
[root@netkiller ~]# fluent-bit -i forward -o stdout
Fluent Bit v1.9.7
* Copyright (C) 2015-2022 The Fluent Bit Authors
* Fluent Bit is a CNCF sub-project under the umbrella of
Fluentd
* https://fluentbit.io

[2022/09/24 23:25:25] [ info] [fluent bit] version=1.9.7,
commit=, pid=1191240
[2022/09/24 23:25:25] [ info] [storage] version=1.2.0,
type=memory-only, sync=normal, checksum=disabled,
max_chunks_up=128
[2022/09/24 23:25:25] [ info] [cmetrics] version=0.3.5
[2022/09/24 23:25:25] [ info] [input:forward:forward.0]
listening on 0.0.0.0:24224
[2022/09/24 23:25:25] [ info] [sp] stream processor started
[2022/09/24 23:25:25] [ info] [output:stdout:stdout.0] worker
#0 started
```

配置 logback-spring.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="true">
```

```
    <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%date - %level - [%thread] - %logger -
[%file:%line] - %msg%n</pattern>
        </encoder>
    </appender>

    <appender name="FLUENT"
class="ch.qos.logback.more.appenders.DataFluentAppender">
        <tag>development</tag>
        <label>normal</label>
        <remoteHost>localhost</remoteHost>
        <port>24224</port>
        <maxQueueSize>20</maxQueueSize>
    </appender>

    <logger name="cn.netkiller.log" level="DEBUG"/>


    <root level="DEBUG">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="FLUENT" />
    </root>

</configuration>
```

查看 **fluent** 输出

```
[0] development.normal: [1664033186.000000000,
{"level"=>"INFO", "logger"=>"cn.netkiller.Application",
"thread"=>"main", "message"=>"Starting Application using Java
18 on MacBook-Pro-Neo.local with PID 85696
(/Users/neo/workspace/bottleneck/target/classes started by neo
in /Users/neo/workspace/bottleneck)"}]
[1] development.normal: [1664033186.000000000,
{"level"=>"INFO", "logger"=>"cn.netkiller.Application",
"thread"=>"main", "message"=>"The following 1 profile is
```

```
active: "prod""}]
[0] development.normal: [1664033187.000000000,
{"level"=>"INFO",
"logger"=>"org.springframework.data.repository.config.Repositor
yConfigurationDelegate", "thread"=>"main", "message"=>"Multiple
Spring Data modules found, entering strict repository
configuration mode"}]
[1] development.normal: [1664033187.000000000,
{"level"=>"INFO",
"logger"=>"org.springframework.data.repository.config.Repositor
yConfigurationDelegate", "thread"=>"main",
"message"=>"Bootstrapping Spring Data Redis repositories in
DEFAULT mode."}]
[2] development.normal: [1664033187.000000000,
{"level"=>"INFO",
"logger"=>"org.springframework.data.repository.config.Repositor
yConfigurationDelegate", "thread"=>"main", "message"=>"Finished
Spring Data repository scanning in 6 ms. Found 0 Redis
repository interfaces."}]
[0] development.normal: [1664033188.000000000,
{"level"=>"INFO",
"logger"=>"org.springframework.boot.web.embedded.tomcat.TomcatW
ebServer", "thread"=>"main", "message"=>"Tomcat initialized
with port(s): 8080 (http)"}]
[1] development.normal: [1664033188.000000000,
{"level"=>"INFO",
"logger"=>"org.apache.catalina.core.StandardService",
"thread"=>"main", "message"=>"Starting service [Tomcat]"}]
[2] development.normal: [1664033188.000000000,
{"level"=>"INFO",
"logger"=>"org.apache.catalina.core.StandardEngine",
"thread"=>"main", "message"=>"Starting Servlet engine: [Apache
Tomcat/9.0.65]"}]
[3] development.normal: [1664033188.000000000,
{"level"=>"INFO",
"logger"=>"org.apache.catalina.core.ContainerBase.[Tomcat].
[localhost].[/]", "thread"=>"main", "message"=>"Initializing
Spring embedded WebApplicationContext"}]
[4] development.normal: [1664033188.000000000,
{"level"=>"INFO",
"logger"=>"org.springframework.boot.web.servlet.context.Servlet
WebServerApplicationContext", "thread"=>"main",
"message"=>"Root WebApplicationContext: initialization
completed in 2133 ms"}]
[0] development.normal: [1664033189.000000000,
```

```
{"level"=>"INFO",
"logger"=>"org.springframework.boot.actuate.endpoint.web.EndpointLinksResolver", "thread"=>"main", "message"=>"Exposing 14 endpoint(s) beneath base path '/actuator'"}]
[0] development.normal: [1664033189.000000000,
{"level"=>"INFO",
"logger"=>"org.springframework.boot.web.embedded.tomcat.TomcatWebServer", "thread"=>"main", "message"=>"Tomcat started on port(s): 8080 (http) with context path ''"}]
[1] development.normal: [1664033189.000000000,
{"level"=>"INFO", "logger"=>"cn.netkiller.Application",
"thread"=>"main", "message"=>"Started Application in 4.224 seconds (JVM running for 4.918)"}]
```

Loki4j Logback

https://loki4j.github.io/loki-logback-appender/

Maven

```
<dependency>
    <groupId>com.github.loki4j</groupId>
    <artifactId>loki-logback-appender</artifactId>
    <version>1.3.2</version>
</dependency>
```

logback.xml

```
<appender name="LOKI"
class="com.github.loki4j.logback.Loki4jAppender">
    <http>
        <url>http://localhost:3100/loki/api/v1/push</url>
    </http>
    <format>
```

```
        <label>
            <pattern>app=my-
app,host=${HOSTNAME},level=%level</pattern>
        </label>
        <message>
            <pattern>l=%level h=${HOSTNAME} c=%logger{20}
t=%thread | %msg %ex</pattern>
        </message>
        <sortByTime>true</sortByTime>
    </format>
</appender>

<root level="DEBUG">
    <appender-ref ref="LOKI" />
</root>
```

6.4. Log4j2 + Gelf + Logstash

https://logging.paluch.biz/examples/log4j-2.x.html

Maven 配置

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                        <!-- Exclude the Tomcat dependency -->
                        <exclusions>
                                <exclusion>

<groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-
boot-starter-tomcat</artifactId>
                                </exclusion>
                                <!-- 禁用 logback -->
                                <exclusion>
```

```
<groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-
boot-starter-logging</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <!-- 添加Log4j2 依赖 -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
log4j2</artifactId>
                </dependency>
                <dependency>
                        <groupId>biz.paluch.logging</groupId>
                        <artifactId>logstash-gelf</artifactId>
                        <version>1.15.0</version>
                </dependency>
```

log4j2.xml 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
        <properties>
                <property name="log.pattern">[%d{yyyy-MM-dd
HH:mm:ss}] [${hostName}] [%p] [%t] %l - %m%n</property>
                <property name="log.dir">/tmp/logs</property>
                <property name="log.level">info</property>
        </properties>
        <Appenders>
                <!-- 控制台 -->
                <Console name="Console" target="SYSTEM_OUT">
                        <PatternLayout pattern="${log.pattern}"
/>
                </Console>

                <!-- INFO级别日志 -->
                <RollingFile name="RollingFileInfo"
fileName="${log.dir}/info.log"
```

```
filePattern="${log.dir}/info.%d{yyyy-MM-dd}.log">
                        <Filters>
                                <ThresholdFilter level="INFO"
/>
                                <ThresholdFilter level="WARN"
onMatch="DENY" onMismatch="NEUTRAL" />
                        </Filters>
                        <PatternLayout pattern="${log.pattern}"
/>
                        <Policies>
                                <TimeBasedTriggeringPolicy
interval="1" modulate="false" />
                        </Policies>
                </RollingFile>

                <!-- WARN级别日志 -->
                <RollingFile name="RollingFileWarn"
fileName="${log.dir}/warn.log"
filePattern="${log.dir}/warn.%d{yyyy-MM-dd}.log">
                        <Filters>
                                <ThresholdFilter level="WARN"
/>
                                <ThresholdFilter level="ERROR"
onMatch="DENY" onMismatch="NEUTRAL" />
                        </Filters>
                        <PatternLayout pattern="${log.pattern}"
/>
                        <Policies>
                                <TimeBasedTriggeringPolicy
interval="1" modulate="false" />
                        </Policies>
                </RollingFile>

                <!-- ERROR级别日志 -->
                <RollingFile name="RollingFileError"
fileName="${log.dir}/error.log"
filePattern="${log.dir}/error.%d{yyyy-MM-dd}.log">
                        <Filters>
                                <ThresholdFilter level="ERROR"
/>
                        </Filters>
                        <PatternLayout pattern="${log.pattern}"
/>
                        <Policies>
                                <TimeBasedTriggeringPolicy
```

```
interval="1" modulate="false" />
                        </Policies>
                </RollingFile>

                <Gelf name="Gelf" host="udp:172.18.200.10"
port="12201" version="1.1" extractStackTrace="true"
filterStackTrace="true" mdcProfiling="true"
includeFullMdc="true" maximumMessageSize="8192"
originHost="%host{fqdn}">
                        <Field name="timestamp"
pattern="%d{yyyy-MM-dd HH:mm:ss.SSS}" />
                        <Field name="logger" pattern="%logger"
/>
                        <Field name="level" pattern="%level" />
                        <Field name="class" pattern="%C{1}" />
                        <Field name="method" pattern="%M" />
                        <Field name="line" pattern="%L" />
                        <Field name="marker" pattern="%marker"
/>
                        <Filters>
                                <MarkerFilter marker="finance"
onMatch="ACCEPT" onMismatch="NEUTRAL" />
                                <MarkerFilter marker="market"
onMatch="ACCEPT" onMismatch="DENY" />
                        </Filters>
                </Gelf>
        </Appenders>
        <Loggers>
                <Root level="${sys:log.level}">
                        <AppenderRef ref="Console" />
                        <AppenderRef ref="Gelf" />
                        <!-- <AppenderRef ref="RollingFileInfo"
/> <AppenderRef ref="RollingFileWarn" /> <AppenderRef
ref="RollingFileError" /> -->
                </Root>
        </Loggers>
</Configuration>
```

Java 测试代码

```
        @GetMapping("/log")
        public String log() {
                Marker finance =
MarkerFactory.getMarker(LogMarker.finance.toString());
                Marker customer =
MarkerFactory.getMarker(LogMarker.customer.toString());
                Marker market =
MarkerFactory.getMarker(LogMarker.market.toString());
                logger.info("常规日志");
                logger.info(finance, "test");
                logger.info(finance, "finance");
                logger.info(customer, "customer");
                logger.info(market, "market");
                return "OK!!!\r\n";
        }

        @GetMapping("/log/marker")
        public String marker(@RequestParam("marker") String
marker, @RequestParam("msg") String msg) {
                logger.info(MarkerFactory.getMarker(marker),
msg);
                msg += "\r\n";
                return msg;
        }
```

Logstash 配置

```
[root@netkiller log]# cat /etc/logstash/conf.d/file.conf
input {
  tcp {
        port => 4567
        codec => json_lines
  }
  gelf {
    port => 12201
    use_udp => true
    #use_tcp => true
```

```
  }
}

filter {
        ruby {
            code => "event.set('datetime',
event.get('@timestamp').time.localtime.strftime('%Y-%m-%d
%H:%M:%S'))"
        }
}

output {

    file {
        path => "/opt/log/%{marker}.%{+yyyy}-%{+MM}-%{+dd}.log"
        codec => line { format => "[%{datetime}] %{level} %
{message}"}
    }

    file {
        path => "/opt/log/origin.%{+yyyy}-%{+MM}-%{+dd}.log.gz"
                codec => json_lines
        gzip => true
    }

}
```

测试结果

```
[root@netkiller log]# ls
finance.2022-11-16.log  market.2022-11-16.log  origin.2022-11-
16.log.gz

[root@netkiller log]# cat finance.2022-11-16.log
[2022-11-16 15:02:36] INFO test
[2022-11-16 15:02:36] INFO finance
[2022-11-16 15:21:34] INFO test
[2022-11-16 15:21:34] INFO finance
```

```
[root@netkiller log]# cat market.2022-11-16.log
[2022-11-16 15:02:36] INFO market
[2022-11-16 15:21:34] INFO market

[root@netkiller log]# zcat origin.2022-11-16.log.gz |jq
{
  "datetime": "2022-11-16 15:21:34",
  "timestamp": "2022-11-16 15:21:34.185",
  "message": "market",
  "host": "macbook-pro-neo.local",
  "level": "INFO",
  "line": 53,
  "@version": "1",
  "@timestamp": "2022-11-16T07:21:34.185Z",
  "marker": "market",
  "logger": "cn.netkiller.controller.HomeController",
  "version": "1.1",
  "method": "log",
  "class": "HomeController",
  "source_host": "172.18.5.142",
  "facility": "logstash-gelf"
}
{
  "datetime": "2022-11-16 15:21:34",
  "timestamp": "2022-11-16 15:21:34.143",
  "message": "test",
  "host": "macbook-pro-neo.local",
  "level": "INFO",
  "line": 49,
  "@version": "1",
  "@timestamp": "2022-11-16T07:21:34.143Z",
  "marker": "finance",
  "logger": "cn.netkiller.controller.HomeController",
  "version": "1.1",
  "method": "log",
  "class": "HomeController",
  "source_host": "172.18.5.142",
  "facility": "logstash-gelf"
}
{
  "datetime": "2022-11-16 15:21:34",
  "timestamp": "2022-11-16 15:21:34.184",
  "message": "finance",
  "host": "macbook-pro-neo.local",
  "level": "INFO",
```

```
  "line": 51,
  "@version": "1",
  "@timestamp": "2022-11-16T07:21:34.184Z",
  "marker": "finance",
  "logger": "cn.netkiller.controller.HomeController",
  "version": "1.1",
  "method": "log",
  "class": "HomeController",
  "source_host": "172.18.5.142",
  "facility": "logstash-gelf"
}
```

Log4j2 更多技巧

多环境配置

方案一

```
logging:
  config: classpath:log4j2-${spring.profiles.active}.xml
```

方案二

```
@SpringBootApplication
public class Application implements CommandLineRunner {

    @Autowired
    private Environment env;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
```

```
    @Override
    public void run(String... param) {
        if
(Arrays.asList(env.getActiveProfiles()).contains("dev")) {
            Configurator.initialize(null, "/path/to/log4j2-
dev.xml");
        } else {
            Configurator.initialize(null,
"/path/to/log4j2.xml");
        }
    }
}
```

控制 **class** 日志输出级别

```
#日志配置 无特殊需求无需更改
logging:
  config:  classpath:log4j2.xml
  level:
    root: INFO
    javax.activation: info
    org.apache.catalina: INFO
    org.apache.commons.beanutils.converters: INFO
    org.apache.coyote.http11.Http11Processor: INFO
    org.apache.http: INFO
    org.apache.tomcat: INFO
    org.springframework: INFO
    com.chinamobile.cmss.bdpaas.resource.monitor: DEBUG
```

日志输出级别

```
    <Loggers>
        <Logger
name="com.ensd.service.sharding.MonthShardingAlgorithm"
level="ERROR" />
```

```
        <Root level="${sys:log.level}">
            <AppenderRef ref="Console"/>
            <AppenderRef ref="File"/>
            <AppenderRef ref="Logstash"/>
        </Root>
    </Loggers>
```

读取系统变量/环境变量

```
${sys:catalina.home}/logs
${env:log.home}/logs
```

读取 **spring** 配置

引入依赖

```
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-spring-
boot</artifactId>
                </dependency>
```

例子

```
<Configuration name="ConfigTest" status="ERROR"
monitorInterval="5">
        <properties>
                <property
name="applicationName">${spring:spring.application.name}
```

```
</property>
        </properties>
        <Appenders>

        <SpringProfile name="dev | staging">
                <Console name="Out">
                        <PatternLayout pattern="%m%n"/>
                </Console>
        </SpringProfile>
        <SpringProfile name="prod">
                <List name="Out">
                </List>
        </SpringProfile>

        </Appenders>
        <Loggers>
                <Logger name="org.apache.test" level="trace"
additivity="false">
                <AppenderRef ref="Out"/>
                </Logger>
                <Root level="error">
                <AppenderRef ref="Out"/>
                </Root>
        </Loggers>
</Configuration>
```

读取方法 ${spring:spring.profiles.active}

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
        <properties>
                <property name="log.pattern">[%d{yyyy-MM-dd
HH:mm:ss}] [${hostName}] [%p] [%t] %l - %m%n</property>
                <property name="log.home">/tmp/logs</property>
                <property name="log.level">info</property>
        </properties>
        <Appenders>
                <Console name="Console" target="SYSTEM_OUT">
                        <PatternLayout pattern="${log.pattern}"
/>
```

```
                </Console>
                <Gelf name="dev" host="udp:172.18.200.10"
port="12201" version="1.1" extractStackTrace="true"
filterStackTrace="true" mdcProfiling="true"
includeFullMdc="true" maximumMessageSize="8192"
originHost="%host{fqdn}">
                        <Field name="timestamp"
pattern="%d{yyyy-MM-dd HH:mm:ss.SSS}" />
                        <Field name="logger" pattern="%logger"
/>
                        <Field name="level" pattern="%level" />
                        <Field name="class" pattern="%C{1}" />
                        <Field name="method" pattern="%M" />
                        <Field name="line" pattern="%L" />
                        <Field name="marker" pattern="%marker"
/>
                        <Filters>
                                <MarkerFilter marker="finance"
onMatch="ACCEPT" onMismatch="NEUTRAL" />
                                <MarkerFilter marker="market"
onMatch="ACCEPT" onMismatch="DENY" />
                        </Filters>
                </Gelf>
        </Appenders>
        <Loggers>
                <Root level="${sys:log.level}">
                        <AppenderRef ref="Console" />
                        <AppenderRef
ref="${spring:spring.profiles.active:-dev}" />
                </Root>
        </Loggers>
</Configuration>
```

Spring 2.1.4 无法获取配置，解决方法使用 sys，同时启动的时候增加系统配置项 java -Dspring.application.name=netkiller -Dspring.profiles.active=dev -jar netkiller.jar

```
<property name="service">${sys:spring.application.name}
</property>
<property name="environment">${sys:spring.profiles.active}
```

```
</property>
```

变量默认值

格式是 ${变量名:-默认值}

<property name="service">${sys:spring.application.name:-dev} </property> <property name="environment">${sys:spring.profiles.active:-dev}</property>

6.5. 日志报警

Logstash 配置

将 ERROR 和 WARN 级别的日志发送到钉钉群

```
[root@netkiller ~]# cat /etc/logstash/conf.d/file.conf
input {
        tcp {
        port => 4567
        codec => json_lines
        }
        gelf {
        port => 12201
        use_udp => true
        #use_tcp => true
        }
}

filter {
        ruby {
                code => "event.set('datetime',
event.get('@timestamp').time.localtime.strftime('%Y-%m-%d
%H:%M:%S'))"
        }
}
```

```
output {

        file {
                path => "/opt/log/%{marker}.%{+yyyy}-%{+MM}-%
{+dd}.log"
                codec => line { format => "[%{datetime}] %
{level} %{message}"}
        }

        file {
                path => "/opt/log/origin.%{+yyyy}-%{+MM}-%
{+dd}.log.gz"
                codec => json_lines
                gzip => true
        }
        if "ERROR" in [level] or "WARN" in [level] {
                http {
                        url =>
"https://oapi.dingtalk.com/robot/send?
access_token=56c27cb734a56cf549f6977ecc2761c4a16473db02d9d2881d
008f9a239ba3e0"
                        http_method => "post"
                        content_type => "application/json;
charset=utf-8"
                        format => "message"
                        message => '{"msgtype":"text","text":
{"content":"Monitor: %{host}  - %{message}"}}'
                }
        }
}
```

监控 SpringBootApplication 的启动和退出

```
neo@MacBook-Pro-M2 ~ % cat
workspace/bottleneck/src/main/java/cn/netkiller/Application.jav
a
package cn.netkiller;

import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import org.slf4j.MarkerFactory;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryClien
t;

import jakarta.annotation.PostConstruct;
import jakarta.annotation.PreDestroy;

@EnableDiscoveryClient
@SpringBootApplication
public class Application {
        private static final Logger logger =
LoggerFactory.getLogger(Application.class);

        @PostConstruct
        public void init() {
                System.out.printf("==================== init
====================");
                logger.warn(MarkerFactory.getMarker("finance"),
"XXX 系统启动");
        }

        @PreDestroy
        public void destroy() {
                System.out.printf("==================== destroy
====================");

logger.error(MarkerFactory.getMarker("finance"), "XXX 系统销
毁");
        }

        public static void main(String[] args) {
                System.out.println("Netkiller bottleneck
tool!");
                SpringApplication.run(Application.class, args);
        }
}
```

@PostConstruct 可以监控 启动情况

@PreDestroy 可以监控 退出情况

6.6. Spring boot with ELK(Elasticsearch + Logstash + Kibana)

将 Spring boot 日志写入 ELK 有多种实现方式，这里仅提供三种方案:

1. Spring boot -> logback -> Tcp/IP -> logstash -> elasticsearch

   这种方式实现非常方便不需要而外包或者软件

2. Spring boot -> logback -> Redis -> logstash -> elasticsearch

   利用 Redis 提供的发布订阅功能将日志投递到 elasticsearch

3. Spring boot -> logback -> Kafka -> logstash -> elasticsearch

   Kafka 方法适合大数据的情况。

TCP 方案

logstash 配置

```
input {
  tcp {
    host => "172.16.1.16"
    port => 9250
    mode => "server"
    tags => ["tags"]
    codec => json_lines  //可能需要更新logstash插件
  }
}

output {
 stdout{codec =>rubydebug}
```

```
  elasticsearch {
   hosts => ["localhost:9200"]  //这块配置需要带端口号
    flush_size => 1000

  }
}
```

Spring boot logback.xml 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property resource="properties/logback-
variables.properties" />

    <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
        <encoder charset="UTF-8">
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level
%logger - %msg%n
            </pattern>
        </encoder>
    </appender>
    <appender name="LOGSTASH"
class="net.logstash.logback.appender.LogstashTcpSocketAppende
r">
        <destination>172.16.1.16:9250</destination>
        <encoder charset="UTF-8"
class="net.logstash.logback.encoder.LogstashEncoder" />
    </appender>

    <!--<appender name="async"
class="ch.qos.logback.classic.AsyncAppender">-->
        <!--<appender-ref ref="stash" />-->
    <!--</appender>-->

    <root level="info">                    <!-- 设置日志级别 --
>
        <appender-ref ref="STDOUT" />
        <appender-ref ref="LOGSTASH" />
```

```
    </root>
</configuration>
```

Redis 方案

https://github.com/kmtong/logback-redis-appender

Maven pom.xml 增加 Logback Redis 依赖

```
<!-- https://mvnrepository.com/artifact/com.cwbase/logback-
redis-appender -->
<dependency>
    <groupId>com.cwbase</groupId>
    <artifactId>logback-redis-appender</artifactId>
    <version>1.1.5</version>
</dependency>
```

Spring boot logback.xml 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
        <include
resource="org/springframework/boot/logging/logback/defaults.x
ml" />
        <include
resource="org/springframework/boot/logging/logback/file-
appender.xml" />
        <property name="type.name" value="test" />
        <appender name="LOGSTASH"
class="com.cwbase.logback.RedisAppender">
                <source>spring-application</source>
                <type>${type.name}</type>
                <host>localhost</host>
```

```
                <key>logstash:redis</key>
                <tags>test-2</tags>
                <mdc>true</mdc>
                <location>true</location>
                <callerStackIndex>0</callerStackIndex>
                <!--additionalField添加附加字段 用于head插件显示
-->
                <additionalField>
                        <key>MyKey</key>
                        <value>MyValue</value>
                </additionalField>
                <additionalField>
                        <key>MySecondKey</key>
                        <value>MyOtherValue</value>
                </additionalField>
        </appender>
        <root level="INFO">
                <appender-ref ref="FILE" />
                <appender-ref ref="LOGSTASH" />
        </root>
</configuration>
```

logstash 配置

```
input {
    redis {
        host => 'localhost'
        data_type => 'list'
        port => "6379"
        key => 'logstash:redis' #自定义
        type => 'redis-input'   #自定义
    }
}
output {
    elasticsearch {
        host => "localhost"
        codec => "json"
        protocol => "http"
    }
```

```
}
```

Kafka 方案

```

```

Other

```

```

7. Undertow

Undertow 是红帽公司开发的一款基于 NIO 的高性能 Web 嵌入式服务器

7.1. Maven 依赖

```
<dependencies>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
            <!-- Exclude the Tomcat dependency -->
            <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
tomcat</artifactId>
            </exclusion>
        </exclusions>
    </dependency>

    <!-- Use Undertow instead -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-undertow</artifactId>
    </dependency>

</dependencies>
```

7.2. Application

```
import org.springframework.boot.SpringApplication;
import
```

```
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class);
    }

    @GetMapping(value = "/undertow/test")
    public String undertow() {
        return "hello undertow";
    }

}
```

7.3. 相关配置

Undertow 日志配置

```
#存放目录
server.undertow.accesslog.dir=
# 是否启用日志
server.undertow.accesslog.enabled=false
# 日志格式
server.undertow.accesslog.pattern=common
# 日志文件名前缀
server.undertow.accesslog.prefix=access_log
# 日志文件名后缀
server.undertow.accesslog.suffix=log
```

HTTP 相关配置

```
# HTTP POST请求最大的大小
server.undertow.max-http-post-size=0
# 设置IO线程数, 它主要执行非阻塞的任务,它们会负责多个连接, 默认设置每个CPU核心一个线程
server.undertow.io-threads=4
# 阻塞任务线程池, 当执行类似servlet请求阻塞操作, undertow会从这个线程池中取得线程,它的值设置取决于系统的负载
server.undertow.worker-threads=20
# 以下的配置会影响buffer,这些buffer会用于服务器连接的IO操作,有点类似netty的池化内存管理
# 每块buffer的空间大小,越小的空间被利用越充分
server.undertow.buffer-size=1024
# 每个区分配的buffer数量 , 所以pool的大小是buffer-size * buffers-per-region
server.undertow.buffers-per-region=1024
# 是否分配的直接内存
server.undertow.direct-buffers=true
```

8. Spring boot with Jetty

使用 Jetty 替代 Tomcat

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<exclusions>
        <!-- Exclude the Tomcat dependency -->
        <exclusion>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
tomcat</artifactId>
        </exclusion>
</exclusions>
</dependency>
<!-- Use Jetty instead -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

9. Spring boot with HTTP2 SSL

9.1. 生成自签名证书

```
keytool -genkey -alias www.netkiller.cn -keyalg RSA -keystore
/www/netkiller.cn/www.netkiller.cn.keystore
```

导入证书（Windows）

```
keytool -selfcert -alias www.netkiller.cn -keystore
www.netkiller.cn.keystore
keytool -export -alias www.netkiller.cn -keystore
www.netkiller.cn.keystore -storepass passw0rd -rfc -file
www.netkiller.cn.cer
```

找到 Java 安装路径

```
[root@localhost ~]# alternatives --list
libnssckbi.so.x86_64    auto    /usr/lib64/pkcs11/p11-kit-
trust.so
python                  auto    /usr/libexec/no-python
cifs-idmap-plugin       auto    /usr/lib64/cifs-
utils/cifs_idmap_sss.so
ifup                    auto    /usr/libexec/nm-ifup
ld                      auto    /usr/bin/ld.bfd
python3                 auto    /usr/bin/python3.6
dockerd                 auto    /usr/bin/dockerd-ce
java                    manual  /usr/lib/jvm/java-14-openjdk-
14.0.2.12-1.rolling.el8.x86_64/bin/java
```

```
jre_openjdk              auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.262.b10-0.el8_2.x86_64/jre
jre_14                   auto    /usr/lib/jvm/java-14-openjdk-
14.0.2.12-1.rolling.el8.x86_64
jre_14_openjdk           auto    /usr/lib/jvm/jre-14-openjdk-
14.0.2.12-1.rolling.el8.x86_64
javac                    auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.262.b10-0.el8_2.x86_64/bin/javac
java_sdk_openjdk         auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.262.b10-0.el8_2.x86_64
java_sdk_14              auto    /usr/lib/jvm/java-14-openjdk-
14.0.2.12-1.rolling.el8.x86_64
java_sdk_14_openjdk      auto    /usr/lib/jvm/java-14-openjdk-
14.0.2.12-1.rolling.el8.x86_64
jre_1.8.0                auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.262.b10-0.el8_2.x86_64/jre
jre_1.8.0_openjdk        auto    /usr/lib/jvm/jre-1.8.0-openjdk-
1.8.0.262.b10-0.el8_2.x86_64
java_sdk_1.8.0           auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.262.b10-0.el8_2.x86_64
java_sdk_1.8.0_openjdk   auto    /usr/lib/jvm/java-1.8.0-
openjdk-1.8.0.262.b10-0.el8_2.x86_64
mvn                      auto    /usr/share/maven/bin/mvn
```

导入证书（JVM）

```
keytool -importcert -alias www.netkiller.cn -file
www.netkiller.cn.cer -keystore
/srv/java/jre/lib/security/cacerts
```

9.2. application.properties 配置文件

配置Tomcat HTTPS 端口 8443（由于JVM不能fork和setuid，所以无法向nginx,apache httpd 那样设置 80 端口，除非你使用root用户运行，但这样做是不安全的。）

```
server.port=8443
server.ssl.enabled=true
server.ssl.key-
store=/www/netkiller.cn/www.netkiller.cn.keystore
server.ssl.key-store-password=passw0rd
server.ssl.key-store-type=JKS
server.ssl.key-alias=www.netkiller.cn
```

keystore 文件可以放到 classpath 中，首先将证书文件放到 src/main/resources 目录中，然后配置 application.properties 如下：

```
server.port=8443
server.ssl.enabled=true
server.ssl.key-store=classpath:www.netkiller.cn.keystore
server.ssl.key-store-password=passw0rd
server.ssl.key-store-type=JKS
server.ssl.key-alias=www.netkiller.cn
```

9.3. 启动 Spring boot

```
/srv/java/bin/java -server -Xms2048m -Xmx8192m -
Djava.security.egd=file:/dev/./urandom -jar
/www/netkiller.cn/www.netkiller.cn/www.netkiller.cn-0.0.1.war
```

9.4. restTemplate 调用实例

```
```

```
String url =
"https://www.netkiller.cn:8443/public/test/version.json";
ResponseEntity<RestResponse<String>> result =
restTemplate.exchange(url, HttpMethod.GET, null, new
ParameterizedTypeReference<RestResponse<String>>() {});
```

9.5. HTTP2

启用 HTTP2 必须使用 Tomcat 9 以上， Springboot 2.1

创建证书

```
keytool -genkey -alias localhost -storetype PKCS12 -keyalg RSA
-keysize 2048 -storepass passw0rd -keystore localhost.p12 -
dname "CN=localhost, OU=netkiller, O=netkiller.cn, L=Guangdong,
ST=Shenzhen, C=CN"
keytool -selfcert -alias localhost -storepass passw0rd -
keystore localhost.p12
keytool -export -alias localhost -keystore localhost.p12 -
storepass passw0rd -rfc -file localhost.cer
keytool -importcert -trustcacerts -alias localhost -file
localhost.cer -storepass passw0rd -keystore
/etc/pki/java/cacerts
```

如果你是自己安装的JDK，需要找到cacerts安装路径

```
keytool -importcert -trustcacerts -alias localhost -file
localhost.cer -storepass passw0rd -keystore
/srv/java/jre/lib/security/cacerts
```

MacOS 添加方法，当提示你输入密码的时候，输入：changeit

```
iMac:resources neo$ sudo keytool -importcert -trustcacerts -
alias localhost -file localhost.cer -cacerts
Password:
输入密钥库口令:
所有者: CN=localhost, OU=netkiller, O=netkiller.cn, L=Guangdong,
ST=Shenzhen, C=CN
发布者: CN=localhost, OU=netkiller, O=netkiller.cn, L=Guangdong,
ST=Shenzhen, C=CN
序列号: ffd28d78add2b56c
生效时间: Mon Sep 07 16:55:39 CST 2020, 失效时间: Sun Dec 06
16:55:39 CST 2020
证书指纹:
         SHA1:
A0:DB:69:34:66:EA:16:A3:AF:65:31:F9:5D:6E:C0:70:CA:5F:0E:22
         SHA256:
2C:04:B7:BB:28:25:B5:E6:7C:0F:73:4B:02:38:6E:04:80:42:E2:F7:61:
5C:91:4D:A8:EA:5E:20:2E:82:4F:0C
签名算法名称: SHA256withRSA
主体公共密钥算法: 2048 位 RSA 密钥
版本: 3

扩展:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 4E 30 9A EC C1 9D FB C2   CC 55 B2 6D 0D F4 01 CE
N0.......U.m....
0010: 13 C6 62 38                                        ..b8
]
]

是否信任此证书? [否]:  Y
证书已添加到密钥库中

iMac:resources neo$ keytool -list -cacerts -alias localhost
输入密钥库口令:
localhost, 2020年9月8日, trustedCertEntry,
证书指纹 (SHA-256):
2C:04:B7:BB:28:25:B5:E6:7C:0F:73:4B:02:38:6E:04:80:42:E2:F7:61:
5C:91:4D:A8:EA:5E:20:2E:82:4F:0C
```

```

```

配置启用 http2

```
server:
  port: 8443
  servlet:
    context-path: /
  ssl:
    enabled: true
    key-store: classpath:ssl/localhost.p12
    key-store-type: PKCS12
    key-store-password: 123456
  http2:
    enabled: true
```

我的配置

```
spring.application.name=web
server.port=8443
#server.servlet.context-path=/
server.ssl.enabled=true
server.ssl.key-store=classpath:localhost.p12
server.ssl.key-store-type=PKCS12
server.ssl.key-store-password=123456
server.http2.enabled=true
```

使用 curl 访问可以看到 HTTP/2 字样，表示成功

```
neo@MacBook-Pro ~ % curl -i -k https://localhost:8443/ping
HTTP/2 200
content-type: text/plain;charset=UTF-8
```

```
content-length: 4
date: Tue, 09 Apr 2019 08:41:29 GMT

Pong%
```

10. Spring boot with Webpage

ViewResolver

10.1. Maven

```
                <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>javax.servlet-
api</artifactId>
                </dependency>
                <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>jstl</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
tomcat</artifactId>
                </dependency>
                <dependency>

<groupId>org.apache.tomcat.embed</groupId>
                        <artifactId>tomcat-embed-
jasper</artifactId>
                </dependency>
```

10.2. application.properties

```
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

10.3. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
@EnableScheduling
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }
}
```

10.4. IndexController

```
package cn.netkiller.web;

import java.util.HashMap;
import java.util.Map;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class IndexController {

        @RequestMapping("/welcome")
        @ResponseBody
        public String welcome() {
                String message = "Welcome";
                return message;
        }

        @RequestMapping("/index")
        public ModelAndView index() {
                String message = "Helloworld";
                return new
ModelAndView("index").addObject("message", message);
        }
}
```

10.5. src/main/webapp/WEB-INF/jsp/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Home</title>
</head>
```

```
<body>
${message}
</body>
</html>
```

10.6. 集成模板引擎

如果你需要使用其他模板引擎可以采用 Bean 注解方式。

```
package cn.netkiller.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.DefaultServ
letHandlerConfigurer;
import
org.springframework.web.servlet.config.annotation.EnableWebMv
c;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;
import
org.springframework.web.servlet.view.InternalResourceViewReso
lver;

@Configuration
@EnableWebMvc
public class WebMvcConfig extends WebMvcConfigurerAdapter {

        @Override
        public void
configureDefaultServletHandling(DefaultServletHandlerConfigur
er configurer) {
                configurer.enable();
        }

        @Bean
        public InternalResourceViewResolver viewResolver() {
```

```
                InternalResourceViewResolver resolver = new
InternalResourceViewResolver();
                resolver.setPrefix("WEB-INF/jsp/");
                resolver.setSuffix(".jsp");
                return resolver;
        }

}
```

11. Spring boot with Velocity template

11.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
velocity</artifactId>
                </dependency>
                <dependency>

<groupId>org.apache.velocity</groupId>
                        <artifactId>velocity</artifactId>
                </dependency>
```

例 5.1. Spring boot with Velocity template (pom.xml)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>api.netkiller.cn</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
```

```
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.3.1.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency> -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
```

```
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
amqp</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
devtools</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
mongodb</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
oracle</artifactId>
                        <version>1.0.0.RELEASE</version>
                </dependency>

                <dependency>
                        <groupId>com.oracle</groupId>
                        <artifactId>ojdbc6</artifactId>
                        <!-- <version>12.1.0.1</version> -->
                        <version>11.2.0.3</version>
                        <scope>system</scope>

<systemPath>${basedir}/lib/ojdbc6.jar</systemPath>
                </dependency>
```

```
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
mail</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
velocity</artifactId>
                </dependency>
                <dependency>

<groupId>org.apache.velocity</groupId>
                        <artifactId>velocity</artifactId>
                </dependency>
                <dependency>

<groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                        <scope>compile</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                    <artifactId>spring-boot-
maven-plugin</artifactId>
```

```
                        </plugin>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-war-
plugin</artifactId>
                                <version>2.6</version>
                                <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>
                                </configuration>
                        </plugin>
                </plugins>
        </build>

</project>
```

11.2. Resource

src/main/resources/application.properties

```
spring.velocity.resourceLoaderPath=classpath:/templates/
spring.velocity.prefix=
spring.velocity.suffix=.vm
spring.velocity.cache=false
spring.velocity.check-template-location=true
spring.velocity.content-type=text/html
spring.velocity.charset=UTF-8
spring.velocity.properties.input.encoding=UTF-8
spring.velocity.properties.output.encoding=UTF-8
```

```

```

src/main/resources/templates/email.vm

```
<html>
<body>
    <h3>${title}!</h3>
    <p>${body}</p>
</body>
</html>
```

11.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.web.servlet.config.annotation.CorsRegistr
y;
```

```
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;

import api.ApplicationConfiguration;

@SpringBootApplication
@EnableConfigurationProperties(ApplicationConfiguration.class
)
@EnableAutoConfiguration
@ComponentScan({ "api.web", "api.rest", "api.service" })
@EnableMongoRepositories
@EnableJpaRepositories
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

11.4. RestController

```
package api.rest;

import java.io.File;
import java.util.HashMap;
import java.util.Map;

import javax.mail.internet.MimeMessage;

import org.apache.velocity.app.VelocityEngine;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.FileSystemResource;
```

```
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.ui.velocity.VelocityEngineUtils;
import org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.ResponseStatus;
import
org.springframework.web.bind.annotation.RestController;

import api.pojo.Email;

@RestController
@RequestMapping("/v1/email")
public class EmailRestController extends CommonRestController
{

        @Autowired
        private JavaMailSender javaMailSender;

        @Autowired
        private VelocityEngine velocityEngine;

        @RequestMapping("version")
        @ResponseStatus(HttpStatus.OK)
        public String version() {
                return "[OK] Welcome to withdraw Restful
version 1.0";
        }

        @RequestMapping(value = "send", method =
RequestMethod.POST, produces = { "application/xml",
"application/json" })
        public ResponseEntity<Email>
sendSimpleMail(@RequestBody Email email) {
                SimpleMailMessage message = new
SimpleMailMessage();
                message.setFrom(email.getFrom());
                message.setTo(email.getTo());
                message.setSubject(email.getSubject());
```

```
                message.setText(email.getText());
                javaMailSender.send(message);
                email.setStatus(true);

                return new ResponseEntity<Email>(email,
HttpStatus.OK);
        }

        @RequestMapping(value = "attachments", method =
RequestMethod.POST, produces = { "application/xml",
"application/json" })
        public ResponseEntity<Email> attachments(@RequestBody
Email email) throws Exception {

                MimeMessage mimeMessage =
javaMailSender.createMimeMessage();

                MimeMessageHelper mimeMessageHelper = new
MimeMessageHelper(mimeMessage, true);
                mimeMessageHelper.setFrom(email.getFrom());
                mimeMessageHelper.setTo(email.getTo());

mimeMessageHelper.setSubject(email.getSubject());
                mimeMessageHelper.setText("<html><body><img
src=\"cid:banner\" >" + email.getText() + "</body></html>",
true);

                FileSystemResource file = new
FileSystemResource(new File("banner.jpg"));
                mimeMessageHelper.addInline("banner", file);

                FileSystemResource fileSystemResource = new
FileSystemResource(new File("Attachment.jpg"));

mimeMessageHelper.addAttachment("Attachment.jpg",
fileSystemResource);

                javaMailSender.send(mimeMessage);
                email.setStatus(true);

                return new ResponseEntity<Email>(email,
HttpStatus.OK);
        }

        @RequestMapping(value = "template", method =
```

```
RequestMethod.POST, produces = { "application/xml",
"application/json" })
        public ResponseEntity<Email> template(@RequestBody
Email email) throws Exception {

                Map<String, Object> model = new
HashMap<String, Object>();
                model.put("title", email.getSubject());
                model.put("body", email.getText());
                String text =
VelocityEngineUtils.mergeTemplateIntoString(velocityEngine,
"email.vm", "UTF-8", model);

                System.out.println(text);

                MimeMessage mimeMessage =
javaMailSender.createMimeMessage();

                MimeMessageHelper mimeMessageHelper = new
MimeMessageHelper(mimeMessage, true);
                mimeMessageHelper.setFrom(email.getFrom());
                mimeMessageHelper.setTo(email.getTo());

mimeMessageHelper.setSubject(email.getSubject());
                mimeMessageHelper.setText(text, true);

                javaMailSender.send(mimeMessage);

                email.setStatus(true);

                return new ResponseEntity<Email>(email,
HttpStatus.OK);
        }
}
```

11.5. Test

```
$ curl -i -H "Accept: application/json" -H "Content-Type:
```

```
application/json" -X POST -d '{"from":"www@netkiller.cn",
"to":"21214094@qq.com","subject":"Hello","text":"Hello
world!!!"}' http://172.16.0.20:8080/v1/email/template.json
```

12. Spring boot with Thymeleaf

12.1. Maven

```
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
thymeleaf</artifactId>
        </dependency>
```

12.2. application.properties

创建目录 src/main/resources/templates/ 用户存放模板文件

```
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
spring.thymeleaf.mode=HTML5
spring.thymeleaf.encoding=UTF-8
spring.thymeleaf.content-type=text/html
spring.thymeleaf.cache=false
```

12.3. Controller

```
package cn.netkiller.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
```

```
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/")
public class HelloController {

        @GetMapping("/hello")
        // 如果此处使用 @ResponseBody，将会返回 "hello" 字符串，而
不是模板
        public String test() {
                return "hello";

        }

        @RequestMapping("/hello1")
        public String hello1(Map<String, Object> map) {
                // 传递参数测试
                map.put("name", "Neo");
                return "thymeleaf";
        }

        @RequestMapping("/hello2")
        public ModelAndView hello2() {
                ModelAndView mv = new ModelAndView();
                mv.addObject("name", "Amy");
                mv.setViewName("thymeleaf");
                return mv;
        }

        @RequestMapping(value = "/{name}", method =
RequestMethod.GET)
        public String getMovie(@PathVariable String name,
ModelMap model) {
                model.addAttribute("name", name);
                return "hello";
        }

}
```

12.4. HTML5 Template

在 src/main/resources/templates/ 目录下创建模板文件 hello.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<title>Spring MVC + Thymeleaf Example</title>
</head>
<body>
        <h1>Welcome to Thymeleaf</h1>
        <span th:text="${name}"></span>
</body>
</html>
```

13. Spring boot with MongoDB

13.1. Maven

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>api.netkiller.cn</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.0.2.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
<!--
```

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>
                -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
amqp</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
devtools</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
mongodb</artifactId>
                </dependency>

                <dependency>

<groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                        <scope>compile</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                    <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
                                </configuration>
                        </plugin>
```

```
                        <plugin>
                                <artifactId>maven-war-
plugin</artifactId>
                                <version>2.6</version>
                                <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>
                                </configuration>
                        </plugin>
                </plugins>
        </build>

</project>
```

13.2. Application

Application.java

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoCon
figuration;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.authentication.UserCredentials;
import org.springframework.data.mongodb.MongoDbFactory;
import org.springframework.data.mongodb.core.MongoTemplate;
import
org.springframework.data.mongodb.core.SimpleMongoDbFactory;
import
```

```
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;

import com.mongodb.Mongo;

@Configuration
@SpringBootApplication
@EnableAutoConfiguration(exclude = {
DataSourceAutoConfiguration.class })
@ComponentScan({ "web", "rest" })
@EnableMongoRepositories
public class Application {

        @SuppressWarnings("deprecation")
        public @Bean MongoDbFactory mongoDbFactory() throws
Exception {
                UserCredentials userCredentials = new
UserCredentials("finance",
"En7d0l0wssXQ8owzedjb82I0BMd4pFoZ");
                return new SimpleMongoDbFactory(new
Mongo("db.netkiller.cn"), "finance", userCredentials);
        }

        public @Bean MongoTemplate mongoTemplate() throws
Exception {
                return new MongoTemplate(mongoDbFactory());
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

13.3. MongoTemplate

```
package web;
```

```
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import api.domain.City;
import api.domain.Article;
import api.ApplicationConfiguration;
import api.repository.CityRepository;
import api.repository.ArticleRepository;
import api.service.TestService;

@Controller
public class IndexController {

        @Autowired
        private CityRepository repository;

        @Autowired
        private TestService testService;

        @Autowired
        private ApplicationConfiguration propertie;

        @RequestMapping("/repository")
        @ResponseBody
        public String repository() {

                repository.deleteAll();

                // save a couple of city
                repository.save(new City("Shenzhen",
"China"));
                repository.save(new City("Beijing",
"China"));

                System.out.println("-------------------------
-------");
                // fetch all city
                for (City city : repository.findAll()) {
                        System.out.println(city);
```

```
                }
                // fetch an individual city
                System.out.println("-------------------------
-------");

System.out.println(repository.findByName("Shenzhen"));
                System.out.println("-------------------------
-------");
                for (City city :
repository.findByCountry("China")) {
                        System.out.println(city);
                }

                String message = "Hello";
                return message;
        }

}
```

13.4. Repository

在上一节 MongoTemplate 中，继续添加下面代码。

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.web.servlet.config.annotation.CorsRegistr
y;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
```

```
gurer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;

@SpringBootApplication
@EnableAutoConfiguration(exclude = {
DataSourceAutoConfiguration.class })
@ComponentScan({ "api.web", "api.rest","api.service" })
@EnableMongoRepositories
public class Application {


    public @Bean WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry
registry) {
                registry.addMapping("/**");
            }
        };
    }
        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

Resource: src/main/resources/application.properties

```
spring.data.mongodb.uri=mongodb://finance:XQ8os82I0pFoZBMd4@m
db.netkiller.cn/finance
spring.data.mongodb.repositories.enabled=true
```

CityRepository.java

```
package repository;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import
org.springframework.data.mongodb.repository.MongoRepository;

import domain.City;

public interface CityRepository extends MongoRepository<City,
String> {
        public Page<City> findAll(Pageable pageable);

        public City findByNameAndCountry(String name, String
country);

        public City findByName(String name);

        public List<City> findByCountry(String country);

}
```

City.java

```
package domain;

import org.springframework.data.annotation.Id;
import
org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "city")
public class City {

        @Id
    private String id;
```

```
        public String name;
        public String country;

        public City(String name, String country){
                this.setName(name);
                this.setCountry(country);
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getCountry() {
                return country;
        }
        public void setCountry(String country) {
                this.country = country;
        }
        @Override
        public String toString() {
                return "City [id=" + id + ", name=" + name +
", country=" + country + "]";
        }

}
```

在 IndexController 中调用 CityRepository

```
package web;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import repository.CityRepository;
```

```
import domain.City;
import repository.CityRepository;

@Controller
public class IndexController {

        @Autowired
        private CityRepository repository;

        @RequestMapping("/index")
        @ResponseBody
        public ModelAndView index() {
                String message = "Hello";
                return new ModelAndView("home/welcome",
"variable", message);
        }

        @RequestMapping("/curd")
        @ResponseBody
        public String curd() {

                repository.deleteAll();

                // save a couple of city
                repository.save(new City("Shenzhen",
"China"));
                repository.save(new City("Beijing",
"China"));

                System.out.println("-------------------------
-------");
                // fetch all city
                for (City city : repository.findAll()) {
                        System.out.println(city);
                }
                // fetch an individual city
                System.out.println("-------------------------
-------");

System.out.println(repository.findByName("Shenzhen"));
                System.out.println("-------------------------
-------");
                for (City city :
repository.findByCountry("China")) {
```

```
                        System.out.println(city);
                }

                String message = "Hello";
                return message;
        }
}
```

14. Spring boot with MySQL

14.1. Maven

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>api.netkiller.cn</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.0.2.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
<!--
```

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
amqp</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
devtools</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
mongodb</artifactId>
                </dependency>
                <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
                <dependency>
                        <groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                        <scope>compile</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
```

```
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-war-
plugin</artifactId>
                                <version>2.6</version>
                                <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>
                                </configuration>
                        </plugin>
                </plugins>
        </build>

</project>
```

14.2. Resource

src/main/resources/application.properties

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://192.168.6.1:3306/test
spring.datasource.username=root
spring.datasource.password=password

spring.jpa.database=MYSQL
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
#spring.jpa.hibernate.ddl-auto=create-drop
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/test
spring.datasource.username=root
spring.datasource.password=pasword
```

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.max-idle=10
spring.datasource.max-wait=10000
spring.datasource.min-idle=5
spring.datasource.initial-size=5
spring.datasource.validation-query=SELECT 1
spring.datasource.test-on-borrow=false
spring.datasource.test-while-idle=true
spring.datasource.time-between-eviction-runs-millis=18800
spring.datasource.jdbc-
interceptors=ConnectionState;SlowQueryReport(threshold=0)
```

14.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;

import api.ApplicationConfiguration;

@SpringBootApplication
@EnableConfigurationProperties(ApplicationConfiguration.class
)
@EnableAutoConfiguration
@ComponentScan({ "api.web", "api.rest","api.service" })
@EnableMongoRepositories
public class Application {

        public static void main(String[] args) {
```

```
                SpringApplication.run(Application.class,
args);
        }

}
```

14.4. JdbcTemplate

```
package api.web;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import api.domain.City;
import api.domain.Article;
import api.ApplicationConfiguration;
import api.repository.CityRepository;
import api.repository.ArticleRepository;
import api.service.TestService;

@Controller
public class IndexController {

        @Autowired
        private CityRepository repository;

        @Autowired
        private TestService testService;

        @Autowired
        private ApplicationConfiguration propertie;

        @Autowired
```

```
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value = "/article")
        public @ResponseBody String dailyStats(@RequestParam
Integer id) {
                String query = "SELECT id, title, content
from article where id = " + id;

                return jdbcTemplate.queryForObject(query,
(resultSet, i) -> {

System.out.println(resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                        return (resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                });
        }
}
```

14.5. CrudRepository

ArticleRepository

```
package api.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import api.domain.Article;

@Repository
public interface ArticleRepository extends
CrudRepository<Article, Long> {

    Page<Article> findAll(Pageable pageable);
```

```
}
```

Article.java

```
package api.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Table;

@Entity
@Table(name = "article")
public class Article implements Serializable {
	private static final long serialVersionUID =
7998903421265538801L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name = "id", unique=true, nullable=false,
insertable=true, updatable = false)
	private Long id;
	private String title;
	private String content;

	public Article(){

	}
	public Article(String title, String content) {
		this.title = title;
		this.content = content;
	}

	public Long getId() {
		return id;
```

```
	}

	public void setId(long id) {
		this.id = id;
	}

	public String getTitle() {
		return title;
	}

	public void setTitle(String title) {
		this.title = title;
	}

	public String getContent() {
		return content;
	}

	public void setContent(String content) {
		this.content = content;
	}

	@Override
	public String toString() {
		return "Article [id=" + id + ", title=" +
title + ", content=" + content + "]";
	}

}
```

```
package api.web;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
import api.domain.City;
import api.domain.Article;
import api.ApplicationConfiguration;
import api.repository.CityRepository;
import api.repository.ArticleRepository;
import api.service.TestService;

@Controller
public class IndexController {

        @Autowired
        private CityRepository repository;

        @Autowired
        private TestService testService;

        @Autowired
        private ApplicationConfiguration propertie;

        @Autowired
        private ArticleRepository articleRepository;

        @RequestMapping("/save")
        @ResponseBody
        public String save() {
                articleRepository.save(new Article("Neo",
"Chen"));
                return "OK";
        }

        @RequestMapping("/mysql")
        @ResponseBody
        public String mysql() {

                for (Article article :
articleRepository.findAll()) {
                        System.out.println(article);
                }
                return "OK";
        }
}
```

15. Spring boot with Oracle

15.1. Maven

首先到oracle官网，根据你的Oracle数据库，下载ojdbc6.jar(Oracle 11) 或者 ojdbc7.jar (Oracle 12)

```
        <dependency>
            <groupId>com.oracle</groupId>
            <artifactId>ojdbc7</artifactId>
            <version>12.1.0.1</version>
        </dependency>
```

```
                        mvn install:install-file -
DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=11.2.0.3 -
Dpackaging=jar -Dfile=ojdbc6.jar -DgeneratePom=true
                        mvn install:install-file -
Dfile=ojdbc7.jar -DgroupId=com.oracle -DartifactId=ojdbc7 -
Dversion=12.1.0.1 -Dpackaging=jar
```

另一种方案是在项目根目录下创建一个/lib文件夹，将下载的驱动放入该文件夹中。然后pom.xml 加入下面代码

ojdbc6.jar 例子

```
        <dependency>
            <groupId>com.oracle</groupId>
            <artifactId>ojdbc6</artifactId>
            <version>11.2.0.3</version>
            <scope>system</scope>
```

```
<systemPath>${basedir}/lib/ojdbc6.jar</systemPath>
        </dependency>
```

ojdbc7.jar 例子

```
        <dependency>
            <groupId>com.oracle</groupId>
            <artifactId>ojdbc7</artifactId>
            <version>12.1.0.1</version>
            <scope>system</scope>

<systemPath>${basedir}/lib/ojdbc7.jar</systemPath>
        </dependency>
```

例 5.2. Example Spring boot with Oracle

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>api.netkiller.cn</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
```

```
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.0.2.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <!-- <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency> -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
```

```
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
amqp</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
devtools</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
mongodb</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
oracle</artifactId>
                        <version>1.0.0.RELEASE</version>
                </dependency>

        <dependency>
            <groupId>com.oracle</groupId>
            <artifactId>ojdbc6</artifactId>
            <!-- <version>12.1.0.1</version> -->
            <version>11.2.0.3</version>
            <scope>system</scope>

<systemPath>${basedir}/lib/ojdbc6.jar</systemPath>
        </dependency>
```

```
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                </dependency>
                <dependency>

<groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                        <scope>compile</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-war-
plugin</artifactId>
                                <version>2.6</version>
                                <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>
```

```
<failOnMissingWebXml>false</failOnMissingWebXml>
                                    </configuration>
                            </plugin>
                    </plugins>
            </build>

</project>
```

15.2. application.properties

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@//192.168.4.9:1521/or
cl.example.com
spring.datasource.username=www
spring.datasource.password=123123
spring.jpa.database-
platform=org.hibernate.dialect.Oracle10gDialect

spring.jpa.show-sql=true
#spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.ddl-auto=create-drop
```

其他配置

```
                        spring.datasource.connection-test-
query="SELECT 1 FROM DUAL"
                        spring.datasource.test-while-idle=true
                        spring.datasource.test-on-borrow=true
```

Oracle RAC

```
                        jdbc:oracle:thin@(DESCRIPTION=
```

```
                        (LOAD_BALANCE=on)
                        (ADDRESS=(PROTOCOL=TCP)(HOST=racnode1)
(PORT=1521))
                        (ADDRESS=(PROTOCOL=TCP)(HOST=racnode2)
(PORT=1521))
                        (CONNECT_DATA=
(SERVICE_NAME=service_name)))

                        jdbc:oracle:thin:@(DESCRIPTION=
                        (ADDRESS=(PROTOCOL=TCP)
(HOST=125.22.42.68)(PORT=1521))
                        (LOAD_BALANCE=on)
                        (FAILOVER=ON)
                        (CONNECT_DATA=
                        (SERVER=DEDICATED)
                        (SERVICE_NAME=service_name)
                        (FAILOVER_MODE=(TYPE=SESSION)
(METHOD=BASIC))
                        )
                        )
```

15.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
```

```
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.web.servlet.config.annotation.CorsRegistr
y;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;

import api.ApplicationConfiguration;

@SpringBootApplication
@EnableConfigurationProperties(ApplicationConfiguration.class
)
@EnableAutoConfiguration
@ComponentScan({ "api.web", "api.rest", "api.service" })
@EnableMongoRepositories
@EnableJpaRepositories
public class Application {

        public @Bean WebMvcConfigurer corsConfigurer() {
                return new WebMvcConfigurerAdapter() {
                        @Override
                        public void
addCorsMappings(CorsRegistry registry) {
                                registry.addMapping("/**");
                        }
                };
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

15.4. CrudRepository

```
package api.repository;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import api.domain.Article;

@Repository
public interface ArticleRepository extends
CrudRepository<Article, Long> {

    Page<Article> findAll(Pageable pageable);

    Article findByTitle(String title);

    //@Query("select id,title,content from Article where id >
?1")
    //public List<Article> findBySearch(@Param("id")long id);

}
```

```
package api.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Table;

@Entity
@Table(name = "article")
public class Article implements Serializable {
	private static final long serialVersionUID =
7998903421265538801L;

	@Id
	@Column(name = "ID")
	@GeneratedValue(strategy=GenerationType.SEQUENCE,
generator = "id_Sequence")
	@SequenceGenerator(name = "id_Sequence", sequenceName
= "ID_SEQ")
	private Long id;
	private String title;
	private String content;

	public Article(){

	}
	public Article(String title, String content) {
		this.title = title;
		this.content = content;
	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getTitle() {
		return title;
	}

	public void setTitle(String title) {
		this.title = title;
	}

	public String getContent() {
		return content;
	}
```

```
        public void setContent(String content) {
                this.content = content;
        }

        @Override
        public String toString() {
                return "Article [id=" + id + ", title=" +
title + ", content=" + content + "]";
        }

}
```

15.5. JdbcTemplate

```
        @Autowired
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value = "/article")
        public @ResponseBody String dailyStats(@RequestParam
Integer id) {
                String query = "SELECT id, title, content
from article where id = " + id;

                return jdbcTemplate.queryForObject(query,
(resultSet, i) -> {

System.out.println(resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                        return (resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                });
        }
```

15.6. Controller

```
package api.web;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import api.domain.Article;
import api.repository.ArticleRepository;

@Controller
public class IndexController {

        @Autowired
        private ArticleRepository articleRepository;

        @RequestMapping("/mysql")
        @ResponseBody
        public String mysql() {
                repository.deleteAll();
                return "Deleted"
        }

        @RequestMapping("/mysql")
        @ResponseBody
        public String mysql() {
                articleRepository.save(new Article("Neo",
"Chen"));
                for (Article article :
articleRepository.findAll()) {
                        System.out.println(article);
                }
                Article tmp =
articleRepository.findByTitle("Neo");
                return tmp.getTitle();
        }
```

```
        /*
        @RequestMapping("/search")
        @ResponseBody
        public String search() {

                /*for (Article article :
articleRepository.findBySearch(1)) {
                        System.out.println(article);
                }*/
                List<Article> tmp =
articleRepository.findBySearch(1L);

                tmp.forEach((temp) -> {
                        System.out.println(temp.toString());
                });

                return tmp.get(0).getTitle();
        }
        */

        @Autowired
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value = "/article")
        public @ResponseBody String dailyStats(@RequestParam
Integer id) {
                String query = "SELECT id, title, content
from article where id = " + id;

                return jdbcTemplate.queryForObject(query,
(resultSet, i) -> {

System.out.println(resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                        return (resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                });

        }
}
```

16. Spring boot with PostgreSQL

16.1. pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<!--
https://mvnrepository.com/artifact/org.postgresql/postgresql
-->
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>9.4.1212</version>
</dependency>
```

16.2. application.properties

```
spring.datasource.url=jdbc:postgresql://localhost:5432/your-
database
spring.datasource.username=postgres
spring.datasource.password=postgres

spring.jpa.database=POSTGRESQL
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.generate-ddl=true
```

16.3. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
@EnableMongoRepositories
@EnableJpaRepositories
@EnableScheduling
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }
}
```

16.4. CrudRepository

Model Class

```
package cn.netkiller.model;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "customer")
public class Customer implements Serializable {

        private static final long serialVersionUID =
-3009077722242246666L;
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private long id;

        @Column(name = "firstname")
        private String firstName;

        @Column(name = "lastname")
        private String lastName;

        protected Customer() {
        }

        public Customer(String firstName, String lastName) {
                this.firstName = firstName;
                this.lastName = lastName;
        }

        @Override
        public String toString() {
                return String.format("Customer[id=%d,
firstName='%s', lastName='%s']", id, firstName, lastName);
        }
```

```
}
```

CrudRepository

```
package cn.netkiller.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import cn.netkiller.model.Customer;

public interface CustomerRepository extends
CrudRepository<Customer, Long>{
        List<Customer> findByFirstName(String firstName);
    List<Customer> findByLastName(String lastName);
}
```

16.5. JdbcTemplate

```
        @Autowired
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value = "/jdbc")
        public @ResponseBody String dailyStats(@RequestParam
Integer id) {
                String query = "SELECT id, firstname,
lastname from customer where id = " + id;

                return jdbcTemplate.queryForObject(query,
(resultSet, i) -> {

System.out.println(resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
```

```
                        return (resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                });
        }
```

16.6. Controller

```
package cn.netkiller.web;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import cn.netkiller.model.Customer;
import cn.netkiller.repository.CustomerRepository;

@Controller
@RequestMapping("/test/pgsql")
public class TestPostgreSQLController {

        @Autowired
        private CustomerRepository customerRepository;

        @RequestMapping("/save")
        public @ResponseBody String process() {
                customerRepository.save(new Customer("Neo",
"Chan"));
                customerRepository.save(new Customer("Luke",
"Liu"));
                customerRepository.save(new Customer("Ran",
"Guo"));
                customerRepository.save(new Customer("Joey",
"Chen"));
                customerRepository.save(new Customer("Larry",
```

```
"Huang"));
                return "Done";
        }

        @RequestMapping("/findall")
        public @ResponseBody String findAll() {
                String result = "<html>";

                for (Customer cust :
customerRepository.findAll()) {
                        result += "<div>" + cust.toString() +
"</div>";
                }

                return result + "</html>";
        }

        @RequestMapping("/findbyid")
        public @ResponseBody String
findById(@RequestParam("id") long id) {
                String result = "";
                result =
customerRepository.findOne(id).toString();
                return result;
        }

        @RequestMapping("/findbylastname")
        public @ResponseBody String
fetchDataByLastName(@RequestParam("lastname") String
lastName) {
                String result = "<html>";

                for (Customer cust :
customerRepository.findByLastName(lastName)) {
                        result += "<div>" + cust.toString() +
"</div>";
                }

                return result + "</html>";
        }

        @Autowired
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value = "/jdbc")
```

```
        public @ResponseBody String dailyStats(@RequestParam
Integer id) {
                String query = "SELECT id, firstname,
lastname from customer where id = " + id;

                return jdbcTemplate.queryForObject(query,
(resultSet, i) -> {

System.out.println(resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                         return (resultSet.getLong(1)+","+
resultSet.getString(2)+","+ resultSet.getString(3));
                });
        }
}
```

16.7. Test

```
                        curl
http://127.0.0.1:7000/test/pgsql/save
                        curl
http://127.0.0.1:7000/test/pgsql/findall
                        curl
http://127.0.0.1:7000/test/pgsql/findbyid?id=1
                        curl
http://127.0.0.1:7000/test/pgsql/jdbc?id=1
```

17. Spring boot with Elasticsearch

17.1. Maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/ma ... gt%3B
    <modelVersion>4.0.0</modelVersion>
    <groupId>cn.netkiller.springboot</groupId>
    <artifactId>elasticsearch</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>elasticsearch</name>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.1.RELEASE</version>
    </parent>
    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-
elasticsearch</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
```

```
    </dependencies>
</project>
```

17.2. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
@EnableMongoRepositories
@EnableJpaRepositories
@EnableScheduling
public class Application {

       public static void main(String[] args) {
               SpringApplication.run(Application.class,
args);
       }
}
```

17.3. application.properties

```
spring.data.elasticsearch.repositories.enabled=true
spring.data.elasticsearch.cluster-nodes=127.0.0.1:9300
```

17.4. Domain

```
@Document(indexName = "province", type = "city")
public class City implements Serializable {
    private static final long serialVersionUID = -1L;
    private Long id;
    private String name;
    private String description;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}
```

17.5. ElasticsearchRepository

```
public interface CityRepository extends
ElasticsearchRepository<City, Long> {
    List<City> findByNameLike(String name);
    Page<City> findByDescription(String description, Pageable
page);
    Page<City> findByDescriptionNot(String description,
Pageable page);
    Page<City> findByDescriptionLike(String description,
Pageable page);
}
```

18. Spring boot with Elasticsearch TransportClient

Spring data 目前还不支持 Elasticsearch 5.5.x 所以需要通过注入 TransportClient 这就意味着使用 5.5.x 版本你无法使用 ElasticsearchRepository 这种特性，只能通过官方的 TransportClient 操作 Elasticsearch。

18.1. Maven

Elasticsearch 依赖下来四个包

```
                <dependency>
                        <groupId>org.elasticsearch</groupId>

<artifactId>elasticsearch</artifactId>
                        <version>5.5.1</version>
                </dependency>
                <dependency>

<groupId>org.elasticsearch.client</groupId>
                        <artifactId>transport</artifactId>
                        <version>5.5.1</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-api</artifactId>
                        <version>2.8.2</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-core</artifactId>
                        <version>2.8.2</version>
                </dependency>
```

下面是我的完整例子

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.example</groupId>
        <artifactId>api</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>api</name>
        <description>Demo project for Spring
Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.3.1.RELEASE</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>
```

```
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
                        </dependency> -->
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
```

```
                <!--
https://mvnrepository.com/artifact/javax.persistence/persiste
nce-api -->
                <dependency>
                        <groupId>javax.persistence</groupId>
                        <artifactId>persistence-
api</artifactId>
                        <version>1.0.2</version>
                </dependency>
                <!--
https://mvnrepository.com/artifact/org.json/json -->
                <dependency>
                        <groupId>org.json</groupId>
                        <artifactId>json</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.elasticsearch</groupId>

<artifactId>elasticsearch</artifactId>
                        <version>5.5.1</version>
                </dependency>
                <dependency>

<groupId>org.elasticsearch.client</groupId>
                        <artifactId>transport</artifactId>
                        <version>5.5.1</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-api</artifactId>
                        <version>2.8.2</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-core</artifactId>
                        <version>2.8.2</version>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
```

```
<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>

                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

18.2. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;
```

```
@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
@EnableMongoRepositories
@EnableJpaRepositories
@EnableScheduling
public class Application {

	public static void main(String[] args) {
		SpringApplication.run(Application.class,
args);
	}
}
```

18.3. application.properties

注意：Elasticsearch 连接地址是 9300， 而不是 9200

```
spring.data.elasticsearch.cluster-nodes=172.0.0.1:9300
spring.data.elasticsearch.local=false
spring.data.elasticsearch.properties.transport.tcp.connect_ti
meout=60s
```

18.4. ElasticsearchConfiguration

```
package com.example.api.config;

import java.net.InetAddress;
import java.net.UnknownHostException;

import org.elasticsearch.client.transport.TransportClient;
import
org.elasticsearch.common.transport.InetSocketTransportAddress
```

```
;
import
org.elasticsearch.transport.client.PreBuiltTransportClient;
import org.elasticsearch.common.settings.Settings;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.DisposableBean;
import org.springframework.beans.factory.FactoryBean;
import org.springframework.beans.factory.InitializingBean;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ElasticsearchConfiguration implements
FactoryBean<TransportClient>, InitializingBean,
DisposableBean {
        private static final Logger logger =
LoggerFactory.getLogger(ElasticsearchConfiguration.class);

        @Value("${spring.data.elasticsearch.cluster-nodes}")
        private String clusterNodes;

        private TransportClient transportClient;
        private PreBuiltTransportClient
preBuiltTransportClient;

        @Override
        public void destroy() throws Exception {
                try {
                        logger.info("Closing elasticSearch
client");
                        if (transportClient != null) {
                                transportClient.close();
                        }
                } catch (final Exception e) {
                        logger.error("Error closing
ElasticSearch client: ", e);
                }
        }

        @Override
        public TransportClient getObject() throws Exception {
                return transportClient;
        }
```

```
        @Override
        public Class<TransportClient> getObjectType() {
                return TransportClient.class;
        }

        @Override
        public boolean isSingleton() {
                return false;
        }

        @Override
        public void afterPropertiesSet() throws Exception {
                buildClient();
        }

        protected void buildClient() {
                try {
                        preBuiltTransportClient = new
PreBuiltTransportClient(settings());

                        String InetSocket[] =
clusterNodes.split(":");
                        String address = InetSocket[0];
                        Integer port =
Integer.valueOf(InetSocket[1]);
                        transportClient =
preBuiltTransportClient.addTransportAddress(new
InetSocketTransportAddress(InetAddress.getByName(address),
port));

                } catch (UnknownHostException e) {
                        logger.error(e.getMessage());
                }
        }

        /**
         * 初始化默认的client
         */
        private Settings settings() {
//              Settings settings =
Settings.builder().put("cluster.name",
clusterName).put("client.transport.sniff", true).build();
                Settings settings =
Settings.builder().put("cluster.name",
"elasticsearch").build();
```

```
                return settings;
        }
}
```

18.5. RestController

```
package com.example.api.restful;

import org.elasticsearch.action.get.GetResponse;
import org.elasticsearch.client.transport.TransportClient;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/restful/search")
public class SearchRestController {

        @Autowired
        private TransportClient client;

        @RequestMapping(value = "/client/{articleId}")
        public GetResponse test(@PathVariable String
articleId) {
                GetResponse response =
client.prepareGet("information", "article", articleId).get();
                return response;
        }
}
```

使用 Curl 测试 restful 接口

```
MacBook-Pro:~ neo$ curl -k
https://test:test@localhost:8443/restful/search/client/1093.jso
n
{"fields":{},"id":"1093","type":"article","source":
{"@timestamp":"2017-07-
31T05:41:00.248Z","author":"test","@version":"1","description":
"test","ctime":"2017-07-
31T05:40:35.000Z","id":1093,"source":"test","title":"test11111"
,"content":"<p>test</p>
<p>aaaaaaaaaaaaaa</p>"},"version":3,"index":"information","sour
ceAsBytes":"eyJAdGltZXN0YW1wIjoiMjAxNy0wNy0zMVQwNTo0MTowMC4yNDh
aIiwiYXV0aG9yIjoidGVzdCIsIkB2ZXJzaW9uIjoiMSIsImRlc2NyaXB0aW9uIj
oidGVzdCIsImN0aW1lIjoiMjAxNy0wNy0zMVQwNTo0MDozNS4wMDBaIiwiaWQiO
jEwOTMsInNvdXJjZSI6InRlc3QiLCJ0aXRsZSI6InRlc3QxMTExMSIsImNvbnRl
bnQiOiI8cD50ZXN0PC9wPjxwPmFhYWFhYWFhYWFhYWFhPC9wPiJ9","sourceIn
ternal":{"childResources":[]},"sourceAsString":"
{\"@timestamp\":\"2017-07-
31T05:41:00.248Z\",\"author\":\"test\",\"@version\":\"1\",\"des
cription\":\"test\",\"ctime\":\"2017-07-
31T05:40:35.000Z\",\"id\":1093,\"source\":\"test\",\"title\":\"
test11111\",\"content\":\"<p>test</p>
<p>aaaaaaaaaaaaaa</p>\"}","sourceEmpty":false,"sourceAsMap":
{"@timestamp":"2017-07-
31T05:41:00.248Z","author":"test","@version":"1","description":
"test","ctime":"2017-07-
31T05:40:35.000Z","id":1093,"source":"test","title":"test11111"
,"content":"<p>test</p>
<p>aaaaaaaaaaaaaa</p>"},"exists":true,"sourceAsBytesRef":
{"childResources":[]},"fragment":false}
```

19. Spring boot with Apache Hive

19.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
hadoop</artifactId>
                        <version>2.5.0.RELEASE</version>
                </dependency>
                <!--
https://mvnrepository.com/artifact/org.apache.hive/hive-jdbc
-->
                <dependency>
                        <groupId>org.apache.hive</groupId>
                        <artifactId>hive-jdbc</artifactId>
                        <version>2.3.0</version>
                        <exclusions>
                                <exclusion>

<groupId>org.eclipse.jetty.aggregate</groupId>
                                        <artifactId>*
</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>org.apache.tomcat</groupId>
                        <artifactId>tomcat-jdbc</artifactId>
                        <version>8.5.20</version>
                </dependency>
```

19.2. application.properties

hive 数据源配置项

```
hive.url=jdbc:hive2://172.16.0.10:10000/default
hive.driver-class-name=org.apache.hive.jdbc.HiveDriver
hive.username=hadoop
hive.password=
```

用户名是需要具有 hdfs 写入权限，密码可以不用写

如果使用 yaml 格式 application.yml 配置如下

```
hive:
  url: jdbc:hive2://172.16.0.10:10000/default
  driver-class-name: org.apache.hive.jdbc.HiveDriver
  type: com.alibaba.druid.pool.DruidDataSource
  username: hive
  password: hive
```

19.3. Configuration

```
package cn.netkiller.config;

import org.apache.tomcat.jdbc.pool.DataSource;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.env.Environment;
import org.springframework.jdbc.core.JdbcTemplate;

@Configuration
public class HiveConfig {
        private static final Logger logger =
LoggerFactory.getLogger(HiveConfig.class);

        @Autowired
        private Environment env;

        @Bean(name = "hiveJdbcDataSource")
        @Qualifier("hiveJdbcDataSource")
        public DataSource dataSource() {
                DataSource dataSource = new DataSource();

dataSource.setUrl(env.getProperty("hive.url"));

dataSource.setDriverClassName(env.getProperty("hive.driver-
class-name"));

dataSource.setUsername(env.getProperty("hive.username"));

dataSource.setPassword(env.getProperty("hive.password"));
                logger.debug("Hive DataSource");
                return dataSource;
        }

        @Bean(name = "hiveJdbcTemplate")
        public JdbcTemplate
hiveJdbcTemplate(@Qualifier("hiveJdbcDataSource") DataSource
dataSource) {
                return new JdbcTemplate(dataSource);
        }
}
```

你也可以使用 DruidDataSource

```
package cn.netkiller.api.config;

@Configuration
public class HiveDataSource {

    @Autowired
    private Environment env;

    @Bean(name = "hiveJdbcDataSource")
    @Qualifier("hiveJdbcDataSource")
    public DataSource dataSource() {
        DruidDataSource dataSource = new DruidDataSource();
        dataSource.setUrl(env.getProperty("hive.url"));

dataSource.setDriverClassName(env.getProperty("hive.driver-
class-name"));

dataSource.setUsername(env.getProperty("hive.username"));

dataSource.setPassword(env.getProperty("hive.password"));
        return dataSource;
    }

    @Bean(name = "hiveJdbcTemplate")
    public JdbcTemplate
hiveJdbcTemplate(@Qualifier("hiveJdbcDataSource") DataSource
dataSource) {
        return new JdbcTemplate(dataSource);
    }

}
```

19.4. CURD 操作实例

Hive 数据库的增删插改操作与其他数据库没有什么不同。

```
package cn.netkiller.web;

import java.util.Iterator;
import java.util.List;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/hive")
public class HiveController {
        private static final Logger logger =
LoggerFactory.getLogger(HiveController.class);

        @Autowired
        @Qualifier("hiveJdbcTemplate")
        private JdbcTemplate hiveJdbcTemplate;

        @RequestMapping("/create")
        public ModelAndView create() {

                StringBuffer sql = new StringBuffer("create
table IF NOT EXISTS ");
                sql.append("HIVE_TEST");
                sql.append("(KEY INT, VALUE STRING)");
                sql.append("PARTITIONED BY (CTIME DATE)"); //
分区存储
                sql.append("ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' LINES TERMINATED BY '\n' "); // 定义分隔符
                sql.append("STORED AS TEXTFILE"); // 作为文本存
```

储

```
                logger.info(sql.toString());
                hiveJdbcTemplate.execute(sql.toString());

                return new ModelAndView("index");

        }

        @RequestMapping("/insert")
        public String insert() {
                hiveJdbcTemplate.execute("insert into
hive_test(key, value) values('Neo','Chen')");
                return "Done";
        }

        @RequestMapping("/select")
        public String select() {
                String sql = "select * from HIVE_TEST";
                List<Map<String, Object>> rows =
hiveJdbcTemplate.queryForList(sql);
                Iterator<Map<String, Object>> it =
rows.iterator();
                while (it.hasNext()) {
                        Map<String, Object> row = it.next();

System.out.println(String.format("%s\t%s", row.get("key"),
row.get("value")));
                }
                return "Done";
        }

        @RequestMapping("/delete")
        public String delete() {
                StringBuffer sql = new StringBuffer("DROP
TABLE IF EXISTS ");
                sql.append("HIVE_TEST");
                logger.info(sql.toString());
                hiveJdbcTemplate.execute(sql.toString());
                return "Done";
        }
}
```

20. Spring boot with Phoenix

20.1. Maven

```
<properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
</properties>

<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-hadoop</artifactId>
        <version>2.4.0.RELEASE</version>
    </dependency>
        <!--
https://mvnrepository.com/artifact/org.apache.phoenix/phoenix
-queryserver-client -->
        <dependency>
            <groupId>org.apache.phoenix</groupId>
            <artifactId>phoenix-queryserver-
client</artifactId>
            <version>4.12.0-HBase-1.3</version>
        </dependency>
        <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid</artifactId>
        <version>1.1.0</version>
    </dependency>
</dependencies>
```

20.2. application.properties

phoenix 数据源配置项

```
phoenix:
  enable: true
  url: jdbc:phoenix:172.16.0.20
  type: com.alibaba.druid.pool.DruidDataSource
  driver-class-name: org.apache.phoenix.jdbc.PhoenixDriver
  username: //phoenix的用户名默认为空
  password: //phoenix的密码默认为空
  default-auto-commit: true
```

20.3. Configuration

```
package cn.netkiller.api.config;

@Configuration
public class PhoenixDataSource {

    @Autowired
    private Environment env;

    @Bean(name = "phoenixJdbcDataSource")
    @Qualifier("phoenixJdbcDataSource")
    public DataSource dataSource() {
        DruidDataSource dataSource = new DruidDataSource();
        dataSource.setUrl(env.getProperty("phoenix.url"));

dataSource.setDriverClassName(env.getProperty("phoenix.driver
-class-name"));

dataSource.setUsername(env.getProperty("phoenix.username"));

dataSource.setPassword(env.getProperty("phoenix.password"));

dataSource.setDefaultAutoCommit(Boolean.valueOf(env.getProper
ty("phoenix.default-auto-commit")));
```

```
        return dataSource;
    }

     @Bean(name = "phoenixJdbcTemplate")
     public JdbcTemplate
phoenixJdbcTemplate(@Qualifier("phoenixJdbcDataSource")
DataSource dataSource) {
          return new JdbcTemplate(dataSource);
     }
}
```

21. Spring boot with Datasource

数据源配置

21.1. Master / Slave 主从数据库数据源配置

application.properties

```
spring.datasource.master.driverClassName =
com.mysql.cj.jdbc.Driver
spring.datasource.master.url=jdbc:mysql://192.168.1.240:3306/te
st?useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC&useSSL=false
spring.datasource.master.username = root
spring.datasource.master.password = password

spring.datasource.slave.driverClassName =
com.mysql.cj.jdbc.Driver
spring.datasource.slave.url=jdbc:mysql://192.168.1.250:3306/tes
t?useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC&useSSL=false
spring.datasource.slave.username = root
spring.datasource.slave.password = password

spring.jpa.database-
platform=org.hibernate.dialect.MySQL5Dialect
```

配置主从数据源

```
package cn.netkiller.config;

import javax.sql.DataSource;
```

```
import
org.springframework.beans.factory.annotation.Qualifier;
import
org.springframework.boot.autoconfigure.jdbc.DataSourcePropert
ies;
import
org.springframework.boot.context.properties.ConfigurationProp
erties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.jdbc.core.JdbcTemplate;

@Configuration
public class MultiDataSourceConfig {

        @Bean
        @Primary
        @ConfigurationProperties("spring.datasource.master")
        public DataSourceProperties
masterDataSourceProperties() {
                return new DataSourceProperties();
        }

        @Bean("Master")
        @Primary
        @ConfigurationProperties("spring.datasource.master")
        public DataSource masterDataSource() {
                return
masterDataSourceProperties().initializeDataSourceBuilder().bu
ild();
        }

        @Bean("masterJdbcTemplate")
        @Primary
        public JdbcTemplate
masterJdbcTemplate(@Qualifier("Master") DataSource Master) {
                return new JdbcTemplate(Master);
        }

        @Bean
        @ConfigurationProperties("spring.datasource.slave")
        public DataSourceProperties
slaveDataSourceProperties() {
                return new DataSourceProperties();
```

```
        }

        @Bean(name = "Slave")
        @ConfigurationProperties("spring.datasource.slave")
        public DataSource slaveDataSource() {
                return
slaveDataSourceProperties().initializeDataSourceBuilder().bui
ld();
        }

        @Bean("slaveJdbcTemplate")
        public JdbcTemplate
slaveJdbcTemplate(@Qualifier("Slave") DataSource Master) {
                return new JdbcTemplate(Master);
        }

}
```

选择数据源

```
        // 默认是 Master
        @Autowired
        private JdbcTemplate jdbcTemplate;

        // 或者这样写
        @Qualifier("masterJdbcTemplate")
        @Autowired
        private JdbcTemplate masterJdbcTemplate;

        // 下面是 Slave 数据源
        @Qualifier("slaveJdbcTemplate")
        @Autowired
        private JdbcTemplate slaveJdbcTemplate;
```

21.2. 多数据源配置

```
package cn.netkiller.project.config;

import javax.sql.DataSource;

import
org.springframework.beans.factory.annotation.Qualifier;
import
org.springframework.boot.context.properties.ConfigurationProp
erties;
import org.springframework.boot.jdbc.DataSourceBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.jdbc.core.JdbcTemplate;
import
org.springframework.jdbc.datasource.DataSourceTransactionMana
ger;
import
org.springframework.transaction.PlatformTransactionManager;

@Configuration
public class DataSourceConfig {

        @Bean
        @Primary
        @ConfigurationProperties("spring.datasource")
        public DataSourceProperties
defaultDataSourceProperties() {
                return new DataSourceProperties();
        }

        @Bean
        @Primary
        @ConfigurationProperties("spring.datasource")
        public DataSource defaultDataSource() {
                return
defaultDataSourceProperties().initializeDataSourceBuilder().b
uild();
        }

        @Bean("JdbcTemplate")
        @Primary
```

```
        public JdbcTemplate
defaultJdbcTemplate(@Qualifier("defaultDataSource")
DataSource Master) {
                return new JdbcTemplate(Master);
        }

        @Bean
        // @Primary
        @ConfigurationProperties("spring.datasource.master")
        public DataSourceProperties
masterDataSourceProperties() {
                return new DataSourceProperties();
        }

        @Bean("Master")
        // @Primary
        @ConfigurationProperties("spring.datasource.master")
        public DataSource masterDataSource() {
                return
masterDataSourceProperties().initializeDataSourceBuilder().bu
ild();
        }

        @Bean("masterJdbcTemplate")
        // @Primary
        public JdbcTemplate
masterJdbcTemplate(@Qualifier("Master") DataSource Master) {
                return new JdbcTemplate(Master);
        }

        @Bean
        @ConfigurationProperties("spring.datasource.slave")
        public DataSourceProperties
slaveDataSourceProperties() {
                return new DataSourceProperties();
        }

        @Bean(name = "Slave")
        @ConfigurationProperties("spring.datasource.slave")
        public DataSource slaveDataSource() {
                return
slaveDataSourceProperties().initializeDataSourceBuilder().bui
ld();
        }
```

```
        @Bean("slaveJdbcTemplate")
        public JdbcTemplate
slaveJdbcTemplate(@Qualifier("Slave") DataSource Master) {
                return new JdbcTemplate(Master);
        }

        @Bean(name = "wwwDataSource")
        @Qualifier("wwwDataSource")
        @ConfigurationProperties(prefix =
"spring.datasource.www")
        public DataSource wwwDataSource() {
                return DataSourceBuilder.create().build();
        }

        @Bean(name = "apiDataSource")
        @Qualifier("apiDataSource")
        @ConfigurationProperties(prefix =
"spring.datasource.api")
        public DataSource apiDataSource() {
                return DataSourceBuilder.create().build();
        }

        @Bean(name = "cmsDataSource")
        @Qualifier("cmsDataSource")
        //@Primary
        @ConfigurationProperties(prefix =
"spring.datasource.cms")
        public DataSource cmsDataSource() {
                return DataSourceBuilder.create().build();
        }

        @Bean
        PlatformTransactionManager transactionManager() {
                return new
DataSourceTransactionManager(apiDataSource());
        }

        @Bean(name = "wwwJdbcTemplate")
        public JdbcTemplate
wwwJdbcTemplate(@Qualifier("wwwDataSource") DataSource
dataSource) {
                return new JdbcTemplate(dataSource);
        }

        @Bean(name = "appJdbcTemplate")
```

```
        public JdbcTemplate
appJdbcTemplate(@Qualifier("apiDataSource") DataSource
dataSource) {
                return new JdbcTemplate(dataSource);
        }

        @Bean(name = "cmsJdbcTemplate")
        public JdbcTemplate
cmsJdbcTemplate(@Qualifier("cmsDataSource") DataSource
dataSource) {
                return new JdbcTemplate(dataSource);
        }
}
```

对应 application.properties 的配置方法

```
spring.datasource.www.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.www.url=jdbc:mysql://localhost:3306/www?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC&useSSL=false
spring.datasource.www.username=www
spring.datasource.www.password=passw0rd
spring.datasource.www.max-idle=10
spring.datasource.www.max-wait=10000
spring.datasource.www.min-idle=5
spring.datasource.www.initial-size=5
spring.datasource.www.validation-query=SELECT 1
spring.datasource.www.test-on-borrow=false
spring.datasource.www.test-while-idle=true
spring.datasource.www.time-between-eviction-runs-millis=18800
spring.datasource.www.jdbc-
interceptors=ConnectionState;SlowQueryReport(threshold=0)

spring.datasource.api.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.api.url=jdbc:mysql://localhost:3306/api?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC&useSSL=false
spring.datasource.api.username=api
spring.datasource.api.password=passw0rd
spring.datasource.api.max-idle=10
```

```
spring.datasource.api.max-wait=10000
spring.datasource.api.min-idle=5
spring.datasource.api.initial-size=5
spring.datasource.api.validation-query=SELECT 1
spring.datasource.api.test-on-borrow=false
spring.datasource.api.test-while-idle=true
spring.datasource.api.time-between-eviction-runs-millis=18800
spring.datasource.api.jdbc-
interceptors=ConnectionState;SlowQueryReport(threshold=0)
```

选择数据库

```
        @Autowired
        @Qualifier("apiJdbcTemplate")
        JdbcTemplate jdbcTempalte;
```

21.3. JPA 多数据源

多个 JPA 数据配置非常简单，请参考下面的例子。注意两点，第一点是设置Repository的位置：

```
@EnableJpaRepositories(
        entityManagerFactoryRef="entityManagerFactoryPrimary",
        transactionManagerRef="transactionManagerPrimary",
        basePackages= { "cn.netkiller.repository.primary" }) //
设置Repository所在位置
```

第二点是设置 Domain 位置，与 Repository 成套出现

```
```

```
        public LocalContainerEntityManagerFactoryBean
entityManagerFactoryPrimary (EntityManagerFactoryBuilder
builder) {
        return builder
                .dataSource(primaryDataSource)

.properties(getVendorProperties(primaryDataSource))
                .packages("cn.netkiller.domain.primary") //设置
实体类所在包
                .persistenceUnit("primaryPersistenceUnit")
                .build();
    }
```

首先配置第一组数据源。

```
package cn.netkiller.project.config;

@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(

entityManagerFactoryRef="entityManagerFactoryPrimary",
        transactionManagerRef="transactionManagerPrimary",
        basePackages= { "cn.netkiller.repository.primary" })
//设置Repository所在位置
public class PrimaryConfig {

    @Autowired @Qualifier("primaryDataSource")
    private DataSource primaryDataSource;

    @Primary
    @Bean(name = "entityManagerPrimary")
    public EntityManager
entityManager(EntityManagerFactoryBuilder builder) {
        return
entityManagerFactoryPrimary(builder).getObject().createEntity
Manager();
    }

    @Primary
```

```
    @Bean(name = "entityManagerFactoryPrimary")
    public LocalContainerEntityManagerFactoryBean
entityManagerFactoryPrimary (EntityManagerFactoryBuilder
builder) {
        return builder
                .dataSource(primaryDataSource)

.properties(getVendorProperties(primaryDataSource))
                .packages("cn.netkiller.domain.primary") //设
置实体类所在包
                .persistenceUnit("primaryPersistenceUnit")
                .build();
    }

    @Autowired
    private JpaProperties jpaProperties;

    private Map<String, String>
getVendorProperties(DataSource dataSource) {
        return
jpaProperties.getHibernateProperties(dataSource);
    }

    @Primary
    @Bean(name = "transactionManagerPrimary")
    public PlatformTransactionManager
transactionManagerPrimary(EntityManagerFactoryBuilder
builder) {
        return new
JpaTransactionManager(entityManagerFactoryPrimary(builder).ge
tObject());
    }

}
```

设置第二组数据源

```
package cn.netkiller.project.config;
```

```
@Configuration
@EnableTransactionManagement
@EnableJpaRepositories(

entityManagerFactoryRef="entityManagerFactorySecondary",
        transactionManagerRef="transactionManagerSecondary",
        basePackages= { "cn.netkiller.repository.secondary"
}) //设置Repository所在位置
public class SecondaryConfig {

    @Autowired @Qualifier("secondaryDataSource")
    private DataSource secondaryDataSource;

    @Bean(name = "entityManagerSecondary")
    public EntityManager
entityManager(EntityManagerFactoryBuilder builder) {
        return
entityManagerFactorySecondary(builder).getObject().createEnti
tyManager();
    }

    @Bean(name = "entityManagerFactorySecondary")
    public LocalContainerEntityManagerFactoryBean
entityManagerFactorySecondary (EntityManagerFactoryBuilder
builder) {
        return builder
                .dataSource(secondaryDataSource)

.properties(getVendorProperties(secondaryDataSource))

.packages("cn.netkiller.repository.domain.secondary") //设置
Domain实体类所在位置
                .persistenceUnit("secondaryPersistenceUnit")
                .build();
    }

    @Autowired
    private JpaProperties jpaProperties;

    private Map<String, String>
getVendorProperties(DataSource dataSource) {
        return
jpaProperties.getHibernateProperties(dataSource);
    }
```

```
    @Bean(name = "transactionManagerSecondary")
    PlatformTransactionManager
transactionManagerSecondary(EntityManagerFactoryBuilder
builder) {
        return new
JpaTransactionManager(entityManagerFactorySecondary(builder).
getObject());
    }

}
```

22. 连接池配置

Connection and Statement Pooling

注意：下面的实例仅限 Spring boot 2.0.2.RELEASE

22.1. org.apache.tomcat.jdbc.pool.DataSource

默认连接池，可以忽略配置

```
spring.datasource.type =
org.apache.tomcat.jdbc.pool.DataSource
```

22.2. druid

pom.xml

```
                <dependency>
                        <groupId>com.alibaba</groupId>
                        <artifactId>druid</artifactId>
                        <version>1.2.6</version>
                </dependency>
```

application.properties

```
spring.jpa.database=MYSQL
```

```
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
#驱动配置信息
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
#基本连接信息
spring.datasource.username = root
spring.datasource.password = root
spring.datasource.url=jdbc:mysql://192.168.153.23:3306/mytest
?useUnicode=true&characterEncoding=utf-8

#连接池属性
spring.datasource.druid.initial-size=15
spring.datasource.druid.max-active=100
spring.datasource.druid.min-idle=15
spring.datasource.druid.max-wait=60000
spring.datasource.druid.time-between-eviction-runs-
millis=60000
spring.datasource.druid.min-evictable-idle-time-millis=300000
spring.datasource.druid.test-on-borrow=false
spring.datasource.druid.test-on-return=false
spring.datasource.druid.test-while-idle=true
spring.datasource.druid.validation-query=SELECT 1
spring.datasource.druid.validation-query-timeout=1000
spring.datasource.druid.keep-alive=true
spring.datasource.druid.remove-abandoned=true
spring.datasource.druid.remove-abandoned-timeout=180
spring.datasource.druid.log-abandoned=true
spring.datasource.druid.pool-prepared-statements=true
spring.datasource.druid.max-pool-prepared-statement-per-
connection-size=20
spring.datasource.druid.filters=stat,wall,slf4j
spring.datasource.druid.use-global-data-source-stat=true
spring.datasource.druid.preparedStatement=true
spring.datasource.druid.maxOpenPreparedStatements=100
spring.datasource.druid.connect-properties.mergeSql=true
spring.datasource.druid.connect-properties.slowSqlMillis=5000
```

加密数据库密码

创建私钥、公钥和密码

```
neo@MacBook-Pro-Neo ~ % java -cp
~/.m2/repository/com/alibaba/druid/1.2.6/druid-1.2.6.jar
com.alibaba.druid.filter.config.ConfigTools you_password
privateKey:MIIBVAIBADANBgkqhkiG9w0BAQEFAASCAT4wggE6AgEAAkEA0dcA
Ph18Jqob83AdY9SeO1mP1UOvivoDenH7nHOXk2ti5B4Q5A/6MYLnbLANHLTFACJ
e7+4ZaFK0qbzixdIqkQIDAQABAkAA9YbRjIczcootlPTkw/VOr7hmc5h0bfOGM7
SVk2+Ci8RFHtzQw9MGHvOCX3NHYA6fqmT4oer/z+GljuF4YeqZAiEA/N6Jvw1Wx
q8EC+4EpRsjCg1eYbOV2kYYBKip0lfDzVkCIQDUcBURmKSDpLPOE+jq4SBZXV3H
TJs5IfmgtTzGWZIH+QIhAOVTQOMGSttXH7ld+9JsgON96kl6330bsm6PM6vyMj3
JAiAUvgjgmfXeQLOpuHnyjR66ewpQDmPNlUqpbWjMuSwwCQIgSEXSKgW+Fd2aIj
B6TrXjUTRsJy3B7OwB3Jfdu4GSioc=
publicKey:MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBANHXAD4dfCaqG/NwHWPUn
jtZj9VDr4r6A3px+5xzl5NrYuQeEOQP+jGC52ywDRy0xQAiXu/uGWhStKm84sXS
KpECAwEAAQ==
password:BMxzWjQmHsQwzNmWPPBn94Vdz8Czi6fDIOHJcqXBGzkAiKsI5cb2NM
a/wtmZY2AEXinaivtiJvqYMwWUPVzRYg==
```

```
spring.datasource.name=druidDataSource
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
spring.datasource.url=jdbc:mysql://localhost:3306/test?
characterEncoding=utf-
8&allowMultiQueries=true&autoReconnect=true
#---------密码加密-------------------------
spring.datasource.username=netkiller
spring.datasource.password=BMxzWjQmHsQwzNmWPPBn94Vdz8Czi6fDIOHJ
cqXBGzkAiKsI5cb2NMa/wtmZY2AEXinaivtiJvqYMwWUPVzRYg==
#---------启用ConfigFilter支持-----------
spring.datasource.druid.filter.config.enabled=true
#---------设置公钥-------------------------
spring.datasource.druid.publicKey=MFwwDQYJKoZIhvcNAQEBBQADSwAwS
AJBANHXAD4dfCaqG/NwHWPUnjtZj9VDr4r6A3px+5xzl5NrYuQeEOQP+jGC52yw
DRy0xQAiXu/uGWhStKm84sXSKpECAwEAAQ==
#---------设置连接属性---------------------
spring.datasource.druid.connection-
properties=config.decrypt=true;config.decrypt.key=${spring.data
source.druid.publicKey}
```

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class ApplicationTests {

    @Test
    public void druidEncrypt() throws Exception {
        //密码明文
        String password = "123456";
        String[] keyPair = ConfigTools.genKeyPair(512);
        //私钥
        String privateKey = keyPair[0];
        //公钥
        String publicKey = keyPair[1];

        //用私钥加密后的密文
        password = ConfigTools.encrypt(privateKey, password);
        System.out.println("privateKey:" + privateKey);
        System.out.println("publicKey:" + publicKey);
        System.out.println("password:" + password);

                //用公钥解密密码
        String decryptPassword = ConfigTools.decrypt(publicKey,
password);
        System.out.println("解密后:" + decryptPassword);
    }
}
```

22.3. c3p0 - JDBC3 Connection and Statement Pooling

pom.xml

```
                <dependency>
                    <groupId>org.hibernate</groupId>
                    <artifactId>hibernate-c3p0</artifactId>
                    <version>4.3.6.Final</version>
                </dependency>
                <dependency>
```

```
                    <groupId>c3p0</groupId>
                    <artifactId>c3p0</artifactId>
                    <version>0.9.1.2</version>
                </dependency>
```

application.properties

```
spring.datasource.type=com.mchange.v2.c3p0.ComboPooledDataSou
rce

spring.datasource.initialSize=5
spring.datasource.minIdle=5
spring.datasource.maxActive=20
spring.datasource.maxWait=60000
spring.datasource.timeBetweenEvictionRunsMillis=60000
spring.datasource.minEvictableIdleTimeMillis=300000
spring.datasource.validationQuery=SELECT 1 FROM DUAL
spring.datasource.testWhileIdle=true
spring.datasource.testOnBorrow=false
spring.datasource.testOnReturn=false
spring.datasource.poolPreparedStatements=true
spring.datasource.maxPoolPreparedStatementPerConnectionSize=2
0
spring.datasource.filters=stat,wall,log4j
spring.datasource.connectionProperties=druid.stat.mergeSql=tr
ue;druid.stat.slowSqlMillis=5000
#spring.datasource.useGlobalDataSourceStat=true

spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://192.168.6.1:3306/test
spring.datasource.username=inf
spring.datasource.password=inf
```

22.4. dbcp2

```
spring.datasource.type =
org.apache.commons.dbcp2.BasicDataSource
```

22.5. bonecp

```
spring.datasource.type = com.jolbox.bonecp.BoneCPDataSource
```

22.6. HikariPool

```
# Hikari will use the above plus the following to setup
connection pooling
spring.datasource.type=com.zaxxer.hikari.HikariDataSource
#最小空闲连接，默认值10，小于0或大于maximum-pool-size，都会重置为
maximum-pool-size
spring.datasource.hikari.minimum-idle=5
#最大连接数，小于等于0会被重置为默认值10；大于零小于1会被重置为minimum-
idle的值
spring.datasource.hikari.maximum-pool-size=15
#自动提交从池中返回的连接，默认值为true
spring.datasource.hikari.auto-commit=true
#空闲连接超时时间，默认值600000（10分钟），大于等于max-lifetime且max-
lifetime>0，会被重置为0；不等于0且小于10秒，会被重置为10秒。
#只有空闲连接数大于最大连接数且空闲时间超过该值，才会被释放
spring.datasource.hikari.idle-timeout=30000
#连接池名称，默认HikariPool-1
spring.datasource.hikari.pool-name=Hikari
#连接最大存活时间.不等于0且小于30秒，会被重置为默认值30分钟.设置应该比
mysql设置的超时时间短；单位ms
spring.datasource.hikari.max-lifetime=55000
#连接超时时间:毫秒，小于250毫秒，会被重置为默认值30秒
spring.datasource.hikari.connection-timeout=30000
```

```
#连接测试查询
spring.datasource.hikari.connection-test-query=SELECT 1
```

23. Spring boot with Redis

23.1. Spring boot with Redis

maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

application.properties

```
spring.redis.database=10
spring.redis.host=localhost
spring.redis.port=6379
spring.redis.password=
spring.redis.pool.max-active=8
spring.redis.pool.max-wait=-1
spring.redis.pool.max-idle=8
spring.redis.pool.min-idle=0
spring.redis.timeout=0
```

JUnit

```
@RunWith(SpringJUnit4ClassRunner.class)
```

```
@SpringApplicationConfiguration(Application.class)
public class ApplicationTests {
        @Autowired
        private StringRedisTemplate stringRedisTemplate;
        @Test
        public void test() throws Exception {
                // 保存字符串
                stringRedisTemplate.opsForValue().set("neo",
"chen");
                Assert.assertEquals("chen",
stringRedisTemplate.opsForValue().get("neo"));
    }
}
```

Controller

stringRedisTemplate模板用于存储key,value为字符串的数据

```
        @Autowired
        private StringRedisTemplate stringRedisTemplate;

        @RequestMapping("/test")
        @ResponseBody
        public String test() {
                String message = "";

stringRedisTemplate.opsForValue().set("hello", "world");
                message =
stringRedisTemplate.opsForValue().get("hello");
                return message;
        }
```

等同于

```
        @Autowired
        private RedisTemplate<String, String> redisTemplate;
```

例 5.3. RedisTemplate

```
        @Autowired
        private RedisTemplate<String, String> redisTemplate;

        public List<Protocol> getProtocol() {
                List<Protocol> protocols = new
ArrayList<Protocol>();
                Gson gson = new Gson();
                Type type = new TypeToken<List<Protocol>>()
{}.getType();
                redisTemplate.setKeySerializer(new
StringRedisSerializer());
                redisTemplate.setValueSerializer(new
StringRedisSerializer());

                String cacheKey = String.format("%s:%s",
this.getClass().getName(),
Thread.currentThread().getStackTrace()[1].getMethodName());
                long expireTime = 5;

                if(redisTemplate.hasKey(cacheKey)){
                        String cacheValue =
redisTemplate.opsForValue().get(cacheKey);
                        System.out.println(cacheValue);
                        protocols = gson.fromJson(cacheValue,
type);
                }else{
                        Protocol protocol = new Protocol();
                        protocol.setRequest(new
Date().toString());
                        protocols.add(protocol);

                        String jsonString =
```

```
gson.toJson(protocols, type);
                System.out.println( jsonString );


redisTemplate.opsForValue().set(cacheKey, jsonString);
                        redisTemplate.expire(cacheKey,
expireTime, TimeUnit.SECONDS);
                }
                return protocols;
        }
```

23.2.

获取过期时间

```
    @Autowired
    private RedisTemplate<String, String> redisTemplate;

    Long ttl =
redisTemplate.getExpire(String.format("lock:%s", device));
```

列表操作

ListOperations

```
public class Example {

    // inject the actual template
    @Autowired
    private RedisTemplate<String, String> template;

    // inject the template as ListOperations
```

```
    // can also inject as Value, Set, ZSet, and
HashOperations
    @Resource(name="redisTemplate")
    private ListOperations<String, String> listOps;

    public void addLink(String userId, URL url) {
        listOps.leftPush(userId, url.toExternalForm());
        // or use template directly

redisTemplate.boundListOps(userId).leftPush(url.toExternalFor
m());
    }
}
```

23.3. Redis Pub/Sub

Redis配置类

```
package cn.netkiller.wallet.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.redis.connection.RedisConnectionFact
ory;

import
org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.listener.ChannelTopic;
import
org.springframework.data.redis.listener.RedisMessageListenerC
ontainer;
import
org.springframework.data.redis.listener.adapter.MessageListen
erAdapter;

import cn.netkiller.wallet.redis.RedisMessageSubscriber;
```

```
@Configuration
public class RedisConfig {

        public RedisConfig() {
        }

        @Bean
        public StringRedisTemplate
stringRedisTemplate(RedisConnectionFactory connectionFactory)
{
                StringRedisTemplate redisTemplate = new
StringRedisTemplate();

redisTemplate.setConnectionFactory(connectionFactory);
                return redisTemplate;
        }

        @Bean
        public MessageListenerAdapter messageListener() {
                return new MessageListenerAdapter(new
RedisMessageSubscriber());
        }

        @Bean
        public ChannelTopic topic() {
                return new ChannelTopic("demo");
        }

        @Bean
        public RedisMessageListenerContainer
redisContainer(RedisConnectionFactory connectionFactory,
MessageListenerAdapter messageListener) {
                RedisMessageListenerContainer container = new
RedisMessageListenerContainer();


container.setConnectionFactory(connectionFactory);

container.addMessageListener(messageListener(), topic());

container.addMessageListener(messageListener(), new
ChannelTopic("test"));
                return container;
        }
```

```
}
```

订阅和发布类

```
package cn.netkiller.wallet.redis;

import java.nio.charset.StandardCharsets;

import org.springframework.data.redis.connection.Message;
import
org.springframework.data.redis.connection.MessageListener;

public class RedisMessageSubscriber implements
MessageListener {
        public void onMessage(final Message message, final
byte[] pattern) {
                System.out.println("Topic : " + new
String(message.getChannel(), StandardCharsets.UTF_8));
                System.out.println("Message : " +
message.toString());
        }
}
```

```
package cn.netkiller.wallet.redis;

import
org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.listener.ChannelTopic;

public class RedisMessagePublisher {

        private final StringRedisTemplate redisTemplate;
```

```
        private final ChannelTopic topic;

        public RedisMessagePublisher(StringRedisTemplate
redisTemplate, ChannelTopic topic) {
                this.redisTemplate = redisTemplate;
                this.topic = topic;
        }

        public void publish(String message) {

redisTemplate.convertAndSend(topic.getTopic(), message);
        }
}
```

消息发布演示

```
        @Autowired
        private StringRedisTemplate stringRedisTemplate;

        @GetMapping("/pub/demo")
        public String pub() {

                RedisMessagePublisher publisher = new
RedisMessagePublisher(stringRedisTemplate, new
ChannelTopic("demo"));
                String message = "Message " +
UUID.randomUUID();
                publisher.publish(message);
                return message;
        }

        @GetMapping("/pub/test")
        public String pub(@RequestParam String message) {

                RedisMessagePublisher publisher = new
RedisMessagePublisher(stringRedisTemplate, new
ChannelTopic("test"));
```

```
                publisher.publish(message);
                return message;
        }
```

23.4. Spring Redis Lock

Maven 依赖

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
integration</artifactId>
        </dependency>

        <dependency>

<groupId>org.springframework.integration</groupId>
            <artifactId>spring-integration-redis</artifactId>
        </dependency>
```

配置锁

```
package cn.netkiller.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.redis.connection.RedisConnectionFact
ory;
import
org.springframework.integration.redis.util.RedisLockRegistry;
```

```
@Configuration
public class RedisLockRegistryConfiguration {
    @Bean
    public RedisLockRegistry
redisLockRegistry(RedisConnectionFactory
redisConnectionFactory) {
        return new RedisLockRegistry(redisConnectionFactory,
"netkiller-lock");
    }
}
```

```
  @Bean(destroyMethod = "destroy")
  public RedisLockRegistry
redisLockRegistry(RedisConnectionFactory
redisConnectionFactory) {
    return new RedisLockRegistry(redisConnectionFactory,
"neo-lock",
        TimeUnit.MINUTES.toMillis(10));
  }
```

使用方法

```
@Autowired
private RedisLockRegistry redisLockRegistry;

                Lock lock = redisLockRegistry.obtain(device);
        if (lock.tryLock()) {
            try {
                // manipulate protected state
            } finally {
                lock.unlock();
            }
        } else {
```

```
            // perform alternative actions
        }
```

```
@Autowired
private RedisLockRegistry redisLockRegistry;

Lock lock = redisLockRegistry.obtain(key);
boolean locked = false;
try {
  locked = lock.tryLock();
  if (!locked) {
    // 没有获取到锁的逻辑
  }

  // 获取锁的逻辑
} finally {
  if (locked) {
    lock.unlock();
  }
}
```

如果没有上锁，上锁后返回 true 状态。如果已经上锁阻塞等待10秒，然后再返回锁状态

```
    public boolean isLock(String device) {

        Lock lock = redisLockRegistry.obtain(device);
        boolean status = false;
        try {
            status = lock.tryLock(10, TimeUnit.SECONDS);

        } catch (Exception e) {
            log.info(e.getMessage());
        }
```

```
        log.warn("status: {} <<<<<<<<<<", status);
        return status;
    }
```

23.5. Sprint boot with Redisson

Springboot 3.x

```
```

Springboot 2.1

注意：排除 redisson-spring-data-23，引用 redisson-spring-data-21

```
            <dependency>
            <groupId>org.redisson</groupId>
            <artifactId>redisson-spring-boot-
starter</artifactId>
            <version>3.14.0</version>
            <exclusions>
                <exclusion>
                    <groupId>org.redisson</groupId>
                    <artifactId>redisson-spring-data-
23</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.redisson</groupId>
            <artifactId>redisson-spring-data-21</artifactId>
            <version>3.14.0</version>
        </dependency>
```

24. Spring boot with RabbitMQ(AMQP)

24.1. maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

24.2. RabbitMQConfig

```
@Configuration
public class RabbitMQConfig {

    public final static String QUEUE_NAME = "spring-boot-
queue";
    public final static String EXCHANGE_NAME = "spring-boot-
exchange";
    public final static String ROUTING_KEY = "spring-boot-
key";

    // 创建队列
    @Bean
    public Queue queue() {
        return new Queue(QUEUE_NAME);
    }

    // 创建一个 topic 类型的交换器
    @Bean
    public TopicExchange exchange() {
        return new TopicExchange(EXCHANGE_NAME);
    }

    // 使用路由键（ROUTING_KEY）把队列（Queue）绑定到交换器
```

```
 (Exchange)
    @Bean
    public Binding binding(Queue queue, TopicExchange
exchange) {
        return
BindingBuilder.bind(queue).to(exchange).with(ROUTING_KEY);
    }

    @Bean
    public ConnectionFactory connectionFactory() {
        CachingConnectionFactory connectionFactory = new
CachingConnectionFactory("127.0.0.1", 5672);
        connectionFactory.setUsername("guest");
        connectionFactory.setPassword("guest");
        return connectionFactory;
    }

    @Bean
    public RabbitTemplate rabbitTemplate(ConnectionFactory
connectionFactory) {
        return new RabbitTemplate(connectionFactory);
    }

}
```

24.3. 生产者

```


@RestController
public class ProducerController {

    @Autowired
    private RabbitTemplate rabbitTemplate;

    @GetMapping("/sendMessage")
    public String sendMessage() {
        new Thread(() -> {
            for (int i = 0; i < 100; i++) {
                String value = new DateTime().toString("yyyy-
MM-dd HH:mm:ss");
```

```
                System.out.println("send message {}", value);

rabbitTemplate.convertAndSend(RabbitMQConfig.EXCHANGE_NAME,
RabbitMQConfig.ROUTING_KEY, value);
            }
        }).start();
        return "ok";
    }

}
```

24.4. 消费者

```
@Component
public class Consumer {

    @RabbitListener(queues = RabbitMQConfig.QUEUE_NAME)
    public void consumeMessage(String message) {
        System.out.println("consume message {}", message);
    }
}
```

25. Spring boot with Apache Kafka

Spring boot 1.5.1

25.1. 安装 kafka

一下安装仅仅适合开发环境，生产环境请使用这个脚本安装 https://github.com/oscm/shell/tree/master/mq/kafka

```
cd /usr/local/src
wget http://apache.communilink.net/kafka/0.10.2.0/kafka_2.12-
0.10.2.0.tgz
tar zxvf kafka_2.12-0.10.2.0.tgz
mv kafka_2.12-0.10.2.0 /srv/
cp /srv/kafka_2.12-
0.10.2.0/config/server.properties{,.original}
echo "advertised.host.name=localhost" >> /srv/kafka_2.12-
0.10.2.0/config/server.properties
ln -s /srv/kafka_2.12-0.10.2.0 /srv/kafka
```

启动 Kafka 服务

```
/srv/kafka/bin/zookeeper-server-start.sh
config/zookeeper.properties
/srv/kafka/bin/kafka-server-start.sh
/srv/kafka/config/server.properties
```

-daemon 表示守护进程方式在后台启动

```
```

```
/srv/kafka/bin/zookeeper-server-start.sh -daemon
config/zookeeper.properties
/srv/kafka/bin/kafka-server-start.sh -daemon
/srv/kafka/config/server.properties
```

停止 Kafka 服务

```
/srv/kafka/bin/kafka-server-stop.sh
/srv/kafka/bin/zookeeper-server-stop.sh
```

25.2. maven

```
            <dependency>
            <groupId>org.springframework.kafka</groupId>
            <artifactId>spring-kafka</artifactId>
        </dependency>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>deploy</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>
```

```
        <name>deploy.netkiller.cn</name>
        <description>Deploy project for Spring
Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.1.RELEASE</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency> -->
                <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency> -->
                <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency> -->
                <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-redis</artifactId>
</dependency> -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.session</groupId>
                        <artifactId>spring-session-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
cache</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
websocket</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.webjars</groupId>
                        <artifactId>webjars-
locator</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.webjars</groupId>
                        <artifactId>sockjs-
client</artifactId>
                        <version>1.0.2</version>
                </dependency>
                <dependency>
                        <groupId>org.webjars</groupId>
                        <artifactId>stomp-
websocket</artifactId>
                        <version>2.3.3</version>
```

```
                </dependency>
                <dependency>
                        <groupId>org.webjars</groupId>
                        <artifactId>bootstrap</artifactId>
                        <version>3.3.7</version>
                </dependency>
                <dependency>
                        <groupId>org.webjars</groupId>
                        <artifactId>jquery</artifactId>
                        <version>3.1.0</version>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
mail</artifactId>
                </dependency>
                <dependency>

<groupId>org.apache.tomcat.embed</groupId>
                        <artifactId>tomcat-embed-
jasper</artifactId>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>jstl</artifactId>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                </dependency>
                <dependency>

<groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
```

```
                        <!-- <version>2.7</version> -->
                </dependency>
                <dependency>
                        <groupId>com.caucho</groupId>
                        <artifactId>hessian</artifactId>
                        <version>4.0.38</version>
                </dependency>

                <dependency>
            <groupId>org.springframework.kafka</groupId>
            <artifactId>spring-kafka</artifactId>
        </dependency>

                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>

        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                                <configuration>

<mainClass>cn.netkiller.Application</mainClass>
                                </configuration>
                        </plugin>
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
```

```
        <repositories>
                <repository>
                        <id>spring-snapshots</id>
                        <name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>
                        <snapshots>
                                <enabled>true</enabled>
                        </snapshots>
                </repository>
                <repository>
                        <id>spring-milestones</id>
                        <name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
        </repositories>
        <pluginRepositories>
                <pluginRepository>
                        <id>spring-snapshots</id>
                        <name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>
                        <snapshots>
                                <enabled>true</enabled>
                        </snapshots>
                </pluginRepository>
                <pluginRepository>
                        <id>spring-milestones</id>
                        <name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </pluginRepository>
        </pluginRepositories>

</project>
```

25.3. Spring boot Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
@EnableScheduling
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);

        }
}
```

25.4. EnableKafka

```
package cn.netkiller.kafka;

import org.apache.kafka.clients.consumer.ConsumerConfig;
```

```
import
org.apache.kafka.common.serialization.IntegerDeserializer;
import
org.apache.kafka.common.serialization.StringDeserializer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.annotation.EnableKafka;
import
org.springframework.kafka.config.ConcurrentKafkaListenerConta
inerFactory;
import
org.springframework.kafka.config.KafkaListenerContainerFactor
y;
import org.springframework.kafka.core.ConsumerFactory;
import
org.springframework.kafka.core.DefaultKafkaConsumerFactory;
import
org.springframework.kafka.listener.ConcurrentMessageListenerC
ontainer;

import java.util.HashMap;
import java.util.Map;

@Configuration
@EnableKafka
public class KafkaConsumerConfig {

        public KafkaConsumerConfig() {
                // TODO Auto-generated constructor stub
        }

        @Bean

KafkaListenerContainerFactory<ConcurrentMessageListenerContai
ner<String, String>> kafkaListenerContainerFactory() {

ConcurrentKafkaListenerContainerFactory<String, String>
factory = new ConcurrentKafkaListenerContainerFactory<String,
String>();

factory.setConsumerFactory(consumerFactory());
                // factory.setConcurrency(1);
                //
factory.getContainerProperties().setPollTimeout(3000);
                return factory;
```

```
        }

        @Bean
        public ConsumerFactory<String, String>
consumerFactory() {
                return new
DefaultKafkaConsumerFactory<String, String>
(consumerConfigs());
        }

        @Bean
        public Map<String, Object> consumerConfigs() {
                Map<String, Object> propsMap = new
HashMap<String, Object>();

propsMap.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"www.netkiller.cn:9092");
                propsMap.put(ConsumerConfig.GROUP_ID_CONFIG,
"test-consumer-group");

propsMap.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
"latest");

propsMap.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, true);

propsMap.put(ConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG,
"100");

propsMap.put(ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG,
"15000");

propsMap.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
IntegerDeserializer.class);

propsMap.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class);
                return propsMap;
        }

        @Bean
        public Listener listener() {
                return new Listener();
        }

}
```

25.5. KafkaListener

```
package cn.netkiller.kafka;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.springframework.kafka.annotation.KafkaListener;
import java.util.concurrent.CountDownLatch;
import java.util.logging.Logger;

public class Listener {

        public Listener() {
                // TODO Auto-generated constructor stub
        }

        protected Logger logger =
Logger.getLogger(Listener.class.getName());

        public CountDownLatch getCountDownLatch1() {
                return countDownLatch1;
        }

        private CountDownLatch countDownLatch1 = new
CountDownLatch(1);

        @KafkaListener(topics = "test")
        public void listen(ConsumerRecord<?, ?> record) {
                logger.info("Received message: " +
record.toString());
                System.out.println("Received message: " +
record);
                countDownLatch1.countDown();
        }
}
```

25.6. 测试

```
$ cd /srv/kafka
$ bin/kafka-console-producer.sh --broker-list
47.89.35.55:9092 --topic test
This is test message.
```

每输入一行回车后发送到你的Spring boot kafka 程序

25.7. 完整的发布订阅实例

上面的例子仅仅是做了一个热身，现在我们将实现 一个完整的例子。

Consumer

例 5.4. Spring boot with Apache kafka.

SpringApplication

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
//import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
//import
```

```
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
// @EnableMongoRepositories
// @EnableJpaRepositories
@EnableScheduling
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);

        }
}
```

Consumer configuration

```
package cn.netkiller.kafka.config;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.consumer.ConsumerConfig;
import
org.apache.kafka.common.serialization.IntegerDeserializer;
import
org.apache.kafka.common.serialization.StringDeserializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.annotation.EnableKafka;
import
org.springframework.kafka.config.ConcurrentKafkaListenerConta
inerFactory;
```

```
import org.springframework.kafka.core.ConsumerFactory;
import
org.springframework.kafka.core.DefaultKafkaConsumerFactory;

import cn.netkiller.kafka.consumer.Consumer;

@Configuration
@EnableKafka
public class ConsumerConfiguration {

        public ConsumerConfiguration() {
                // TODO Auto-generated constructor stub
        }

        @Bean
        public Map<String, Object> consumerConfigs() {
                HashMap<String, Object> props = new HashMap<>
();
                // list of host:port pairs used for
establishing the initial connections
                // to the Kakfa cluster

props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"www.netkiller.cn:9092");

props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
IntegerDeserializer.class);

props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class);
                // consumer groups allow a pool of processes
to divide the work of
                // consuming and processing records
                props.put(ConsumerConfig.GROUP_ID_CONFIG,
"helloworld");

                return props;
        }

        @Bean
        public ConsumerFactory<String, String>
consumerFactory() {
                return new
DefaultKafkaConsumerFactory<String, String>
(consumerConfigs());
```

```
        }

        @Bean
        public
ConcurrentKafkaListenerContainerFactory<String, String>
kafkaListenerContainerFactory() {

ConcurrentKafkaListenerContainerFactory<String, String>
factory = new ConcurrentKafkaListenerContainerFactory<String,
String>();

factory.setConsumerFactory(consumerFactory());
                return factory;
        }

        @Bean
        public Consumer receiver() {
                return new Consumer();
        }
}
```

Consumer

```
package cn.netkiller.kafka.consumer;
import java.util.concurrent.CountDownLatch;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.kafka.annotation.KafkaListener;

public class Consumer {

        public Consumer() {
                // TODO Auto-generated constructor stub
        }
        private static final Logger logger = LoggerFactory
            .getLogger(Consumer.class);
```

```
    private CountDownLatch latch = new CountDownLatch(1);

    @KafkaListener(topics = "helloworld.t")
    public void receiveMessage(String message) {
        logger.info("received message='{}'", message);
        latch.countDown();
    }

    public CountDownLatch getLatch() {
        return latch;
    }
}
```

Producer

例 5.5. Spring boot with Apache kafka.

Producer configuration

```
package cn.netkiller.kafka.config;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.producer.ProducerConfig;
import
org.apache.kafka.common.serialization.IntegerSerializer;
import
org.apache.kafka.common.serialization.StringSerializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;
```

```
import cn.netkiller.kafka.producer.Producer;

@Configuration
public class ProducerConfiguration {

        public ProducerConfiguration() {
                // TODO Auto-generated constructor stub
        }

        @Bean
        public Map<String, Object> producerConfigs() {
                HashMap<String, Object> props = new HashMap<>
();
                // list of host:port pairs used for
establishing the initial connections
                // to the Kakfa cluster

props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"www.netkiller.cn:9092");

props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
IntegerSerializer.class);

props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class);
                // value to block, after which it will throw
a TimeoutException
                props.put(ProducerConfig.MAX_BLOCK_MS_CONFIG,
5000);

                return props;
        }

        @Bean
        public ProducerFactory<String, String>
producerFactory() {
                return new
DefaultKafkaProducerFactory<String, String>
(producerConfigs());
        }

        @Bean
        public KafkaTemplate<String, String> kafkaTemplate()
{
                return new KafkaTemplate<String, String>
```

```
(producerFactory());
        }

        @Bean
        public Producer sender() {
                return new Producer();
        }
}
```

Producer

```
package cn.netkiller.kafka.producer;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.support.SendResult;
import org.springframework.util.concurrent.ListenableFuture;
import
org.springframework.util.concurrent.ListenableFutureCallback;

public class Producer {

        private static final Logger logger =
LoggerFactory.getLogger(Producer.class);

        /*
         * public Sender() { // TODO Auto-generated
constructor stub }
         */

        @Autowired
        private KafkaTemplate<String, String> kafkaTemplate;

        public void sendMessage(String topic, String message)
{
```

```
                // the KafkaTemplate provides asynchronous
send methods returning a
                // Future
                ListenableFuture<SendResult<String, String>>
future = kafkaTemplate.send(topic, message);

                // you can register a callback with the
listener to receive the result
                // of the send asynchronously
                future.addCallback(new
ListenableFutureCallback<SendResult<String, String>>() {

                        @Override
                        public void
onSuccess(SendResult<String, String> result) {
                                logger.info("sent
message='{}' with offset={}", message,
result.getRecordMetadata().offset());
                        }

                        @Override
                        public void onFailure(Throwable ex) {
                                logger.error("unable to send
message='{}'", message, ex);
                        }
                });

                // alternatively, to block the sending
thread, to await the result,
                // invoke the future's get() method
        }
}
```

Controller

```
package cn.netkiller.web;

import java.util.concurrent.TimeUnit;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import cn.netkiller.kafka.consumer.Consumer;
import cn.netkiller.kafka.producer.Producer;

@Controller
@RequestMapping("/test")
public class KafkaTestController {

        private static final Logger logger =
LoggerFactory.getLogger(IndexController.class);

        public KafkaTestController() {
                // TODO Auto-generated constructor stub
        }

        @Autowired
        private Producer sender;

        @Autowired
        private Consumer receiver;

        @RequestMapping("/ping")
        @ResponseBody
        public String ping() {
                String message = "PONG";
                return message;
        }

        @RequestMapping("/kafka/send")
        @ResponseBody
        public String testReceiver() throws Exception {
                sender.sendMessage("helloworld.t", "Hello
Spring Kafka!");

                receiver.getLatch().await(10000,
TimeUnit.MILLISECONDS);
                logger.info(receiver.getLatch().getCount() +
```

```
"");
                return "OK";
        }

}
```

Test

例 5.6. Test Spring Kafka

SpringBootTest

```
package cn.netkiller;
import static org.assertj.core.api.Assertions.assertThat;

import java.util.concurrent.TimeUnit;

import org.junit.Test;
import org.junit.runner.RunWith;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

import cn.netkiller.kafka.consumer.Consumer;
import cn.netkiller.kafka.producer.Producer;


@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringKafkaApplicationTests {

        public SpringKafkaApplicationTests() {
                // TODO Auto-generated constructor stub
        }
        @Autowired
    private Producer sender;
```

```
    @Autowired
    private Consumer receiver;

    @Test
    public void testReceiver() throws Exception {
        sender.sendMessage("helloworld.t", "Hello Spring
Kafka!");

        receiver.getLatch().await(10000,
TimeUnit.MILLISECONDS);

assertThat(receiver.getLatch().getCount()).isEqualTo(0);
    }
}
```

```
```

25.8. Spring cloud with Kafka

Application 主文件

```


package schedule;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.domain.EntityScan;
import
org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableScheduling
```

```
@EnableEurekaClient
@EntityScan("common.domain")
public class Application {

        public static void main(String[] args) {
                System.out.println("Service Schedule
Starting...");
                SpringApplication.run(Application.class,
args);
        }
}
```

资源配置文件

application.properties

只需要两行，其余所有配置均放在配置中心。

```
# ==============================
spring.application.name=schedule
eureka.client.serviceUrl.defaultZone=http://eureka:s3cr3t@172
.16.0.10:8761/eureka/
# ==============================
```

bootstrap.properties

配置中心服务器相关配置

```
#spring.application.name=schedule
spring.cloud.config.profile=development
spring.cloud.config.label=master
spring.cloud.config.uri=http://172.16.0.10:8888
```

```
management.security.enabled=false
spring.cloud.config.username=cfg
spring.cloud.config.password=s3cr3t
```

Git 仓库

在 git 仓库中加入 spring.kafka.bootstrap_servers 配置项

```
spring.kafka.bootstrap_servers=172.16.0.10:9092
```

启用 kafka

使用 @EnableKafka 启用 Kafka 不需要其他@Bean等。这个配置文件可以省略，可以将 @EnableKafka 放到 Application.java 中。我还是喜欢独立配置。

```
package schedule.config;
@Configuration
@EnableKafka
public class KafkaConfiguration {
}
```

消息发布主程序

```
package schedule.task;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.support.SendResult;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;
import org.springframework.util.concurrent.ListenableFuture;
import
org.springframework.util.concurrent.ListenableFutureCallback;
import org.springframework.web.client.RestTemplate;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;

import schedule.repository.CmsTrashRepository;
import schedule.repository.ArticleRepository;
import common.domain.Article;
import common.domain.CmsTrash;
import common.pojo.ResponseRestful;

@Component
public class CFPushTasks {
        private static final Logger logger =
LoggerFactory.getLogger(CFPushTasks.class);

        private static final String TOPIC = "test";
        private static final SimpleDateFormat
simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
        private static final ObjectMapper mapper = new
ObjectMapper();

        @Autowired
        private ArticleRepository articleRepository;
```

```
        @Autowired
        private CmsTrashRepository cmsTrashRepository;

        @Autowired
        private KafkaTemplate<String, String> kafkaTemplate;

        @Autowired
        private RedisTemplate<String, String> redisTemplate;

        @Value("${cf.cms.site_id}")
        private int siteId;

        public CFPushTasks() {
        }

        private Date getDate() {

                Calendar calendar = Calendar.getInstance();
                calendar.add(Calendar.MINUTE, -1);
                Date date = calendar.getTime();
                return date;
        }

        private boolean setPostionDate(String key, Date
value) {
                String cacheKey =
String.format("schedule:CFPushTasks:%s", key);
                String date = simpleDateFormat.format(value);
                logger.info("setPostion({},{})", cacheKey,
date);
                redisTemplate.opsForValue().set(cacheKey,
date);

                if (value == this.getPostionDate(cacheKey)) {
                        return true;
                }
                return false;
        }

        private Date getPostionDate(String key) {
                String cacheKey =
String.format("schedule:CFPushTasks:%s", key);
                Date date = null;
                if (redisTemplate.hasKey(cacheKey)) {
```

```
                        try {
                                date =
simpleDateFormat.parse(redisTemplate.opsForValue().get(cacheK
ey));
                        } catch (ParseException e) {
                                // TODO Auto-generated catch
block
                                // e.printStackTrace();
                                logger.warn(e.getMessage());
                        }
                }
                logger.debug("getPostion({}) => {}",
cacheKey, date);
                return date;
        }

        private boolean setPostionId(String key, int id) {
                String cacheKey =
String.format("schedule:CFPushTasks:PostionId:%s", key);
                logger.info("setPostionId({},{})", cacheKey,
id);
                redisTemplate.opsForValue().set(cacheKey,
String.valueOf(id));
                if (id == this.getPostionId(cacheKey)) {
                        return true;
                }
                return false;
        }

        private int getPostionId(String key) {
                String cacheKey =
String.format("schedule:CFPushTasks:PostionId:%s", key);
                int id = 0;
                if (redisTemplate.hasKey(cacheKey)) {
                        id =
Integer.valueOf(redisTemplate.opsForValue().get(cacheKey));
                }
                logger.debug("getPostion({}) => {}",
cacheKey, id);
                return id;
        }

        @Scheduled(fixedRate = 1000 * 50)
        public void insert() {
                Iterable<Article> articles = null;
```

```
                int id = this.getPostionId("insert");

                if (id == 0) {
                        articles =
articleRepository.findBySiteId(this.siteId);

                } else {
                        articles =
articleRepository.findBySiteIdAndIdGreaterThan(this.siteId,
id);
                }
                if (articles != null) {
                        for (Article article : articles) {
                                ResponseRestful
responseRestful = new ResponseRestful(true,
this.getPostionId("insert"), "INSERT", article);
                                String jsonString;
                                try {
                                        jsonString =
mapper.writeValueAsString(responseRestful);
                                        this.send(TOPIC,
jsonString);
                                        if
(!this.setPostionId("insert", article.getId())) {
                                                return;
                                        }

                                } catch
(JsonProcessingException e) {
                                        // TODO Auto-
generated catch block
                                        e.printStackTrace();
                                }
                        }
                }

        }

        @Scheduled(fixedRate = 1000 * 50)
        public void update() {
                String message = "Hello";
                this.send(TOPIC, message);

        }
```

```
        @Scheduled(fixedRate = 1000 * 50)
        public void delete() {
                Date date = this.getPostionDate("delete");
                Iterable<CmsTrash> cmsTrashs;
                if (date == null) {
                        cmsTrashs =
cmsTrashRepository.findBySiteIdAndTypeOrderByCtime(this.siteI
d, "delete");
                } else {
                        cmsTrashs =
cmsTrashRepository.findBySiteIdAndTypeAndCtimeGreaterThanOrde
rByCtime(this.siteId, "delete", date);
                }
                if (cmsTrashs != null) {

                        for (CmsTrash cmsTrash : cmsTrashs) {
                                ResponseRestful
responseRestful = new ResponseRestful(true,
this.getPostionId("delete"), "DELETE", cmsTrash);
                                String jsonString;
                                try {
                                        jsonString =
mapper.writeValueAsString(responseRestful);
                                        this.send(TOPIC,
jsonString);

this.setPostionId("delete", cmsTrash.getId());
                                        if
(!this.setPostionDate("delete", cmsTrash.getCtime())) {
                                                return;
                                        } else {

                                        }
                                } catch
(JsonProcessingException e) {
                                        // TODO Auto-
generated catch block
                                        e.printStackTrace();
                                }

                        }
                }

        }
```

```
        private void send(String topic, String message) {

                ListenableFuture<SendResult<String, String>>
future = kafkaTemplate.send(topic, message);

                future.addCallback(new
ListenableFutureCallback<SendResult<String, String>>() {

                        @Override
                        public void
onSuccess(SendResult<String, String> result) {
                                logger.debug("sent
message='{}' with offset={}", message,
result.getRecordMetadata().offset());
                        }

                        @Override
                        public void onFailure(Throwable ex) {
                                logger.error("unable to send
message='{}'", message, ex);
                        }
                });
        }

        private void post(ResponseRestful responseRestful) {
                RestTemplate restTemplate = new
RestTemplate();
                String response =
restTemplate.postForObject("http://localhost:8440/test/cf/pos
t.json", responseRestful, String.class);

                // logger.info(article.toString());
                if (response != null) {
                        logger.info(response);
                }
        }
}
```

26. Spring boot with Scheduling

项目中经常会用到计划任务，spring Boot 中通过@EnableScheduling启用计划任务，再通过@Scheduled注解配置计划任务如果运行。

26.1. Application.java

Application.java

启用计划任务, 在Spring Boot启动类加上注解@EnableScheduling，表示该项目启用计划任务

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;
import
org.springframework.web.servlet.config.annotation.CorsRegistr
y;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;
```

```
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan({ "api.config", "api.web", "api.rest",
"api.service","api.schedule" })
@EnableMongoRepositories
@EnableJpaRepositories
@EnableScheduling
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

开启计划任务 @EnableScheduling

确保你的计划任务在 @ComponentScan 包中。

26.2. Component

在计划任务方法上加上@Scheduled注解，表示该方法是一个计划任务，项目启动后会去扫描该注解的方法并加入计划任务列表。

```
package api.schedule;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.scheduling.annotation.Scheduled;
```

```
import org.springframework.stereotype.Component;

@Component
public class ScheduledTasks {

        private static final Logger log =
LoggerFactory.getLogger(ScheduledTasks.class);
        private static final SimpleDateFormat dateFormat =
new SimpleDateFormat("HH:mm:ss");
        public final static long ONE_DAY = 24 * 60 * 60 *
1000;
        public final static long ONE_HOUR = 60 * 60 * 1000;

        public ScheduledTasks() {
        // TODO Auto-generated constructor stub
        }

        @Scheduled(fixedRate = 5000) //5秒运行一次调度任务
                public void echoCurrentTime() {
                log.info("The time is now {}",
dateFormat.format(new Date()));
        }

        @Scheduled(fixedRate = ONE_DAY)
                public void scheduledTask() {
                System.out.println("每隔一天执行一次调度任务");
        }

        @Scheduled(fixedDelay = ONE_HOUR)
                public void scheduleTask2() {
                System.out.println("运行完后隔一小时就执行任务");
        }

        @Scheduled(initialDelay = 1000, fixedRate = 5000)
        public void doSomething() {
                // something that should execute periodically
        }

        @Scheduled(cron = "0 0/1 * * * ? ")
                public void ScheduledTask3() {
                System.out.println(" 每隔一分钟执行一次任务");
        }

}
```

```

```

26.3. 查看日志

```
tail -f spring.log
```

26.4. 计划任务控制开关

matchIfMissing = true， 如果改属性条目不存在返回 true

```
@ConditionalOnProperty("batch.metrics.export.influxdb.enabled
")

# mybean.enabled = true
@ConditionalOnProperty(value='mybean.enabled')
@ConditionalOnProperty(value = "endpoints.hal.enabled",
matchIfMissing = true)

# server.host = localhost
@ConditionalOnProperty(name="server.host",
havingValue="localhost")
@ConditionalOnExpression("'${server.host}'=='localhost'")

# spring.rabbitmq.dynamic = true
@ConditionalOnProperty(prefix = "spring.rabbitmq", name =
"dynamic", matchIfMissing = true)
@ConditionalOnProperty(prefix = "extension.security.cors",
name = "enabled", matchIfMissing = false)
@ConditionalOnProperty(prefix = "camunda.bpm.job-execution",
name = "enabled", havingValue = "true", matchIfMissing =
true)

# spring.social.auto-connection-views = true
@ConditionalOnProperty(prefix = "spring.social.", value =
```

```
"auto-connection-views")
```

使用案例

```
package mis.schedule;

import java.text.SimpleDateFormat;
import java.util.Date;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.boot.autoconfigure.condition.ConditionalO
nProperty;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

@ConditionalOnProperty("mis.schedule.enabled")
@Component
public class ScheduledTasks {
        private static final Logger logger =
LoggerFactory.getLogger(ScheduledTasks.class);
        private static final SimpleDateFormat dateFormat =
new SimpleDateFormat("yyyy-mm-dd HH:mm:ss");
        public final static long ONE_DAY = 24 * 60 * 60 *
1000;
        public final static long ONE_HOUR = 60 * 60 * 1000;
        public final static long ONE_SECOND = 1000;

        public ScheduledTasks() {
                // TODO Auto-generated constructor stub
        }

        @Scheduled(fixedDelay = ONE_SECOND)
        public void scheduleTaskSplitLine() {
                logger.info("==================== {}
====================", dateFormat.format(new Date()));
        }
}
```

```

```

application.properties 配置如下

```
mis.schedule.enabled=true
```

26.5. @Scheduled 详解

@Scheduled参数说明

```
@Scheduled注解有一些参数，用于配置计划任务执行频率，执行时段等。

cron ：cron表达式，e.g. {@code "0 * * * * ? "}从前到后依次表示秒 分 时 日 月 年
zone：设置时区，指明计划任务运行在哪个时区下，默认为空，采用操作系统默认时区
fixedDelay：同一个计划任务两次执行间隔固定时间，单位毫秒，上次执行结束到下次开始执行的时间，以long类型复制
fixedDelayString：同一个计划任务两次执行间隔固定时间，单位毫秒，上次执行结束到下次开始执行的时间，以String类型赋值
fixedRate：以一个固定频率执行，单位毫秒，表示每隔多久执行一次，以long类型赋值
fixedRateString：以一个固定频率执行，单位毫秒，表示每隔多久执行一次，以String类型赋值
initialDelay：延迟启动计划任务，单位毫秒，表示执行第一次计划任务前先延迟一段时间，以long类型赋值
initialDelayString：延迟启动计划任务，单位毫秒，表示执行第一次计划任务前先延迟一段时间，以String赋值
```

cron表达式使用空格分隔的时间元素。

```

```

字段　　允许值　　允许的特殊字符
秒　　0-59　　, - * /
分　　0-59　　, - * /
小时　　0-23　　, - * /
日期　　1-31　　, - * ? / L W C
月份　　1-12 或者 JAN-DEC　　, - * /
星期　　1-7 或者 SUN-SAT　　, - * ? / L C #
年（可选）　　留空, 1970-2099　　, - * /

按顺序依次为

秒（0~59）

分钟（0~59）

小时（0~23）

天（月）（0~31，但是你需要考虑你月的天数）

月（0~11）

天（星期）（1~7 1=SUN 或 SUN, MON, TUE, WED, THU, FRI, SAT）

7.年份（1970－2099）

其中每个元素可以是一个值(如6),一个连续区间(9-12),一个间隔时间(8-18/4)(/表示每隔4小时),一个列表(1,3,5),通配符。由于"月份中的日期"和"星期中的日期"这两个元素互斥的,必须要对其中一个设置?.

0 0 10,14,16 * * ? 每天上午10点，下午2点，4点
0 0/30 9-17 * * ?　朝九晚五工作时间内每半小时
0 0 12 ? * WED 表示每个星期三中午12点
"0 0 12 * * ?" 每天中午12点触发
"0 15 10 ? * *" 每天上午10:15触发
"0 15 10 * * ?" 每天上午10:15触发
"0 15 10 * * ? *" 每天上午10:15触发
"0 15 10 * * ? 2005" 2005年的每天上午10:15触发
"0 * 14 * * ?" 在每天下午2点到下午2:59期间的每1分钟触发
"0 0/5 14 * * ?" 在每天下午2点到下午2:55期间的每5分钟触发
"0 0/5 14,18 * * ?" 在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发
"0 0-5 14 * * ?" 在每天下午2点到下午2:05期间的每1分钟触发
"0 10,44 14 ? 3 WED" 每年三月的星期三的下午2:10和2:44触发
"0 15 10 ? * MON-FRI" 周一至周五的上午10:15触发

"0 15 10 15 * ?" 每月15日上午10:15触发
"0 15 10 L * ?" 每月最后一日的上午10:15触发
"0 15 10 ? * 6L" 每月的最后一个星期五上午10:15触发
"0 15 10 ? * 6L 2002-2005" 2002年至2005年的每月的最后一个星期五上午10:15触发
"0 15 10 ? * 6#3" 每月的第三个星期五上午10:15触发

有些子表达式能包含一些范围或列表

例如：子表达式（天（星期））可以为 “MON-FRI”，“MON，WED，FRI”，“MON-WED,SAT”

“*”字符代表所有可能的值

因此，“*”在子表达式（月）里表示每个月的含义，“*”在子表达式（天（星期））表示星期的每一天

“/”字符用来指定数值的增量

例如：在子表达式（分钟）里的“0/15”表示从第0分钟开始，每15分钟

在子表达式（分钟）里的“3/20”表示从第3分钟开始，每20分钟（它和“3，23，43”）的含义一样

“？”字符仅被用于天（月）和天（星期）两个子表达式，表示不指定值

当2个子表达式其中之一被指定了值以后，为了避免冲突，需要将另一个子表达式的值设为“？”

“L” 字符仅被用于天（月）和天（星期）两个子表达式，它是单词“last”的缩写

但是它在两个子表达式里的含义是不同的。

在天（月）子表达式中，“L”表示一个月的最后一天

在天（星期）自表达式中，“L”表示一个星期的最后一天，也就是SAT

如果在“L”前有具体的内容，它就具有其他的含义了

例如：“6L”表示这个月的倒数第６天，“ＦＲＩＬ”表示这个月的最一个星期五

注意：在使用“L”参数时，不要指定列表或范围，因为这会导致问题

每3秒钟一运行一次

```
@Component
@EnableScheduling
public class MyTask {

    @Scheduled(cron="*/3 * * * * *")
    public void myTaskMethod(){
        //do something
    }
}
```

凌晨23点运行

```
        @Scheduled(cron = "0 0 23 * * ?")
        private void cleanNewToday() {
                long begin = System.currentTimeMillis();

                redisTemplate.delete("news:today");

        long end = System.currentTimeMillis();
                logger.info("Schedule clean redis {} 耗时 {}
秒", "cleanNewFlash()", (end-begin) / 1000 );
        }
```

26.6. Timer 例子

```
package cn.netkiller.schedule;

import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;

public class TimerTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                TimerTask timerTask = new TimerTask() {
                        @Override
                        public void run() {
                                System.out.println("task
run:" + new Date());
                        }

                };

                Timer timer = new Timer();

                // 每3秒执行一次
                timer.schedule(timerTask, 10, 3000);
        }

}
```

26.7. ScheduledExecutorService 例子

```
package cn.netkiller.schedule;

import java.util.Date;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
```

```
import java.util.concurrent.TimeUnit;

public class ScheduledExecutorServiceTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ScheduledExecutorService service =
Executors.newSingleThreadScheduledExecutor();

                // 参数：执行命令，初始执行的延时时间，任务执行间隔，
间隔时间单位
                service.scheduleAtFixedRate(() ->
System.out.println("ScheduledExecutorService " + new Date()),
0, 3, TimeUnit.SECONDS);

        }

}
```

27. Spring boot with Swagger

27.1. Swagger3

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-boot-starter</artifactId>
    <version>3.0.0</version>
</dependency>
```

27.2. Swagger2

Maven 文件

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>swagger2</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>swagger2</name>
        <url>http://www.netkiller.cn</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
```

```
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>

                <dependency>
                        <groupId>io.springfox</groupId>
                        <artifactId>springfox-
swagger2</artifactId>
                        <version>2.9.2</version>
                </dependency>

                <dependency>
                        <groupId>io.springfox</groupId>
                        <artifactId>springfox-swagger-
ui</artifactId>
                        <version>2.9.2</version>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
</project>
```

SpringApplication

```
package cn.netkiller.swagger2;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
```

```
public class Application {

        public static void main(String[] args) {
                System.out.println("Swagger2!");
                SpringApplication.run(Application.class,
args);
        }
}
```

EnableSwagger2

```
package cn.netkiller.swagger2;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import com.google.common.base.Predicate;

import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import
springfox.documentation.swagger2.annotations.EnableSwagger2;
import static
springfox.documentation.builders.PathSelectors.regex;
import static com.google.common.base.Predicates.or;

@Configuration
@EnableSwagger2
public class Swagger2Configuration {
        @Bean
        public Docket postsApi() {
                return new
Docket(DocumentationType.SWAGGER_2).groupName("public").apiIn
fo(apiInfo()).select().paths(postPaths()).build();
        }
```

```
        private Predicate<String> postPaths() {
                return or(regex("/api/.*"),
regex("/public/api/.*"));
        }

        private ApiInfo apiInfo() {
                return new ApiInfoBuilder().title("Open
API").description("Open API reference for
developers").termsOfServiceUrl("http://api.netkiller.cn").con
tact(new Contact("Neo Chen", "http://www.netkiller.cn",
"netkiller@msn.com")).license("Mit
License").licenseUrl("").version("1.0").build();
        }

}
```

RestController

```
package cn.netkiller.swagger2;

import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController {

        @RequestMapping(method = RequestMethod.GET, value =
"/api/hello")
        public String sayHello() {
                return "Swagger Hello World";
        }
}
```

27.3. @Api()

用于类；表示标识这个类是swagger的资源tags,value 是说明，可以使用tags替代, tags如果有多个值，生成多个list

```
@Api(value="用户控制器",tags={"用户操作接口"})
@RestController
public class UserController {

}
```

27.4. @ApiOperation()

用于方法；表示一个http请求的操作，value用于方法描述，notes用于提示内容，tags可以重新分组

```
@ApiImplicitParams() 请求参数，包含多个 @ApiImplicitParam
@ApiImplicitParam()
name–参数ming
value–参数说明
dataType–数据类型
paramType–参数类型
example–举例说明
```

```
package cn.netkiller.swagger2;

import
```

```
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;

@Api(value = "test", tags = "test")
@RestController
@RequestMapping("/api/test")
public class TestController {
        @ApiOperation(value = "接口说明", notes = "接口说明")
        @ApiImplicitParams({ @ApiImplicitParam(name = "id",
value = "唯一ID", dataType = "Integer"),
@ApiImplicitParam(name = "name", value = "名字") })
        @RequestMapping(value = "/name", method = {
RequestMethod.GET, RequestMethod.POST })
        public String test(@RequestParam(value = "id",
required = true) String id, @RequestParam(value = "name",
required = true) String name) {
                return String.format("%s:%s", id, name);
        }

        @ApiOperation(value = "getGreeting", notes="get
greeting",nickname = "getGreeting")
        @RequestMapping(method = RequestMethod.GET, value =
"/api/javainuse")
        public <Hello> sayHello() {
                ArrayList<Hello> arrayList= new ArrayList<>
();
                        arrayList.add(new Hello());
                return arrayList;
        }

}
```

27.5. @ApiResponses

```
        @ApiOperation(value = "getGreeting", nickname =
"getGreeting")
        @ApiResponses(value = {
                            @ApiResponse(code = 500, message =
"Server error"),
                             @ApiResponse(code = 404, message =
"Service not found"),
                            @ApiResponse(code = 200, message =
"Successful retrieval",
                                response = Hello.class,
responseContainer = "List") })
        @RequestMapping(method = RequestMethod.GET, value =
"/api/javainuse")
        public <Hello> sayHello() {
                            ArrayList<Hello> arrayList= new
ArrayList<>();
                            arrayList.add(new Hello());
                  return arrayList;
        }
```

27.6. @ApiModel 实体类

```
@ApiModel()用于类 ； 表示对类进行说明，用于参数用实体类接收
value–表示对象名,description–描述,都可省略

@ApiModelProperty()用于方法，字段； 表示对model属性的说明或者数据操作更
改
value 字段说明
name 重写属性名字
dataType 重写属性类型
required 是否必填
example 举例说明
hidden 隐藏
```

```
package cn.netkiller.swagger2;

import java.io.Serializable;
import java.util.List;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

@ApiModel(value = "User", description = "通用用户对象")
public class User implements Serializable {
        private static final long serialVersionUID = 1L;
        @ApiModelProperty(value = "用户名", name = "username",
example = "neo")
        private String username = "Neo";
        private String password = "passw0rd";
        private String nickname = "netkiller";
        @ApiModelProperty(value = "状态", name = "state",
example = "false", required = true)
        private boolean state = false;

        @ApiModelProperty(value = "字符串数组", hidden = true)
        private String[] array;
        private List<String> list;

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        public String getNickname() {
                return nickname;
```

```
        }

        public void setNickname(String nickname) {
                this.nickname = nickname;
        }

        public boolean isState() {
                return state;
        }

        public void setState(boolean state) {
                this.state = state;
        }

        public String[] getArray() {
                return array;
        }

        public void setArray(String[] array) {
                this.array = array;
        }

        public List<String> getList() {
                return list;
        }

        public void setList(List<String> list) {
                this.list = list;
        }

}
```

```
package cn.netkiller.swagger2;

import java.io.Serializable;
import java.util.List;

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import
```

```
org.springframework.web.bind.annotation.RestController;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;

@Api(value = "测试", tags = "test")
@RestController
@RequestMapping("/api/test")
public class TestController {
        @ApiOperation("更改用户信息")
        @PostMapping("/user/info")
        public User userInfo(@RequestBody @ApiParam(name = "用
户对象", value = "传入json格式", required = true) User user) {

                return user;
        }
}
```

28. Spring boot with knife4j

28.1. maven

```
        <dependency>
            <groupId>com.github.xiaoymin</groupId>
            <artifactId>knife4j-openapi3-jakarta-spring-boot-
starter</artifactId>
            <version>4.4.0</version>
        </dependency>
```

28.2. Knife4jConfiguration

```
package cn.netkiller.config;

import io.swagger.v3.oas.models.ExternalDocumentation;
import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Contact;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.info.License;
import org.springdoc.core.models.GroupedOpenApi;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class Knife4jConfiguration {
    @Bean
    public GroupedOpenApi adminApi() {      // 创建了一个api接口
的分组
        return GroupedOpenApi.builder()
                .group("default")          // 分组名称
                .pathsToMatch("/**")  // 接口请求路径规则
                .build();
    }
```

```
    @Bean
    public OpenAPI openAPI() {
        return new OpenAPI()
                .info(new Info()
                        .title("netkiller")
                        .description("接口描述")
                        .version("v1.0.0")
                        .contact(new
Contact().name("neo").email("netkiller@msn.com"))
                        .license(new License().name("Apache
2.0").url("https://www.netkiller.cn/docs"))
                ).externalDocs(new ExternalDocumentation()
                        .description("外部文档")

.url("https://www.netkiller.cn/docs"));

    }

}
```

28.3. application.properties

```
# springdoc-openapi项目配置
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.swagger-ui.tags-sorter=alpha
springdoc.swagger-ui.operations-sorter=alpha
springdoc.api-docs.path=/v3/api-docs
# knife4j的增强配置，不需要增强可以不配
knife4j.enable=true
knife4j.setting.language=zh_cn
```

29. Spring boot with lombok

```
<dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
</dependency>
```

常用的 lombok 注解:

```
@EqualsAndHashCode：实现equals()方法和hashCode()方法 @ToString：实现toString()方法
@Data  ：注解在类上；提供类所有属性的 getting 和 setting 方法，此外还提供了equals、canEqual、hashCode、toString 方法
@Setter：注解在属性上；为属性提供 setting 方法
@Getter：注解在属性上；为属性提供 getting 方法
@Log4j ：注解在类上；为类提供一个 属性名为log 的 log4j 日志对象
@NoArgsConstructor：注解在类上；为类提供一个无参的构造方法
@AllArgsConstructor：注解在类上；为类提供一个全参的构造方法
@Cleanup：关闭流 @Synchronized：对象同步 @SneakyThrows：抛出异常
```

29.1. @Builder

```
package cn.netkiller.graphql.domain;

import lombok.Builder;
import lombok.Data;

@Builder
@Data
```

```
public class Author {
        private Integer id;
        private String name;
        private Integer age;

        public Author() {
                // TODO Auto-generated constructor stub
        }

        @Override
        public String toString() {
                return "Author [id=" + id + ", name=" + name
+ ", age=" + age + "]";
        }

}
```

```
Author author = Author.builder().id(1).name("Neo
Chen").age(40).build();
```

29.2. @Slf4j 注解

如果不想每次都写

```
private  final Logger logger =
LoggerFactory.getLogger(CLASSNAME.class);
```

可以用注解 @Slf4j

```
```

```
package cn.netkiller.service;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import lombok.extern.slf4j.Slf4j;

@RestController
@Slf4j
public class HelloController {

//      private static final Log log =
LogFactory.getLog(HelloController.class);

        @GetMapping("/")
        public String hello() {
                Log.info("@Slf4j Test OK");
                return "Hello World";
        }

}
```

30. Spring boot with Docker

30.1. 通过 Docker 命令构建镜像

手工编译镜像

在项目根目录创建 Dockerfile 文件

```
% cat Dockerfile
FROM openjdk
VOLUME /tmp
COPY target/*.jar app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
```

编译镜像

```
mvn package
docker build -t netkiller/docker .

% docker images | grep netkiller
netkiller/docker                            latest
ed359b6ffcad        16 seconds ago      105MB

% docker run -ti --entrypoint /bin/sh netkiller/docker
sh-4.2# ls
app.jar  bin  boot  dev  etc  home  lib  lib64  media  mnt  opt
proc  root  run  sbin   srv  sys  tmp  usr  var
sh-4.2#
```

启动镜像测试

```
docker run -p 8080:8080 netkiller/docker

neo@MacBook-Pro ~ % curl http://localhost:8080
Hello Docker World
```

Dockerfile 放在 src/main/docker/Dockerfile 下

```
% cat src/main/docker/Dockerfile
FROM openjdk
VOLUME /tmp
COPY target/*.jar app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
```

```
mvn package
docker rmi netkiller/docker -f
docker build -t netkiller/docker -f src/main/docker/Dockerfile
.
docker run -p 8080:8080 netkiller/docker
```

```
neo@MacBook-Pro ~ % curl http://localhost:8080
Hello Docker World
```

通过参数指定 Springboot 文件

```
% cat src/main/docker/Dockerfile
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-
jar","/app.jar"]
```

```
mvn package
docker rmi netkiller/docker -f
docker build --build-arg JAR_FILE=target/*.jar -t
netkiller/docker -f src/main/docker/Dockerfile .
docker run -p 8080:8080 netkiller/docker
```

SPRING_PROFILES_ACTIVE 指定配置文件

```
% docker run -e "SPRING_PROFILES_ACTIVE=prod" -p 8080:8080
netkiller/docker
```

推送镜像到仓库

```
neo@MacBook-Pro ~ % docker push netkiller/docker
The push refers to repository [docker.io/netkiller/docker]
100ff47f36fe: Pushed
a7aafc769de1: Mounted from library/openjdk
2666aafcfdd9: Mounted from library/openjdk
c4a7cf6a6169: Mounted from library/openjdk
```

```
latest: digest:
sha256:3078fea95c633f007be33b829efae0ff8e9d78ad463925af7d07752c
95eb43a3 size: 1165
```

30.2. 通过 Maven 构建 Docker 镜像

Maven + Dockerfile 方案一

项目地址 https://github.com/spotify/dockerfile-maven

```
        <build>
                <plugins>
                        <plugin>
<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <configuration>
<mainClass>cn.netkiller.docker.Application</mainClass>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>dockerfile-maven-
plugin</artifactId>
                                <version>1.4.10</version>
                                <executions>
                                        <execution>
<id>default</id>
                                                <goals>
<goal>build</goal>
<goal>push</goal>
                                                </goals>
                                        </execution>
                                </executions>
```

```
                                <configuration>

<dockerfile>${project.basedir}/src/main/docker/Dockerfile</dock
erfile>

<repository>${docker.image.prefix}/${project.artifactId}
</repository>
                                        <tag>${project.version}
</tag>
                                        <buildArgs>

<JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
                                        </buildArgs>
                                        <resources>
                                                <resource>

<targetPath>/</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>

                                </configuration>

                        </plugin>
                </plugins>
        </build>
```

```
neo@MacBook-Pro ~/git/springcloud/docker % mvn dockerfile:build
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< cn.netkiller:docker >---------
----------------
[INFO] Building docker 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]-----------------
----------------
[INFO]
[INFO] --- dockerfile-maven-plugin:1.4.10:build (default-cli) @
```

```
docker ---
[INFO] dockerfile:
/Users/neo/git/springcloud/docker/src/main/docker/Dockerfile
[INFO] contextDirectory: /Users/neo/git/springcloud/docker
[INFO] Building Docker context
/Users/neo/git/springcloud/docker
[INFO] Path(dockerfile):
/Users/neo/git/springcloud/docker/src/main/docker/Dockerfile
[INFO] Path(contextDirectory):
/Users/neo/git/springcloud/docker
[INFO]
[INFO] Image will be built as netkiller/docker:0.0.1-SNAPSHOT
[INFO]
[INFO] Step 1/7 : FROM openjdk
[INFO]
[INFO] Pulling from library/openjdk
[INFO] Digest:
sha256:38ec2c78a60ec4d5773c93534e433237be154ff5afa476965a68837b
43ef2f19
[INFO] Status: Image is up to date for openjdk:latest
[INFO]  ---> b697a97ee8e1
[INFO] Step 2/7 : MAINTAINER Netkiller <netkiller@msn.com>
[INFO]
[INFO]  ---> Using cache
[INFO]  ---> e6fd68ec1ce8
[INFO] Step 3/7 : VOLUME /tmp
[INFO]
[INFO]  ---> Using cache
[INFO]  ---> 78b146e1a8a0
[INFO] Step 4/7 : ARG JAR_FILE
[INFO]
[INFO]  ---> Using cache
[INFO]  ---> 2c60b65d49dc
[INFO] Step 5/7 : COPY ${JAR_FILE} app.jar
[INFO]
[INFO]  ---> Using cache
[INFO]  ---> 3186f0425f1d
[INFO] Step 6/7 : CMD ["java", "-version"]
[INFO]
[INFO]  ---> Using cache
[INFO]  ---> d14b8d6360fe
[INFO] Step 7/7 : ENTRYPOINT ["java","-
Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
[INFO]
[INFO]  ---> Using cache
```

```
[INFO]  ---> 68e424cf5eab
[INFO] Successfully built 68e424cf5eab
[INFO] Successfully tagged netkiller/docker:0.0.1-SNAPSHOT
[INFO]
[INFO] Detected build of image with id 68e424cf5eab
[INFO] Building jar:
/Users/neo/git/springcloud/docker/target/docker-0.0.1-SNAPSHOT-
docker-info.jar
[INFO] Successfully built netkiller/docker:0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time:  9.413 s
[INFO] Finished at: 2019-04-13T05:39:07+08:00
[INFO] ------------------------------------------------------------
----------------
```

Maven + Dockerfile 方案二

```
<build>
  <plugins>
    ...
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>docker-maven-plugin</artifactId>
      <version>VERSION GOES HERE</version>
      <configuration>
        <imageName>example</imageName>
        <dockerDirectory>docker</dockerDirectory>
        <resources>
           <resource>
             <targetPath>/</targetPath>
             <directory>${project.build.directory}</directory>
             <include>${project.build.finalName}.jar</include>
           </resource>
        </resources>
```

```
      </configuration>
    </plugin>
    ...
  </plugins>
</build>
```

Maven 不使用 Dockerfile 文件

项目地址 https://github.com/spotify/docker-maven-plugin

```
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.0</version>
                                <configuration>

<imageName>${docker.image.prefix}/${project.artifactId}
</imageName>

<baseImage>openjdk</baseImage>
                                        <tag>${project.version}
</tag>

<maintainer>${docker.maintainer}</maintainer>
                                        <volumes>/tmp</volumes>
                                        <workdir>/</workdir>
                                        <cmd>["java", "-
version"]</cmd>
                                        <entryPoint>["java", "-
Djava.security.egd=file:/dev/./urandom", "-jar",
"/${project.build.finalName}.jar"]</entryPoint>
                                        <!-- copy the service's
jar file from target into the root directory of the image -->
                                        <resources>
                                                <resource>

<targetPath>/</targetPath>
```

```
<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                    </resource>
                                                </resources>
                                            </configuration>
                                        </plugin>
```

构建镜像 mvn clean package docker:build

```
neo@MacBook-Pro ~/git/springcloud/webflux % mvn docker:build
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< cn.netkiller:webflux >--------
----------------
[INFO] Building webflux 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]-----------------
----------------
[INFO]
[INFO] --- docker-maven-plugin:1.2.0:build (default-cli) @
webflux ---
[INFO] Using authentication suppliers:
[ConfigFileRegistryAuthSupplier]
[INFO] Copying
/Users/neo/git/springcloud/webflux/target/webflux-0.0.1-
SNAPSHOT.jar ->
/Users/neo/git/springcloud/webflux/target/docker/webflux-0.0.1-
SNAPSHOT.jar
[INFO] Building image netkiller/webflux
Step 1/7 : FROM openjdk

 ---> b697a97ee8e1
Step 2/7 : MAINTAINER netkiller

 ---> Using cache
 ---> c275f5dc2815
Step 3/7 : WORKDIR /

 ---> Using cache
 ---> 27815e0b4455
```

```
Step 4/7 : ADD /webflux-0.0.1-SNAPSHOT.jar //

 ---> Using cache
 ---> 78b0fe2a827d
Step 5/7 : ENTRYPOINT ["java", "-
Djava.security.egd=file:/dev/./urandom", "-jar", "/webflux-
0.0.1-SNAPSHOT.jar"]

 ---> Using cache
 ---> 66d5499c8ba3
Step 6/7 : CMD ["java", "-version"]

 ---> Using cache
 ---> 080a1468d88b
Step 7/7 : VOLUME /tmp

 ---> Using cache
 ---> 60debfac7b7c
ProgressMessage{id=null, status=null, stream=null, error=null,
progress=null, progressDetail=null}
Successfully built 60debfac7b7c
Successfully tagged netkiller/webflux:latest
[INFO] Built netkiller/webflux
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time:  4.485 s
[INFO] Finished at: 2019-04-13T05:41:41+08:00
[INFO] ------------------------------------------------------------
----------------
```

推送镜像

```
neo@MacBook-Pro ~ % vim
/usr/local/Cellar/maven/3.6.0/libexec/conf/settings.xml

  <!-- servers
```

```
   | This is a list of authentication profiles, keyed by the
server-id used within the system.
   | Authentication profiles can be used whenever maven must
make a connection to a remote server.
   |-->
  <servers>
    <!-- server
     | Specifies the authentication information to use when
connecting to a particular server, identified by
     | a unique name within the system (referred to by the 'id'
attribute below).
     |
     | NOTE: You should either specify username/password OR
privateKey/passphrase, since these pairings are
     |       used together.
     |
    <server>
      <id>deploymentRepo</id>
      <username>repouser</username>
      <password>repopwd</password>
    </server>
    -->

    <!-- Another sample, using keys to authenticate.
    <server>
      <id>siteServer</id>
      <privateKey>/path/to/private/key</privateKey>
      <passphrase>optional; leave empty if not used.
</passphrase>
    </server>
    -->
    <server>
      <id>docker-hub</id>
      <username>netkiller</username>
      <password>******</password>
      <configuration>
        <email>netkiller@msn.com</email>
      </configuration>
    </server>
  </servers>
```

****** 修改为你的密码

查看 Docker Registry 地址

```
neo@MacBook-Pro ~ % docker info | grep Registry
Registry: https://index.docker.io/v1/
```

maven docker 插件配置

```
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.0</version>
                                <configuration>

<imageName>${docker.image.prefix}/${project.artifactId}
</imageName>

<baseImage>openjdk</baseImage>
                                        <tag>${project.version}
</tag>

<maintainer>${docker.maintainer}</maintainer>
                                        <volumes>/tmp</volumes>
                                        <workdir>/srv</workdir>
                                        <cmd>["java", "-
version"]</cmd>
                                        <entryPoint>["java", "-
Djava.security.egd=file:/dev/./urandom", "-jar",
"/srv/${project.build.finalName}.jar"]</entryPoint>
                                        <!-- copy the service's
jar file from target into the root directory of the image -->
                                        <resources>
                                                <resource>

<targetPath>/</targetPath>

<directory>${project.build.directory}</directory>
```

```
<include>${project.build.finalName}.jar</include>
                                                    </resource>
                                            </resources>

<image>${docker.image.prefix}/${project.artifactId}</image>

<newName>${docker.image.prefix}/${project.artifactId}:${project
.version}</newName>
                                            <serverId>docker-
hub</serverId>

<registryUrl>https://index.docker.io/v1/</registryUrl>
                                    </configuration>
                            </plugin>
```

```
docker:build -DpushImage or docker:push
```

使用加密的密码

```
neo@MacBook-Pro ~ % mvn --encrypt-master-password
Master password:
{r7kkN/XCOXYHqwRqE30k6Bz+pNGsB7/UogGTqqo+G2A=}
```

```
vim /usr/local/Cellar/maven/3.6.0/libexec/conf/settings.xml

<servers>
  <server>
    <id>docker-hub</id>
    <username>netkiller</username>
    <password>{r7kkN/XCOXYHqwRqE30k6Bz+pNGsB7/UogGTqqo+G2A=}
</password>
  </server>
```

```
</servers>
```

```
vim ~/.m2/settings-security.xml

<settingsSecurity>
  <master>{r7kkN/XCOXYHqwRqE30k6Bz+pNGsB7/UogGTqqo+G2A=}
</master>
</settingsSecurity>
```

30.3. [ERROR] No plugin found for prefix 'dockerfile' in the current project and in the plugin groups [org.apache.maven.plugins, org.codehaus.mojo] available from the repositories [local (/Users/neo/.m2/repository), central (https://repo.maven.apache.org/maven2)] -> [Help 1]

```
在maven的conf/setting.xml中要加入:

neo@MacBook-Pro ~ % mvn -version
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3;
2018-10-25T02:41:47+08:00)
Maven home: /usr/local/Cellar/maven/3.6.0/libexec
Java version: 12, vendor: Oracle Corporation, runtime:
/Library/Java/JavaVirtualMachines/jdk-12.jdk/Contents/Home
Default locale: en_CN, platform encoding: UTF-8
OS name: "mac os x", version: "10.14.5", arch: "x86_64",
family: "mac"


vim /usr/local/Cellar/maven/3.6.0/libexec/conf/settings.xml

<pluginGroups>
    <pluginGroup>com.spotify</pluginGroup>
</pluginGroups>
```

30.4. curl: (35) LibreSSL SSL_connect: SSL_ERROR_SYSCALL in connection to localhost:8888

```
iMac:config neo$ curl -k -i -H HOST:sss
https://config:s3cr3t@localhost:8888/netkiller-dev.json
curl: (35) LibreSSL SSL_connect: SSL_ERROR_SYSCALL in
connection to localhost:8888
```

检查发现 8888 端口已经启动，SSL证书读不到

```
iMac:config neo$ openssl s_client -connect localhost:8888
CONNECTED(00000005)
140735970464712:error:140790E5:SSL routines:SSL23_WRITE:ssl
handshake
failure:/BuildRoot/Library/Caches/com.apple.xbs/Sources/libress
l/libressl-22.50.3/libressl/ssl/s23_lib.c:124:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 0 bytes and written 318 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
---
```

我开始怀疑是泛域名问题

```
keytool -genkey -alias *.netkiller.cn -storetype PKCS12 -keyalg
RSA -keysize 2048 -storepass passw0rd -keystore allhost.p12 -
dname "CN=*.netkiller.cn, OU=netkiller, O=netkiller.cn,
L=Guangdong, ST=Shenzhen, C=CN"
keytool -selfcert -alias *.netkiller.cn -storepass passw0rd -
keystore allhost.p12
```

测试后发现跟证书无关。

经过曲折的排查发现绑定了地址，在本地启动是正常的，一旦放入 Docker 容器就无法工作。

```
#server.address=localhost
server.port=8888

server.ssl.enabled=true
server.ssl.key-store-type=PKCS12
server.ssl.key-store=classpath:localhost.p12
server.ssl.key-store-password=123456
#server.ssl.key-store=classpath:allhost.p12
#server.ssl.key-store-password=passw0rd
server.http2.enabled=true

#logging.file=target/spring.log

spring.application.name=config-server
spring.profiles.active=native
spring.security.user.name=config
spring.security.user.password=s3cr3t

#spring.cloud.config.server.git.uri=/opt/config
spring.cloud.config.server.native.search-
locations=classpath:/shared
```

去掉 server.address=localhost 即可，在 build docker 镜像，然后启动容器。可以正常获取证书

```
iMac:config neo$ openssl s_client -connect localhost:8888
CONNECTED(00000005)
depth=0 C = CN, ST = Shenzhen, L = Guangdong, O = netkiller.cn,
OU = netkiller, CN = localhost
verify error:num=18:self signed certificate
verify return:1
depth=0 C = CN, ST = Shenzhen, L = Guangdong, O = netkiller.cn,
OU = netkiller, CN = localhost
verify return:1
---
Certificate chain
 0
s:/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkiller/CN=
localhost

i:/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkiller/CN=
localhost
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDijCCAnKgAwIBAgIJAP/SjXit0rVsMA0GCSqGSIb3DQEBCwUAMHMxCzAJBgN
V
BAYTAkNOMREwDwYDVQQIEwhTaGVuemhlbjESMBAGA1UEBxMJR3Vhbmdkb25nMRU
w
EwYDVQQKEwxuZXRraWxsZXIuY24xEjAQBgNVBAsTCW5ldGtpbGxlcjESMBAGA1U
E
AxMJbG9jYWxob3N0MB4XDTIwMDkwNzA4NTUzOVoXDTIwMTIwNjA4NTUzOVowczE
L
MAkGA1UEBhMCQ04xETAPBgNVBAgTCFNoZW56aGVuMRIwEAYDVQQHEwlHdWFuZ2R
v
bmcxFTATBgNVBAoTDG5ldGtpbGxlci5jbjESMBAGA1UECxMJbmV0a2lsbGVyMRI
w
EAYDVQQDEwlsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoI
B
AQCTZUtf/siYQr3MBstphQsBceRxvflDm2C4ztZ8OemDzH2avhI7edD6rzrJQ0V
2
1n1XlTRgwoYqoTgeIdQ1DbzgrCliBYy+3E9vcp8WWzYz9o2YZRphYUr37iWonP+
b
ZkLqzmRLuASRNZ8sBrwD7Mvs5IXfJZQ8wruO0V4oJQ5NOzcxDmbA0WGJn/0QZDK
```

```
N
/tR7Rw3g9B96fFYGI/T7g4nuteEiUqQ9GJ1gx3utBd31Z1m8cV59ZsWd+Y2P14L
O
W+YxkpB56OZKWWr1ExxQdZmLIME+D0d40M8At6rCAvclMKa7dva6+ZRxPlizVkQ
S
L4JNT1WOMtVaUUhFX5xlhsBtAgMBAAGjITAfMB0GA1UdDgQWBBROMJrswZ37wsx
V
sm0N9AHOE8ZiODANBgkqhkiG9w0BAQsFAAOCAQEAPIgc6ZcQueQTEym36gx2IRW
T
wLVQEabyS4/xeu89aRfbGDOavBajNwStqGdWUE8PRb95bhfvziZ61c6gBO9IE23
j
GOmIQTW5RvZL6HLJgqR3LngZUiV/Ugwuno5Uo8IN25duq993tNmdCG8YeBtfuy/
j
OFRrn96OT/Trj04NfYmC7nqBThyNmLPY5Oeo0XkhIAqqcLJE8/SJ9zd16vmgVhP
M
UlsFJcZoL1uhbNXQuLPv8id8tntH+Lli39RVwd56CgTW7k9YFfFNV0mCeWBsAYl
3
74R8l4ClV15o31wH/qPLg0F6uE/M/xsz56WIu2e5Oa30issz0DjYrG9GiQ2kDA=
=
-----END CERTIFICATE-----
subject=/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkill
er/CN=localhost
issuer=/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkille
r/CN=localhost
---
No client certificate CA names sent
---
SSL handshake has read 2631 bytes and written 512 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID:
856BA1E0CFE8AC65AEC838C0A4DA0503C7A05F0BA803B127D3B1EBBB8FF1A34
4
    Session-ID-ctx:
    Master-Key:
2DCA2747330C8008958B1A4F3EF340044FE69455EA730DA0E30DF97A13E6EB7
BCABDFDF5CA0FA5B278701EA25D694CAB
```

```
    TLS session ticket lifetime hint: 86400 (seconds)
    TLS session ticket:
    0000 - 93 c1 d2 63 4d ef 37 a9-47 d1 72 2e ee 07 5a e2
...cM.7.G.r...Z.
    0010 - b7 40 aa 89 db 70 64 88-86 ad 65 2e e9 f8 2a de
.@...pd...e...*.
    0020 - 02 03 7f d3 5d 22 c2 e1-48 5a 43 59 7d 0f ef cc
....]"..HZCY}...
    0030 - cc fa 08 f9 bd 23 70 bb-82 8b d8 29 c8 42 e8 ed
.....#p....).B..
    0040 - 12 6d ae 99 c8 74 c0 87-d9 a0 c0 27 ae 92 d9 71
.m...t.....'...q
    0050 - ab 14 da d1 c6 9f 6f ba-7b 2f 6a 39 af c3 81 09
......o.{/j9....
    0060 - bd 8a ac 55 d0 9f e4 32-d7 a6 1f 10 29 0d 07 f0
...U...2....)...
    0070 - 09 d2 54 35 a8 d5 9e 9c-e1 5b 7b dd cc de eb 2a
..T5.....[{....*
    0080 - 94 f9 56 41 df 14 85 37-b3 c1 28 be fe 1b ae 64
..VA...7..(....d
    0090 - 68 c9 b3 12 8b 78 28 d4-16 f3 28 3e 0e c3 e2 e3
h....x(...(>....
    00a0 - 0d d5 42 46 37 3a 62 11-38 d4 68 59 77 01 2f 12
..BF7:b.8.hYw./.
    00b0 - 29 b1 3f ab 3d c2 0b be-f0 df 87 43 ae 89 99 35
).?.=......C...5
    00c0 - 19 eb fc 00 38 fa cc 5e-bb 0c 81 7f ae ee 8f 0e
....8..^........
    00d0 - c5 82 00 4f bc f4 c6 a7-b0 3e 27 a8 0a 7e 57 a0
...O.....>'..~W.
    00e0 - b8 c9 4a 04 49 61 db 62-cd bc a2 3d c4 32 a0 74
..J.Ia.b...=.2.t
    00f0 - 11 0a ee c0 99 58 7a ce-99 30 7f a2 90 a0 50 30
.....Xz..0....P0
    0100 - fe df 5e 57 d5 e3 fb 6f-20 64 eb 8e ef da 95 6b
..^W...o d.....k
    0110 - 5c 20 38 62 75 5b d0 b6-4a 38 12 4b 8e be 6c 03   \
8bu[..J8.K..l.
    0120 - 14 b1 e9 05 cf b7 8c 12-e4 b6 2e 84 c3 14 57 4b
..............WK
    0130 - 56 a6 47 f6 2f 06 81 12-a5 d8 88 8e 2f dd 40 43
V.G./......./.@C
    0140 - 31 c3 0b 85 7d 26 ef b2-4d 9d aa 40 f4 e4 1c bd
1...}&..M..@....
    0150 - 03 8e 61 b6 da d0 05 49-32 7a 26 44 7c 8e 69 c5
```

```
..a....I2z&D|.i.
    0160 - 9c 41 30 e3 0f 08 8f 57-1e 70 13 ff c2 cc f2 53
.A0....W.p.....S
    0170 - 44 ed d2 9f c0 1c 5a 49-1a e3 88 94 84 15 7d c1
D.....ZI......}.
    0180 - a7 e5 fc 39 70 92 c1 6f-77 64 dc 93 aa af 81 ad
...9p..owd......
    0190 - 64 50 c6 f9 3e da 4f 62-60 21 df 78 98 ca 78 6e
dP..>.Ob`!.x..xn
    01a0 - d0 43 14 12 54 ae 4b e0-f4 4b 70 06 1e 26 6a 17
.C..T.K..Kp..&j.
    01b0 - af b2 7c 76 75 ce 4f 60-79 5d a8 4d 8f e7 22 75
..|vu.O`y].M.."u
    01c0 - 5b 65 db 42 5e b5 c0 05-9e ef f1 38 e4 e8 b0 a2
[e.B^......8....
    01d0 - 89 60 fa 43 18 e3 89 e9-4d d2 52 87 8c a3 73 16
.`.C....M.R...s.
    01e0 - f6 9b d4 0f 72 b3 22 e1-86 87 b1 85 c4 b0 b6 36
....r."........6
    01f0 - 1f 83 1f 87 76 28 20 9f-64 ca f0 1e 11 da 0b bf
....v( .d.......
    0200 - 75 df a9 77 48 84 6d a1-5e 2d 3c f7 d6 df 3e d8
u..wH.m.^-<...>.
    0210 - 6e 18 6f 53 eb c1 86 9e-cb a8 e1 19 e7 f4 5c b9
n.oS..........\.
    0220 - 58 c9 d4 38 b1 4a 3b ff-a0 16 34 2f 69 67 28 b4
X..8.J;...4/ig(.
    0230 - e9 72 f8 97 75 6d a0 15-5c 16 cf 28 33 2f c1 37
.r..um..\..(3/.7
    0240 - ca 09 07 2b 5f 5f e7 6b-94 19 9c 95 5c 2c d1 54
...+__.k....\,.T
    0250 - 69 3f cd d5 63 9f 75 6c-26 53 cd 57 3a 9b 7b 02
i?..c.ul&S.W:.{.
    0260 - 6e 79 5c e5 36 9d 90 1a-d2 8a 0b b2 6f 03 5a fd
ny\.6.......o.Z.
    0270 - b0 3b d1 b8 68 be 1f 99-05 e2 52 a5 96 99 bd bf
.;..h.....R.....
    0280 - bd 84 06 b9 ed fb bb 2e-fd 9b 14 1b ca 7c 07 eb
.............|..
    0290 - a6 ff 07 ce d3 6b 48 26-b2 f0 67 c2 96 6d 4b 00
.....kH&..g..mK.
    02a0 - 77 d3 59 e0 fc 48 19 29-23 1a 9a 30 b6 3f 2a 12
w.Y..H.)#..0.?*.
    02b0 - 80 b4 f7 5e 33 85 42 da-c2 b9 42 dd 30 73 f1 15
...^3.B...B.0s..
```

```
    02c0 - f2 16 49 f7 24 39 77 61-e4 90 7c 32 f1 e9 0e fb
..I.$9wa..|2....
    02d0 - 7b a7 02 db 91 3a 16 8c-85 d2 2a 38 ad 3c a8 a9
{....:....*8.<..
    02e0 - 0b a8 3f 5b 49 92 de 45-41 74 60 dd 41 66 8f ac
..?[I..EAt`.Af..
    02f0 - d2 23 60 25 99 6f 73 8b-8c f1 88 6c 67 36 b7 e0
.#`%.os....lg6..
    0300 - 60 d1 2a 77 b4 3e 29 bb-90 dc 7f f2 30 2e e7 de
`.*w.>).....0...
    0310 - dd 48 f6 dc 59 30 89 fe-1f 90 ac a6 10 42 96 ab
.H..Y0.......B..
    0320 - a7 84 34 2c 2e 54 d1 1b-65 48 a9 47 63 3f ff 2a
..4,.T..eH.Gc?.*
    0330 - a1 66 b7 6d d6 f7 d3 11-d3 6a 21 33 a4 99 5c a4
.f.m.....j!3..\.
    0340 - e3 a1 b8 5a 1b 7a d9 45-89 fa 12 ee 5f 5b 69 6e
...Z.z.E...._[in
    0350 - 7b 77 ba c9 3a 3c 09 b0-db 16 ad ac 66 6e 36 5a
{w..:<......fn6Z
    0360 - 48 c9 9a e7 6c a7 2f 10-31 33 9c 3f e1 18 9c af
H...l./.13.?....
    0370 - dc a1 f9 26 50 2a 66 e8-62 da fb 51 ad dc d6 72
...&P*f.b..Q...r
    0380 - ca 53 4c 7b 72 e6 2b ee-f9 fd 97 f3 c4 67 dc c6
.SL{r.+......g..
    0390 - f1 38 d1 58 d5 df 02 a5-1c f0 3d 5b 6d 01 be ff
.8.X......=[m...
    03a0 - a7 d1 0b 68 04 22 2b ab-ee a6 0a c3 98 80 04 bf
...h."+.........
    03b0 - 99 8b 9b 67 6e d3 fc 25-ab 87 01 74 8c 29 c8 8b
...gn..%...t.)..
    03c0 - 10 f0 b5 24 a9 71 e9 66-a4 65 cf a8 ee 2f ab 4c
...$.q.f.e.../.L
    03d0 - 0a c0 08 87 1e 34 84 c1-a6 fe 7b 55 42 bb b2 0c
.....4....{UB...
    03e0 - 46 c4 1a 77 df cb 9c 8f-9f de 9d 57 8a 5c e1 12
F..w.......W.\..
    03f0 - 43 8e f3 fe 09 63 7f 47-c0 31 bc 51 f1 59 2e fb
C....c.G.1.Q.Y..
    0400 - 89 f7 16 99 20 eb 52 e3-5f 11 70 4a c4 9e 19 5d
.... .R._.pJ...]
    0410 - 29 11 23 f6 9b f9 d1 2f-6c f9 55 54 53 c5 65 6a
).#..../l.UTS.ej
    0420 - c7 b0 26 cc 42 b6 8d c3-19 d8 f0 57 7d 55 59 65
```

```
..&.B......W}UYe
    0430 - 6c 39 8c a0 69 51 d2 3d-d4 d4 71 c5 7f 6e eb f3
l9..iQ.=..q..n..
    0440 - 46 45 2a 73 a6 1c cb ec-47 35 13 05 81 53 02 6f
FE*s....G5...S.o
    0450 - f1 ae 8c 27 a2 b7 05 0d-e3 f9 20 46 1d 4a d6 ce
...'...... F.J..
    0460 - b6 19 72 0f 3f 60 1e 65-57 5c 55 a3 b5 4d f1 05
..r.?`.eW\U..M..
    0470 - 2b 41 a2 47 2e a9 63 42-be 37 e1 d2 28 92
+A.G..cB.7..(.

    Start Time: 1600656460
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
closed
```

工作正常

```
iMac:config neo$ curl -k -i
https://config:s3cr3t@192.168.3.85:8888/netkiller-dev.json
HTTP/2 200
set-cookie: JSESSIONID=75D0C2900D87C789DF596220FA77012D;
Path=/; Secure; HttpOnly
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
cache-control: no-cache, no-store, max-age=0, must-revalidate
pragma: no-cache
expires: 0
strict-transport-security: max-age=31536000 ; includeSubDomains
x-frame-options: DENY
content-type: application/json
content-length: 100
date: Mon, 21 Sep 2020 02:51:11 GMT

{"sms":{"gateway":
{"url":"https://sms.netkiller.cn/v1","username":"netkiller","pa
ssword":"123456"}}}
```

31. Spring boot with Docker stack

31.1. 编译 Docker 镜像

```
iMac:config neo$ mvn docker:build
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< cn.netkiller:config >---------
----------------
[INFO] Building config 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]-----------------
----------------
[INFO]
[INFO] --- docker-maven-plugin:1.2.2:build (default-cli) @
config ---
[INFO] Using authentication suppliers:
[ConfigFileRegistryAuthSupplier, FixedRegistryAuthSupplier]
[INFO] Copying
/Users/neo/workspace/Microservice/config/target/config-0.0.1-
SNAPSHOT.jar ->
/Users/neo/workspace/Microservice/config/target/docker/srv/conf
ig-0.0.1-SNAPSHOT.jar
[INFO] Building image netkiller/config
Step 1/7 : FROM openjdk

 ---> b2324c52d969
Step 2/7 : WORKDIR /srv

 ---> Using cache
 ---> f7c1730935c6
Step 3/7 : ADD /srv/config-0.0.1-SNAPSHOT.jar /srv/

 ---> Using cache
 ---> 8b5a053550ba
Step 4/7 : EXPOSE 8888

 ---> Running in 7f4e35b3564f
Removing intermediate container 7f4e35b3564f
 ---> a968ea58ba64
Step 5/7 : ENTRYPOINT ["java", "-jar", "-
```

```
Djava.security.egd=file:/dev/./urandom", "/srv/config-0.0.1-
SNAPSHOT.jar"]

 ---> Running in 6b110b5d16b7
Removing intermediate container 6b110b5d16b7
 ---> a8ab10c1c186
Step 6/7 : CMD ["java", "-version"]

 ---> Running in 4f2dc6e08404
Removing intermediate container 4f2dc6e08404
 ---> a74bbf7b6c30
Step 7/7 : VOLUME /tmp

 ---> Running in 0a3836ea768f
Removing intermediate container 0a3836ea768f
 ---> 5e13d81a9dea
ProgressMessage{id=null, status=null, stream=null, error=null,
progress=null, progressDetail=null}
Successfully built 5e13d81a9dea
Successfully tagged netkiller/config:latest
[INFO] Built netkiller/config
[INFO] Tagging netkiller/config with 0.0.1-SNAPSHOT
[INFO] Tagging netkiller/config with latest
[INFO] ------------------------------------------------------------
----------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
----------------
[INFO] Total time:  20.516 s
[INFO] Finished at: 2020-09-20T21:49:28+08:00
[INFO] ------------------------------------------------------------
----------------
```

31.2.

初始化 Swarm

```
iMac:springboot neo$ docker swarm init
Swarm initialized: current node (qvqez97c8ja014ktmroy9sw47) is
```

```
now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-
49w6mcdjvj9nhb1go4wiazygupvj6qmjy7mgdb7x5bzqspldss-
6yfvnij63it1qbs2nwvqw6xv0 192.168.65.3:2377

To add a manager to this swarm, run 'docker swarm join-token
manager' and follow the instructions.
```

创建 docker-compose.yml 文件

```
version: '3.8'

services:
  config:
    image: netkiller/config:latest
    ports:
      - "8888"
    volumes:
      - /tmp/config:/tmp
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
```

部署服务

```
iMac:springboot neo$ docker stack deploy -c docker-compose.yml
springboot
Creating network springboot_default
```

```
Creating service springboot_config
```

查看部署情况

```
iMac:springboot neo$ docker stack ls
NAME                SERVICES            ORCHESTRATOR
springboot          1                   Swarm
iMac:springboot neo$ docker stack services springboot
ID                  NAME                MODE
REPLICAS            IMAGE                     PORTS
viavpkzk6lvo        springboot_config   replicated          0/1
netkiller/config:latest   *:30001->8888/tcp
```

查看服务运行状态

```
iMac:springboot neo$ docker stack ps springboot
ID                  NAME                    IMAGE
NODE                DESIRED STATE       CURRENT STATE
ERROR               PORTS
mr30ujsdbti4        springboot_config.1
netkiller/config:latest   docker-desktop      Running
Preparing 4 minutes ago
```

32. Spring boot with Kubernetes

首先你需要构建 docker 镜像，并且 push 到 registry 参考这里

32.1. Kubernetes 编排脚本

创建密钥

```
kubectl create secret docker-registry docker-hub \
--docker-server=https://index.docker.io/v1/ \
--docker-username=netkiller \
--docker-password=passw0rd \
--docker-email=netkiller@msn.com
```

查看是否创建成功

```
iMac:spring neo$ kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-fhfn8   kubernetes.io/service-account-token   3      2d23h
docker-hub            kubernetes.io/dockerconfigjson        1      15s
```

springboot.yml 编排脚本

```
apiVersion: v1
kind: Service
metadata:
  name: springboot
```

```
  namespace: default
  labels:
    app: springboot
spec:
  type: NodePort
  ports:
  - port: 8888
    nodePort: 30000
  selector:
    app: springboot
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: springboot
spec:
  replicas: 3
  selector:
    matchLabels:
      app: springboot
  template:
    metadata:
      labels:
        app: springboot
    spec:
      containers:
      - name: springboot
        image: netkiller/config:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8888
      imagePullSecrets:
        - name: docker-hub
```

32.2. 部署镜像

```
iMac:spring neo$ kubectl create -f springboot.yml
deployment.apps/springboot created

iMac:spring neo$ kubectl expose deployment springboot --
```

```
type="LoadBalancer"
service/springboot exposed

iMac:spring neo$ minikube service list
|-------------------|--------------------------|-----------
---|--------------------------|
|      NAMESPACE       |           NAME           | TARGET
PORT  |            URL           |
|-------------------|--------------------------|-----------
---|--------------------------|
| default              | kubernetes               | No node
port |
| default              | springboot               |
8888 | http://192.168.64.2:30000 |
| kube-system          | kube-dns                 | No node
port |
| kube-system          | registry                 | No node
port |
| kubernetes-dashboard | dashboard-metrics-scraper | No node
port |
| kubernetes-dashboard | kubernetes-dashboard     | No node
port |
|-------------------|--------------------------|-----------
---|--------------------------|

iMac:spring neo$ minikube service springboot --url
http://192.168.64.2:30000
```

http://192.168.64.2:30000 是访问地址，Kubernetes 会负载均衡到后面的三个 pod 上。

```
iMac:config neo$ curl -k
https://config:s3cr3t@192.168.64.2:30000/netkiller-dev.json
{"sms":{"gateway":
{"url":"https://sms.netkiller.cn/v1","username":"netkiller","pa
ssword":"123456"}}}
```

删除服务

```
iMac:spring neo$ kubectl delete -f springboot.yml
service "springboot" deleted
deployment.apps "springboot" deleted
```

33. Spring boot with command line

33.1. Maven

开发命令行程序通常我们不需要 Tomcat，所以不需要引入 spring-boot-starter-web 依赖，spring-boot-starter 依赖不含Tomcat。

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>command</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Command Line</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
starter</artifactId>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
```

```
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

33.2. CommandLineRunner 例子

```
package cn.netkiller.cmd;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application implements CommandLineRunner {

    private static Logger logger =
LoggerFactory.getLogger(Sb2runnerApplication.class);

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        logger.info("服务已启动，执行command line runner。");
```

```
        for (int i = 0; i < args.length; ++i) {
            logger.info("args[{}]: {}", i, args[i]);
        }
    }
}
```

```
% java -jar target/command-0.0.1-SNAPSHOT.jar --
host=ww.netkiller.cn java spring boot --help -v
```

33.3. ApplicationRunner 例子

```
package cn.netkiller.component;

import java.util.Arrays;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.core.annotation.Order;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Component;

@Component
@Order(1)
public class Command implements ApplicationRunner {
        private final static Logger logger =
LoggerFactory.getLogger(Command.class);

        @Override
        public void run(ApplicationArguments args) throws
Exception {
                System.out.println("==ApplicationRunner====="
```

```
+ Arrays.asList(args.getSourceArgs()));
                System.out.println("==getOptionNames========"
+ args.getOptionNames());
                System.out.println("==getOptionValues======="
+ args.getOptionValues("foo"));
                System.out.println("==getOptionValues======="
+ args.getOptionValues("developer.name"));
//              System.exit(0);
        }

}
```

34. Spring Boot Actuator

健康检查、审计、统计和监控

34.1. Maven 依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

34.2. 与 Spring Boot Actuator 有关的配置

application.properties

跨域配置

```
management.endpoints.web.cors.allowed-
origins=https://example.com
management.endpoints.web.cors.allowed-methods=GET,POST
```

禁用HTTP端点

如果您不想通过HTTP公开端点，则可以将管理端口设置为-1，如以下示例所示：

```
management.server.port=-1
```

安全配置

```
security.basic.enabled=true
security.basic.path=/admin     #针对/admin路径进行认证
security.user.name=admin      #认证使用的用户名
security.user.password=password    #认证使用的密码
management.security.roles=SUPERUSER

management.port=11111    #actuator暴露接口使用的端口，为了和api接口使用的端口进行分离
management.context-path=/admin    #actuator暴露接口的前缀
management.security.enabled=true    #actuator是否需要安全保证

endpoints.metrics.sensitive=false    #actuator的metrics接口是否需要安全保证
endpoints.metrics.enabled=true

endpoints.health.sensitive=false  #actuator的health接口是否需要安全保证
endpoints.health.enabled=true
```

修改 **actuator** 地址

```
management:
  endpoints:
    web:
      base-path: /monitor
```

关机

配置文件中加入

```
management.endpoint.shutdown.enabled=true
```

```
curl -X POST www.netkiller.cn:8080/actuator/shutdown
```

34.3. actuator 接口

常规接口

```
neo@MacBook-Pro ~ % curl -s
http://www.netkiller.cn:8080/actuator | jq
{
  "_links": {
    "self": {
      "href": "http://www.netkiller.cn:8080/actuator",
      "templated": false
    },
    "health": {
      "href": "http://www.netkiller.cn:8080/actuator/health",
      "templated": false
    },
    "health-component": {
      "href":
"http://www.netkiller.cn:8080/actuator/health/{component}",
      "templated": true
    },
    "health-component-instance": {
      "href":
"http://www.netkiller.cn:8080/actuator/health/{component}/{inst
ance}",
      "templated": true
    },
```

```
    "info": {
      "href": "http://www.netkiller.cn:8080/actuator/info",
      "templated": false
    }
  }
}
```

暴漏所有接口

```
management:
  endpoints:
    web:
      exposure:
        include: "*"
```

再次访问 /actuator

```
neo@MacBook-Pro-Neo ~/w/Architect (master)> curl -s
https://www.netkiller.cn/actuator/ | jq
{
  "_links": {
    "self": {
      "href": "http://pre.ejiayou.com/ensd-channel-
service/monitor",
      "templated": false
    },
    "archaius": {
      "href": "https://www.netkiller.cn/actuator/archaius",
      "templated": false
    },
    "nacosconfig": {
      "href": "https://www.netkiller.cn/actuator/nacosconfig",
      "templated": false
    },
    "nacosdiscovery": {
```

```
      "href":
"https://www.netkiller.cn/actuator/nacosdiscovery",
      "templated": false
    },
    "auditevents": {
      "href": "https://www.netkiller.cn/actuator/auditevents",
      "templated": false
    },
    "beans": {
      "href": "https://www.netkiller.cn/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href":
"https://www.netkiller.cn/actuator/caches/{cache}",
      "templated": true
    },
    "caches": {
      "href": "https://www.netkiller.cn/actuator/caches",
      "templated": false
    },
    "health-component": {
      "href":
"https://www.netkiller.cn/actuator/health/{component}",
      "templated": true
    },
    "health": {
      "href": "https://www.netkiller.cn/actuator/health",
      "templated": false
    },
    "health-component-instance": {
      "href":
"https://www.netkiller.cn/actuator/health/{component}/{instance
}",
      "templated": true
    },
    "conditions": {
      "href": "https://www.netkiller.cn/actuator/conditions",
      "templated": false
    },
    "configprops": {
      "href": "https://www.netkiller.cn/actuator/configprops",
      "templated": false
    },
    "env-toMatch": {
```

```
      "href":
"https://www.netkiller.cn/actuator/env/{toMatch}",
      "templated": true
    },
    "env": {
      "href": "https://www.netkiller.cn/actuator/env",
      "templated": false
    },
    "info": {
      "href": "https://www.netkiller.cn/actuator/info",
      "templated": false
    },
    "loggers-name": {
      "href":
"https://www.netkiller.cn/actuator/loggers/{name}",
      "templated": true
    },
    "loggers": {
      "href": "https://www.netkiller.cn/actuator/loggers",
      "templated": false
    },
    "heapdump": {
      "href": "https://www.netkiller.cn/actuator/heapdump",
      "templated": false
    },
    "threaddump": {
      "href": "https://www.netkiller.cn/actuator/threaddump",
      "templated": false
    },
    "metrics": {
      "href": "https://www.netkiller.cn/actuator/metrics",
      "templated": false
    },
    "metrics-requiredMetricName": {
      "href":
"https://www.netkiller.cn/actuator/metrics/{requiredMetricName}
",
      "templated": true
    },
    "scheduledtasks": {
      "href":
"https://www.netkiller.cn/actuator/scheduledtasks",
      "templated": false
    },
    "httptrace": {
```

```
      "href": "https://www.netkiller.cn/actuator/httptrace",
      "templated": false
    },
    "mappings": {
      "href": "https://www.netkiller.cn/actuator/mappings",
      "templated": false
    },
    "refresh": {
      "href": "https://www.netkiller.cn/actuator/refresh",
      "templated": false
    },
    "features": {
      "href": "https://www.netkiller.cn/actuator/features",
      "templated": false
    },
    "service-registry": {
      "href": "https://www.netkiller.cn/actuator/service-
registry",
      "templated": false
    }
  }
}
```

Spring boot 2.3.0

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator |jq
{
  "_links": {
    "self": {
      "href": "http://www.netkiller.cn:8080/actuator",
      "templated": false
    },
    "beans": {
      "href": "http://www.netkiller.cn:8080/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href":
"http://www.netkiller.cn:8080/actuator/caches/{cache}",
```

```
      "templated": true
    },
    "caches": {
      "href": "http://www.netkiller.cn:8080/actuator/caches",
      "templated": false
    },
    "health": {
      "href": "http://www.netkiller.cn:8080/actuator/health",
      "templated": false
    },
    "health-path": {
      "href":
"http://www.netkiller.cn:8080/actuator/health/{*path}",
      "templated": true
    },
    "info": {
      "href": "http://www.netkiller.cn:8080/actuator/info",
      "templated": false
    },
    "conditions": {
      "href":
"http://www.netkiller.cn:8080/actuator/conditions",
      "templated": false
    },
    "configprops": {
      "href":
"http://www.netkiller.cn:8080/actuator/configprops",
      "templated": false
    },
    "configprops-prefix": {
      "href":
"http://www.netkiller.cn:8080/actuator/configprops/{prefix}",
      "templated": true
    },
    "env": {
      "href": "http://www.netkiller.cn:8080/actuator/env",
      "templated": false
    },
    "env-toMatch": {
      "href":
"http://www.netkiller.cn:8080/actuator/env/{toMatch}",
      "templated": true
    },
    "logfile": {
      "href": "http://www.netkiller.cn:8080/actuator/logfile",
```

```
      "templated": false
    },
    "loggers": {
      "href": "http://www.netkiller.cn:8080/actuator/loggers",
      "templated": false
    },
    "loggers-name": {
      "href":
"http://www.netkiller.cn:8080/actuator/loggers/{name}",
      "templated": true
    },
    "heapdump": {
      "href": "http://www.netkiller.cn:8080/actuator/heapdump",
      "templated": false
    },
    "threaddump": {
      "href":
"http://www.netkiller.cn:8080/actuator/threaddump",
      "templated": false
    },
    "metrics-requiredMetricName": {
      "href":
"http://www.netkiller.cn:8080/actuator/metrics/{requiredMetricN
ame}",
      "templated": true
    },
    "metrics": {
      "href": "http://www.netkiller.cn:8080/actuator/metrics",
      "templated": false
    },
    "scheduledtasks": {
      "href":
"http://www.netkiller.cn:8080/actuator/scheduledtasks",
      "templated": false
    },
    "mappings": {
      "href": "http://www.netkiller.cn:8080/actuator/mappings",
      "templated": false
    }
  }
}
```

34.4. 健康状态

curl www.netkiller.cn:8080/actuator/health

```
neo@MacBook-Pro ~ % curl -s
http://www.netkiller.cn:8080/actuator/health | jq
{
  "status": "UP"
}
```

健康状态

详细的健康状态信息

```
management.endpoint.health.show-details=always
```

```
neo@MacBook-Pro ~ % curl -s
http://www.netkiller.cn:8080/actuator/health | jq
{
  "status": "UP",
  "details": {
    "diskSpace": {
      "status": "UP",
      "details": {
        "total": 250790436864,
        "free": 23556112384,
        "threshold": 10485760
      }
    }
  }
}
```

34.5. info 配置信息

返回 application.properties 文件中定义的 info 配置信息，如:

```
# info端点信息配置
info.app.name=spring-boot-example
info.app.version=v1.0.0
```

```
neo@MacBook-Pro ~ % curl -s
http://www.netkiller.cn:8080/actuator/info | jq
{
  "app": {
    "name": "spring-boot-example",
    "version": "v1.0.0"
  }
}
```

34.6. beans 信息

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/beans |jq
{
  "contexts": {
    "watch-production": {
      "beans": {
        "spring.jpa-
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties":
{
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
          "dependencies": []
        },
        "applicationTaskExecutor": {
          "aliases": [
            "taskExecutor"
          ],
          "scope": "singleton",
          "type":
"org.springframework.scheduling.concurrent.ThreadPoolTaskExecut
or",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/task/TaskExecutorConfig
urations$TaskExecutorConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$TaskExecutorConfiguration",
            "taskExecutorBuilder"
          ]
        },
        "healthEndpointGroups": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.AutoConf
iguredHealthEndpointGroups",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties"
          ]
        },
        "webConversionServiceProvider": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.providers.WebConversionServiceProvider",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration$WebCon
versionServiceConfiguration.class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration$WebCon
versionServiceConfiguration"
          ]
        },
        "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointProperties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.servlet.MultipartAu
toConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.MultipartAu
toConfiguration",
          "dependencies": [
            "spring.servlet.multipart-
org.springframework.boot.autoconfigure.web.servlet.MultipartPro
perties"
          ]
        },
        "hikariDataSourceMeterBinder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsCon
figuration$HikariDataSourceMeterBinder",
          "resource": "class path resource
```

```
[org/springframework/boot/actuate/autoconfigure/metrics/jdbc/Da
taSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsCon
figuration"
          ]
        },
        "jdbcTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.jdbc.core.JdbcTemplate",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/JdbcTemplateConfig
uration.class]",
          "dependencies": [
            "dataSourceScriptDatabaseInitializer",

"org.springframework.boot.autoconfigure.jdbc.JdbcTemplateConfig
uration",
            "dataSource",
            "spring.jdbc-
org.springframework.boot.autoconfigure.jdbc.JdbcProperties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration",
          "dependencies": [

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties"
          ]
        },
        "webEndpointPathMapper": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.Ma
ppingWebEndpointPathMapper",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaCon
figuration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaCon
figuration",
          "dependencies": [
            "dataSource",
            "spring.jpa-
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",

"org.springframework.beans.factory.support.DefaultListableBeanF
actory@37efd131",
            "spring.jpa.hibernate-
org.springframework.boot.autoconfigure.orm.jpa.HibernatePropert
ies"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.cache.CachesEnd
pointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.cache.CachesEnd
pointAutoConfiguration",
          "dependencies": []
        },
```

```
"org.springframework.boot.autoconfigure.aop.AopAutoConfiguratio
n$AspectJAutoProxyingConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.aop.AopAutoConfiguratio
n$AspectJAutoProxyingConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.transaction.Transaction
AutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.Transaction
AutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.transaction.Transaction
AutoConfiguration$EnableTransactionManagementConfiguration$Cgli
bAutoProxyConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.Transaction
AutoConfiguration$EnableTransactionManagementConfiguration$Cgli
bAutoProxyConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.ClientHttpConnectorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.ClientHttpConnectorAutoConfiguration",
          "dependencies": []
        },

"org.springdoc.webmvc.core.configuration.MultipleOpenApiSupport
Configuration": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.core.configuration.MultipleOpenApiSupport
Configuration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.WebClientObservationConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web
.client.WebClientObservationConfiguration",
          "dependencies": []
        },
        "management.simple.metrics.export-
org.springframework.boot.actuate.autoconfigure.metrics.export.s
imple.SimpleProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.export.
simple.SimpleProperties",
          "dependencies": []
        },
        "beanNameViewResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.view.BeanNameViewResolver",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration$WhitelabelErrorViewConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$WhitelabelErrorViewConfiguration"
          ]
        },
        "reactiveRedisTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.ReactiveRedisTemplate",
```

```
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/RedisReactiv
eAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.RedisReactiv
eAutoConfiguration",
            "redisConnectionFactory",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "swaggerResourceResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.ui.SwaggerResourceResolver",
          "resource": "class path resource
[org/springdoc/webmvc/ui/SwaggerConfig.class]",
          "dependencies": [
            "org.springdoc.webmvc.ui.SwaggerConfig",

"org.springdoc.core.properties.SwaggerUiConfigProperties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.endpoint.web.se
rvlet.WebMvcEndpointManagementContextConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.se
rvlet.WebMvcEndpointManagementContextConfiguration",
          "dependencies": []
        },
        "viewResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.view.ContentNegotiatingViewRes
olver",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter.class]",
```

```
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter",

"org.springframework.beans.factory.support.DefaultListableBeanF
actory@37efd131"
          ]
        },
        "toolsController": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.controller.ToolsController",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/ToolsController.class]",
          "dependencies": [
            "aliyunService",
            "psychoanalysisService",
            "baiduService",
            "audioService",
            "mqttService",
            "config"
          ]
        },
        "reactorResourceFactory": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.http.client.ReactorResourceFactory",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/reactor/netty/ReactorNe
ttyConfigurations$ReactorResourceFactoryConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.reactor.netty.ReactorNe
ttyConfigurations$ReactorResourceFactoryConfiguration",
            "spring.reactor.netty-
org.springframework.boot.autoconfigure.reactor.netty.ReactorNet
tyProperties"
          ]
        },
        "projectingArgumentResolverBeanPostProcessor": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.data.web.config.ProjectingArgumentResolver
Registrar$ProjectingArgumentResolverBeanPostProcessor",
          "resource": "class path resource
[org/springframework/data/web/config/ProjectingArgumentResolver
Registrar.class]",
          "dependencies": []
        },
        "tomcatServletWebServerFactoryCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.TomcatServl
etWebServerFactoryCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/ServletWebS
erverFactoryAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryAutoConfiguration",
            "server-
org.springframework.boot.autoconfigure.web.ServerProperties"
          ]
        },
        "server-
org.springframework.boot.autoconfigure.web.ServerProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.ServerProperties",
          "dependencies": []
        },
        "redisConnectionDetails": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.redis.PropertiesRe
disConnectionDetails",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/RedisAutoCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.RedisAutoCon
```

```
figuration",
            "spring.data.redis-
org.springframework.boot.autoconfigure.data.redis.RedisProperti
es"
          ]
        },
        "messageConverters": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ers",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/http/HttpMessageConvert
ersAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ersAutoConfiguration"
          ]
        },
        "websocketServletWebServerCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.websocket.servlet.Tomca
tWebSocketServletWebServerCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/websocket/servlet/WebSo
cketServletAutoConfiguration$TomcatWebSocketConfiguration.class
]",
          "dependencies": [

"org.springframework.boot.autoconfigure.websocket.servlet.WebSo
cketServletAutoConfiguration$TomcatWebSocketConfiguration"
          ]
        },
        "configurationPropertiesReportEndpointWebExtension": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.context.properties.Configurat
ionPropertiesReportEndpointWebExtension",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/context/propert
```

```
ies/ConfigurationPropertiesReportEndpointAutoConfiguration.clas
s]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.context.propert
ies.ConfigurationPropertiesReportEndpointAutoConfiguration",
            "configurationPropertiesReportEndpoint",
            "management.endpoint.configprops-
org.springframework.boot.actuate.autoconfigure.context.properti
es.ConfigurationPropertiesReportEndpointProperties"
          ]
        },
        "redisCustomConversions": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.convert.RedisCustomConvers
ions",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.sql.init.SqlInitializat
ionAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.sql.init.SqlInitializat
ionAutoConfiguration",
          "dependencies": []
        },
        "dataSourceScriptDatabaseInitializer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.sql.init.SqlDataSourceS
criptDatabaseInitializer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/sql/init/DataSourceInit
ializationConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.sql.init.DataSourceInit
ializationConfiguration",
            "dataSource",
            "spring.sql.init-
```

```
org.springframework.boot.autoconfigure.sql.init.SqlInitializati
onProperties"
          ]
        },
        "meterRegistryPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.MeterRe
gistryPostProcessor",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
AutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },

"org.springdoc.core.configuration.SpringDocSortConfiguration":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocSortConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.JdbcClientAutoConf
iguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.JdbcClientAutoConf
iguration",
          "dependencies": []
        },
        "endpointMediaTypes": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.EndpointMediaTyp
es",
          "resource": "class path resource
```

```
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration"
          ]
        },
        "jvmCompilationMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.jvm.JvmCompilationMetrics
",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },
        "cacheMetricsRegistrar": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.cache.CacheMetricsReg
istrar",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/cache/C
acheMetricsRegistrarConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMetricsRegistrarConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointWebExtensionConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
```

```
dpointWebExtensionConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientAutoConfiguration",
          "dependencies": []
        },
        "jdbcConnectionDetailsHikariBeanPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.HikariJdbcConnecti
onDetailsBeanPostProcessor",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/DataSourceConfigur
ation$Hikari.class]",
          "dependencies": []
        },

"org.springdoc.core.configuration.SpringDocKotlinConfiguration"
: {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocKotlinConfiguration"
,
          "dependencies": []
        },
        "spring.ssl-
org.springframework.boot.autoconfigure.ssl.SslProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.ssl.SslProperties",
          "dependencies": []
        },
        "repositoryTagsProvider": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.boot.actuate.metrics.data.DefaultRepositor
yTagsProvider",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/data/Re
positoryMetricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.data.Re
positoryMetricsAutoConfiguration"
          ]
        },
        "dbHealthContributor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.jdbc.DataSourceHealthIndicato
r",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/jdbc/DataSource
HealthContributorAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.jdbc.DataSource
HealthContributorAutoConfiguration",
            "dataSource",
            "management.health.db-
org.springframework.boot.actuate.autoconfigure.jdbc.DataSourceH
ealthIndicatorProperties"
          ]
        },
        "dataSourcePoolMetadataMeterBinder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$DataSourcePoolMetadataMetr
icsConfiguration$DataSourcePoolMetadataMeterBinder",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/jdbc/Da
taSourcePoolMetricsAutoConfiguration$DataSourcePoolMetadataMetr
icsConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
```

```
taSourcePoolMetricsAutoConfiguration$DataSourcePoolMetadataMetr
icsConfiguration",
            "dataSource"
          ]
        },
        "webServerFactoryCustomizerBeanPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.server.WebServerFactoryCustomizer
BeanPostProcessor",
          "dependencies": []
        },
        "metricsHttpClientUriTagFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.config.MeterFilter$9",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/web
/client/HttpClientObservationsAutoConfiguration$MeterFilterConf
iguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.HttpClientObservationsAutoConfiguration$MeterFilterConf
iguration",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties",
            "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties"
          ]
        },
        "timedAspect": {
          "aliases": [],
          "scope": "singleton",
          "type": "io.micrometer.core.aop.TimedAspect",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
AspectsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
```

```
AspectsAutoConfiguration",
            "simpleMeterRegistry"
          ]
        },

"org.springframework.boot.autoconfigure.websocket.servlet.WebSo
cketServletAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.websocket.servlet.WebSo
cketServletAutoConfiguration",
          "dependencies": []
        },
        "management.health.diskspace-
org.springframework.boot.actuate.autoconfigure.system.DiskSpace
HealthIndicatorProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.system.DiskSpac
eHealthIndicatorProperties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.gson.GsonAutoConfigurat
ion": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.gson.GsonAutoConfigurat
ion",
          "dependencies": []
        },
        "controllerEndpointHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.servlet.Controll
erEndpointHandlerMapping",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/se
rvlet/WebMvcEndpointManagementContextConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.actuate.autoconfigure.endpoint.web.se
rvlet.WebMvcEndpointManagementContextConfiguration",
            "controllerEndpointDiscoverer",
            "management.endpoints.web.cors-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Cor
sEndpointProperties",
            "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties"
          ]
        },
        "webFluxSupportConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.WebFluxSupportConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration$Spring
DocWebFluxSupportConfiguration.class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration$Spring
DocWebFluxSupportConfiguration",
            "springdocObjectMapperProvider"
          ]
        },
        "management.endpoint.env-
org.springframework.boot.actuate.autoconfigure.env.EnvironmentE
ndpointProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.env.Environment
EndpointProperties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.JdbcTemplateAutoCo
nfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.JdbcTemplateAutoCo
nfiguration",
          "dependencies": []
```

```
        },

"org.springframework.boot.actuate.autoconfigure.management.Thre
adDumpEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.management.Thre
adDumpEndpointAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.availability.Applicatio
nAvailabilityAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.availability.Applicatio
nAvailabilityAutoConfiguration",
          "dependencies": []
        },
        "config": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.config.Config",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/config/Config.class]",
          "dependencies": []
        },
        "jdbcClient": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.jdbc.core.simple.DefaultJdbcClient",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/JdbcClientAutoConf
iguration.class]",
          "dependencies": [
            "dataSourceScriptDatabaseInitializer",

"org.springframework.boot.autoconfigure.jdbc.JdbcClientAutoConf
iguration",
            "namedParameterJdbcTemplate"
          ]
```

```
        },
        "observabilitySchedulingConfigurer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksObservabilityAutoConfiguration$ObservabilitySchedulin
gConfigurer",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/scheduling/Sche
duledTasksObservabilityAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksObservabilityAutoConfiguration",
            "observationRegistry"
          ]
        },
        "metricsEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.MetricsEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
EndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
EndpointAutoConfiguration",
            "simpleMeterRegistry"
          ]
        },
        "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationProperties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.aop.AopAutoConfiguratio
```

```
n": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.aop.AopAutoConfiguratio
n",
          "dependencies": []
        },
        "spring.cache-
org.springframework.boot.autoconfigure.cache.CacheProperties":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.cache.CacheProperties",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.info.InfoContri
butorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.info.InfoContri
butorAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$ObservedAspectConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$ObservedAspectConfiguration",
          "dependencies": []
        },
        "metricsObservationHandlerGrouping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationHandlerGrouping",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/Obs
```

```
ervationAutoConfiguration$OnlyMetricsConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$OnlyMetricsConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryAutoConfiguration",
          "dependencies": []
        },
        "spring.jpa.hibernate-
org.springframework.boot.autoconfigure.orm.jpa.HibernatePropert
ies": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.orm.jpa.HibernateProper
ties",
          "dependencies": []
        },
        "environmentEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.env.EnvironmentEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/env/Environment
EndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.env.Environment
EndpointAutoConfiguration",
            "environment",
            "management.endpoint.env-
org.springframework.boot.actuate.autoconfigure.env.EnvironmentE
ndpointProperties"
          ]
        },
```

```
        "kotlinCoroutinesReturnTypeParser": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.parsers.KotlinCoroutinesReturnTypeParser",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocKotlinConfiguration.
class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocKotlinConfiguration"
          ]
        },
        "springDocCustomizers": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.customizers.SpringDocCustomizers",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration"
          ]
        },
        "jacksonCodecCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$JacksonCodecConfiguration$$Lambda/0x00007ff400a3506
8",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/http/codec/CodecsAutoCo
nfiguration$JacksonCodecConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$JacksonCodecConfiguration",
            "jacksonObjectMapper"
          ]
        },
        "conventionErrorViewResolver": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Defau
ltErrorViewResolver",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration$DefaultErrorViewResolverConfiguration.clas
s]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$DefaultErrorViewResolverConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
          "dependencies": [
            "spring.mvc-
org.springframework.boot.autoconfigure.web.servlet.WebMvcProper
ties",
            "spring.web-
org.springframework.boot.autoconfigure.web.WebProperties",

"org.springframework.beans.factory.support.DefaultListableBeanF
actory@37efd131",

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter",
            "swaggerWebMvcConfigurer",

"org.springframework.data.web.config.SpringDataWebConfiguration
",
            "endpointObjectMapperWebMvcConfigurer"
          ]
        },

"org.springdoc.core.configuration.SpringDocConfiguration$OpenAp
iResourceAdvice": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocConfiguration$OpenAp
iResourceAdvice",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.Logback
MetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.Logback
MetricsAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.HttpClientObservationsAutoConfiguration$MeterFilterConf
iguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web
.client.HttpClientObservationsAutoConfiguration$MeterFilterConf
iguration",
          "dependencies": []
        },
        "localeCharsetMappingsCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.HttpEncodin
gAutoConfiguration$LocaleCharsetMappingsCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/HttpEncodin
gAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.HttpEncodin
gAutoConfiguration"
          ]
```

```
        },

"org.springframework.boot.actuate.autoconfigure.logging.Loggers
EndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.logging.Loggers
EndpointAutoConfiguration",
          "dependencies": []
        },
        "jsonMixinModuleEntries": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.jackson.JsonMixinModuleEntries",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConf
iguration$JacksonMixinConfiguration.class]",
          "dependencies": [

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "formContentFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.servlet.filter.OrderedFormContent
Filter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration"
          ]
        },
        "mockController": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.controller.MockController",
          "resource": "URL [jar:nested:/app/watch-1.0-
```

```
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/MockController.class]",
          "dependencies": []
        },
        "pictureService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.service.PictureService",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/PictureService.class]",
          "dependencies": [
            "pictureRepository",
            "mqttService",
            "baiduService",
            "sessionStatusService"
          ]
        },
        "defaultViewResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.view.InternalResourceViewResol
ver",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter"
          ]
        },
        "keyValueMappingContext": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.mapping.RedisMappingContex
t",
          "dependencies": [
            "redisMappingConfiguration#0"
          ]
        },

"org.springframework.boot.autoconfigure.transaction.Transaction
```

```
AutoConfiguration$EnableTransactionManagementConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.TransactionAutoConfiguration$EnableTransactionManagementConfiguration",
          "dependencies": []
        },
        "routerFunctionMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.function.support.RouterFunctionMapping",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfiguration$EnableWebMvcConfiguration",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },
        "jacksonObjectMapperBuilder": {
          "aliases": [],
          "scope": "prototype",
          "type":
"org.springframework.http.converter.json.Jackson2ObjectMapperBuilder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConfiguration$JacksonObjectMapperBuilderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConfiguration$JacksonObjectMapperBuilderConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@18e36d14",
            "standardJacksonObjectMapperBuilderCustomizer"
          ]
        },
```

```
"org.springframework.boot.autoconfigure.dao.PersistenceExceptio
nTranslationAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.dao.PersistenceExceptio
nTranslationAutoConfiguration",
          "dependencies": []
        },
        "cacheManager": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.cache.RedisCacheManager",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/cache/RedisCacheConfigu
ration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.cache.RedisCacheConfigu
ration",
            "spring.cache-
org.springframework.boot.autoconfigure.cache.CacheProperties",
            "cacheManagerCustomizers",
            "redisConnectionFactory",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "spring.task.scheduling-
org.springframework.boot.autoconfigure.task.TaskSchedulingPrope
rties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskSchedulingProp
erties",
          "dependencies": []
        },
        "spring.reactor-
org.springframework.boot.autoconfigure.reactor.ReactorPropertie
s": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.boot.autoconfigure.reactor.ReactorProperti
es",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.data.redis.RedisAutoCon
figuration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.redis.RedisAutoCon
figuration",
          "dependencies": []
        },
        "healthEndpointWebExtension": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.health.HealthEndpointWebExten
sion",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointWebExtensionConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointWebExtensionConfiguration",
            "healthContributorRegistry",
            "healthEndpointGroups",
            "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties"
          ]
        },
        "multipartResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.multipart.support.StandardServletMulti
partResolver",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/MultipartAu
toConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.autoconfigure.web.servlet.MultipartAu
toConfiguration"
          ]
        },
        "sessionStatusService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.service.SessionStatusService",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/SessionStatusService.class]"
,
          "dependencies": [
            "sessionStatusRepository"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration$WebEndpointServletConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration$WebEndpointServletConfiguration",
          "dependencies": []
        },
        "requestMappingHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.mvc.method.annotation.RequestM
appingHandlerMapping",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcContentNegotiationManager",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },
```

```
        "webExposeExcludePropertyEndpointFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.expose
.IncludeExcludeEndpointFilter",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration"
          ]
        },
        "lifecycleProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.context.support.DefaultLifecycleProcessor"
,
          "resource": "class path resource
[org/springframework/boot/autoconfigure/context/LifecycleAutoCo
nfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.context.LifecycleAutoCo
nfiguration",
            "spring.lifecycle-
org.springframework.boot.autoconfigure.context.LifecyclePropert
ies"
          ]
        },
        "requestMappingHandlerAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.mvc.method.annotation.RequestM
appingHandlerAdapter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
```

```
onfiguration$EnableWebMvcConfiguration",
            "mvcContentNegotiationManager",
            "mvcConversionService",
            "mvcValidator"
          ]
        },
        "tomcatMetricsBinder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.web.tomcat.TomcatMetr
icsBinder",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/web/tom
cat/TomcatMetricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.web.tom
cat.TomcatMetricsAutoConfiguration",
            "simpleMeterRegistry"
          ]
        },

"org.springdoc.core.configuration.SpringDocConfiguration$Spring
DocSpringDataWebPropertiesProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocConfiguration$Spring
DocSpringDataWebPropertiesProvider",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$DataSourcePoolMetadataMetr
icsConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$DataSourcePoolMetadataMetr
icsConfiguration",
          "dependencies": []
        },
```

```
"org.springframework.boot.actuate.autoconfigure.endpoint.web.Se
rvletEndpointManagementContextConfiguration$WebMvcServletEndpoi
ntManagementContextConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.Se
rvletEndpointManagementContextConfiguration$WebMvcServletEndpoi
ntManagementContextConfiguration",
          "dependencies": []
        },

"org.springframework.data.jpa.repository.support.JpaEvaluationC
ontextExtension": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.jpa.repository.support.JpaEvaluationC
ontextExtension",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.context.LifecycleAutoCo
nfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.context.LifecycleAutoCo
nfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.logging.LogFile
WebEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.logging.LogFile
WebEndpointAutoConfiguration",
          "dependencies": []
        },
        "spring.info-
org.springframework.boot.autoconfigure.info.ProjectInfoProperti
es": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.info.ProjectInfoPropert
ies",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceTransact
ionManagerAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceTransact
ionManagerAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
EndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
EndpointAutoConfiguration",
          "dependencies": []
        },
        "infoEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.info.InfoEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/info/InfoEndpoi
ntAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.info.InfoEndpoi
ntAutoConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.observation.web
.servlet.WebMvcObservationAutoConfiguration$MeterFilterConfigur
ation": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web
.servlet.WebMvcObservationAutoConfiguration$MeterFilterConfigur
ation",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceTransact
ionManagerAutoConfiguration$JdbcTransactionManagerConfiguration
": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceTransact
ionManagerAutoConfiguration$JdbcTransactionManagerConfiguration
",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConf
iguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConf
iguration",
          "dependencies": []
        },
        "lettuceMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.redis.L
ettuceMetricsAutoConfiguration$$Lambda/0x00007ff40055ba28",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/redis/L
ettuceMetricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.redis.L
ettuceMetricsAutoConfiguration",
            "simpleMeterRegistry",
            "micrometerOptions"
          ]
```

```
        },

"metricsRepositoryMethodInvocationListenerBeanPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.data.Me
tricsRepositoryMethodInvocationListenerBeanPostProcessor",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/data/Re
positoryMetricsAutoConfiguration.class]",
          "dependencies": []
        },

"org.springdoc.core.configuration.SpringDocConfiguration$Queryd
slProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocConfiguration$Queryd
slProvider",
          "dependencies": []
        },
        "redisCacheMeterBinderProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.cache.RedisCacheMeter
BinderProvider",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/cache/C
acheMeterBinderProvidersConfiguration$RedisCacheMeterBinderProv
iderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMeterBinderProvidersConfiguration$RedisCacheMeterBinderProv
iderConfiguration"
          ]
        },
        "classLoaderMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.jvm.ClassLoaderMetrics",
```

```
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },
        "observationRestTemplateCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.web.client.Observatio
nRestTemplateCustomizer",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/web
/client/RestTemplateObservationConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.RestTemplateObservationConfiguration",
            "observationRegistry",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties"
          ]
        },
        "knife4j-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jPropert
ies": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.github.xiaoymin.knife4j.spring.configuration.Knife4jProper
ties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.context.ConfigurationPr
opertiesAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.context.ConfigurationPr
```

```
opertiesAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.ssl.SslAutoConfiguratio
n": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.ssl.SslAutoConfiguratio
n",
          "dependencies": [
            "spring.ssl-
org.springframework.boot.autoconfigure.ssl.SslProperties"
          ]
        },

"org.springframework.boot.autoconfigure.internalCachingMetadata
ReaderFactory": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.core.type.classreading.CachingMetadataRead
erFactory",
          "dependencies": []
        },
        "knife4jJakartaOperationCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.github.xiaoymin.knife4j.spring.extension.Knife4jJakartaOpe
rationCustomizer",
          "resource": "class path resource
[com/github/xiaoymin/knife4j/spring/configuration/Knife4jAutoCo
nfiguration.class]",
          "dependencies": [

"com.github.xiaoymin.knife4j.spring.configuration.Knife4jAutoCo
nfiguration"
          ]
        },
        "fileWatcher": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.autoconfigure.ssl.FileWatcher",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/ssl/SslAutoConfiguratio
n.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.ssl.SslAutoConfiguratio
n"
          ]
        },

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$ParameterNamesModuleConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$ParameterNamesModuleConfiguration",
          "dependencies": []
        },
        "sortResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.web.SortHandlerMethodArgumentResolver
",
          "resource": "class path resource
[org/springframework/data/web/config/SpringDataWebConfiguration
.class]",
          "dependencies": [

"org.springframework.data.web.config.SpringDataWebConfiguration
"
          ]
        },

"org.springframework.boot.autoconfigure.sql.init.DataSourceInit
ializationConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.sql.init.DataSourceInit
ializationConfiguration",
          "dependencies": []
        },
```

```
        "baiduService": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.service.BaiduService$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/BaiduService.class]",
          "dependencies": [
            "sessionStatusService",
            "psychoanalysisService"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.audit.AuditEven
tsEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.audit.AuditEven
tsEndpointAutoConfiguration",
          "dependencies": []
        },
        "errorAttributes": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.servlet.error.DefaultErrorAttribu
tes",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration"
          ]
        },
        "observationWebClientCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.web.reactive.client.O
bservationWebClientCustomizer",
          "resource": "class path resource
```

```
[org/springframework/boot/actuate/autoconfigure/observation/web
/client/WebClientObservationConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.WebClientObservationConfiguration",
            "observationRegistry",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties",
            "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties"
          ]
        },
        "beanNameHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.handler.BeanNameUrlHandlerMapp
ing",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMeterBinderProvidersConfiguration$RedisCacheMeterBinderProv
```

```
iderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMeterBinderProvidersConfiguration$RedisCacheMeterBinderProv
iderConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.info.ProjectInfoAutoCon
figuration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.info.ProjectInfoAutoCon
figuration",
          "dependencies": [
            "spring.info-
org.springframework.boot.autoconfigure.info.ProjectInfoPropertie
s"
          ]
        },
        "psychoanalysisService": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.service.PsychoanalysisService$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/PsychoanalysisService.class]
",
          "dependencies": [
            "psychoanalysisRepository",
            "chatGPT"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryConfiguration$EmbeddedTomcat": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryConfiguration$EmbeddedTomcat",
```

```
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.management.Heap
DumpWebEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.management.Heap
DumpWebEndpointAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceConfigur
ation$Hikari": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceConfigur
ation$Hikari",
          "dependencies": []
        },
        "springDataWebPropertiesProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.providers.SpringDataWebPropertiesProvider",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration$Spring
DocSpringDataWebPropertiesProvider.class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration$Spring
DocSpringDataWebPropertiesProvider"
          ]
        },
        "responseBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.service.GenericResponseService",
          "resource": "class path resource
[org/springdoc/webmvc/core/configuration/SpringDocWebMvcConfigu
ration.class]",
          "dependencies": [
```

```
"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration",
            "operationBuilder",
            "genericReturnTypeParser",
            "kotlinCoroutinesReturnTypeParser",

"org.springdoc.core.properties.SpringDocConfigProperties",
            "propertyResolverUtils"
          ]
        },

"org.springframework.data.web.config.SpringDataJacksonConfigura
tion": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.web.config.SpringDataJacksonConfigura
tion",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMetricsRegistrarConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMetricsRegistrarConfiguration",
          "dependencies": [
            "simpleMeterRegistry",
            "redisCacheMeterBinderProvider",
            "cacheManager"
          ]
        },
        "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
Properties",
          "dependencies": []
        },
```

```
        "swaggerWelcome": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.ui.SwaggerWelcomeWebMvc",
          "resource": "class path resource
[org/springdoc/webmvc/ui/SwaggerConfig.class]",
          "dependencies": [
            "org.springdoc.webmvc.ui.SwaggerConfig",

"org.springdoc.core.properties.SwaggerUiConfigProperties",

"org.springdoc.core.properties.SpringDocConfigProperties",

"org.springdoc.core.properties.SwaggerUiConfigParameters",
            "springWebProvider"
          ]
        },
        "flashMapManager": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.support.SessionFlashMapManager
",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$TaskSchedulerBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$TaskSchedulerBuilderConfiguration",
          "dependencies": []
        },
        "pictureRepository": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type": "cn.netkiller.repository.PictureRepository",
          "resource":
"cn.netkiller.repository.PictureRepository defined in
@EnableJpaRepositories declared on Application",
          "dependencies": [
            "jpa.named-queries#2",
            "jpa.PictureRepository.fragments#0",
            "jpaSharedEM_entityManagerFactory",
            "jpaMappingContext"
          ]
        },
        "requestBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.core.service.RequestService",
          "resource": "class path resource
[org/springdoc/webmvc/core/configuration/SpringDocWebMvcConfigu
ration.class]",
          "dependencies": [

"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration",
            "parameterBuilder",
            "requestBodyBuilder",
            "operationBuilder",
            "localSpringDocParameterNameDiscoverer"
          ]
        },
        "redisHealthContributor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.data.redis.RedisReactiveHealt
hIndicator",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/data/redis/Redi
sReactiveHealthContributorAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.data.redis.Redi
sReactiveHealthContributorAutoConfiguration"
          ]
        },
```

```
"org.springframework.boot.autoconfigure.netty.NettyAutoConfigur
ation": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.netty.NettyAutoConfigur
ation",
          "dependencies": [
            "spring.netty-
org.springframework.boot.autoconfigure.netty.NettyProperties"
          ]
        },
        "healthHttpCodeStatusMapper": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.health.SimpleHttpCodeStatusMa
pper",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",
            "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties"
          ]
        },
        "psychoanalysisRepository": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.repository.PsychoanalysisRepository",
          "resource":
"cn.netkiller.repository.PsychoanalysisRepository defined in
@EnableJpaRepositories declared on Application",
          "dependencies": [
            "jpa.named-queries#4",
            "jpa.PsychoanalysisRepository.fragments#0",
            "jpaSharedEM_entityManagerFactory",
            "jpaMappingContext"
          ]
```

```
        },
        "defaultCodecCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$DefaultCodecsConfiguration$$Lambda/0x00007ff400a34e
48",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/http/codec/CodecsAutoCo
nfiguration$DefaultCodecsConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$DefaultCodecsConfiguration",
            "spring.codec-
org.springframework.boot.autoconfigure.codec.CodecProperties"
          ]
        },
        "statusController": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.controller.StatusController",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/StatusController.class]",
          "dependencies": [
            "sessionStatusRepository"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration",
          "dependencies": []
        },
        "metricsRepositoryMethodInvocationListener": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.data.MetricsRepositor
```

```
yMethodInvocationListener",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/data/Re
positoryMetricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.data.Re
positoryMetricsAutoConfiguration",
            "repositoryTagsProvider"
          ]
        },
        "uptimeMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.system.UptimeMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/SystemM
etricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.SystemM
etricsAutoConfiguration"
          ]
        },
        "controllerExposeExcludePropertyEndpointFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.expose
.IncludeExcludeEndpointFilter",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration"
          ]
        },
        "pathMappedEndpoints": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.PathMappedEndpoi
```

```
nts",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration",
            "servletEndpointDiscoverer",
            "webEndpointDiscoverer",
            "controllerEndpointDiscoverer"
          ]
        },
        "jvmThreadMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.jvm.JvmThreadMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },
        "scheduledTasksEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.scheduling.ScheduledTasksEndp
oint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/scheduling/Sche
duledTasksEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksEndpointAutoConfiguration"
          ]
        },
        "indexPageTransformer": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springdoc.webmvc.ui.SwaggerIndexPageTransformer",
          "resource": "class path resource
[org/springdoc/webmvc/ui/SwaggerConfig.class]",
          "dependencies": [
            "org.springdoc.webmvc.ui.SwaggerConfig",

"org.springdoc.core.properties.SwaggerUiConfigProperties",

"org.springdoc.core.properties.SwaggerUiOAuthProperties",

"org.springdoc.core.properties.SwaggerUiConfigParameters",
            "swaggerWelcome",
            "springdocObjectMapperProvider"
          ]
        },

"org.springframework.data.web.config.ProjectingArgumentResolver
Registrar": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.web.config.ProjectingArgumentResolver
Registrar",
          "dependencies": []
        },
        "heapDumpWebEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.management.HeapDumpWebEndpoin
t",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/management/Heap
DumpWebEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.management.Heap
DumpWebEndpointAutoConfiguration"
          ]
        },
        "managementServletContext": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.actuate.autoconfigure.web.servlet.Ser
vletManagementContextAutoConfiguration$$Lambda/0x00007ff400ae9a
58",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/web/servlet/Ser
vletManagementContextAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.web.servlet.Ser
vletManagementContextAutoConfiguration",
            "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties"
          ]
        },
        "spring.gson-
org.springframework.boot.autoconfigure.gson.GsonProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.gson.GsonProperties",
          "dependencies": []
        },
        "fileDescriptorMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.system.FileDescriptorMetr
ics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/SystemM
etricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.SystemM
etricsAutoConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.aop.AopAutoConfiguratio
n$AspectJAutoProxyingConfiguration$CglibAutoProxyConfiguration"
: {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.autoconfigure.aop.AopAutoConfiguratio
n$AspectJAutoProxyingConfiguration$CglibAutoProxyConfiguration"
,
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.health.Reactive
HealthEndpointConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.Reactive
HealthEndpointConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMeterBinderProvidersConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.cache.C
acheMeterBinderProvidersConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$MeterObservationHandlerConfiguration$
OnlyMetricsMeterObservationHandlerConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$MeterObservationHandlerConfiguration$
OnlyMetricsMeterObservationHandlerConfiguration",
          "dependencies": []
        },
        "environmentEndpointWebExtension": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.env.EnvironmentEndpointWebExt
ension",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/env/Environment
```

```
EndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.env.Environment
EndpointAutoConfiguration",
            "environmentEndpoint",
            "management.endpoint.env-
org.springframework.boot.actuate.autoconfigure.env.EnvironmentE
ndpointProperties"
          ]
        },
        "restClientBuilderConfigurer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.client.RestClientBu
ilderConfigurer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/client/RestClientAu
toConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.client.RestClientAu
toConfiguration"
          ]
        },
        "mvcValidator": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.validation.ValidatorAda
pter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
        "conditionsReportEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.actuate.autoconfigure.condition.Condi
tionsReportEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/condition/Condi
tionsReportEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.condition.Condi
tionsReportEndpointAutoConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "applicationAvailability": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.availability.ApplicationAvailabilityB
ean",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/availability/Applicatio
nAvailabilityAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.availability.Applicatio
nAvailabilityAutoConfiguration"
          ]
        },
        "org.springdoc.webmvc.ui.SwaggerConfig": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springdoc.webmvc.ui.SwaggerConfig",
          "dependencies": []
        },
        "mvcResourceUrlProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.resource.ResourceUrlProvider",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementContextAutoConfiguration$SameManagementContextConfigurati
on$EnableSameManagementContextConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementContextAutoConfiguration$SameManagementContextConfigurati
on$EnableSameManagementContextConfiguration",
          "dependencies": []
        },
        "reactorClientHttpConnectorFactory": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.ReactorClientHttpConnectorFactory",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/reactive/function/c
lient/ClientHttpConnectorFactoryConfiguration$ReactorNetty.clas
s]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.ClientHttpConnectorFactoryConfiguration$ReactorNetty",
            "reactorResourceFactory"
          ]
        },

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$SimpleAsyncTaskExecutorBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$SimpleAsyncTaskExecutorBuilderConfiguration",
          "dependencies": [
            "spring.task.execution-
org.springframework.boot.autoconfigure.task.TaskExecutionProper
ties"
```

```
          ]
        },
        "servletEndpointRegistrar": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.ServletEndpointR
egistrar",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/Se
rvletEndpointManagementContextConfiguration$WebMvcServletEndpoi
ntManagementContextConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.Se
rvletEndpointManagementContextConfiguration$WebMvcServletEndpoi
ntManagementContextConfiguration",
            "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties",
            "servletEndpointDiscoverer",
            "dispatcherServletRegistration"
          ]
        },
        "dumpEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.management.ThreadDumpEndpoint
",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/management/Thre
adDumpEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.management.Thre
adDumpEndpointAutoConfiguration"
          ]
        },
        "springdocObjectMapperProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.providers.ObjectMapperProvider",
          "resource": "class path resource
```

```
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",

"org.springdoc.core.properties.SpringDocConfigProperties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.endpoint.web.Se
rvletEndpointManagementContextConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.Se
rvletEndpointManagementContextConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.data.redis.RedisReactiv
eAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.redis.RedisReactiv
eAutoConfiguration",
          "dependencies": []
        },
        "knife4j.basic-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jHttpBas
ic": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.github.xiaoymin.knife4j.spring.configuration.Knife4jHttpBa
sic",
          "dependencies": []
        },
        "hikariPoolDataSourceMetadataProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.metadata.DataSourc
ePoolMetadataProvidersConfiguration$HikariPoolDataSourceMetadat
```

```
aProviderConfiguration$$Lambda/0x00007ff4005b7c00",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/metadata/DataSourc
ePoolMetadataProvidersConfiguration$HikariPoolDataSourceMetadat
aProviderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jdbc.metadata.DataSourc
ePoolMetadataProvidersConfiguration$HikariPoolDataSourceMetadat
aProviderConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$SimpleAsyncTaskSchedulerBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$SimpleAsyncTaskSchedulerBuilderConfiguration",
          "dependencies": [
            "spring.task.scheduling-
org.springframework.boot.autoconfigure.task.TaskSchedulingPrope
rties"
          ]
        },

"org.springframework.boot.autoconfigure.cache.CacheAutoConfigur
ation$CacheManagerEntityManagerFactoryDependsOnPostProcessor":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.cache.CacheAutoConfigur
ation$CacheManagerEntityManagerFactoryDependsOnPostProcessor",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$TaskExecutorBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$TaskExecutorBuilderConfiguration",
```

```
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.context.PropertyPlaceho
lderAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.context.PropertyPlaceho
lderAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$MeterObservationHandlerConfiguration"
: {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$MeterObservationHandlerConfiguration"
,
          "dependencies": []
        },
        "nettyWebServerFactoryCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.embedded.NettyWebSe
rverFactoryCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/embedded/EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$NettyWebServerFactory
CustomizerConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$NettyWebServerFactory
CustomizerConfiguration",
            "environment",
            "server-
org.springframework.boot.autoconfigure.web.ServerProperties"
          ]
        },
```

```
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter",
          "dependencies": [
            "spring.web-
org.springframework.boot.autoconfigure.web.WebProperties",
            "spring.mvc-
org.springframework.boot.autoconfigure.web.servlet.WebMvcProper
ties",

"org.springframework.beans.factory.support.DefaultListableBeanF
actory@37efd131"
          ]
        },

"org.springframework.data.web.config.SpringDataWebConfiguration
": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.web.config.SpringDataWebConfiguration
",
          "dependencies": [

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "errorPageRegistrarBeanPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.server.ErrorPageRegistrarBeanPost
Processor",
          "dependencies": []
        },
        "mvcConversionService": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.format.WebConversio
```

```
nService",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
        "redisTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.RedisTemplate",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/RedisAutoCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.RedisAutoCon
figuration",
            "redisConnectionFactory"
          ]
        },
        "memberService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.service.MemberService",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/MemberService.class]",
          "dependencies": []
        },
        "spring.codec-
org.springframework.boot.autoconfigure.codec.CodecProperties":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.codec.CodecProperties",
          "dependencies": []
        },
        "controllerEndpointDiscoverer": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.annotation.Contr
ollerEndpointDiscoverer",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration"
          ]
        },
        "diskSpaceHealthIndicator": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.system.DiskSpaceHealthIndicat
or",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/system/DiskSpac
eHealthContributorAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.system.DiskSpac
eHealthContributorAutoConfiguration",
            "management.health.diskspace-
org.springframework.boot.actuate.autoconfigure.system.DiskSpace
HealthIndicatorProperties"
          ]
        },
        "jvmMemoryMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.jvm.JvmMemoryMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },
```

```
        "storyService": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.service.StoryService$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/StoryService.class]",
          "dependencies": [
            "pictureRepository",
            "mqttService",
            "baiduService",
            "sessionStatusService",
            "pictureService",
            "psychoanalysisService",
            "stableDiffusionService",
            "audioService",
            "warningService",
            "chatService",
            "chatGPT",
            "aliyunService"
          ]
        },
        "modelConverterRegistrar": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.ModelConverterRegistrar",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",

"org.springdoc.core.properties.SpringDocConfigProperties"
          ]
        },
        "restClientSsl": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.client.AutoConfigur
edRestClientSsl",
          "resource": "class path resource
```

```
[org/springframework/boot/autoconfigure/web/client/RestClientAu
toConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.client.RestClientAu
toConfiguration",
            "sslBundleRegistry"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.condition.Condi
tionsReportEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.condition.Condi
tionsReportEndpointAutoConfiguration",
          "dependencies": []
        },

"com.github.xiaoymin.knife4j.spring.configuration.Knife4jAutoCo
nfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.github.xiaoymin.knife4j.spring.configuration.Knife4jAutoCo
nfiguration$$SpringCGLIB$$0",
          "dependencies": [
            "knife4j-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jPropert
ies",
            "environment"
          ]
        },
        "logFileWebEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.logging.LogFileWebEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/logging/LogFile
WebEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.logging.LogFile
```

```
WebEndpointAutoConfiguration",
            "management.endpoint.logfile-
org.springframework.boot.actuate.autoconfigure.logging.LogFileW
ebEndpointProperties"
          ]
        },
        "mvcPathMatcher": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.util.AntPathMatcher",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.redis.L
ettuceMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.redis.L
ettuceMetricsAutoConfiguration",
          "dependencies": []
        },
        "handlerExceptionResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.handler.HandlerExceptionResolv
erComposite",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcContentNegotiationManager"
          ]
        },
```

```
        "routerFunctionProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.core.providers.RouterFunctionWebMvcProvid
er",
          "resource": "class path resource
[org/springdoc/webmvc/core/configuration/SpringDocWebMvcConfigu
ration$SpringDocWebMvcRouterConfiguration.class]",
          "dependencies": [

"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration$SpringDocWebMvcRouterConfiguration"
          ]
        },
        "redisKeyValueTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.RedisKeyValueTemplate",
          "dependencies": [
            "redisKeyValueAdapter",
            "keyValueMappingContext"
          ]
        },
        "basicErrorController": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Basic
ErrorController",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration",
            "errorAttributes"
          ]
        },
        "pingHealthContributor": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.actuate.health.PingHealthIndicator",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthCo
ntributorAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthCo
ntributorAutoConfiguration"
          ]
        },

"org.springdoc.core.configuration.SpringDocConfiguration$Spring
DocWebFluxSupportConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocConfiguration$Spring
DocWebFluxSupportConfiguration",
          "dependencies": []
        },
        "mappingsEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.web.mappings.MappingsEndpoint
",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/web/mappings/Ma
ppingsEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "sortOpenAPIConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.SortOpenAPIConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocSortConfiguration.cl
```

```
ass]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocSortConfiguration",
            "springdocObjectMapperProvider"
          ]
        },
        "healthEndpointGroupMembershipValidator": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration$HealthEndpointGroupMembershipValidator",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",
            "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties",
            "healthContributorRegistry"
          ]
        },

"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$JacksonCodecConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$JacksonCodecConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.cache.CacheAutoConfigur
ation": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.cache.CacheAutoConfigur
ation",
          "dependencies": []
```

```
        },

"org.springframework.boot.autoconfigure.http.GsonHttpMessageCon
vertersConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.GsonHttpMessageCon
vertersConfiguration",
          "dependencies": []
        },
        "spring.datasource-
org.springframework.boot.autoconfigure.jdbc.DataSourcePropertie
s": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceProperti
es",
          "dependencies": []
        },
        "namedParameterJdbcTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.jdbc.core.namedparam.NamedParameterJdbcTem
plate",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/NamedParameterJdbc
TemplateConfiguration.class]",
          "dependencies": [
            "dataSourceScriptDatabaseInitializer",

"org.springframework.boot.autoconfigure.jdbc.NamedParameterJdbc
TemplateConfiguration",
            "jdbcTemplate"
          ]
        },
        "simpleAsyncTaskExecutorBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.task.SimpleAsyncTaskExecutorBuilder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/task/TaskExecutorConfig
```

```
urations$SimpleAsyncTaskExecutorBuilderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$SimpleAsyncTaskExecutorBuilderConfiguration"
          ]
        },
        "spring.web-
org.springframework.boot.autoconfigure.web.WebProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.WebProperties",
          "dependencies": []
        },
        "mvcViewResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.view.ViewResolverComposite",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcContentNegotiationManager"
          ]
        },
        "simpleConfig": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.export.
simple.SimplePropertiesConfigAdapter",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/export/
simple/SimpleMetricsExportAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.export.
simple.SimpleMetricsExportAutoConfiguration",
            "management.simple.metrics.export-
org.springframework.boot.actuate.autoconfigure.metrics.export.s
```

```
imple.SimpleProperties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.RestClientObservationConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web
.client.RestClientObservationConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.web.servlet.Ser
vletManagementContextAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.servlet.Ser
vletManagementContextAutoConfiguration",
          "dependencies": []
        },
        "observationRegistry": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.observation.SimpleObservationRegistry",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/Obs
ervationAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration"
          ]
        },
        "mvcUriComponentsContributor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.method.support.CompositeUriComponentsC
ontributor",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
```

```
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcConversionService",
            "requestMappingHandlerAdapter"
          ]
        },
        "webClientBuilder": {
          "aliases": [],
          "scope": "prototype",
          "type":
"org.springframework.web.reactive.function.client.WebClient$Bui
lder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/reactive/function/c
lient/WebClientAutoConfiguration.class]",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementContextAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementContextAutoConfiguration",
          "dependencies": []
        },
        "endpointObjectMapper": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.jackso
n.JacksonEndpointAutoConfiguration$$Lambda/0x00007ff400a51c30",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/jackso
n/JacksonEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.jackso
n.JacksonEndpointAutoConfiguration"
          ]
        },
```

```
        "jpaSharedEM_entityManagerFactory": {
          "aliases": [],
          "scope": "singleton",
          "type": "jdk.proxy2.$Proxy155",
          "dependencies": [
            "entityManagerFactory"
          ]
        },
        "application": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.Application$$SpringCGLIB$$0",
          "dependencies": []
        },
        "httpMessageConvertersRestClientCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.client.HttpMessageC
onvertersRestClientCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/client/RestClientAu
toConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.client.RestClientAu
toConfiguration"
          ]
        },
        "redisMappingConfiguration#0": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.convert.MappingConfigurati
on",
          "dependencies": [
            "redisIndexConfiguration#0",
            "redisKeyspaceConfiguration#0"
          ]
        },
        "taskExecutorBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.task.TaskExecutorBuilder",
```

```
          "resource": "class path resource
[org/springframework/boot/autoconfigure/task/TaskExecutorConfig
urations$TaskExecutorBuilderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$TaskExecutorBuilderConfiguration",
            "spring.task.execution-
org.springframework.boot.autoconfigure.task.TaskExecutionProper
ties"
          ]
        },

"org.springframework.boot.autoconfigure.jdbc.JdbcTemplateConfig
uration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.JdbcTemplateConfig
uration",
          "dependencies": []
        },
        "queryDslQuerydslPredicateOperationCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.customizers.QuerydslPredicateOperationCusto
mizer",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration$Queryd
slProvider.class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration$Queryd
slProvider",

"org.springdoc.core.properties.SpringDocConfigProperties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.endpoint.jackso
n.JacksonEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.actuate.autoconfigure.endpoint.jackso
n.JacksonEndpointAutoConfiguration",
          "dependencies": []
        },
        "multipleOpenApiResource": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.api.MultipleOpenApiWebMvcResource",
          "resource": "class path resource
[org/springdoc/webmvc/core/configuration/MultipleOpenApiSupport
Configuration.class]",
          "dependencies": [

"org.springdoc.webmvc.core.configuration.MultipleOpenApiSupport
Configuration",
            "adminApi",
            "requestBuilder",
            "responseBuilder",
            "operationBuilder",

"org.springdoc.core.properties.SpringDocConfigProperties",
            "springDocProviders",
            "springDocCustomizers"
          ]
        },
        "platformTransactionManagerCustomizers": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.Transaction
ManagerCustomizers",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/transaction/Transaction
ManagerCustomizationAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.transaction.Transaction
ManagerCustomizationAutoConfiguration"
          ]
        },
        "requestBodyBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springdoc.core.service.RequestBodyService",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "parameterBuilder"
          ]
        },
        "endpointCachingOperationInvokerAdvisor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.invoker.cache.Cachin
gOperationInvokerAdvisor",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/Endpoi
ntAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.Endpoi
ntAutoConfiguration",
            "environment"
          ]
        },
        "defaultServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
        "swaggerWebMvcConfigurer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.ui.SwaggerWebMvcConfigurer",
```

```
          "resource": "class path resource
[org/springdoc/webmvc/ui/SwaggerConfig.class]",
          "dependencies": [
            "org.springdoc.webmvc.ui.SwaggerConfig",

"org.springdoc.core.properties.SwaggerUiConfigParameters",
            "indexPageTransformer",
            "swaggerResourceResolver"
          ]
        },
        "persistenceExceptionTranslationPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.dao.annotation.PersistenceExceptionTransla
tionPostProcessor",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/dao/PersistenceExceptio
nTranslationAutoConfiguration.class]",
          "dependencies": [
            "environment"
          ]
        },
        "management.health.db-
org.springframework.boot.actuate.autoconfigure.jdbc.DataSourceH
ealthIndicatorProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.jdbc.DataSource
HealthIndicatorProperties",
          "dependencies": []
        },
        "observationRegistryPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationRegistryPostProcessor",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/Obs
ervationAutoConfiguration.class]",
          "dependencies": []
        },
        "spring.data.web-
```

```
org.springframework.boot.autoconfigure.data.web.SpringDataWebPr
operties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.web.SpringDataWebP
roperties",
          "dependencies": []
        },
        "characterEncodingFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.servlet.filter.OrderedCharacterEn
codingFilter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/HttpEncodin
gAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.HttpEncodin
gAutoConfiguration"
          ]
        },
        "sortCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.web.SpringDataWebA
utoConfiguration$$Lambda/0x00007ff400a7b1f8",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/web/SpringDataWebA
utoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.web.SpringDataWebA
utoConfiguration"
          ]
        },
        "webEndpointDiscoverer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.annotation.WebEn
dpointDiscoverer",
```

```
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration",
            "endpointOperationParameterMapper",
            "endpointMediaTypes"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration$DispatcherServletRegistrationConfigurat
ion": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration$DispatcherServletRegistrationConfigurat
ion",
          "dependencies": []
        },
        "preserveErrorControllerTargetClassPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$PreserveErrorControllerTargetClassPostProc
essor",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration.class]",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonObjectMapperBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonObjectMapperBuilderConfiguration",
          "dependencies": []
        },
```

```
"org.springframework.boot.autoconfigure.web.reactive.function.client.ClientHttpConnectorFactoryConfiguration$ReactorNetty": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.client.ClientHttpConnectorFactoryConfiguration$ReactorNetty",
          "dependencies": []
        },
        "openAPIBuilder": {
          "aliases": [],
          "scope": "prototype",
          "type": "org.springdoc.core.service.OpenAPIService",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "securityParser",

"org.springdoc.core.properties.SpringDocConfigProperties",
            "propertyResolverUtils"
          ]
        },
        "logbackMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.logging.LogbackMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/LogbackMetricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.LogbackMetricsAutoConfiguration"
          ]
        },
        "management.info-
org.springframework.boot.actuate.autoconfigure.info.InfoContributorProperties": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.boot.actuate.autoconfigure.info.InfoContri
butorProperties",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.data.Re
positoryMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.data.Re
positoryMetricsAutoConfiguration",
          "dependencies": [
            "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties"
          ]
        },
        "propertySourcesPlaceholderConfigurer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.context.support.PropertySourcesPlaceholder
Configurer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/context/PropertyPlaceho
lderAutoConfiguration.class]",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.transaction.Transaction
ManagerCustomizationAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.Transaction
ManagerCustomizationAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.transaction.jta.JtaAuto
Configuration": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.boot.autoconfigure.transaction.jta.JtaAuto
Configuration",
          "dependencies": []
        },
        "management.endpoints.web.cors-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Cor
sEndpointProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.Co
rsEndpointProperties",
          "dependencies": []
        },
        "stringHttpMessageConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.http.converter.StringHttpMessageConverter"
,
          "resource": "class path resource
[org/springframework/boot/autoconfigure/http/HttpMessageConvert
ersAutoConfiguration$StringHttpMessageConverterConfiguration.cl
ass]",
          "dependencies": [

"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ersAutoConfiguration$StringHttpMessageConverterConfiguration",
            "environment"
          ]
        },

"org.springframework.boot.autoconfigure.data.redis.RedisReposit
oriesAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.redis.RedisReposit
oriesAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$TomcatWebServerFactor
```

```
yCustomizerConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$TomcatWebServerFactor
yCustomizerConfiguration",
          "dependencies": []
        },
        "chatGPT": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.component.ChatGPT$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/component/ChatGPT.class]",
          "dependencies": []
        },
        "jsonComponentModule": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.jackson.JsonComponentModule",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConf
iguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration"
          ]
        },

"org.springdoc.core.configuration.SpringDocUIConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocUIConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration$SpringM
vcConfiguration": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration$SpringM
vcConfiguration",
          "dependencies": []
        },
        "entityManagerFactoryBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.orm.jpa.EntityManagerFactoryBuilder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaCon
figuration",
            "jpaVendorAdapter"
          ]
        },

"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration",
          "dependencies": []
        },
        "audioService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.component.AudioService",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/component/AudioService.class]",
          "dependencies": []
        },
        "mappingJackson2HttpMessageConverter": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.http.converter.json.MappingJackson2HttpMes
sageConverter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/http/JacksonHttpMessage
ConvertersConfiguration$MappingJackson2HttpMessageConverterConf
iguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.http.JacksonHttpMessage
ConvertersConfiguration$MappingJackson2HttpMessageConverterConf
iguration",
            "jacksonObjectMapper"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.env.Environment
EndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.env.Environment
EndpointAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceJmxConfi
guration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceJmxConfi
guration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.data.redis.Redi
sReactiveHealthContributorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.data.redis.Redi
sReactiveHealthContributorAutoConfiguration",
          "dependencies": [
            "redisConnectionFactory"
```

```
          ]
        },
        "stableDiffusionService": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.service.StableDiffusionService$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/StableDiffusionService.class
]",
          "dependencies": [
            "aliyunService",
            "pictureService",
            "sessionStatusService",
            "mqttService"
          ]
        },
        "spring.reactor.netty-
org.springframework.boot.autoconfigure.reactor.netty.ReactorNet
tyProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.reactor.netty.ReactorNe
ttyProperties",
          "dependencies": []
        },
        "healthStatusAggregator": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.health.SimpleStatusAggregator
",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",
            "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties"
          ]
```

```
        },

"org.springframework.boot.actuate.autoconfigure.metrics.SystemM
etricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.SystemM
etricsAutoConfiguration",
          "dependencies": []
        },
        "management.server-
org.springframework.boot.actuate.autoconfigure.web.server.Manag
ementServerProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementServerProperties",
          "dependencies": []
        },
        "servletWebServerFactoryCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/ServletWebS
erverFactoryAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryAutoConfiguration",
            "server-
org.springframework.boot.autoconfigure.web.ServerProperties"
          ]
        },
        "mvcUrlPathHelper": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.util.UrlPathHelper",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
```

```
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
        "transactionTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.transaction.support.TransactionTemplate",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/transaction/Transaction
AutoConfiguration$TransactionTemplateConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.transaction.Transaction
AutoConfiguration$TransactionTemplateConfiguration",
            "transactionManager"
          ]
        },
        "jpa.named-queries#4": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.repository.core.support.PropertiesBas
edNamedQueries",
          "dependencies": []
        },
        "nullableKotlinRequestParameterCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocKotlinConfiguration$
$Lambda/0x00007ff400aa6460",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocKotlinConfiguration.
class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocKotlinConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.startup
```

```
.StartupTimeMetricsListenerAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.startup
.StartupTimeMetricsListenerAutoConfiguration",
          "dependencies": []
        },
        "servletMappingDescriptionProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.web.mappings.servlet.Servlets
MappingDescriptionProvider",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/web/mappings/Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration.class]"
,
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$ThreadPoolTaskSchedulerBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$ThreadPoolTaskSchedulerBuilderConfiguration",
          "dependencies": []
        },

"org.springdoc.core.configuration.SpringDocConfiguration$WebCon
versionServiceConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocConfiguration$WebCon
versionServiceConfiguration",
          "dependencies": []
        },
```

```
"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration$SpringDocWebMvcActuatorConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration$SpringDocWebMvcActuatorConfiguration",
          "dependencies": []
        },
        "stringRedisTemplate": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.StringRedisTemplate",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/RedisAutoCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.RedisAutoCon
figuration",
            "redisConnectionFactory"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksObservabilityAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksObservabilityAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration",
          "dependencies": []
        },
        "spring.sql.init-
```

```
org.springframework.boot.autoconfigure.sql.init.SqlInitializati
onProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.sql.init.SqlInitializat
ionProperties",
          "dependencies": []
        },
        "reactiveHealthContributorRegistry": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.AutoConf
iguredReactiveHealthContributorRegistry",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/Reactive
HealthEndpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.Reactive
HealthEndpointConfiguration",
            "diskSpaceHealthIndicator",
            "pingHealthContributor",
            "dbHealthContributor",
            "redisHealthContributor",
            "healthEndpointGroups"
          ]
        },
        "adminApi": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springdoc.core.models.GroupedOpenApi",
          "resource": "class path resource
[cn/netkiller/config/Knife4jConfiguration.class]",
          "dependencies": [
            "knife4jConfiguration"
          ]
        },
        "jpa.named-queries#1": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.repository.core.support.PropertiesBas
edNamedQueries",
```

```
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration",
          "dependencies": []
        },
        "jpa.named-queries#0": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.repository.core.support.PropertiesBas
edNamedQueries",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.reactor.ReactorAutoConf
iguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.reactor.ReactorAutoConf
iguration",
          "dependencies": [
            "spring.reactor-
org.springframework.boot.autoconfigure.reactor.ReactorPropertie
s"
          ]
        },
        "jpa.named-queries#3": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.repository.core.support.PropertiesBas
edNamedQueries",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$Jackson2ObjectMapperBuilderCustomizerConfiguration":
```

```
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$Jackson2ObjectMapperBuilderCustomizerConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ersAutoConfiguration$StringHttpMessageConverterConfiguration":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ersAutoConfiguration$StringHttpMessageConverterConfiguration",
          "dependencies": []
        },
        "jpa.named-queries#2": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.repository.core.support.PropertiesBas
edNamedQueries",
          "dependencies": []
        },
        "standardJacksonObjectMapperBuilderCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$Jackson2ObjectMapperBuilderCustomizerConfiguration$St
andardJackson2ObjectMapperBuilderCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConf
iguration$Jackson2ObjectMapperBuilderCustomizerConfiguration.cl
ass]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$Jackson2ObjectMapperBuilderCustomizerConfiguration",
            "spring.jackson-
org.springframework.boot.autoconfigure.jackson.JacksonPropertie
s"
```

```
          ]
        },
        "taskSchedulerBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.task.TaskSchedulerBuilder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/task/TaskSchedulingConf
igurations$TaskSchedulerBuilderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$TaskSchedulerBuilderConfiguration",
            "spring.task.scheduling-
org.springframework.boot.autoconfigure.task.TaskSchedulingPrope
rties"
          ]
        },

"org.springframework.boot.autoconfigure.task.TaskExecutionAutoC
onfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskExecutionAutoC
onfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration",
          "dependencies": []
        },
        "spring.data.redis-
org.springframework.boot.autoconfigure.data.redis.RedisProperti
es": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.autoconfigure.data.redis.RedisPropert
ies",
          "dependencies": []
        },
        "simpleMeterRegistry": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.simple.SimpleMeterRegistry",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/export/
simple/SimpleMetricsExportAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.export.
simple.SimpleMetricsExportAutoConfiguration",
            "simpleConfig",
            "micrometerClock"
          ]
        },

"org.springdoc.core.configuration.SpringDocPageableConfiguratio
n": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocPageableConfiguratio
n",
          "dependencies": []
        },
        "entityManagerFactory": {
          "aliases": [],
          "scope": "singleton",
          "type": "jdk.proxy2.$Proxy148",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaCon
figuration.class]",
          "dependencies": [
            "cacheManager",
            "dataSourceScriptDatabaseInitializer",

"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaCon
figuration",
            "entityManagerFactoryBuilder",
            "persistenceManagedTypes"
```

```
          ]
        },
        "webClientSsl": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.AutoConfiguredWebClientSsl",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/reactive/function/c
lient/WebClientAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientAutoConfiguration",
            "reactorClientHttpConnectorFactory",
            "sslBundleRegistry"
          ]
        },
        "startupTimeMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.startup.StartupTimeMe
tricsListener",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/startup
/StartupTimeMetricsListenerAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.startup
.StartupTimeMetricsListenerAutoConfiguration",
            "simpleMeterRegistry"
          ]
        },
        "observedAspect": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.observation.aop.ObservedAspect",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/Obs
ervationAutoConfiguration$ObservedAspectConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$ObservedAspectConfiguration",
            "observationRegistry"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.web.tom
cat.TomcatMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.web.tom
cat.TomcatMetricsAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.metadata.DataSourc
ePoolMetadataProvidersConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.metadata.DataSourc
ePoolMetadataProvidersConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.web
.servlet.WebMvcObservationAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web
.servlet.WebMvcObservationAutoConfiguration",
          "dependencies": []
        },
        "spring.mvc-
org.springframework.boot.autoconfigure.web.servlet.WebMvcProper
ties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.WebMvcPrope
rties",
          "dependencies": []
        },
```

```
        "knife4j.setting-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jSetting
": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.github.xiaoymin.knife4j.spring.configuration.Knife4jSettin
g",
          "dependencies": []
        },

"org.springdoc.core.configuration.SpringDocConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configuration.SpringDocConfiguration",
          "dependencies": []
        },
        "configurationPropertiesReportEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.context.properties.Configurat
ionPropertiesReportEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/context/propert
ies/ConfigurationPropertiesReportEndpointAutoConfiguration.clas
s]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.context.propert
ies.ConfigurationPropertiesReportEndpointAutoConfiguration",
            "management.endpoint.configprops-
org.springframework.boot.actuate.autoconfigure.context.properti
es.ConfigurationPropertiesReportEndpointProperties"
          ]
        },
        "multipartConfigElement": {
          "aliases": [],
          "scope": "singleton",
          "type": "jakarta.servlet.MultipartConfigElement",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/MultipartAu
toConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.autoconfigure.web.servlet.MultipartAu
toConfiguration"
          ]
        },
        "requestContextFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.servlet.filter.OrderedRequestCont
extFilter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$WebMvcAutoConfigurationAdapter.class]",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.orm.jpa.JpaBaseConfigur
ation$PersistenceManagedTypesConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.orm.jpa.JpaBaseConfigur
ation$PersistenceManagedTypesConfiguration",
          "dependencies": []
        },
        "mqttService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.utils.MqttService",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/utils/MqttService.class]",
          "dependencies": []
        },
        "operationBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.service.OperationService",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [
```

```
"org.springdoc.core.configuration.SpringDocConfiguration",
            "parameterBuilder",
            "requestBodyBuilder",
            "securityParser",
            "propertyResolverUtils"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration",
          "dependencies": []
        },
        "beansEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.beans.BeansEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/beans/BeansEndp
ointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.beans.BeansEndp
ointAutoConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "lettuceClientResources": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.lettuce.core.resource.DefaultClientResources",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/LettuceConne
ctionConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.LettuceConne
```

```
ctionConfiguration"
          ]
        },
        "pageableResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.web.PageableHandlerMethodArgumentReso
lver",
          "resource": "class path resource
[org/springframework/data/web/config/SpringDataWebConfiguration
.class]",
          "dependencies": [

"org.springframework.data.web.config.SpringDataWebConfiguration
"
          ]
        },
        "exchangeStrategiesCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientCodecCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/reactive/function/c
lient/WebClientAutoConfiguration$WebClientCodecsConfiguration.c
lass]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientAutoConfiguration$WebClientCodecsConfiguration"
          ]
        },
        "localeResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolve
r",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
        "handlerFunctionAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.function.support.HandlerFuncti
onAdapter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
        "localSpringDocParameterNameDiscoverer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.discoverer.SpringDocParameterNameDiscoverer
",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration"
          ]
        },
        "schemaPropertyDeprecatingConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.SchemaPropertyDeprecatingConvert
er",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration"
```

```
          ]
        },
        "pictureController": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.controller.PictureController",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/PictureController.class]"
,
          "dependencies": [
            "pictureService",
            "jdbcTemplate"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementContextAutoConfiguration$SameManagementContextConfigurati
on": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.server.Mana
gementContextAutoConfiguration$SameManagementContextConfigurati
on",
          "dependencies": [
            "environment"
          ]
        },

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonMixinConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonMixinConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.NamedParameterJdbc
TemplateConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.autoconfigure.jdbc.NamedParameterJdbc
TemplateConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientAutoConfiguration$WebClientCodecsConfiguration":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.reactive.function.c
lient.WebClientAutoConfiguration$WebClientCodecsConfiguration",
          "dependencies": []
        },
        "spring.transaction-
org.springframework.boot.autoconfigure.transaction.TransactionP
roperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.Transaction
Properties",
          "dependencies": []
        },
        "observationRestClientCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.web.client.Observatio
nRestClientCustomizer",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/web
/client/RestClientObservationConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.RestClientObservationConfiguration",
            "observationRegistry",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties"
          ]
        },
        "warningService": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.service.WarningService$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/WarningService.class]",
          "dependencies": [
            "mqttService"
          ]
        },
        "jvmHeapPressureMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.jvm.JvmHeapPressureMetric
s",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.HttpEncodin
gAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.HttpEncodin
gAutoConfiguration",
          "dependencies": [
            "server-
org.springframework.boot.autoconfigure.web.ServerProperties"
          ]
        },
        "welcomePageNotAcceptableHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.WelcomePage
NotAcceptableHandlerMapping",
```

```
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.observation.web
.client.HttpClientObservationsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web
.client.HttpClientObservationsAutoConfiguration",
          "dependencies": []
        },
        "endpointOperationParameterMapper": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.invoke.convert.Conve
rsionServiceParameterValueMapper",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/Endpoi
ntAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.Endpoi
ntAutoConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$NettyWebServerFactory
CustomizerConfiguration": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$NettyWebServerFactory
CustomizerConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.health.HealthCo
ntributorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthCo
ntributorAutoConfiguration",
          "dependencies": []
        },
        "sslBundleRegistry": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.ssl.DefaultSslBundleRegistry",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/ssl/SslAutoConfiguratio
n.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.ssl.SslAutoConfiguratio
n"
          ]
        },
        "jpaVendorAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaCon
figuration"
          ]
        },
        "reactiveStringRedisTemplate": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.ReactiveStringRedisTemplat
e",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/RedisReactiv
eAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.RedisReactiv
eAutoConfiguration",
            "redisConnectionFactory"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.info.InfoEndpoi
ntAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.info.InfoEndpoi
ntAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$ThreadPoolTaskExecutorBuilderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$ThreadPoolTaskExecutorBuilderConfiguration",
          "dependencies": []
        },
        "cacheAutoConfigurationValidator": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.cache.CacheAutoConfigur
ation$CacheManagerValidator",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/cache/CacheAutoConfigur
ation.class]",
          "dependencies": [
```

```
"org.springframework.boot.autoconfigure.cache.CacheAutoConfigur
ation",
            "spring.cache-
org.springframework.boot.autoconfigure.cache.CacheProperties"
          ]
        },
        "servletWebChildContextFactory": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.ManagementC
ontextFactory",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/web/servlet/Ser
vletManagementContextAutoConfiguration.class]",
          "dependencies": []
        },
        "countedAspect": {
          "aliases": [],
          "scope": "singleton",
          "type": "io.micrometer.core.aop.CountedAspect",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
AspectsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AspectsAutoConfiguration",
            "simpleMeterRegistry"
          ]
        },
        "additionalModelsConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.AdditionalModelsConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "springdocObjectMapperProvider"
          ]
```

```
        },
        "jacksonGeoModule": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.data.geo.GeoModule",
          "resource": "class path resource
[org/springframework/data/web/config/SpringDataJacksonConfigura
tion.class]",
          "dependencies": [

"org.springframework.data.web.config.SpringDataJacksonConfigura
tion"
          ]
        },
        "meterRegistryCloser": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AutoConfiguration$MeterRegistryCloser",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
AutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AutoConfiguration"
          ]
        },
        "mvcContentNegotiationManager": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.accept.ContentNegotiationManager",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.Composi
```

```
teMeterRegistryAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.Composi
teMeterRegistryAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsCon
figuration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.jdbc.Da
taSourcePoolMetricsAutoConfiguration$HikariDataSourceMetricsCon
figuration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.task.TaskSchedulingAuto
Configuration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskSchedulingAuto
Configuration",
          "dependencies": []
        },
        "httpRequestHandlerAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter"
,
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },
```

```
"org.springframework.boot.actuate.autoconfigure.web.mappings.MappingsEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.mappings.MappingsEndpointAutoConfiguration",
          "dependencies": []
        },
        "java.lang.Object": {
          "aliases": [],
          "scope": "singleton",
          "type": "java.lang.Object",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.observation.web.client.RestTemplateObservationConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.web.client.RestTemplateObservationConfiguration",
          "dependencies": []
        },
        "spring.servlet.multipart-
org.springframework.boot.autoconfigure.web.servlet.MultipartProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.MultipartProperties",
          "dependencies": []
        },
        "sslPropertiesSslBundleRegistrar": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.ssl.SslPropertiesBundleRegistrar",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/ssl/SslAutoConfiguration.class]",
```

```
          "dependencies": [
            "org.springframework.boot.autoconfigure.ssl.SslAutoConfiguration",
            "fileWatcher"
          ]
        },
        "org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfiguration$SpringDocWebMvcRouterConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfiguration$SpringDocWebMvcRouterConfiguration",
          "dependencies": []
        },
        "defaultMeterObservationHandler": {
          "aliases": [],
          "scope": "singleton",
          "type": "io.micrometer.core.instrument.observation.DefaultMeterObservationHandler",
          "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/observation/ObservationAutoConfiguration$MeterObservationHandlerConfiguration$OnlyMetricsMeterObservationHandlerConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.actuate.autoconfigure.observation.ObservationAutoConfiguration$MeterObservationHandlerConfiguration$OnlyMetricsMeterObservationHandlerConfiguration",
            "simpleMeterRegistry"
          ]
        },
        "resourceHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.servlet.handler.SimpleUrlHandlerMapping",
          "resource": "class path resource [org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [
```

```
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcContentNegotiationManager",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },
        "simpleControllerHandlerAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.mvc.SimpleControllerHandlerAda
pter",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.export.
simple.SimpleMetricsExportAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.export.
simple.SimpleMetricsExportAutoConfiguration",
          "dependencies": []
        },
        "propertyResolverUtils": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.utils.PropertyResolverUtils",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
```

```
"org.springframework.beans.factory.support.DefaultListableBeanF
actory@37efd131",
            "messageSource",

"org.springdoc.core.properties.SpringDocConfigProperties"
          ]
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceJmxConfi
guration$Hikari": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceJmxConfi
guration$Hikari",
          "dependencies": [
            "dataSource"
          ]
        },

"org.springdoc.core.properties.SwaggerUiConfigProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.properties.SwaggerUiConfigProperties",
          "dependencies": []
        },
        "cachesEndpointWebExtension": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.cache.CachesEndpointWebExtens
ion",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/cache/CachesEnd
pointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.cache.CachesEnd
pointAutoConfiguration",
            "cachesEndpoint"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.context.propert
```

```
ies.ConfigurationPropertiesReportEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.context.propert
ies.ConfigurationPropertiesReportEndpointAutoConfiguration",
          "dependencies": []
        },
        "simpleAsyncTaskSchedulerBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.task.SimpleAsyncTaskSchedulerBuilder"
,
          "resource": "class path resource
[org/springframework/boot/autoconfigure/task/TaskSchedulingConf
igurations$SimpleAsyncTaskSchedulerBuilderConfiguration.class]"
,
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$SimpleAsyncTaskSchedulerBuilderConfiguration"
          ]
        },
        "spring.lifecycle-
org.springframework.boot.autoconfigure.context.LifecyclePropert
ies": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.context.LifecycleProper
ties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$DefaultCodecsConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration$DefaultCodecsConfiguration",
          "dependencies": []
        },
```

```
"org.springframework.boot.actuate.autoconfigure.metrics.cache.CacheMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.cache.CacheMetricsAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.data.redis.LettuceConnectionConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.redis.LettuceConnectionConfiguration",
          "dependencies": [
            "spring.data.redis-
org.springframework.boot.autoconfigure.data.redis.RedisProperties",
            "redisConnectionDetails"
          ]
        },
        "badgesController": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.controller.BadgesController",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/BadgesController.class]",
          "dependencies": [
            "baiduService",
            "sessionStatusService",
            "storyService"
          ]
        },
        "healthContributorRegistry": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.AutoConfiguredHealthContributorRegistry",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
```

```
dpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "healthEndpointGroups",
            "diskSpaceHealthIndicator",
            "pingHealthContributor",
            "dbHealthContributor",
            "redisHealthContributor"
          ]
        },
        "micrometerOptions": {
          "aliases": [],
          "scope": "singleton",
          "type": "io.lettuce.core.metrics.MicrometerOptions",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/redis/L
ettuceMetricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.redis.L
ettuceMetricsAutoConfiguration"
          ]
        },
        "management.endpoint.configprops-
org.springframework.boot.actuate.autoconfigure.context.properti
es.ConfigurationPropertiesReportEndpointProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.context.propert
ies.ConfigurationPropertiesReportEndpointProperties",
          "dependencies": []
        },
        "parameterNamesModule": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.fasterxml.jackson.module.paramnames.ParameterNamesModule",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConf
```

```
iguration$ParameterNamesModuleConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$ParameterNamesModuleConfiguration"
          ]
        },
        "propertiesObservationFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Pro
pertiesObservationFilterPredicate",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/Obs
ervationAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties"
          ]
        },
        "sessionStatusRepository": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.repository.SessionStatusRepository",
          "resource":
"cn.netkiller.repository.SessionStatusRepository defined in
@EnableJpaRepositories declared on Application",
          "dependencies": [
            "jpa.named-queries#3",
            "jpa.SessionStatusRepository.fragments#0",
            "jpaSharedEM_entityManagerFactory",
            "jpaMappingContext"
          ]
        },
        "persistenceManagedTypes": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.orm.jpa.persistenceunit.SimplePersistenceM
```

```
anagedTypes",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/orm/jpa/JpaBaseConfigur
ation$PersistenceManagedTypesConfiguration.class]",
          "dependencies": [

"org.springframework.beans.factory.support.DefaultListableBeanF
actory@37efd131",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "micrometerClock": {
          "aliases": [],
          "scope": "singleton",
          "type": "io.micrometer.core.instrument.Clock$1",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
AutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AutoConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.beans.BeansEndp
ointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.beans.BeansEndp
ointAutoConfiguration",
          "dependencies": []
        },
        "responseSupportConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.ResponseSupportConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [
```

```
"org.springdoc.core.configuration.SpringDocConfiguration",
            "springdocObjectMapperProvider"
          ]
        },
        "pageableOpenAPIConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.PageableOpenAPIConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocPageableConfiguratio
n.class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocPageableConfiguratio
n",
            "springdocObjectMapperProvider"
          ]
        },
        "redisReferenceResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.convert.ReferenceResolverI
mpl",
          "dependencies": [
            "redisTemplate"
          ]
        },
        "propertiesMeterFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.Propert
iesMeterFilter",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/Metrics
AutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AutoConfiguration",
            "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
```

```
roperties"
          ]
        },

"org.springframework.boot.autoconfigure.web.client.RestTemplate
AutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.client.RestTemplate
AutoConfiguration",
          "dependencies": []
        },
        "gsonBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type": "com.google.gson.GsonBuilder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/gson/GsonAutoConfigurat
ion.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.gson.GsonAutoConfigurat
ion",
            "standardGsonBuilderCustomizer"
          ]
        },
        "aliyunService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.service.AliyunService",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/AliyunService.class]",
          "dependencies": []
        },
        "diskSpaceMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.metrics.system.DiskSpaceMetri
csBinder",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/SystemM
etricsAutoConfiguration.class]",
```

```
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.SystemM
etricsAutoConfiguration",
            "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties"
          ]
        },
        "securityParser": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springdoc.core.service.SecurityService",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "propertyResolverUtils"
          ]
        },

"org.springframework.data.jpa.util.JpaMetamodelCacheCleanup": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.jpa.util.JpaMetamodelCacheCleanup",
          "dependencies": []
        },
        "homeController": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.controller.HomeController",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/HomeController.class]",
          "dependencies": []
        },

"org.springdoc.core.properties.SpringDocConfigProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.properties.SpringDocConfigProperties",
```

```
          "dependencies": []
        },

"org.springdoc.core.properties.SwaggerUiConfigParameters": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.properties.SwaggerUiConfigParameters",
          "dependencies": [

"org.springdoc.core.properties.SwaggerUiConfigProperties"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration",
          "dependencies": [
            "server-
org.springframework.boot.autoconfigure.web.ServerProperties"
          ]
        },
        "emBeanDefinitionRegistrarPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.jpa.repository.support.EntityManagerB
eanDefinitionRegistrarPostProcessor",
          "dependencies": []
        },
        "servletEndpointDiscoverer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.annotation.Servl
etEndpointDiscoverer",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/We
bEndpointAutoConfiguration$WebEndpointServletConfiguration.clas
s]",
          "dependencies": [
```

```
"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointAutoConfiguration$WebEndpointServletConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14"
          ]
        },
        "metricsHttpServerUriTagFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.config.MeterFilter$9",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/web
/servlet/WebMvcObservationAutoConfiguration$MeterFilterConfigur
ation.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.web
.servlet.WebMvcObservationAutoConfiguration$MeterFilterConfigur
ation",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties",
            "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties"
          ]
        },
        "eagerTaskExecutorMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.LazyInitializationExcludeFilter$$Lamb
da/0x00007ff400afb340",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/task/Ta
skExecutorMetricsAutoConfiguration.class]",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.websocket.servlet.WebSo
cketServletAutoConfiguration$TomcatWebSocketConfiguration": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.websocket.servlet.WebSo
cketServletAutoConfiguration$TomcatWebSocketConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ersAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.HttpMessageConvert
ersAutoConfiguration",
          "dependencies": []
        },
        "redisConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.convert.MappingRedisConver
ter",
          "dependencies": [
            "keyValueMappingContext",
            "redisReferenceResolver",
            "redisCustomConversions"
          ]
        },
        "knife4jOpenApiCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"com.github.xiaoymin.knife4j.spring.extension.Knife4jOpenApiCus
tomizer",
          "resource": "class path resource
[com/github/xiaoymin/knife4j/spring/configuration/Knife4jAutoCo
nfiguration.class]",
          "dependencies": [

"com.github.xiaoymin.knife4j.spring.configuration.Knife4jAutoCo
nfiguration"
          ]
        },
        "healthEndpoint": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.health.HealthEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration",
            "healthContributorRegistry",
            "healthEndpointGroups",
            "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties"
          ]
        },
        "spring.task.execution-
org.springframework.boot.autoconfigure.task.TaskExecutionProper
ties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskExecutionPrope
rties",
          "dependencies": []
        },
        "viewControllerHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },

"org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConf
```

```
iguration$PooledDataSourceConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConf
iguration$PooledDataSourceConfiguration",
          "dependencies": []
        },
        "openApiResource": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.api.OpenApiWebMvcResource",
          "resource": "class path resource
[org/springdoc/webmvc/core/configuration/SpringDocWebMvcConfigu
ration.class]",
          "dependencies": [

"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration",
            "requestBuilder",
            "responseBuilder",
            "operationBuilder",

"org.springdoc.core.properties.SpringDocConfigProperties",
            "springDocProviders",
            "springDocCustomizers"
          ]
        },
        "themeResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.theme.FixedThemeResolver",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.availability.Av
```

```
ailabilityHealthContributorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.availability.Av
ailabilityHealthContributorAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.jdbc.metadata.DataSourc
ePoolMetadataProvidersConfiguration$HikariPoolDataSourceMetadat
aProviderConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.metadata.DataSourc
ePoolMetadataProvidersConfiguration$HikariPoolDataSourceMetadat
aProviderConfiguration",
          "dependencies": []
        },
        "mvcPatternParser": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.util.pattern.PathPatternParser",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration",
          "dependencies": []
        },
```

```
"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAut
oConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAut
oConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointWebExtensionConfiguration$MvcAdditionalHealthEndpointPath
sConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointWebExtensionConfiguration$MvcAdditionalHealthEndpointPath
sConfiguration",
          "dependencies": []
        },

"org.springframework.boot.sql.init.dependency.DatabaseInitializ
ationDependencyConfigurer$DependsOnDatabaseInitializationPostPr
ocessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.sql.init.dependency.DatabaseInitializ
ationDependencyConfigurer$DependsOnDatabaseInitializationPostPr
ocessor",
          "dependencies": []
        },
        "dispatcherServlet": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.DispatcherServlet",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/DispatcherS
ervletAutoConfiguration$DispatcherServletConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration$DispatcherServletConfiguration",
```

```
            "spring.mvc-
org.springframework.boot.autoconfigure.web.servlet.WebMvcProper
ties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.task.Ta
skExecutorMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.task.Ta
skExecutorMetricsAutoConfiguration",
          "dependencies": [
            "applicationTaskExecutor",
            "simpleMeterRegistry"
          ]
        },
        "restClientBuilder": {
          "aliases": [],
          "scope": "prototype",
          "type":
"org.springframework.web.client.RestClient$Builder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/client/RestClientAu
toConfiguration.class]",
          "dependencies": []
        },
        "springWebProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.core.providers.SpringWebMvcProvider",
          "resource": "class path resource
[org/springdoc/webmvc/core/configuration/SpringDocWebMvcConfigu
ration.class]",
          "dependencies": [

"org.springdoc.webmvc.core.configuration.SpringDocWebMvcConfigu
ration"
          ]
        },
        "jvmInfoMetrics": {
          "aliases": [],
          "scope": "singleton",
```

```
          "type":
"io.micrometer.core.instrument.binder.jvm.JvmInfoMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration",
          "dependencies": []
        },
        "parameterBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.service.GenericParameterService",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "propertyResolverUtils",
            "springdocObjectMapperProvider"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.jdbc.DataSource
HealthContributorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.jdbc.DataSource
HealthContributorAutoConfiguration",
          "dependencies": []
```

```
        },
        "webEndpointServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.servlet.WebMvcEn
dpointHandlerMapping",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/se
rvlet/WebMvcEndpointManagementContextConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.se
rvlet.WebMvcEndpointManagementContextConfiguration",
            "webEndpointDiscoverer",
            "servletEndpointDiscoverer",
            "controllerEndpointDiscoverer",
            "endpointMediaTypes",
            "management.endpoints.web.cors-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Cor
sEndpointProperties",
            "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties",
            "environment"
          ]
        },
        "threadPoolTaskExecutorBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.task.ThreadPoolTaskExecutorBuilder",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/task/TaskExecutorConfig
urations$ThreadPoolTaskExecutorBuilderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$ThreadPoolTaskExecutorBuilderConfiguration",
            "spring.task.execution-
org.springframework.boot.autoconfigure.task.TaskExecutionProper
ties"
          ]
        },
        "processorMetrics": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.system.ProcessorMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/SystemM
etricsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.SystemM
etricsAutoConfiguration"
          ]
        },
        "dispatcherServletMappingDescriptionProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.web.mappings.servlet.Dispatch
erServletsMappingDescriptionProvider",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/web/mappings/Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration$SpringM
vcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.web.mappings.Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration$SpringM
vcConfiguration"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.scheduling.Sche
duledTasksEndpointAutoConfiguration",
          "dependencies": []
        },
        "chatService": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.service.ChatService",
          "resource": "URL [jar:nested:/app/watch-1.0-
```

```
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/service/ChatService.class]",
          "dependencies": [
            "chatRepository"
          ]
        },
        "spring.jdbc-
org.springframework.boot.autoconfigure.jdbc.JdbcProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.JdbcProperties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.cache.RedisCacheConfigu
ration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.cache.RedisCacheConfigu
ration",
          "dependencies": []
        },
        "psychoanalysisController": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.controller.PsychoanalysisController",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/controller/PsychoanalysisController.
class]",
          "dependencies": [
            "psychoanalysisService"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.integra
tion.IntegrationMetricsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.metrics.integra
tion.IntegrationMetricsAutoConfiguration",
```

```
          "dependencies": []
        },
        "transactionExecutionListeners": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.ExecutionLi
stenersTransactionManagerCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/transaction/Transaction
ManagerCustomizationAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.transaction.Transaction
ManagerCustomizationAutoConfiguration"
          ]
        },
        "redisConnectionFactory": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.connection.lettuce.LettuceConne
ctionFactory",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/redis/LettuceConne
ctionConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.redis.LettuceConne
ctionConfiguration",
            "lettuceClientResources"
          ]
        },
        "springDocProviders": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.providers.SpringDocProviders",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "springdocObjectMapperProvider"
```

```
          ]
        },
        "transactionManager": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.orm.jpa.JpaTransactionManager",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/orm/jpa/HibernateJpaCon
figuration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaCon
figuration"
          ]
        },
        "spring.netty-
org.springframework.boot.autoconfigure.netty.NettyProperties":
{
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.netty.NettyProperties",
          "dependencies": []
        },
        "errorPageCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$ErrorPageCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration",
            "dispatcherServletRegistration"
          ]
        },
        "loggersEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.actuate.logging.LoggersEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/logging/Loggers
EndpointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.logging.Loggers
EndpointAutoConfiguration",
            "springBootLoggingSystem"
          ]
        },
        "standardGsonBuilderCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.gson.GsonAutoConfigurat
ion$StandardGsonBuilderCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/gson/GsonAutoConfigurat
ion.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.gson.GsonAutoConfigurat
ion",
            "spring.gson-
org.springframework.boot.autoconfigure.gson.GsonProperties"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$DefaultErrorViewResolverConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$DefaultErrorViewResolverConfiguration",
          "dependencies": [

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "spring.web-
org.springframework.boot.autoconfigure.web.WebProperties"
          ]
        },
        "tomcatWebServerFactoryCustomizer": {
```

```
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.embedded.TomcatWebS
erverFactoryCustomizer",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/embedded/EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$TomcatWebServerFactor
yCustomizerConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.embedded.EmbeddedWe
bServerFactoryCustomizerAutoConfiguration$TomcatWebServerFactor
yCustomizerConfiguration",
            "environment",
            "server-
org.springframework.boot.autoconfigure.web.ServerProperties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$OnlyMetricsConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.observation.Obs
ervationAutoConfiguration$OnlyMetricsConfiguration",
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.endpoint.Endpoi
ntAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.Endpoi
ntAutoConfiguration",
          "dependencies": []
        },
        "threadPoolTaskSchedulerBuilder": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.task.ThreadPoolTaskSchedulerBuilder",
          "resource": "class path resource
```

```
[org/springframework/boot/autoconfigure/task/TaskSchedulingConf
igurations$ThreadPoolTaskSchedulerBuilderConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.task.TaskSchedulingConf
igurations$ThreadPoolTaskSchedulerBuilderConfiguration",
            "spring.task.scheduling-
org.springframework.boot.autoconfigure.task.TaskSchedulingPrope
rties"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.data.redis.Redi
sHealthContributorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.data.redis.Redi
sHealthContributorAutoConfiguration",
          "dependencies": []
        },
        "viewNameTranslator": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.web.servlet.view.DefaultRequestToViewNameT
ranslator",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration"
          ]
        },

"org.springframework.boot.autoconfigure.http.JacksonHttpMessage
ConvertersConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.JacksonHttpMessage
ConvertersConfiguration",
          "dependencies": []
```

```
        },
        "polymorphicModelConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.PolymorphicModelConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "springdocObjectMapperProvider"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.system.DiskSpac
eHealthContributorAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.system.DiskSpac
eHealthContributorAutoConfiguration",
          "dependencies": []
        },
        "fileSupportConverter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.converters.FileSupportConverter",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration",
            "springdocObjectMapperProvider"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springframework.boot.actuate.autoconfigure.metrics.Metrics
AutoConfiguration",
          "dependencies": []
        },
        "cachesEndpoint": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.cache.CachesEndpoint",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/cache/CachesEnd
pointAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.cache.CachesEnd
pointAutoConfiguration",
            "cacheManager"
          ]
        },

"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.codec.CodecsAutoCo
nfiguration",
          "dependencies": []
        },
        "dispatcherServletRegistration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletRegistrationBean",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/DispatcherS
ervletAutoConfiguration$DispatcherServletRegistrationConfigurat
ion.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration$DispatcherServletRegistrationConfigurat
ion",
            "dispatcherServlet",
```

```
            "spring.mvc-
org.springframework.boot.autoconfigure.web.servlet.WebMvcProper
ties"
          ]
        },
        "pageableCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.web.SpringDataWebA
utoConfiguration$$Lambda/0x00007ff400a7e000",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/data/web/SpringDataWebA
utoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.data.web.SpringDataWebA
utoConfiguration"
          ]
        },
        "dataSource": {
          "aliases": [],
          "scope": "singleton",
          "type": "com.zaxxer.hikari.HikariDataSource",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/DataSourceConfigur
ation$Hikari.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jdbc.DataSourceConfigur
ation$Hikari",
            "spring.datasource-
org.springframework.boot.autoconfigure.jdbc.DataSourcePropertie
s",
            "jdbcConnectionDetails"
          ]
        },
        "knife4jConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"cn.netkiller.config.Knife4jConfiguration$$SpringCGLIB$$0",
          "resource": "URL [jar:nested:/app/watch-1.0-
SNAPSHOT.jar/!BOOT-
INF/classes/!/cn/netkiller/config/Knife4jConfiguration.class]",
```

```
          "dependencies": []
        },
        "openAPI": {
          "aliases": [],
          "scope": "prototype",
          "type": "io.swagger.v3.oas.models.OpenAPI",
          "resource": "class path resource
[cn/netkiller/config/Knife4jConfiguration.class]",
          "dependencies": [
            "knife4jConfiguration"
          ]
        },
        "healthEndpointGroupsBeanPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointConfiguration$HealthEndpointGroupsBeanPostProcessor",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointConfiguration.class]",
          "dependencies": []
        },
        "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.web.We
bEndpointProperties",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$TaskExecutorConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.task.TaskExecutorConfig
urations$TaskExecutorConfiguration",
          "dependencies": []
        },
        "jpaMappingContext": {
          "aliases": [],
```

```
          "scope": "singleton",
          "type":
"org.springframework.data.jpa.mapping.JpaMetamodelMappingContex
t",
          "dependencies": []
        },
        "tomcatServletWebServerFactory": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.embedded.tomcat.TomcatServletWebS
erverFactory",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/ServletWebS
erverFactoryConfiguration$EmbeddedTomcat.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.ServletWebS
erverFactoryConfiguration$EmbeddedTomcat"
          ]
        },

"org.springframework.boot.autoconfigure.http.JacksonHttpMessage
ConvertersConfiguration$MappingJackson2HttpMessageConverterConf
iguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.http.JacksonHttpMessage
ConvertersConfiguration$MappingJackson2HttpMessageConverterConf
iguration",
          "dependencies": []
        },
        "spring.jackson-
org.springframework.boot.autoconfigure.jackson.JacksonPropertie
s": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonProperti
es",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.reactor.netty.ReactorNe
```

```
ttyConfigurations$ReactorResourceFactoryConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.reactor.netty.ReactorNe
ttyConfigurations$ReactorResourceFactoryConfiguration",
          "dependencies": []
        },
        "chatRepository": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.repository.ChatRepository",
          "resource": "cn.netkiller.repository.ChatRepository
defined in @EnableJpaRepositories declared on Application",
          "dependencies": [
            "jpa.named-queries#1",
            "jpa.ChatRepository.fragments#0",
            "jpaSharedEM_entityManagerFactory",
            "jpaMappingContext"
          ]
        },

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration",
          "dependencies": []
        },
        "error": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$StaticView",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/error/Error
MvcAutoConfiguration$WhitelabelErrorViewConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$WhitelabelErrorViewConfiguration"
          ]
```

```
        },

"org.springframework.boot.autoconfigure.data.web.SpringDataWebA
utoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.data.web.SpringDataWebA
utoConfiguration",
          "dependencies": [
            "spring.data.web-
org.springframework.boot.autoconfigure.data.web.SpringDataWebPr
operties"
          ]
        },
        "memberRepository": {
          "aliases": [],
          "scope": "singleton",
          "type": "cn.netkiller.repository.MemberRepository",
          "resource": "cn.netkiller.repository.MemberRepository
defined in @EnableJpaRepositories declared on Application",
          "dependencies": [
            "jpa.named-queries#0",
            "jpa.MemberRepository.fragments#0",
            "jpaSharedEM_entityManagerFactory",
            "jpaMappingContext"
          ]
        },

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonObjectMapperConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonObjectMapperConfiguration",
          "dependencies": []
        },
        "swaggerConfigResource": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.webmvc.ui.SwaggerConfigResource",
          "resource": "class path resource
[org/springdoc/webmvc/ui/SwaggerConfig.class]",
```

```
          "dependencies": [
            "org.springdoc.webmvc.ui.SwaggerConfig",
            "swaggerWelcome"
          ]
        },

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointAutoConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$WhitelabelErrorViewConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.error.Error
MvcAutoConfiguration$WhitelabelErrorViewConfiguration",
          "dependencies": []
        },
        "delegatingMethodParameterCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.customizers.DataRestDelegatingMethodParamet
erCustomizer",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocPageableConfiguratio
n.class]",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocPageableConfiguratio
n"
          ]
        },
        "jsonMixinModule": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.jackson.JsonMixinModule",
```

```
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConf
iguration$JacksonMixinConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonMixinConfiguration",

"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "jsonMixinModuleEntries"
          ]
        },

"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration$DispatcherServletConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.DispatcherS
ervletAutoConfiguration$DispatcherServletConfiguration",
          "dependencies": []
        },
        "jdbcConnectionDetails": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.jdbc.PropertiesJdbcConn
ectionDetails",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jdbc/DataSourceAutoConf
iguration$PooledDataSourceConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConf
iguration$PooledDataSourceConfiguration",
            "spring.datasource-
org.springframework.boot.autoconfigure.jdbc.DataSourcePropertie
s"
          ]
        },
        "genericReturnTypeParser": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"org.springdoc.core.configuration.SpringDocConfiguration$1",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": [

"org.springdoc.core.configuration.SpringDocConfiguration"
          ]
        },
        "jvmGcMetrics": {
          "aliases": [],
          "scope": "singleton",
          "type":
"io.micrometer.core.instrument.binder.jvm.JvmGcMetrics",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/metrics/JvmMetr
icsAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.metrics.JvmMetr
icsAutoConfiguration"
          ]
        },
        "offsetResolver": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.web.OffsetScrollPositionHandlerMethod
ArgumentResolver",
          "resource": "class path resource
[org/springframework/data/web/config/SpringDataWebConfiguration
.class]",
          "dependencies": [

"org.springframework.data.web.config.SpringDataWebConfiguration
"
          ]
        },

"org.springframework.boot.autoconfigure.transaction.Transaction
AutoConfiguration$TransactionTemplateConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.transaction.Transaction
```

```
AutoConfiguration$TransactionTemplateConfiguration",
          "dependencies": []
        },

"org.springframework.boot.autoconfigure.web.client.RestClientAu
toConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.client.RestClientAu
toConfiguration",
          "dependencies": []
        },
        "healthEndpointWebMvcHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.endpoint.web.servlet.Addition
alHealthEndpointPathsWebMvcHandlerMapping",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/health/HealthEn
dpointWebExtensionConfiguration$MvcAdditionalHealthEndpointPath
sConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.health.HealthEn
dpointWebExtensionConfiguration$MvcAdditionalHealthEndpointPath
sConfiguration",
            "webEndpointDiscoverer",
            "healthEndpointGroups"
          ]
        },
        "welcomePageHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.web.servlet.WelcomePage
HandlerMapping",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoC
onfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoC
onfiguration$EnableWebMvcConfiguration",
```

```
"org.springframework.boot.web.servlet.context.AnnotationConfigS
ervletWebServerApplicationContext@18e36d14",
            "mvcConversionService",
            "mvcResourceUrlProvider"
          ]
        },
        "redisKeyValueAdapter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.data.redis.core.RedisKeyValueAdapter",
          "dependencies": [
            "redisTemplate",
            "redisConverter"
          ]
        },
        "servletExposeExcludePropertyEndpointFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.endpoint.expose
.IncludeExcludeEndpointFilter",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/endpoint/web/Se
rvletEndpointManagementContextConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.endpoint.web.Se
rvletEndpointManagementContextConfiguration",
            "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties"
          ]
        },
        "filterMappingDescriptionProvider": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.web.mappings.servlet.FiltersM
appingDescriptionProvider",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/web/mappings/Ma
ppingsEndpointAutoConfiguration$ServletWebConfiguration.class]"
,
```

```
          "dependencies": [
"org.springframework.boot.actuate.autoconfigure.web.mappings.MappingsEndpointAutoConfiguration$ServletWebConfiguration"
          ]
        },
        "springdocBeanFactoryPostProcessor": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.configurer.SpringdocBeanFactoryConfigurer",
          "resource": "class path resource
[org/springdoc/core/configuration/SpringDocConfiguration.class]
",
          "dependencies": []
        },
        "cacheManagerCustomizers": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.autoconfigure.cache.CacheManagerCustomizers",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/cache/CacheAutoConfiguration.class]",
          "dependencies": [
"org.springframework.boot.autoconfigure.cache.CacheAutoConfiguration"
          ]
        },
"org.springframework.boot.actuate.autoconfigure.web.exchanges.HttpExchangesEndpointAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.web.exchanges.HttpExchangesEndpointAutoConfiguration",
          "dependencies": []
        },
        "jacksonObjectMapper": {
          "aliases": [],
          "scope": "singleton",
          "type":
```

```
"com.fasterxml.jackson.databind.ObjectMapper",
          "resource": "class path resource
[org/springframework/boot/autoconfigure/jackson/JacksonAutoConf
iguration$JacksonObjectMapperConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.jackson.JacksonAutoConf
iguration$JacksonObjectMapperConfiguration",
            "jacksonObjectMapperBuilder"
          ]
        },
        "webMvcObservationFilter": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.web.servlet.FilterRegistrationBean",
          "resource": "class path resource
[org/springframework/boot/actuate/autoconfigure/observation/web
/servlet/WebMvcObservationAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.actuate.autoconfigure.observation.web
.servlet.WebMvcObservationAutoConfiguration",
            "observationRegistry",
            "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties"
          ]
        },

"org.springdoc.core.properties.SwaggerUiOAuthProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springdoc.core.properties.SwaggerUiOAuthProperties",
          "dependencies": []
        },
        "management.endpoint.logfile-
org.springframework.boot.actuate.autoconfigure.logging.LogFileW
ebEndpointProperties": {
          "aliases": [],
          "scope": "singleton",
          "type":
"org.springframework.boot.actuate.autoconfigure.logging.LogFile
WebEndpointProperties",
```

```
          "dependencies": []
        },

"org.springframework.boot.actuate.autoconfigure.metrics.MetricsAspectsAutoConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.autoconfigure.metrics.MetricsAspectsAutoConfiguration",
          "dependencies": []
        },
        "gson": {
          "aliases": [],
          "scope": "singleton",
          "type": "com.google.gson.Gson",
          "resource": "class path resource [org/springframework/boot/autoconfigure/gson/GsonAutoConfiguration.class]",
          "dependencies": [

"org.springframework.boot.autoconfigure.gson.GsonAutoConfiguration",
            "gsonBuilder"
          ]
        }
      }
    }
  }
}
```

34.7. caches

```
neo@MacBook-Pro-M2 ~ % curl -s http://www.netkiller.cn:8080/actuator/caches | jq
{
  "cacheManagers": {
    "cacheManager": {
      "caches": {
        "censor": {
```

```
          "target":
"org.springframework.data.redis.cache.DefaultRedisCacheWriter"
        },
        "translate": {
          "target":
"org.springframework.data.redis.cache.DefaultRedisCacheWriter"
        }
      }
    }
  }
}
```

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/caches/censor |jq
{
  "target":
"org.springframework.data.redis.cache.DefaultRedisCacheWriter",
  "name": "censor",
  "cacheManager": "cacheManager"
}
```

34.8. conditions

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/conditions | jq | more
{
  "contexts": {
    "watch-production": {
      "positiveMatches": {
        "Knife4jAutoConfiguration": [
          {
            "condition": "OnPropertyCondition",
            "message": "@ConditionalOnProperty
(knife4j.enable=true) matched"
          }
        ],
```

```
"Knife4jAutoConfiguration#knife4jJakartaOperationCustomizer": [
          {
            "condition": "OnBeanCondition",
            "message": "@ConditionalOnMissingBean (types:
com.github.xiaoymin.knife4j.spring.extension.Knife4jJakartaOper
ationCustomizer; SearchStrategy: all) did not find any beans"
          }
        ],
        "Knife4jAutoConfiguration#knife4jOpenApiCustomizer": [
          {
            "condition": "OnBeanCondition",
            "message": "@ConditionalOnMissingBean (types:
com.github.xiaoymin.knife4j.spring.extension.Knife4jOpenApiCust
omizer; SearchStrategy: all) did not find any beans"
          }
        ],
        "SpringDocConfiguration": [
          {
            "condition": "OnWebApplicationCondition",
            "message": "@ConditionalOnWebApplication (required)
found 'session' scope"
          },
          {
            "condition": "OnPropertyCondition",
            "message": "@ConditionalOnProperty (springdoc.api-
docs.enabled) matched"
          }
        ],
        "SpringDocConfiguration#fileSupportConverter": [
          {
            "condition": "OnBeanCondition",
            "message": "@ConditionalOnMissingBean (types:
org.springdoc.core.converters.FileSupportConverter;
SearchStrategy: all) did not find any beans"
          }
        ],
        "SpringDocConfiguration#openAPIBuilder": [
          {
            "condition": "OnBeanCondition",
            "message": "@ConditionalOnMissingBean (types:
org.springdoc.core.service.OpenAPIService; SearchStrategy: all)
did not find any beans"
          }
        ],
```

```
        "SpringDocConfiguration#operationBuilder": [
          {
            "condition": "OnBeanCondition",
            "message": "@ConditionalOnMissingBean (types:
org.springdoc.core.service.OperationService; SearchStrategy:
all) did not find any beans"
          }
        ],
        "SpringDocConfiguration#parameterBuilder": [
          {
            "condition": "OnBeanCondition",
            "message": "@ConditionalOnMissingBean (types:
org.springdoc.core.service.GenericParameterService;
SearchStrategy: all) did not find any beans"
          }
        ],
```

34.9. configprops 配置文件

查看 spring.task.execution 配置

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/configprops/spring.task.e
xecution | jq
{
  "contexts": {
    "watch-production": {
      "beans": {
        "spring.task.execution-
org.springframework.boot.autoconfigure.task.TaskExecutionProper
ties": {
          "prefix": "spring.task.execution",
          "properties": {
            "pool": {
              "queueCapacity": "******",
              "coreSize": "******",
              "maxSize": "******",
              "allowCoreThreadTimeout": "******",
```

```
            "keepAlive": "******"
          },
          "simple": {},
          "threadNamePrefix": "******",
          "shutdown": {
            "awaitTermination": "******",
            "awaitTerminationPeriod": "******"
          }
        },
        "inputs": {
          "pool": {
            "queueCapacity": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 25:43"
            },
            "coreSize": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 23:38"
            },
            "maxSize": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 24:37"
            },
            "allowCoreThreadTimeout": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 27:54"
            },
            "keepAlive": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 26:39"
            }
          },
          "simple": {},
          "threadNamePrefix": {
            "value": "******",
            "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 30:42"
          },
          "shutdown": {
            "awaitTermination": {
```

```
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 28:50"
              },
              "awaitTerminationPeriod": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 29:57"
              }
            }
          }
        }
      }
    }
  }
}
```

查看所有配置

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/configprops | jq
{
  "contexts": {
    "watch-production": {
      "beans": {
        "spring.transaction-
org.springframework.boot.autoconfigure.transaction.TransactionP
roperties": {
          "prefix": "spring.transaction",
          "properties": {},
          "inputs": {}
        },
        "spring.jpa-
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties":
{
          "prefix": "spring.jpa",
          "properties": {
            "mappingResources": [],
            "showSql": "******",
            "openInView": "******",
```

```
            "generateDdl": "******",
            "properties": {}
          },
          "inputs": {
            "mappingResources": [],
            "showSql": {},
            "openInView": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 41:25"
            },
            "generateDdl": {},
            "properties": {}
          }
        },
        "management.observations-
org.springframework.boot.actuate.autoconfigure.observation.Obse
rvationProperties": {
          "prefix": "management.observations",
          "properties": {
            "keyValues": {},
            "http": {
              "client": {
                "requests": {
                  "name": "******"
                }
              },
              "server": {
                "requests": {
                  "name": "******"
                }
              }
            },
            "enable": {}
          },
          "inputs": {
            "keyValues": {},
            "http": {
              "client": {
                "requests": {
                  "name": {}
                }
              },
              "server": {
                "requests": {
```

```
                  "name": {}
                }
              }
            },
            "enable": {}
          }
        },
        "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Web
EndpointProperties": {
          "prefix": "management.endpoints.web",
          "properties": {
            "pathMapping": {},
            "exposure": {
              "include": [
                "******"
              ],
              "exclude": []
            },
            "basePath": "******",
            "discovery": {
              "enabled": "******"
            }
          },
          "inputs": {
            "pathMapping": {},
            "exposure": {
              "include": [
                {
                  "value": "******",
                  "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 20:43"
                }
              ],
              "exclude": []
            },
            "basePath": {},
            "discovery": {
              "enabled": {}
            }
          }
        },
        "spring.cache-
org.springframework.boot.autoconfigure.cache.CacheProperties":
{
```

```
          "prefix": "spring.cache",
          "properties": {
            "caffeine": {},
            "infinispan": {},
            "cacheNames": [],
            "couchbase": {},
            "jcache": {},
            "type": "******",
            "redis": {
              "timeToLive": "******",
              "cacheNullValues": "******",
              "useKeyPrefix": "******",
              "enableStatistics": "******"
            }
          },
          "inputs": {
            "caffeine": {},
            "infinispan": {},
            "cacheNames": [],
            "couchbase": {},
            "jcache": {},
            "type": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 60:19"
            },
            "redis": {
              "timeToLive": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 61:33"
              },
              "cacheNullValues": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 62:38"
              },
              "useKeyPrefix": {},
              "enableStatistics": {}
            }
          }
        },
        "spring.jdbc-
org.springframework.boot.autoconfigure.jdbc.JdbcProperties": {
          "prefix": "spring.jdbc",
```

```
          "properties": {
            "template": {
              "fetchSize": "******",
              "maxRows": "******"
            }
          },
          "inputs": {
            "template": {
              "fetchSize": {},
              "maxRows": {}
            }
          }
        },
        "spring.data.redis-
org.springframework.boot.autoconfigure.data.redis.RedisProperti
es": {
          "prefix": "spring.data.redis",
          "properties": {
            "database": "******",
            "password": "******",
            "port": "******",
            "jedis": {
              "pool": {
                "maxIdle": "******",
                "minIdle": "******",
                "maxActive": "******",
                "maxWait": "******"
              }
            },
            "host": "******",
            "ssl": {
              "enabled": "******"
            },
            "lettuce": {
              "shutdownTimeout": "******",
              "pool": {
                "maxIdle": "******",
                "minIdle": "******",
                "maxActive": "******",
                "maxWait": "******"
              },
              "cluster": {
                "refresh": {
                  "dynamicRefreshSources": "******",
                  "adaptive": "******"
```

```
                }
              }
            },
            "timeout": "******"
          },
          "inputs": {
            "database": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 11:28"
            },
            "password": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 10:28"
            },
            "port": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 9:24"
            },
            "jedis": {
              "pool": {
                "maxIdle": {},
                "minIdle": {},
                "maxActive": {},
                "maxWait": {}
              }
            },
            "host": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 8:24"
            },
            "ssl": {
              "enabled": {}
            },
            "lettuce": {
              "shutdownTimeout": {},
              "pool": {
                "maxIdle": {
                  "value": "******",
                  "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 58:41"
                },
```

```
                "minIdle": {
                  "value": "******",
                  "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 59:41"
                },
                "maxActive": {
                  "value": "******",
                  "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 56:43"
                },
                "maxWait": {
                  "value": "******",
                  "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 57:41"
                }
              },
              "cluster": {
                "refresh": {
                  "dynamicRefreshSources": {},
                  "adaptive": {}
                }
              }
            },
            "timeout": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 12:27"
            }
          }
        },
        "spring.jackson-
org.springframework.boot.autoconfigure.jackson.JacksonPropertie
s": {
          "prefix": "spring.jackson",
          "properties": {
            "serialization": {},
            "visibility": {},
            "parser": {},
            "datatype": {
              "jsonNode": {},
              "enum": {}
            },
            "deserialization": {},
            "generator": {},
            "mapper": {}
```

```
          },
          "inputs": {
            "serialization": {},
            "visibility": {},
            "parser": {},
            "datatype": {
              "jsonNode": {},
              "enum": {}
            },
            "deserialization": {},
            "generator": {},
            "mapper": {}
          }
        },
        "spring.gson-
org.springframework.boot.autoconfigure.gson.GsonProperties": {
          "prefix": "spring.gson",
          "properties": {},
          "inputs": {}
        },
        "management.health.db-
org.springframework.boot.actuate.autoconfigure.jdbc.DataSourceH
ealthIndicatorProperties": {
          "prefix": "management.health.db",
          "properties": {
            "ignoreRoutingDataSources": "******"
          },
          "inputs": {
            "ignoreRoutingDataSources": {}
          }
        },
        "spring.data.web-
org.springframework.boot.autoconfigure.data.web.SpringDataWebPr
operties": {
          "prefix": "spring.data.web",
          "properties": {
            "pageable": {
              "pageParameter": "******",
              "sizeParameter": "******",
              "oneIndexedParameters": "******",
              "prefix": "******",
              "qualifierDelimiter": "******",
              "defaultPageSize": "******",
              "maxPageSize": "******"
            },
```

```
            "sort": {
              "sortParameter": "******"
            }
          },
          "inputs": {
            "pageable": {
              "pageParameter": {},
              "sizeParameter": {},
              "oneIndexedParameters": {},
              "prefix": {},
              "qualifierDelimiter": {},
              "defaultPageSize": {},
              "maxPageSize": {}
            },
            "sort": {
              "sortParameter": {}
            }
          }
        },

"org.springdoc.core.properties.SpringDocConfigProperties": {
          "prefix": "springdoc",
          "properties": {
            "removeBrokenReferenceDefinitions": "******",
            "writerWithOrderByKeys": "******",
            "disableI18n": "******",
            "showLoginEndpoint": "******",
            "nullableRequestParameterEnabled": "******",
            "enableJavadoc": "******",
            "useFqn": "******",
            "useManagementPort": "******",
            "showSpringCloudFunctions": "******",
            "enableDataRest": "******",
            "cache": {
              "disabled": "******"
            },
            "preLoadingEnabled": "******",
            "enableKotlin": "******",
            "enableGroovy": "******",
            "defaultProducesMediaType": "******",
            "modelConverters": {
              "deprecatingConverter": {
                "enabled": "******"
              },
              "pageableConverter": {
```

```
            "enabled": "******"
          },
          "polymorphicConverter": {
            "enabled": "******"
          }
        },
        "enableHateoas": "******",
        "groupConfigs": [
          {
            "pathsToMatch": [
              "******"
            ],
            "group": "******",
            "displayName": "******"
          }
        ],
        "enableSpringSecurity": "******",
        "writerWithDefaultPrettyPrinter": "******",
        "showOauth2Endpoints": "******",
        "showActuator": "******",
        "defaultFlatParamObject": "******",
        "defaultSupportFormData": "******",
        "apiDocs": {
          "path": "******",
          "enabled": "******",
          "resolveSchemaProperties": "******",
          "groups": {
            "enabled": "******"
          }
        },
        "autoTagClasses": "******",
        "webjars": {
          "prefix": "******"
        },
        "modelAndViewAllowed": "******",
        "defaultConsumesMediaType": "******",
        "sortConverter": {
          "enabled": "******"
        }
      },
      "inputs": {
        "removeBrokenReferenceDefinitions": {},
        "writerWithOrderByKeys": {},
        "disableI18n": {},
        "showLoginEndpoint": {},
```

```
            "nullableRequestParameterEnabled": {},
            "enableJavadoc": {},
            "useFqn": {},
            "useManagementPort": {},
            "showSpringCloudFunctions": {},
            "enableDataRest": {},
            "cache": {
              "disabled": {}
            },
            "preLoadingEnabled": {},
            "enableKotlin": {},
            "enableGroovy": {},
            "defaultProducesMediaType": {},
            "modelConverters": {
              "deprecatingConverter": {
                "enabled": {}
              },
              "pageableConverter": {
                "enabled": {}
              },
              "polymorphicConverter": {
                "enabled": {}
              }
            },
            "enableHateoas": {},
            "groupConfigs": [
              {
                "pathsToMatch": [
                  {}
                ],
                "group": {},
                "displayName": {}
              }
            ],
            "enableSpringSecurity": {},
            "writerWithDefaultPrettyPrinter": {},
            "showOauth2Endpoints": {},
            "showActuator": {},
            "defaultFlatParamObject": {},
            "defaultSupportFormData": {},
            "apiDocs": {
              "path": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 38:25"
```

```
            },
            "enabled": {},
            "resolveSchemaProperties": {},
            "groups": {
              "enabled": {}
            }
          },
          "autoTagClasses": {},
          "webjars": {
            "prefix": {}
          },
          "modelAndViewAllowed": {},
          "defaultConsumesMediaType": {},
          "sortConverter": {
            "enabled": {}
          }
        }
      },
      "spring.jpa.hibernate-
org.springframework.boot.autoconfigure.orm.jpa.HibernateProperties": {
        "prefix": "spring.jpa.hibernate",
        "properties": {
          "naming": {},
          "ddlAuto": "******"
        },
        "inputs": {
          "naming": {},
          "ddlAuto": {
            "value": "******",
            "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 43:31"
          }
        }
      },
      "spring.info-
org.springframework.boot.autoconfigure.info.ProjectInfoProperties": {
        "prefix": "spring.info",
        "properties": {
          "build": {
            "location": {},
            "encoding": "******"
          },
          "git": {
```

```
            "location": {},
            "encoding": "******"
          }
        },
        "inputs": {
          "build": {
            "location": {},
            "encoding": {}
          },
          "git": {
            "location": {},
            "encoding": {}
          }
        }
      },
      "spring.datasource-
org.springframework.boot.autoconfigure.jdbc.DataSourcePropertie
s": {
        "prefix": "spring.datasource",
        "properties": {
          "password": "******",
          "embeddedDatabaseConnection": "******",
          "driverClassName": "******",
          "generateUniqueName": "******",
          "xa": {
            "properties": {}
          },
          "url": "******",
          "username": "******"
        },
        "inputs": {
          "password": {
            "value": "******",
            "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 6:28"
          },
          "embeddedDatabaseConnection": {},
          "driverClassName": {
            "value": "******",
            "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 3:37"
          },
          "generateUniqueName": {},
          "xa": {
            "properties": {}
```

```
            },
            "url": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 4:23"
            },
            "username": {
              "value": "******",
              "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 5:28"
            }
          }
        },

"org.springdoc.core.properties.SwaggerUiConfigProperties": {
          "prefix": "springdoc.swagger-ui",
          "properties": {
            "path": "******",
            "operationsSorter": "******",
            "useRootPath": "******",
            "groupsOrder": "******",
            "tagsSorter": "******",
            "disableSwaggerDefaultUrl": "******",
            "csrf": {
              "enabled": "******",
              "useLocalStorage": "******",
              "useSessionStorage": "******",
              "cookieName": "******",
              "localStorageKey": "******",
              "sessionStorageKey": "******",
              "headerName": "******"
            },
            "syntaxHighlight": {},
            "version": "******",
            "enabled": "******"
          },
          "inputs": {
            "path": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 35:27"
            },
            "operationsSorter": {
              "value": "******",
              "origin": "class path resource
```

```
[application.properties] from watch-1.0-SNAPSHOT.jar - 37:40"
            },
            "useRootPath": {},
            "groupsOrder": {},
            "tagsSorter": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 36:34"
            },
            "disableSwaggerDefaultUrl": {},
            "csrf": {
              "enabled": {},
              "useLocalStorage": {},
              "useSessionStorage": {},
              "cookieName": {},
              "localStorageKey": {},
              "sessionStorageKey": {},
              "headerName": {}
            },
            "syntaxHighlight": {},
            "version": {},
            "enabled": {}
          }
        },
        "management.endpoint.health-
org.springframework.boot.actuate.autoconfigure.health.HealthEnd
pointProperties": {
          "prefix": "management.endpoint.health",
          "properties": {
            "logging": {
              "slowIndicatorThreshold": "******"
            },
            "showDetails": "******",
            "status": {
              "order": [],
              "httpMapping": {}
            },
            "roles": [],
            "group": {}
          },
          "inputs": {
            "logging": {
              "slowIndicatorThreshold": {}
            },
            "showDetails": {},
```

```
            "status": {
              "order": [],
              "httpMapping": {}
            },
            "roles": [],
            "group": {}
          }
        },
        "spring.reactor.netty-
org.springframework.boot.autoconfigure.reactor.netty.ReactorNet
tyProperties": {
          "prefix": "spring.reactor.netty",
          "properties": {},
          "inputs": {}
        },
        "spring.lifecycle-
org.springframework.boot.autoconfigure.context.LifecycleProperti
es": {
          "prefix": "spring.lifecycle",
          "properties": {
            "timeoutPerShutdownPhase": "******"
          },
          "inputs": {
            "timeoutPerShutdownPhase": {}
          }
        },
        "spring.web-
org.springframework.boot.autoconfigure.web.WebProperties": {
          "prefix": "spring.web",
          "properties": {
            "localeResolver": "******",
            "resources": {
              "staticLocations": [
                "******",
                "******",
                "******",
                "******"
              ],
              "addMappings": "******",
              "chain": {
                "cache": "******",
                "compressed": "******",
                "strategy": {
                  "fixed": {
                    "enabled": "******",
```

```
                "paths": [
                  "******"
                ]
              },
              "content": {
                "enabled": "******",
                "paths": [
                  "******"
                ]
              }
            }
          },
          "cache": {
            "cachecontrol": {},
            "useLastModified": "******"
          }
        }
      },
      "inputs": {
        "localeResolver": {},
        "resources": {
          "staticLocations": [
            {},
            {},
            {},
            {}
          ],
          "addMappings": {},
          "chain": {
            "cache": {},
            "compressed": {},
            "strategy": {
              "fixed": {
                "enabled": {},
                "paths": [
                  {}
                ]
              },
              "content": {
                "enabled": {},
                "paths": [
                  {}
                ]
              }
            }
```

```
            },
            "cache": {
              "cachecontrol": {},
              "useLastModified": {}
            }
          }
        }
      },
      "management.metrics-
org.springframework.boot.actuate.autoconfigure.metrics.MetricsP
roperties": {
        "prefix": "management.metrics",
        "properties": {
          "system": {
            "diskspace": {
              "paths": [
                "******"
              ]
            }
          },
          "data": {
            "repository": {
              "metricName": "******",
              "autotime": {
                "enabled": "******",
                "percentilesHistogram": "******"
              }
            }
          },
          "web": {
            "client": {
              "maxUriTags": "******"
            },
            "server": {
              "maxUriTags": "******"
            }
          },
          "enable": {},
          "useGlobalRegistry": "******",
          "distribution": {
            "percentilesHistogram": {},
            "percentiles": {},
            "slo": {},
            "minimumExpectedValue": {},
            "maximumExpectedValue": {},
```

```
            "expiry": {},
            "bufferLength": {}
          },
          "tags": {}
        },
        "inputs": {
          "system": {
            "diskspace": {
              "paths": [
                {}
              ]
            }
          },
          "data": {
            "repository": {
              "metricName": {},
              "autotime": {
                "enabled": {},
                "percentilesHistogram": {}
              }
            }
          },
          "web": {
            "client": {
              "maxUriTags": {}
            },
            "server": {
              "maxUriTags": {}
            }
          },
          "enable": {},
          "useGlobalRegistry": {},
          "distribution": {
            "percentilesHistogram": {},
            "percentiles": {},
            "slo": {},
            "minimumExpectedValue": {},
            "maximumExpectedValue": {},
            "expiry": {},
            "bufferLength": {}
          },
          "tags": {}
        }
      },
      "spring.mvc-
```

```
org.springframework.boot.autoconfigure.web.servlet.WebMvcProper
ties": {
          "prefix": "spring.mvc",
          "properties": {
            "contentnegotiation": {
              "favorParameter": "******",
              "mediaTypes": {}
            },
            "servlet": {
              "path": "******",
              "loadOnStartup": "******"
            },
            "format": {},
            "staticPathPattern": "******",
            "dispatchOptionsRequest": "******",
            "dispatchTraceRequest": "******",
            "problemdetails": {
              "enabled": "******"
            },
            "logResolvedException": "******",
            "webjarsPathPattern": "******",
            "async": {},
            "view": {},
            "publishRequestHandledEvents": "******",
            "logRequestDetails": "******",
            "pathmatch": {
              "matchingStrategy": "******"
            },
            "throwExceptionIfNoHandlerFound": "******"
          },
          "inputs": {
            "contentnegotiation": {
              "favorParameter": {},
              "mediaTypes": {}
            },
            "servlet": {
              "path": {},
              "loadOnStartup": {}
            },
            "format": {},
            "staticPathPattern": {},
            "dispatchOptionsRequest": {},
            "dispatchTraceRequest": {},
            "problemdetails": {
              "enabled": {}
```

```
            },
            "logResolvedException": {},
            "webjarsPathPattern": {},
            "async": {},
            "view": {},
            "publishRequestHandledEvents": {},
            "logRequestDetails": {},
            "pathmatch": {
              "matchingStrategy": {}
            },
            "throwExceptionIfNoHandlerFound": {}
          }
        },
        "knife4j.setting-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jSetting
": {
          "prefix": "knife4j.setting",
          "properties": {
            "enableDebug": "******",
            "enableOpenApi": "******",
            "enableHomeCustom": "******",
            "enableFooter": "******",
            "enableFilterMultipartApiMethodType": "******",
            "language": "******",
            "enableReloadCacheParameter": "******",
            "enableDynamicParameter": "******",
            "enableHostText": "******",
            "enableRequestCache": "******",
            "enableFooterCustom": "******",
            "customCode": "******",
            "swaggerModelName": "******",
            "enableResponseCode": "******",
            "enableDocumentManage": "******",
            "enableAfterScript": "******",
            "enableSwaggerModels": "******",
            "enableFilterMultipartApis": "******",
            "enableHost": "******",
            "enableVersion": "******",
            "enableSearch": "******",
            "enableGroup": "******"
          },
          "inputs": {
            "enableDebug": {},
            "enableOpenApi": {},
            "enableHomeCustom": {},
```

```
            "enableFooter": {},
            "enableFilterMultipartApiMethodType": {},
            "language": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 33:26"
            },
            "enableReloadCacheParameter": {},
            "enableDynamicParameter": {},
            "enableHostText": {},
            "enableRequestCache": {},
            "enableFooterCustom": {},
            "customCode": {},
            "swaggerModelName": {},
            "enableResponseCode": {},
            "enableDocumentManage": {},
            "enableAfterScript": {},
            "enableSwaggerModels": {},
            "enableFilterMultipartApis": {},
            "enableHost": {},
            "enableVersion": {},
            "enableSearch": {},
            "enableGroup": {}
          }
        },
        "management.info-
org.springframework.boot.actuate.autoconfigure.info.InfoContrib
utorProperties": {
          "prefix": "management.info",
          "properties": {
            "git": {
              "mode": "******"
            }
          },
          "inputs": {
            "git": {
              "mode": {}
            }
          }
        },
        "spring.netty-
org.springframework.boot.autoconfigure.netty.NettyProperties":
{
          "prefix": "spring.netty",
          "properties": {},
```

```
          "inputs": {}
        },
        "spring.task.execution-
org.springframework.boot.autoconfigure.task.TaskExecutionProper
ties": {
          "prefix": "spring.task.execution",
          "properties": {
            "pool": {
              "queueCapacity": "******",
              "coreSize": "******",
              "maxSize": "******",
              "allowCoreThreadTimeout": "******",
              "keepAlive": "******"
            },
            "simple": {},
            "threadNamePrefix": "******",
            "shutdown": {
              "awaitTermination": "******",
              "awaitTerminationPeriod": "******"
            }
          },
          "inputs": {
            "pool": {
              "queueCapacity": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 25:43"
              },
              "coreSize": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 23:38"
              },
              "maxSize": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 24:37"
              },
              "allowCoreThreadTimeout": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 27:54"
              },
              "keepAlive": {
                "value": "******",
```

```
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 26:39"
              }
            },
            "simple": {},
            "threadNamePrefix": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 30:42"
            },
            "shutdown": {
              "awaitTermination": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 28:50"
              },
              "awaitTerminationPeriod": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 29:57"
              }
            }
          }
        },
        "management.server-
org.springframework.boot.actuate.autoconfigure.web.server.Manag
ementServerProperties": {
          "prefix": "management.server",
          "properties": {
            "basePath": "******"
          },
          "inputs": {
            "basePath": {}
          }
        },
        "knife4j-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jPropert
ies": {
          "prefix": "knife4j",
          "properties": {
            "cors": "******",
            "production": "******",
            "enable": "******",
            "setting": {
              "customCode": "******",
```

```
            "language": "******",
            "enableSwaggerModels": "******",
            "swaggerModelName": "******",
            "enableReloadCacheParameter": "******",
            "enableAfterScript": "******",
            "enableDocumentManage": "******",
            "enableVersion": "******",
            "enableRequestCache": "******",
            "enableFilterMultipartApis": "******",
            "enableFilterMultipartApiMethodType": "******",
            "enableHost": "******",
            "enableHostText": "******",
            "enableDynamicParameter": "******",
            "enableDebug": "******",
            "enableFooter": "******",
            "enableFooterCustom": "******",
            "enableSearch": "******",
            "enableOpenApi": "******",
            "enableHomeCustom": "******",
            "enableGroup": "******",
            "enableResponseCode": "******"
          }
        },
        "inputs": {
          "cors": {},
          "production": {},
          "enable": {
            "value": "******",
            "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 32:16"
          },
          "setting": {
            "customCode": {},
            "language": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 33:26"
            },
            "enableSwaggerModels": {},
            "swaggerModelName": {},
            "enableReloadCacheParameter": {},
            "enableAfterScript": {},
            "enableDocumentManage": {},
            "enableVersion": {},
            "enableRequestCache": {},
```

```
              "enableFilterMultipartApis": {},
              "enableFilterMultipartApiMethodType": {},
              "enableHost": {},
              "enableHostText": {},
              "enableDynamicParameter": {},
              "enableDebug": {},
              "enableFooter": {},
              "enableFooterCustom": {},
              "enableSearch": {},
              "enableOpenApi": {},
              "enableHomeCustom": {},
              "enableGroup": {},
              "enableResponseCode": {}
            }
          }
        },
        "spring.ssl-
org.springframework.boot.autoconfigure.ssl.SslProperties": {
          "prefix": "spring.ssl",
          "properties": {
            "bundle": {
              "pem": {},
              "jks": {},
              "watch": {
                "file": {
                  "quietPeriod": "******"
                }
              }
            }
          },
          "inputs": {
            "bundle": {
              "pem": {},
              "jks": {},
              "watch": {
                "file": {
                  "quietPeriod": {}
                }
              }
            }
          }
        },
        "spring.codec-
org.springframework.boot.autoconfigure.codec.CodecProperties":
{
```

```
          "prefix": "spring.codec",
          "properties": {
            "logRequestDetails": "******"
          },
          "inputs": {
            "logRequestDetails": {}
          }
        },
        "management.endpoint.configprops-
org.springframework.boot.actuate.autoconfigure.context.properti
es.ConfigurationPropertiesReportEndpointProperties": {
          "prefix": "management.endpoint.configprops",
          "properties": {
            "showValues": "******",
            "roles": []
          },
          "inputs": {
            "showValues": {},
            "roles": []
          }
        },
        "management.simple.metrics.export-
org.springframework.boot.actuate.autoconfigure.metrics.export.s
imple.SimpleProperties": {
          "prefix": "management.simple.metrics.export",
          "properties": {
            "mode": "******",
            "enabled": "******",
            "step": "******"
          },
          "inputs": {
            "mode": {},
            "enabled": {},
            "step": {}
          }
        },
        "spring.task.scheduling-
org.springframework.boot.autoconfigure.task.TaskSchedulingPrope
rties": {
          "prefix": "spring.task.scheduling",
          "properties": {
            "pool": {
              "size": "******"
            },
            "simple": {},
```

```
            "threadNamePrefix": "******",
            "shutdown": {
              "awaitTermination": "******"
            }
          },
          "inputs": {
            "pool": {
              "size": {}
            },
            "simple": {},
            "threadNamePrefix": {},
            "shutdown": {
              "awaitTermination": {}
            }
          }
        },
        "spring.reactor-
org.springframework.boot.autoconfigure.reactor.ReactorPropertie
s": {
          "prefix": "spring.reactor",
          "properties": {
            "contextPropagation": "******"
          },
          "inputs": {
            "contextPropagation": {}
          }
        },
        "management.endpoints.web.cors-
org.springframework.boot.actuate.autoconfigure.endpoint.web.Cor
sEndpointProperties": {
          "prefix": "management.endpoints.web.cors",
          "properties": {
            "allowedHeaders": [],
            "allowedMethods": [],
            "allowedOrigins": [],
            "maxAge": "******",
            "exposedHeaders": [],
            "allowedOriginPatterns": []
          },
          "inputs": {
            "allowedHeaders": [],
            "allowedMethods": [],
            "allowedOrigins": [],
            "maxAge": {},
            "exposedHeaders": [],
```

```
            "allowedOriginPatterns": []
          }
        },
        "management.health.diskspace-
org.springframework.boot.actuate.autoconfigure.system.DiskSpace
HealthIndicatorProperties": {
          "prefix": "management.health.diskspace",
          "properties": {
            "path": "******",
            "threshold": "******"
          },
          "inputs": {
            "path": {},
            "threshold": {}
          }
        },

"org.springdoc.core.properties.SwaggerUiOAuthProperties": {
          "prefix": "springdoc.swagger-ui.oauth",
          "properties": {
            "configParameters": {}
          },
          "inputs": {
            "configParameters": {}
          }
        },
        "spring.sql.init-
org.springframework.boot.autoconfigure.sql.init.SqlInitializati
onProperties": {
          "prefix": "spring.sql.init",
          "properties": {
            "mode": "******",
            "separator": "******",
            "platform": "******",
            "continueOnError": "******"
          },
          "inputs": {
            "mode": {},
            "separator": {},
            "platform": {},
            "continueOnError": {}
          }
        },
        "management.endpoint.env-
org.springframework.boot.actuate.autoconfigure.env.EnvironmentE
```

```
ndpointProperties": {
          "prefix": "management.endpoint.env",
          "properties": {
            "showValues": "******",
            "roles": []
          },
          "inputs": {
            "showValues": {},
            "roles": []
          }
        },
        "management.endpoint.logfile-
org.springframework.boot.actuate.autoconfigure.logging.LogFileW
ebEndpointProperties": {
          "prefix": "management.endpoint.logfile",
          "properties": {},
          "inputs": {}
        },
        "knife4j.basic-
com.github.xiaoymin.knife4j.spring.configuration.Knife4jHttpBas
ic": {
          "prefix": "knife4j.basic",
          "properties": {
            "enable": "******"
          },
          "inputs": {
            "enable": {}
          }
        },
        "server-
org.springframework.boot.autoconfigure.web.ServerProperties": {
          "prefix": "server",
          "properties": {
            "maxHttpRequestHeaderSize": "******",
            "reactive": {
              "session": {
                "timeout": "******"
              }
            },
            "undertow": {
              "maxHttpPostSize": "******",
              "eagerFilterInit": "******",
              "maxHeaders": "******",
              "maxCookies": "******",
              "allowEncodedSlash": "******",
```

```
            "decodeUrl": "******",
            "urlCharset": "******",
            "alwaysSetKeepAlive": "******",
            "preservePathOnForward": "******",
            "accesslog": {
              "enabled": "******",
              "pattern": "******",
              "prefix": "******",
              "suffix": "******",
              "dir": "******",
              "rotate": "******"
            },
            "threads": {},
            "options": {
              "socket": {},
              "server": {}
            }
          },
          "port": "******",
          "tomcat": {
            "accesslog": {
              "enabled": "******",
              "pattern": "******",
              "directory": "******",
              "prefix": "******",
              "suffix": "******",
              "checkExists": "******",
              "rotate": "******",
              "renameOnRotate": "******",
              "maxDays": "******",
              "fileDateFormat": "******",
              "ipv6Canonical": "******",
              "requestAttributesEnabled": "******",
              "buffered": "******"
            },
            "threads": {
              "max": "******",
              "minSpare": "******"
            },
            "backgroundProcessorDelay": "******",
            "maxHttpFormPostSize": "******",
            "maxSwallowSize": "******",
            "redirectContextRoot": "******",
            "useRelativeRedirects": "******",
            "uriEncoding": "******",
```

```
      "maxConnections": "******",
      "acceptCount": "******",
      "processorCache": "******",
      "maxKeepAliveRequests": "******",
      "additionalTldSkipPatterns": [],
      "relaxedPathChars": [],
      "relaxedQueryChars": [],
      "rejectIllegalHeader": "******",
      "resource": {
        "allowCaching": "******"
      },
      "mbeanregistry": {
        "enabled": "******"
      },
      "remoteip": {
        "internalProxies": "******",
        "protocolHeaderHttpsValue": "******",
        "hostHeader": "******",
        "portHeader": "******"
      },
      "maxHttpResponseHeaderSize": "******"
    },
    "servlet": {
      "contextParameters": {},
      "applicationDisplayName": "******",
      "registerDefaultServlet": "******"
    },
    "jetty": {
      "accesslog": {
        "enabled": "******",
        "format": "******",
        "retentionPeriod": "******",
        "append": "******"
      },
      "threads": {
        "acceptors": "******",
        "selectors": "******",
        "max": "******",
        "min": "******",
        "idleTimeout": "******"
      },
      "maxHttpFormPostSize": "******",
      "maxHttpResponseHeaderSize": "******",
      "maxConnections": "******"
    },
```

```
      "error": {
        "path": "******",
        "includeException": "******",
        "includeStacktrace": "******",
        "includeMessage": "******",
        "includeBindingErrors": "******",
        "whitelabel": {
          "enabled": "******"
        }
      },
      "shutdown": "******",
      "netty": {
        "h2cMaxContentLength": "******",
        "initialBufferSize": "******",
        "maxInitialLineLength": "******",
        "validateHeaders": "******"
      }
    },
    "inputs": {
      "maxHttpRequestHeaderSize": {},
      "reactive": {
        "session": {
          "timeout": {}
        }
      },
      "undertow": {
        "maxHttpPostSize": {},
        "eagerFilterInit": {},
        "maxHeaders": {},
        "maxCookies": {},
        "allowEncodedSlash": {},
        "decodeUrl": {},
        "urlCharset": {},
        "alwaysSetKeepAlive": {},
        "preservePathOnForward": {},
        "accesslog": {
          "enabled": {},
          "pattern": {},
          "prefix": {},
          "suffix": {},
          "dir": {},
          "rotate": {}
        },
        "threads": {},
        "options": {
```

```
                "socket": {},
                "server": {}
              }
            },
            "port": {
              "value": "******",
              "origin": "\"server.port\" from property source
\"commandLineArgs\""
            },
            "tomcat": {
              "accesslog": {
                "enabled": {},
                "pattern": {},
                "directory": {},
                "prefix": {},
                "suffix": {},
                "checkExists": {},
                "rotate": {},
                "renameOnRotate": {},
                "maxDays": {},
                "fileDateFormat": {},
                "ipv6Canonical": {},
                "requestAttributesEnabled": {},
                "buffered": {}
              },
              "threads": {
                "max": {},
                "minSpare": {}
              },
              "backgroundProcessorDelay": {},
              "maxHttpFormPostSize": {},
              "maxSwallowSize": {},
              "redirectContextRoot": {},
              "useRelativeRedirects": {},
              "uriEncoding": {},
              "maxConnections": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 13:31"
              },
              "acceptCount": {
                "value": "******",
                "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 15:28"
              },
```

```
      "processorCache": {},
      "maxKeepAliveRequests": {},
      "additionalTldSkipPatterns": [],
      "relaxedPathChars": [],
      "relaxedQueryChars": [],
      "rejectIllegalHeader": {},
      "resource": {
        "allowCaching": {}
      },
      "mbeanregistry": {
        "enabled": {}
      },
      "remoteip": {
        "internalProxies": {},
        "protocolHeaderHttpsValue": {},
        "hostHeader": {},
        "portHeader": {}
      },
      "maxHttpResponseHeaderSize": {}
    },
    "servlet": {
      "contextParameters": {},
      "applicationDisplayName": {},
      "registerDefaultServlet": {}
    },
    "jetty": {
      "accesslog": {
        "enabled": {},
        "format": {},
        "retentionPeriod": {},
        "append": {}
      },
      "threads": {
        "acceptors": {},
        "selectors": {},
        "max": {},
        "min": {},
        "idleTimeout": {}
      },
      "maxHttpFormPostSize": {},
      "maxHttpResponseHeaderSize": {},
      "maxConnections": {}
    },
    "error": {
      "path": {},
```

```
            "includeException": {},
            "includeStacktrace": {},
            "includeMessage": {},
            "includeBindingErrors": {},
            "whitelabel": {
              "enabled": {}
            }
          },
          "shutdown": {},
          "netty": {
            "h2cMaxContentLength": {},
            "initialBufferSize": {},
            "maxInitialLineLength": {},
            "validateHeaders": {}
          }
        }
      },
      "spring.servlet.multipart-
org.springframework.boot.autoconfigure.web.servlet.MultipartPro
perties": {
        "prefix": "spring.servlet.multipart",
        "properties": {
          "fileSizeThreshold": "******",
          "maxFileSize": "******",
          "maxRequestSize": "******",
          "strictServletCompliance": "******",
          "enabled": "******",
          "resolveLazily": "******"
        },
        "inputs": {
          "fileSizeThreshold": {},
          "maxFileSize": {
            "value": "******",
            "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 12:40"
          },
          "maxRequestSize": {
            "value": "******",
            "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 11:43"
          },
          "strictServletCompliance": {},
          "enabled": {},
          "resolveLazily": {}
        }
```

```
        },
        "dataSource": {
          "prefix": "spring.datasource.hikari",
          "properties": {
            "validationTimeout": "******",
            "hikariPoolMXBean": {},
            "minimumIdle": "******",
            "password": "******",
            "metricsTrackerFactory": {},
            "dataSourceProperties": {},
            "loginTimeout": "******",
            "autoCommit": "******",
            "connectionTimeout": "******",
            "poolName": "******",
            "initializationFailTimeout": "******",
            "readOnly": "******",
            "registerMbeans": "******",
            "healthCheckProperties": {},
            "isolateInternalQueries": "******",
            "leakDetectionThreshold": "******",
            "maxLifetime": "******",
            "keepaliveTime": "******",
            "allowPoolSuspension": "******",
            "idleTimeout": "******",
            "connectionTestQuery": "******",
            "driverClassName": "******",
            "jdbcUrl": "******",
            "maximumPoolSize": "******",
            "username": "******"
          },
          "inputs": {
            "validationTimeout": {},
            "hikariPoolMXBean": {},
            "minimumIdle": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 48:39"
            },
            "password": {},
            "metricsTrackerFactory": {},
            "dataSourceProperties": {},
            "loginTimeout": {},
            "autoCommit": {
              "value": "******",
              "origin": "class path resource
```

```
[application.properties] from watch-1.0-SNAPSHOT.jar - 50:38"
            },
            "connectionTimeout": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 53:45"
            },
            "poolName": {},
            "initializationFailTimeout": {},
            "readOnly": {},
            "registerMbeans": {},
            "healthCheckProperties": {},
            "isolateInternalQueries": {},
            "leakDetectionThreshold": {},
            "maxLifetime": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 52:39"
            },
            "keepaliveTime": {},
            "allowPoolSuspension": {},
            "idleTimeout": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 51:39"
            },
            "connectionTestQuery": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 54:48"
            },
            "driverClassName": {},
            "jdbcUrl": {},
            "maximumPoolSize": {
              "value": "******",
              "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 49:44"
            },
            "username": {}
          }
        }
      }
    }
  }
}
```

34.10. env 环境变量

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/env |jq
{
  "activeProfiles": [
    "prod"
  ],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": "******"
        }
      }
    },
    {
      "name": "commandLineArgs",
      "properties": {
        "logging.file.name": {
          "value": "******"
        },
        "server.port": {
          "value": "******"
        },
        "spring.profiles.active": {
          "value": "******"
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "systemProperties",
      "properties": {
        "java.specification.version": {
```

```
      "value": "******"
    },
    "sun.jnu.encoding": {
      "value": "******"
    },
    "java.class.path": {
      "value": "******"
    },
    "java.vm.vendor": {
      "value": "******"
    },
    "sun.arch.data.model": {
      "value": "******"
    },
    "java.vendor.url": {
      "value": "******"
    },
    "catalina.useNaming": {
      "value": "******"
    },
    "user.timezone": {
      "value": "******"
    },
    "org.jboss.logging.provider": {
      "value": "******"
    },
    "java.vm.specification.version": {
      "value": "******"
    },
    "os.name": {
      "value": "******"
    },
    "sun.java.launcher": {
      "value": "******"
    },
    "sun.boot.library.path": {
      "value": "******"
    },
    "sun.java.command": {
      "value": "******"
    },
    "jdk.debug": {
      "value": "******"
    },
    "sun.cpu.endian": {
```

```
      "value": "******"
    },
    "user.home": {
      "value": "******"
    },
    "user.language": {
      "value": "******"
    },
    "java.specification.vendor": {
      "value": "******"
    },
    "java.version.date": {
      "value": "******"
    },
    "java.home": {
      "value": "******"
    },
    "file.separator": {
      "value": "******"
    },
    "java.vm.compressedOopsMode": {
      "value": "******"
    },
    "line.separator": {
      "value": "******"
    },
    "java.specification.name": {
      "value": "******"
    },
    "java.vm.specification.vendor": {
      "value": "******"
    },
    "FILE_LOG_CHARSET": {
      "value": "******"
    },
    "java.awt.headless": {
      "value": "******"
    },
    "java.protocol.handler.pkgs": {
      "value": "******"
    },
    "sun.management.compiler": {
      "value": "******"
    },
    "java.runtime.version": {
```

```
      "value": "******"
    },
    "user.name": {
      "value": "******"
    },
    "stdout.encoding": {
      "value": "******"
    },
    "path.separator": {
      "value": "******"
    },
    "os.version": {
      "value": "******"
    },
    "java.runtime.name": {
      "value": "******"
    },
    "file.encoding": {
      "value": "******"
    },
    "java.vm.name": {
      "value": "******"
    },
    "LOG_FILE": {
      "value": "******"
    },
    "java.vendor.url.bug": {
      "value": "******"
    },
    "java.io.tmpdir": {
      "value": "******"
    },
    "catalina.home": {
      "value": "******"
    },
    "com.zaxxer.hikari.pool_number": {
      "value": "******"
    },
    "java.version": {
      "value": "******"
    },
    "user.dir": {
      "value": "******"
    },
    "os.arch": {
```

```
        "value": "******"
      },
      "java.vm.specification.name": {
        "value": "******"
      },
      "PID": {
        "value": "******"
      },
      "CONSOLE_LOG_CHARSET": {
        "value": "******"
      },
      "catalina.base": {
        "value": "******"
      },
      "native.encoding": {
        "value": "******"
      },
      "java.library.path": {
        "value": "******"
      },
      "stderr.encoding": {
        "value": "******"
      },
      "java.vm.info": {
        "value": "******"
      },
      "java.vendor": {
        "value": "******"
      },
      "java.vm.version": {
        "value": "******"
      },
      "sun.io.unicode.encoding": {
        "value": "******"
      },
      "java.class.version": {
        "value": "******"
      },
      "LOGGED_APPLICATION_NAME": {
        "value": "******"
      }
    }
  },
  {
    "name": "systemEnvironment",
```

```
      "properties": {
        "HOME": {
          "value": "******",
          "origin": "System Environment Property \"HOME\""
        },
        "PATH": {
          "value": "******",
          "origin": "System Environment Property \"PATH\""
        },
        "JAVA_VERSION": {
          "value": "******",
          "origin": "System Environment Property
\"JAVA_VERSION\""
        },
        "JAVA_HOME": {
          "value": "******",
          "origin": "System Environment Property \"JAVA_HOME\""
        },
        "TZ": {
          "value": "******",
          "origin": "System Environment Property \"TZ\""
        },
        "JAVA_OPTS": {
          "value": "******",
          "origin": "System Environment Property \"JAVA_OPTS\""
        },
        "LANG": {
          "value": "******",
          "origin": "System Environment Property \"LANG\""
        },
        "HOSTNAME": {
          "value": "******",
          "origin": "System Environment Property \"HOSTNAME\""
        }
      }
    },
    {
      "name": "Config resource 'class path resource
[application-prod.properties]' via location
'optional:classpath:/'",
      "properties": {
        "spring.application.name": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 1:25"
```

```
        },
        "spring.datasource.driver-class-name": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 3:37"
        },
        "spring.datasource.url": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 4:23"
        },
        "spring.datasource.username": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 5:28"
        },
        "spring.datasource.password": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 6:28"
        },
        "spring.data.redis.host": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 8:24"
        },
        "spring.data.redis.port": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 9:24"
        },
        "spring.data.redis.password": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 10:28"
        },
        "spring.data.redis.database": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 11:28"
        },
        "spring.data.redis.timeout": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 12:27"
```

```
        },
        "aliyun.oss.access_key_id": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 14:26"
        },
        "aliyun.oss.secret_access_key": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 15:30"
        },
        "aliyun.oss.endpoint": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 16:21"
        },
        "aliyun.oss.bucket_name": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 17:24"
        },
        "aliyun.oss.callback_url": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 18:25"
        },
        "mqtt.broker": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 20:13"
        },
        "mqtt.username": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 21:15"
        },
        "mqtt.password": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 22:15"
        },
        "mqtt.topic.prefix": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 23:19"
```

```
        },
        "baidu.url": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 25:11"
        },
        "baidu.appid": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 26:13"
        },
        "baidu.apikey": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 27:14"
        },
        "baidu.secretkey": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 28:17"
        },
        "baidu.appsecret": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 29:17"
        },
        "baidu.censor.appid": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 30:20"
        },
        "baidu.censor.appsecret": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 31:24"
        },
        "xfyun.appid": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 33:13"
        },
        "xfyun.stt.apikey": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 34:18"
```

```
        },
        "xfyun.stt.length": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 35:18"
        },
        "xfyun.tts.apikey": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 36:18"
        },
        "xfyun.tts.apisecret": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 37:21"
        },
        "xfyun.tts.vcn": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 42:15"
        },
        "xfyun.tts.pitch": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 43:17"
        },
        "xfyun.tts.speed": {
          "value": "******",
          "origin": "class path resource [application-
prod.properties] from watch-1.0-SNAPSHOT.jar - 44:17"
        }
      }
    },
    {
      "name": "Config resource 'class path resource
[application.properties]' via location 'optional:classpath:/'",
      "properties": {
        "spring.application.name": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 2:25"
        },
        "spring.profiles.active": {
          "value": "******",
          "origin": "class path resource
```

```
[application.properties] from watch-1.0-SNAPSHOT.jar - 3:24"
        },
        "server.port": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 10:13"
        },
        "spring.servlet.multipart.max-request-size": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 11:43"
        },
        "spring.servlet.multipart.max-file-size": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 12:40"
        },
        "server.tomcat.max-connections": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 13:31"
        },
        "server.tomcat.max-threads": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 14:27"
        },
        "server.tomcat.accept-count": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 15:28"
        },
        "server.tomcat.min-spare-threads": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 16:33"
        },
        "endpoints.metrics.enabled": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 18:27"
        },
        "management.endpoints.jmx.exposure.include": {
          "value": "******",
          "origin": "class path resource
```

```
[application.properties] from watch-1.0-SNAPSHOT.jar - 19:43"
        },
        "management.endpoints.web.exposure.include": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 20:43"
        },
        "management.endpoints.health.show-details": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 21:42"
        },
        "spring.task.execution.pool.core-size": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 23:38"
        },
        "spring.task.execution.pool.max-size": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 24:37"
        },
        "spring.task.execution.pool.queue-capacity": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 25:43"
        },
        "spring.task.execution.pool.keep-alive": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 26:39"
        },
        "spring.task.execution.pool.allow-core-thread-timeout":
{
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 27:54"
        },
        "spring.task.execution.shutdown.await-termination": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 28:50"
        },
        "spring.task.execution.shutdown.await-termination-
period": {
```

```
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 29:57"
        },
        "spring.task.execution.thread-name-prefix": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 30:42"
        },
        "knife4j.enable": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 32:16"
        },
        "knife4j.setting.language": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 33:26"
        },
        "springdoc.swagger-ui.path": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 35:27"
        },
        "springdoc.swagger-ui.tags-sorter": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 36:34"
        },
        "springdoc.swagger-ui.operations-sorter": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 37:40"
        },
        "springdoc.api-docs.path": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 38:25"
        },
        "spring.jpa.open-in-view": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 41:25"
        },
        "spring.jpa.hibernate.ddl-auto": {
```

```
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 43:31"
        },
        "spring.datasource.driver-class-name": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 45:37"
        },
        "spring.datasource.hikari.minimum-idle": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 48:39"
        },
        "spring.datasource.hikari.maximum-pool-size": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 49:44"
        },
        "spring.datasource.hikari.auto-commit": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 50:38"
        },
        "spring.datasource.hikari.idle-timeout": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 51:39"
        },
        "spring.datasource.hikari.max-lifetime": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 52:39"
        },
        "spring.datasource.hikari.connection-timeout": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 53:45"
        },
        "spring.datasource.hikari.connection-test-query": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 54:48"
        },
        "spring.data.redis.lettuce.pool.max-active": {
```

```
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 56:43"
        },
        "spring.data.redis.lettuce.pool.max-wait": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 57:41"
        },
        "spring.data.redis.lettuce.pool.max-idle": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 58:41"
        },
        "spring.data.redis.lettuce.pool.min-idle": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 59:41"
        },
        "spring.cache.type": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 60:19"
        },
        "spring.cache.redis.time-to-live": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 61:33"
        },
        "spring.cache.redis.cache-null-values": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 62:38"
        },
        "stablediffusion.url": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 64:21"
        },
        "stablediffusion.negative_prompt": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 66:0"
        },
        "stablediffusion.sampler_index": {
```

```
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 66:31"
        },
        "stablediffusion.seed": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 67:22"
        },
        "stablediffusion.steps": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 68:23"
        },
        "stablediffusion.width": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 69:23"
        },
        "stablediffusion.height": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 70:24"
        },
        "stablediffusion.cfg_scale": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 71:27"
        },
        "chatgpt.url": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 73:13"
        },
        "chatgpt.username": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 74:18"
        },
        "chatgpt.password": {
          "value": "******",
          "origin": "class path resource
[application.properties] from watch-1.0-SNAPSHOT.jar - 75:18"
        }
      }
```

```
    }
  ]
}
```

34.11. logfile 日志

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/logfile | tail -n 5
2024-01-01T10:26:57.967+08:00  INFO 1 --- [watch-production]
[http-nio-8080-exec-3] c.netkiller.service.SessionStatusService
: SessionStatus(id=15476, session=06c27cc8-dbc6-4386-bd69-
e555fa3715cf, progress=5.创建会话, description=画一张猫,
ctime=null)
2024-01-01T10:26:58.091+08:00  INFO 1 --- [watch-production]
[http-nio-8080-exec-3] cn.netkiller.service.BaiduService
:
{"refresh_token":"25.3fe5248bc2006526a59a10973eaa4320.315360000
.2019436018.282335-
45847061","expires_in":2592000,"session_key":"9mzdA5O3tx06PF7hF
uzwL5FsbQDWq0LxF4mpdWWl5josf1ZE0IvZOXsrDS471dAz8F6aq7YbivAAZ7Ws
rliSyZ2Fwtr2iA==","access_token":"24.89b1691bb1d8e1437671365521
b43a2d.2592000.1706668018.282335-45847061","scope":"public
brain_all_scope brain_nlp_sentiment_classify brain_nlp_emotion
solution_face brain_mt_texttrans wise_adapt lebo_resource_base
lightservice_public hetu_basic lightcms_map_poi kaidian_kaidian
ApsMisTest_Test\u6743\u9650 vis-classify_flower
lpq_\u5f00\u653e cop_helloScope ApsMis_fangdi_permission
smartapp_snsapi_base smartapp_mapp_dev_manage iop_autocar
oauth_tp_app smartapp_smart_game_openapi oauth_sessionkey
smartapp_swanid_verify smartapp_opensource_openapi
smartapp_opensource_recapi fake_face_detect_\u5f00\u653eScope
vis-ocr_\u865a\u62df\u4eba\u7269\u52a9\u7406 idl-
video_\u865a\u62df\u4eba\u7269\u52a9\u7406 smartapp_component
smartapp_search_plugin avatar_video_test b2b_tp_openapi
b2b_tp_openapi_online
smartapp_gov_aladin_to_xcx","session_secret":"0bca5fb59addf566e
b2e9038afb09883"}

2024-01-01T10:26:58.468+08:00  INFO 1 --- [watch-production]
```

```
[http-nio-8080-exec-3] cn.netkiller.service.BaiduService
: Translate: {"result":{"from":"zh","trans_result":
[{"dst":"Draw a cat","src":"画一张
猫"}],"to":"en"},"log_id":1741647171642597969}
2024-01-01T10:26:58.499+08:00  INFO 1 --- [watch-production]
[http-nio-8080-exec-3] c.netkiller.service.SessionStatusService
: SessionStatus(id=15477, session=06c27cc8-dbc6-4386-bd69-
e555fa3715cf, progress=6.翻译成功, description=Draw a cat,
ctime=null)
```

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/loggers/cn.netkiller.comp
onent
{"effectiveLevel":"INFO"}%

neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/loggers | tail -n 5
{"levels":
["OFF","ERROR","WARN","INFO","DEBUG","TRACE"],"loggers":
{"ROOT":
{"configuredLevel":"INFO","effectiveLevel":"INFO"},"_org":
{"effectiveLevel":"INFO"},"_org.springframework":
{"effectiveLevel":"INFO"},"_org.springframework.web":
{"effectiveLevel":"INFO"},"_org.springframework.web.servlet":
{"effectiveLevel":"INFO"},"_org.springframework.web.servlet.Han
dlerMapping":
{"effectiveLevel":"INFO"},"_org.springframework.web.servlet.Han
dlerMapping.Mappings":{"effectiveLevel":"INFO"},"cn":
{"effectiveLevel":"INFO"},"cn.netkiller":
{"effectiveLevel":"INFO"},"cn.netkiller.Application":
{"effectiveLevel":"INFO"},"cn.netkiller.ai":
{"effectiveLevel":"INFO"},"cn.netkiller.ai.xfyun":
{"effectiveLevel":"INFO"},"cn.netkiller.ai.xfyun.SpeechRecogniz
erService":
{"effectiveLevel":"INFO"},"cn.netkiller.ai.xfyun.SpeechSynthesi
zerService":{"effectiveLevel":"INFO"},"cn.netkiller.component":
{"effectiveLevel":"INFO"},"cn.netkiller.component.AudioService"
:
```

34.12. threaddump 线程信息

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/threaddump | jq | grep
threadName
      "threadName": "Reference Handler",
      "threadName": "Finalizer",
      "threadName": "Signal Dispatcher",
      "threadName": "Notification Thread",
      "threadName": "Common-Cleaner",
      "threadName": "Cleaner-0",
      "threadName": "Catalina-utility-1",
      "threadName": "Catalina-utility-2",
      "threadName": "container-0",
      "threadName": "lettuce-timer-3-1",
      "threadName": "mysql-cj-abandoned-connection-cleanup",
      "threadName": "HikariPool-1 housekeeper",
      "threadName": "http-nio-8080-exec-1",
      "threadName": "http-nio-8080-exec-2",
      "threadName": "http-nio-8080-exec-3",
      "threadName": "http-nio-8080-exec-4",
      "threadName": "http-nio-8080-exec-5",
      "threadName": "http-nio-8080-exec-7",
      "threadName": "http-nio-8080-exec-8",
      "threadName": "http-nio-8080-exec-10",
      "threadName": "http-nio-8080-Poller",
      "threadName": "http-nio-8080-Acceptor",
      "threadName": "DestroyJavaVM",
      "threadName": "lettuce-epollEventLoop-4-1",
      "threadName": "lettuce-eventExecutorLoop-1-1",
      "threadName": "lettuce-eventExecutorLoop-1-2",
      "threadName": "lettuce-eventExecutorLoop-1-3",
      "threadName": "lettuce-eventExecutorLoop-1-4",
      "threadName": "task-11",
      "threadName": "task-12",
      "threadName": "task-13",
      "threadName": "task-14",
      "threadName": "task-15",
      "threadName": "task-16",
      "threadName": "task-17",
      "threadName": "task-18",
      "threadName": "task-19",
```

```
      "threadName": "task-20",
      "threadName": "MQTT Rec: netkiller-1704016635545",
      "threadName": "MQTT Snd: netkiller-1704016635545",
      "threadName": "MQTT Call: netkiller-1704016635545",
      "threadName": "MQTT Ping: netkiller-1704016635545",
      "threadName": "http-nio-8080-exec-14",
      "threadName": "http-nio-8080-exec-16",
      "threadName": "boundedElastic-evictor-1",
      "threadName": "lettuce-epollEventLoop-4-2",
```

34.13. 计划任务

```
http://www.netkiller.cn:8080/actuator/scheduledtasks
```

34.14. metrics

```
neo@MacBook-Pro-Neo ~/w/Architect (master)> curl -s
https://www.netkiller.cn/actuator/metrics/ | jq
{
  "names": [
    "jvm.threads.states",
    "process.files.max",
    "jvm.gc.memory.promoted",
    "hikaricp.connections.max",
    "hikaricp.connections.min",
    "jvm.memory.committed",
    "system.load.average.1m",
    "http.server.requests",
    "jvm.memory.used",
    "jvm.gc.max.data.size",
    "jdbc.connections.max",
    "jdbc.connections.min",
    "hikaricp.connections.usage",
    "jvm.gc.pause",
    "system.cpu.count",
```

```
    "hikaricp.connections.timeout",
    "tomcat.global.sent",
    "jvm.buffer.memory.used",
    "tomcat.sessions.created",
    "jvm.memory.max",
    "jvm.threads.daemon",
    "hikaricp.connections.acquire",
    "system.cpu.usage",
    "jvm.gc.memory.allocated",
    "tomcat.global.request.max",
    "tomcat.global.request",
    "tomcat.sessions.expired",
    "jvm.threads.live",
    "jvm.threads.peak",
    "tomcat.global.received",
    "process.uptime",
    "tomcat.sessions.rejected",
    "process.cpu.usage",
    "tomcat.threads.config.max",
    "jvm.classes.loaded",
    "jvm.classes.unloaded",
    "tomcat.global.error",
    "tomcat.sessions.active.current",
    "tomcat.sessions.alive.max",
    "jvm.gc.live.data.size",
    "log4j2.events",
    "hikaricp.connections.idle",
    "tomcat.threads.current",
    "hikaricp.connections.pending",
    "process.files.open",
    "jvm.buffer.count",
    "hikaricp.connections",
    "jvm.buffer.total.capacity",
    "tomcat.sessions.active.max",
    "hikaricp.connections.active",
    "hikaricp.connections.creation",
    "tomcat.threads.busy",
    "process.start.time"
  ]
}
```

```
neo@MacBook-Pro-Neo ~/w/Architect (master)> curl -s
https://www.netkiller.cn/actuator/metrics/tomcat.threads.config
.max |jq
{
  "name": "tomcat.threads.config.max",
  "description": null,
  "baseUnit": "threads",
  "measurements": [
    {
      "statistic": "VALUE",
      "value": 4096
    }
  ],
  "availableTags": [
    {
      "tag": "name",
      "values": [
        "http-nio-8080"
      ]
    }
  ]
}
neo@MacBook-Pro-Neo ~/w/Architect (master)> curl -s
https://www.netkiller.cn/actuator/metrics/tomcat.threads.curren
t |jq
{
  "name": "tomcat.threads.current",
  "description": null,
  "baseUnit": "threads",
  "measurements": [
    {
      "statistic": "VALUE",
      "value": 24
    }
  ],
  "availableTags": [
    {
      "tag": "name",
      "values": [
        "http-nio-8080"
      ]
    }
  ]
}
```

34.15. 控制器映射 URL

```
neo@MacBook-Pro-M2 ~ % curl -s
http://www.netkiller.cn:8080/actuator/mappings |jq
```

34.16. 自定义监控指标

```
package cn.netkiller.config;

import
org.springframework.boot.actuate.endpoint.annotation.Endpoint
;
import
org.springframework.boot.actuate.endpoint.annotation.ReadOper
ation;
import org.springframework.context.annotation.Configuration;

import java.util.*;

@Configuration
@Endpoint(id = "netkiller")
public class TestEndpoint {
    @ReadOperation
    public Map<String, Object> threadPoolsMetric() {
        Map<String, Object> metricMap = new HashMap<>();
        List<Map> threadPools = new ArrayList<>();
        Map<String, Object> poolInfo = new HashMap<>();
        poolInfo.put("thread.pool.name", "netkiller");
        poolInfo.put("thread.pool.core.size", 100);
        poolInfo.put("thread.pool.time", new Date());
        threadPools.add(poolInfo);
        metricMap.put("netkiller", threadPools);
        return metricMap;
```

```
    }
}
```

验证

```
neo@MacBook-Pro-M2 ~> curl -s
http://www.netkiller.cn:8080/actuator/netkiller | jq
{
  "netkiller": [
    {
      "thread.pool.time": "2023-04-24T09:08:14.407+00:00",
      "thread.pool.core.size": 100,
      "thread.pool.name": "netkiller"
    }
  ]
}
```

35. String boot with RestTemplate

RestTemplate - Spring Restful

RestTemplate 是 Spring Restful Client 用于调用restful接口

首先我要禁告各位，Spring发展过程中，每个版本都有一定差异。如果你做实验失败后在网上搜索答案，切记看一下版本号还有文章帖子的发布时间。否则你可能按照Spring3配置方法去Spring4。

@RestController 默认返回 @ResponseBody， 所以@ResponseBody可加可不加

35.1. RestTemplate Example

pom.xml

Maven 增加 jackson 开发包

```
                <dependency>
<groupId>com.fasterxml.jackson.dataformat</groupId>
                        <artifactId>jackson-dataformat-
xml</artifactId>
                </dependency>
                <dependency>
<groupId>com.fasterxml.jackson.core</groupId>
                        <artifactId>jackson-core</artifactId>
                </dependency>
                <dependency>
<groupId>com.fasterxml.jackson.core</groupId>
                        <artifactId>jackson-
databind</artifactId>
                </dependency>
```

```
                <dependency>

<groupId>com.fasterxml.jackson.core</groupId>
                        <artifactId>jackson-
annotations</artifactId>
                </dependency>
```

web.xml

url-pattern匹配中增加*.xml跟*.json

```
        <servlet>
        <servlet-name>springframework</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>springframework</servlet-name>
        <url-pattern>/welcome.jsp</url-pattern>
        <url-pattern>/welcome.html</url-pattern>
        <url-pattern>*.json</url-pattern>
        <url-pattern>*.xml</url-pattern>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>
```

springframework.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
```

```
xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:mongo="http://www.springframework.org/schema/data/mongo
" xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-
beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-
mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context.xsd
http://www.springframework.org/schema/data/mongo
http://www.springframework.org/schema/data/mongo/spring-
mongo-1.5.xsd
        ">

        <mvc:resources location="/images/"
mapping="/images/**" />
        <mvc:resources location="/css/" mapping="/css/**" />
        <mvc:resources location="/js/" mapping="/js/**" />
        <mvc:resources location="/zt/" mapping="/zt/**" />
        <mvc:resources location="/sm/" mapping="/sm/**" />
        <mvc:resources location="/module/"
mapping="/module/**" />

        <context:component-scan base-
package="cn.netkiller.controller">

        </context:component-scan>
        <context:annotation-config />
        <mvc:annotation-driven />

        <bean id="viewResolver"
class="org.springframework.web.servlet.view.UrlBasedViewResol
ver">
                <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView" />
                <property name="prefix" value="/WEB-INF/jsp/"
/>
                <property name="suffix" value=".jsp" />
```

```
        </bean>


        <bean id="configuracion"
class="org.springframework.beans.factory.config.PropertyPlace
holderConfigurer">
                <property name="location"
value="classpath:resources/development.properties" />
        </bean>

        <!-- Redis Connection Factory -->
        <bean id="jedisConnFactory"
class="org.springframework.data.redis.connection.jedis.JedisC
onnectionFactory" p:host-name="192.168.2.1" p:port="6379"
p:use-pool="true" />

        <!-- redis template definition -->
        <bean id="redisTemplate"
class="org.springframework.data.redis.core.RedisTemplate"
p:connection-factory-ref="jedisConnFactory" />

        <mongo:db-factory id="mongoDbFactory"
host="${mongo.host}" port="${mongo.port}"
dbname="${mongo.database}" />
        <!-- username="${mongo.username}"
password="${mongo.password}" -->

        <bean id="mongoTemplate"
class="org.springframework.data.mongodb.core.MongoTemplate">
                <constructor-arg name="mongoDbFactory"
ref="mongoDbFactory" />
        </bean>

        <mongo:mapping-converter id="converter" db-factory-
ref="mongoDbFactory" />
        <bean id="gridFsTemplate"
class="org.springframework.data.mongodb.gridfs.GridFsTemplate
">
                <constructor-arg ref="mongoDbFactory" />
                <constructor-arg ref="converter" />
        </bean>

</beans>
```

RestController

```
package cn.netkiller.controller;

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestHeader;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.ResponseStatus;
import
org.springframework.web.bind.annotation.RestController;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ResponseBody;

import cn.netkiller.pojo.Message;

@RestController
@RequestMapping("/rest")
public class TestRestController {

        public TrackerRestController() {
                // TODO Auto-generated constructor stub
        }

        @RequestMapping("welcome")
        @ResponseStatus(HttpStatus.OK)
        public String welcome() {
                return "Welcome to RestTemplate Example.";
        }

        @RequestMapping(value = "test", method =
RequestMethod.GET, produces = { "application/xml",
"application/json" })
        @ResponseStatus(HttpStatus.OK)
        public @ResponseBody Message
```

```
test(@RequestHeader(value = "accept") String accept) {
                Message message = new Message();
                message.setTitle("test");
                message.setText("Helloworld!!!");
                System.out.println("accept: " + accept);
                System.out.println(message.toString());
                return message;
        }

        @RequestMapping("message/{name}")
        public ResponseEntity<Message> message(@PathVariable
String name) {
                Message msg = new Message();
                msg.setTitle(name);
                return new ResponseEntity<Message>(msg,
HttpStatus.OK);
        }

        @RequestMapping(value = "create", method =
RequestMethod.POST, produces = { "application/xml",
"application/json" })
        public ResponseEntity<Tracker> create(@RequestBody
Tracker tracker) {
                this.mongoTemplate.insert(tracker);
                return new ResponseEntity<Tracker>(tracker,
HttpStatus.OK);
        }

        @RequestMapping(value = "read", method =
RequestMethod.GET, produces = { "application/xml",
"application/json" })
        @ResponseStatus(HttpStatus.OK)
        public ArrayList<Tracker> read() {

                ArrayList<Tracker> trackers =
(ArrayList<Tracker>) mongoTemplate.findAll(Tracker.class);
                return trackers;
        }
}
```

POJO

```
package cn.netkiller.pojo;

import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Message {

        String title;
    String text;

        public Message() {
                // TODO Auto-generated constructor stub
        }

        //@XmlElement
        @XmlAttribute
        public String getTitle() {
                return title;
        }
        public void setTitle(String title) {
                this.title = title;
        }

        //@XmlElement
        @XmlAttribute
        public String getText() {
                return text;
        }
        public void setText(String text) {
                this.text = text;
        }
        @Override
        public String toString() {
                return "Message [title=" + title + ", text="
+ text + "]";
        }

}
```

在控制器中完整实例

```
package api.web;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.client.RestTemplate;

import api.domain.City;
import api.repository.CityRepository;

@Controller
public class IndexController {

        @Autowired
        private CityRepository repository;

        // Spring RESTFul Client

        @RequestMapping("/restful/get")
        @ResponseBody
        public String restfulGet() {
                RestTemplate restTemplate = new
RestTemplate();
                String text =
restTemplate.getForObject("http://inf.netkiller.cn/detail/htm
l/2/2/42564.html", String.class);
                return text;
        }

        @RequestMapping("/restful/get/{id}")
```

```
        @ResponseBody
        private static String restfulGetId(@PathVariable
String id) {
                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("tid", "2");
                params.put("cid", "2");
                params.put("id", id);
                RestTemplate restTemplate = new
RestTemplate();
                String result =
restTemplate.getForObject(uri, String.class, params);

                return (result);
        }

        @RequestMapping("/restful/post/{id}")
        @ResponseBody
        private static String restfullPost(@PathVariable
String id) {

                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("tid", "2");
                params.put("cid", "2");
                params.put("id", id);

                City city = new City("Shenzhen",
"Guangdong");

                RestTemplate restTemplate = new
RestTemplate();
                String result =
restTemplate.postForObject(uri, city, String.class, params);
                return result;
        }

        @RequestMapping("/restful/put/{id}")
        private static void restfulPut(@PathVariable String
```

```
id) {
                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("id", id);

                City city = new City("Shenzhen",
"Guangdong");

                RestTemplate restTemplate = new
RestTemplate();
                restTemplate.put(uri, city, params);
        }

        @RequestMapping("/restful/delete/{id}")
        private static void restfulDelete(@PathVariable
String id) {
                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("id", id);

                RestTemplate restTemplate = new
RestTemplate();
                restTemplate.delete(uri, params);
        }
}
```

测试

```
neo@netkiller:~/www.netkiller.cn$ curl
http://172.16.0.1:8080/spring4/rest/welcome.html
Welcome to RestTemplate Example.
```

```
neo@netkiller:~/www.netkiller.cn$ curl
http://172.16.0.1:8080/spring4/rest/test.json
{"title":"test","text":"Helloworld!!!"}

neo@netkiller:~/www.netkiller.cn$ curl
http://172.16.0.1:8080/spring4/rest/test.xml
<Message xmlns=""><title>test</title><text>Helloworld!!!
</text></Message>

neo@netkiller:~/www.netkiller.cn$ curl -i -H "Accept:
application/json" -H "Content-Type: application/json" -X POST
-d '{"login":"neo",
"unique":"356770257607079474","hostname":"www.example.com","r
eferrer":"http://www.netkiller.cn","href":"http://www.netkill
er.cn"}' http://172.16.0.1:8080/spring4/rest/create.json
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Tue, 21 Jun 2016 03:08:26 GMT

{"name":"neo","unique":"356770257607079474","hostname":"www.n
etkiller.cn","referrer":"http://www.netkiller.cn","href":"htt
p://www.netkiller.cn"}
```

35.2. GET 操作

返回字符串

```
        @RequestMapping("/restful/get")
        @ResponseBody
        public String restfulGet() {
                RestTemplate restTemplate = new
RestTemplate();
                String text =
restTemplate.getForObject("http://inf.netkiller.cn/detail/htm
l/2/2/42564.html", String.class);
                return text;
```

```
        }
```

传递 GET 参数

```
        @RequestMapping("/restful/get/{id}")
        @ResponseBody
        private static String restfulGetId(@PathVariable
String id) {
                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("tid", "2");
                params.put("cid", "2");
                params.put("id", id);
                RestTemplate restTemplate = new
RestTemplate();
                String result =
restTemplate.getForObject(uri, String.class, params);

                return (result);
        }
```

35.3. POST 操作

postForObject

传递对象

```
        @RequestMapping("/restful/post/{id}")
        @ResponseBody
```

```
        private static String restfullPost(@PathVariable
String id) {

                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("tid", "2");
                params.put("cid", "2");
                params.put("id", id);

                City city = new City("Shenzhen",
"Guangdong");

                RestTemplate restTemplate = new
RestTemplate();
                String result =
restTemplate.postForObject(uri, city, String.class, params);
                return result;
        }
```

传递数据结构 **MultiValueMap**

```
        @RequestMapping("/findByMobile")
        public String findByMobile() {

System.out.println("*****************************findByMobile*
*****************************");

                final String uri =
"http://www.netkiller.cn/account/getMemberByMobile.json";
                MultiValueMap<String, String> map = new
LinkedMultiValueMap<String, String>();
                try {

                        map.add("prefix", "86");
                        map.add("mobile", "13698041116");
                        map.add("_pretty_", "false");
```

```
                } catch (Exception e) {
                        e.printStackTrace();
                }

                RestTemplate restTemplate = new
RestTemplate();
                String result =
restTemplate.postForObject(uri, map, String.class);

                System.out.println(map.toString());
                System.out.println(result);

                return result;
        }
```

postForEntity

```
        @RequestMapping("/findByMobile")
        @ResponseBody
        public String findByMobile() {

System.out.println("****************************findByMobile*
****************************");

                final String uri =
"https://www.netkiller.cn/account/getMemberByMobile";
                MultiValueMap<String, String> map = new
LinkedMultiValueMap<String, String>();
                try {

                        map.add("prefix", "86");
                        map.add("mobile", "13698041116");
                        map.add("args", "[]");
                        map.add("_pretty_", "false");

                } catch (Exception e) {
                        e.printStackTrace();
                }
```

```
                RestTemplate restTemplate = new
RestTemplate();
                ResponseEntity<String> response =
restTemplate.postForEntity(uri, map, String.class);
                System.out.println(map.toString());
                System.out.println();
                return response.getBody();
        }
```

35.4. PUT 操作

```
        @RequestMapping("/restful/put/{id}")
        private static void restfulPut(@PathVariable String
id) {
                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("id", id);

                City city = new City("Shenzhen",
"Guangdong");

                RestTemplate restTemplate = new
RestTemplate();
                restTemplate.put(uri, city, params);
        }
```

35.5. Delete 操作

```
        @RequestMapping("/restful/delete/{id}")
```

```
        private static void restfulDelete(@PathVariable
String id) {
                final String uri =
"http://inf.netkiller.cn/detail/html/{tid}/{cid}/{id}.html";

                Map<String, String> params = new
HashMap<String, String>();
                params.put("id", id);

                RestTemplate restTemplate = new
RestTemplate();
                restTemplate.delete(uri, params);
        }
```

35.6. 上传文件

```
package cn.netkiller.file;

import org.springframework.core.io.FileSystemResource;
import org.springframework.core.io.Resource;
import org.springframework.http.*;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestTemplate;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;

public class UploadClient {

    public static void main(String[] args) throws IOException
{
        MultiValueMap<String, Object> bodyMap = new
LinkedMultiValueMap<>();
        bodyMap.add("user-file", getUserFileResource());
        HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.MULTIPART_FORM_DATA);
```

```
        HttpEntity<MultiValueMap<String, Object>>
requestEntity = new HttpEntity<>(bodyMap, headers);

        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> response =
restTemplate.exchange("http://localhost:8080/upload",
HttpMethod.POST, requestEntity, String.class);
        System.out.println("response status: " +
response.getStatusCode());
        System.out.println("response body: " +
response.getBody());
    }

    public static Resource getUserFileResource() throws
IOException {
        //todo replace tempFile with a real file
        Path tempFile = Files.createTempFile("hello",
".txt");
        Files.write(tempFile, "Helloworld,
http://www.netkiller.cn".getBytes());
        System.out.println("uploading: " + tempFile);
        File file = tempFile.toFile();
        //to upload in-memory bytes use ByteArrayResource
instead
        return new FileSystemResource(file);
    }
}
```

35.7. HTTP Auth

Client

```
HttpClient client = new HttpClient();
UsernamePasswordCredentials credentials = new
UsernamePasswordCredentials("your_user","your_password");
client.getState().setCredentials(new AuthScope("thehost",
9090, AuthScope.ANY_REALM), credentials);
CommonsClientHttpRequestFactory commons = new
```

```
CommonsClientHttpRequestFactory(client);

RestTemplate template = new RestTemplate(commons);
Example results =
template.getForObject("http://www.netkiller.cn:9090/foo.json"
, Example.class);
```

35.8. PKCS12

```
package example.controller;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.KeyManagementException;
import java.security.KeyStore;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;
import java.security.UnrecoverableKeyException;
import java.security.cert.CertificateException;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLContext;

import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContextBuilder;
import org.apache.http.ssl.SSLContexts;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import
org.springframework.http.client.HttpComponentsClientHttpReque
```

```
stFactory;
import
org.springframework.security.oauth2.client.OAuth2RestOperatio
ns;
import
org.springframework.security.oauth2.common.OAuth2AccessToken;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.client.RestTemplate;

@Controller
public class TestController {
        @Autowired
        private OAuth2RestOperations restTemplate;

        @GetMapping("/")
        @ResponseBody
        public String index() {
                OAuth2AccessToken token =
restTemplate.getAccessToken();
                System.out.println(token.getValue());
                String tmp =
restTemplate.getForObject("http://api.alpha.netkiller.cn/",
String.class);
                System.out.println(tmp);
                return tmp;
        }

        @GetMapping("/ssl")
        @ResponseBody
        public String ssl() throws KeyManagementException,
NoSuchAlgorithmException, KeyStoreException {
                String url =
"https://api.alpha.netkiller.cn/";

                SSLContext sslcontext =
SSLContexts.custom().loadTrustMaterial(null, (chain,
authType) -> true).build();
                SSLConnectionSocketFactory sslsf = new
SSLConnectionSocketFactory(sslcontext, new String[] { "TLSv1"
}, null, new NoopHostnameVerifier());
                CloseableHttpClient httpClient =
HttpClients.custom().setSSLSocketFactory(sslsf).build();
                HttpComponentsClientHttpRequestFactory
```

```
httpComponentsClientHttpRequestFactory = new
HttpComponentsClientHttpRequestFactory(httpClient);

httpComponentsClientHttpRequestFactory.setConnectTimeout(6000
0);

httpComponentsClientHttpRequestFactory.setReadTimeout(180000)
;

                final RestTemplate restTemplate = new
RestTemplate(httpComponentsClientHttpRequestFactory);

                HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_JSON);
                headers.set("Authorization", "Bearer " +
this.restTemplate.getAccessToken().getValue());
                HttpEntity<String> entity = new
HttpEntity<String>(headers);

                ResponseEntity<String> response =
restTemplate.exchange(url, HttpMethod.GET, entity,
String.class);
                String str = response.getBody();
                return str;
        }

        @GetMapping("/pkcs12")
        @ResponseBody
        public String PKCS12(String url, String data) throws
KeyStoreException, NoSuchAlgorithmException,
CertificateException, IOException, KeyManagementException,
UnrecoverableKeyException {
                KeyStore keyStore =
KeyStore.getInstance("PKCS12");
                FileInputStream instream = new
FileInputStream(new File("/opt/xxx.p12"));
                keyStore.load(instream,
"netkiller".toCharArray());
                // Trust own CA and all self-signed certs
                SSLContext sslcontext =
SSLContextBuilder.create().loadKeyMaterial(keyStore,
"netkiller".toCharArray()).build();
                // Allow TLSv1 protocol only
                HostnameVerifier hostnameVerifier =
```

```
NoopHostnameVerifier.INSTANCE;
                SSLConnectionSocketFactory sslsf = new
SSLConnectionSocketFactory(sslcontext, new String[] { "TLSv1"
}, null, hostnameVerifier);
                CloseableHttpClient httpclient =
HttpClients.custom().setSSLSocketFactory(sslsf).build();

                HttpComponentsClientHttpRequestFactory
clientHttpRequestFactory = new
HttpComponentsClientHttpRequestFactory(httpclient);

                RestTemplate restTemplate = new
RestTemplate(clientHttpRequestFactory);

                HttpHeaders httpHeaders = new HttpHeaders();
                httpHeaders.add("Connection", "keep-alive");
                httpHeaders.add("Accept", "*/*");
                httpHeaders.add("Content-Type",
"application/x-www-form-urlencoded;charset=UTF-8");
                httpHeaders.add("Host", "api.netkiller.cn");
                httpHeaders.add("X-Requested-With",
"XMLHttpRequest");
                httpHeaders.add("Cache-Control", "max-
age=0");
                httpHeaders.add("User-Agent", "Mozilla/4.0
(compatible; MSIE 8.0; Windows NT 6.0) ");

                HttpEntity<String> httpEntity = new
HttpEntity<String>(httpHeaders);

                ResponseEntity<String> response =
restTemplate.exchange(url, HttpMethod.POST, httpEntity,
String.class);
                return response.getBody();

        }

}
```

35.9. Timeout 超时设置

JRE 启动参数设置超时时间

```
-Dsun.net.client.defaultConnectTimeout=<TimeoutInMiliSec>
-Dsun.net.client.defaultReadTimeout=<TimeoutInMiliSec>
```

RestTemplate timeout with SimpleClientHttpRequestFactory

```
//Create resttemplate
RestTemplate restTemplate = new
RestTemplate(getClientHttpRequestFactory());

//Override timeouts in request factory
private SimpleClientHttpRequestFactory
getClientHttpRequestFactory()
{
    SimpleClientHttpRequestFactory clientHttpRequestFactory =
new SimpleClientHttpRequestFactory();
    // or
    // HttpComponentsClientHttpRequestFactory
clientHttpRequestFactory = new
HttpComponentsClientHttpRequestFactory();

    //Connect timeout
    clientHttpRequestFactory.setConnectTimeout(10_000);

    //Read timeout
    clientHttpRequestFactory.setReadTimeout(10_000);
    return clientHttpRequestFactory;
}
```

@Configuration 方式

注意下面使用了 Java 11 语法 var factory = new SimpleClientHttpRequestFactory();

```
package cn.netkiller.consul.consumer;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.http.client.SimpleClientHttpRequestFactor
y;
import org.springframework.web.client.RestTemplate;

@Configuration
public class RestTemplateConfiguration {

        @Bean
        public RestTemplate restTemplate() {

                var factory = new
SimpleClientHttpRequestFactory();

                factory.setConnectTimeout(3000);
                factory.setReadTimeout(3000);

                return new RestTemplate(factory);
        }
}
```

36. SpringBootTest

36.1. Maven 依赖

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
```

36.2. 测试类

创建测试类，在测试类的类头部添加：@RunWith(SpringRunner.class)和@SpringBootTest注解，在测试方法的前添加@Test，最后选择方法右键run运行。

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class WalletTest {

        @Autowired
        WalletService walletService;

        public WalletTest() {
                // TODO Auto-generated constructor stub
        }

        @Test
        public void test() throws Exception {

                Assert.assertEquals(5,5);
```

```
        }
}
```

Junit基本注解介绍

```
@RunWith
在JUnit中有很多个Runner，他们负责调用你的测试代码，每一个Runner都有各自的特殊功能，你要根据需要选择不同的Runner来运行你的测试代码。
如果我们只是简单的做普通Java测试，不涉及Spring Web项目，你可以省略@RunWith注解，这样系统会自动使用默认Runner来运行你的代码。

//在所有测试方法前执行一次，一般在其中写上整体初始化的代码
@BeforeClass

//在所有测试方法后执行一次，一般在其中写上销毁和释放资源的代码
@AfterClass

//在每个测试方法前执行，一般用来初始化方法（比如我们在测试别的方法时，类中与其他测试方法共享的值已经被改变，为了保证测试结果的有效性，我们会在@Before注解的方法中重置数据）
@Before

//在每个测试方法后执行，在方法执行完成后要做的事情
@After

// 测试方法执行超过1000毫秒后算超时，测试将失败
@Test(timeout = 1000)

// 测试方法期望得到的异常类，如果方法执行没有抛出指定的异常，则测试失败
@Test(expected = Exception.class)

// 执行测试时将忽略掉此方法，如果用于修饰类，则忽略整个类
@Ignore("not ready yet")
@Test
```

36.3.

Assert.assertEquals 判断相等

Assert.assertTrue

36.4. JPA 测试

```
@RunWith(SpringJUnit4ClassRunner.class)
@SpringApplicationConfiguration(Application.class)
public class ApplicationTests {

    @Autowired
    private UserRepository userRepository;
    @Autowired
    private MessageRepository messageRepository;

    @Test
    public void test() throws Exception {

        userRepository.save(new User("Neo", 10));
        userRepository.save(new User("Jam", 20));
        userRepository.save(new User("Tom", 30));
        userRepository.save(new User("Sam", 40));
        userRepository.save(new User("Leo", 50));

        Assert.assertEquals(5,
userRepository.findAll().size());

        messageRepository.save(new Message("Neo", "How are
you?"));
        messageRepository.save(new Message("Jam", "Hi!"));
        messageRepository.save(new Message("Sam", "What's
going on?"));

        Assert.assertEquals(3,
messageRepository.findAll().size());

    }
}
```

36.5. TestRestTemplate

```
package cn.netkiller.rest;

import java.net.URI;
import java.net.URISyntaxException;

import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.boot.test.context.SpringBootTest.WebEnviron
ment;
import
org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.boot.web.server.LocalServerPort;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.junit4.SpringRunner;

import cn.netkiller.rest.model.Employee;

@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=WebEnvironment.RANDOM_PORT)
public class SpringBootDemoApplicationTests
{
    @Autowired
    private TestRestTemplate restTemplate;

    @LocalServerPort
    int randomServerPort;

    @Test
    public void testAddEmployeeSuccess() throws
URISyntaxException
    {
        final String baseUrl =
"http://localhost:"+randomServerPort+"/employees/";
        URI uri = new URI(baseUrl);
```

```
        Employee employee = new Employee(null, "Adam", "Gilly",
"test@email.com");

        HttpHeaders headers = new HttpHeaders();
        headers.set("X-COM-PERSIST", "true");

        HttpEntity<Employee> request = new HttpEntity<>
(employee, headers);

        ResponseEntity<String> result =
this.restTemplate.postForEntity(uri, request, String.class);

        //Verify request succeed
        Assert.assertEquals(201, result.getStatusCodeValue());
    }

    @Test
    public void testAddEmployeeMissingHeader() throws
URISyntaxException
    {
        final String baseUrl =
"http://localhost:"+randomServerPort+"/employees/";
        URI uri = new URI(baseUrl);
        Employee employee = new Employee(null, "Adam", "Gilly",
"test@email.com");

        HttpHeaders headers = new HttpHeaders();

        HttpEntity<Employee> request = new HttpEntity<>
(employee, headers);

        ResponseEntity<String> result =
this.restTemplate.postForEntity(uri, request, String.class);

        //Verify bad request and missing header
        Assert.assertEquals(400, result.getStatusCodeValue());
        Assert.assertEquals(true,
result.getBody().contains("Missing request header"));
    }

}
```

36.6. Controller单元测试

创建测试类，在测试类的类头部添加:@RunWith(SpringRunner.class)、@SpringBootTest、@AutoConfigureMockMvc注解，在测试方法的前添加@Test，最后选择方法右键run运行。

使用@Autowired 注入MockMvc，在方法中使用 mvc测试功能。示例:

```
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
public class StudentControllerTest {
    @Autowired
    private MockMvc mvc;


    @Test
    public void getAll() throws Exception {


mvc.perform(MockMvcRequestBuilders.get("/student/getAll")).an
dExpect(MockMvcResultMatchers.model().attributeExists("studen
ts"));

    }


     @Test
     public void save() throws Exception {

                Student student = new Student();
                student.setAge(12);
                student.setId("1003");
                student.setName("Neo");

mvc.perform(MockMvcRequestBuilders.post("/student/save",
student));
```

```
    }



    @Test
    public void delete() throws Exception {


mvc.perform(MockMvcRequestBuilders.delete("/student/delete?
id=1002"));

    }



    @Test
    public void index() throws Exception {


mvc.perform(MockMvcRequestBuilders.get("/student/index")).and
Return();

    }

}
```

36.7. WebTestClient

```
package cn.netkiller.webflux;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;
import
org.springframework.test.web.reactive.server.WebTestClient;

@RunWith(SpringRunner.class)
```

```
@SpringBootTest
public class WebfluxApplicationTests {

        @Test
        public void contextLoads() {
        }

        private WebTestClient webTestClient;

        @Before
        public void setUp() {
                this.webTestClient =
WebTestClient.bindToServer().baseUrl("http://localhost:8080")
.build();
        }

        @Test
        public void sample() throws Exception {

this.webTestClient.get().uri("/").exchange().expectStatus().i
sOk().expectBody(String.class).isEqualTo("Hello world!");
        }

        @Test
        public void client() {

        }
}
```

37. Spring boot with Aop

37.1. Aspect

Maven

```
                <dependency>
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
aop</artifactId>
                </dependency>
```

Pojo 类

```
package cn.netkiller.aop.pojo;

import lombok.Data;

@Data
public class Employee {
        private String id;
        private String name;

        public Employee() {
                // TODO Auto-generated constructor stub
        }

}
```

Service 类

```
package cn.netkiller.aop.service;

import org.springframework.stereotype.Service;

import cn.netkiller.aop.pojo.Employee;

@Service
public class EmployeeService {

        public EmployeeService() {
                // TODO Auto-generated constructor stub
        }

        public Employee createEmployee(String id, String
name) {

                Employee emp = new Employee();
                emp.setName(name);
                emp.setId(id);
                return emp;
        }

        public void deleteEmployee(String id) {

        }
}
```

Aspect 类

```
package cn.netkiller.aop.aspect;

import org.aspectj.lang.JoinPoint;
```

```
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class EmployeeServiceAspect {
        public EmployeeServiceAspect() {
        }

        @Before(value = "execution(*
cn.netkiller.aop.service.EmployeeService.*(..)) and args(id,
name)")
        public void beforeAdvice(JoinPoint joinPoint, String
id, String name) {
                System.out.println("Before method:" +
joinPoint.getSignature());

                System.out.println("Creating Employee with
id: " + id + ", name: " + name);
        }

        @After(value = "execution(*
cn.netkiller.aop.service.EmployeeService.*(..)) and
args(id,name)")
        public void afterAdvice(JoinPoint joinPoint, String
id, String name) {
                System.out.println("After method:" +
joinPoint.getSignature());

                System.out.println("Successfully created
Employee with id: " + id + ", name: " + name);
        }
}
```

控制器

```
package cn.netkiller.aop.controller;
```

```
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import cn.netkiller.aop.pojo.Employee;
import cn.netkiller.aop.service.EmployeeService;

@RestController
public class EmployeeController {

        public EmployeeController() {
                // TODO Auto-generated constructor stub
        }

        @Autowired
        private EmployeeService employeeService;

        @RequestMapping(value = "/add/employee", method =
RequestMethod.GET)
        public Employee addEmployee(@RequestParam("id")
String id, @RequestParam("name") String name) {

                return employeeService.createEmployee(id,
name);

        }

        @RequestMapping(value = "/remove/employee", method =
RequestMethod.GET)
        public String removeEmployee(@RequestParam("id")
String id) {

                employeeService.deleteEmployee(id);

                return "Employee removed";
        }

}
```

Application

```
package cn.netkiller.aop;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                SpringApplication.run(Application.class,
args);
        }
}
```

测试

触发 Aspect

```
neo@MacBook-Pro ~ % curl http://localhost:8080/add/employee\?
id\=1\&name\=neo
{"id":"1","name":"neo"}
```

控制台输出效果

```
Before method:Employee
cn.netkiller.aop.service.EmployeeService.createEmployee(String,
String)
Creating Employee with id: 1, name: neo
After method:Employee
cn.netkiller.aop.service.EmployeeService.createEmployee(String,
String)
Successfully created Employee with id: 1, name: neo
```

38. Spring boot with starter

spring-boot-starter-xxxxx 是 Spring boot 子模块，开发中我们可以根据自己的需求开引用所需的功能，这样不必引用所有的 Spring boot 依赖包。

我们也可以开发自己的 starter 模块和自定义注解，将我们的项目化整为零，模块化，随时根据项目的需要引用，并且可以使用自定义注解启用它们。

38.1. 实现 starter

Maven pom.xml 依赖包

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>spring-boot-starter-customize</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>Spring Boot Starter Project</name>

        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
```

```
                <start-class>cn.netkiller.starter.App</start-
class>
                <java.version>11</java.version>
                <lombok.version>1.16.18</lombok.version>
        </properties>

        <dependencies>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
starter</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <version>${lombok.version}</version>
                        <scope>provided</scope>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```

配置文件处理

application.properties 加入短信网关的配置项

```
sms.gateway.url=https://sms.netkiller.cn/v1
sms.gateway.username=netkiller
sms.gateway.password=passw0rd
```

SmsProperties 用于读取前缀为 sms.gateway 的配置项。

```
package cn.netkiller.autoconfigure;

import
org.springframework.boot.context.properties.ConfigurationProp
erties;

import lombok.Data;

@ConfigurationProperties(prefix = "sms.gateway")
@Data
public class SmsProperties {

        private String url;

        private String username;

        private String password;

        public String getUrl() {
                return url;
        }

        public void setUrl(String url) {
```

```
                this.url = url;
        }

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        @Override
        public String toString() {
                return "SmsProperties [url=" + url + ",
username=" + username + ", password=" + password + "]";
        }

}
```

自动配置文件

```
package cn.netkiller.autoconfigure;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
import cn.netkiller.sms.SmsSender;

@EnableConfigurationProperties(value = SmsProperties.class)
@Configuration
public class SmsAutoConfiguration {

        @Autowired
        private SmsProperties smsProperties;

        @Bean
        public SmsSender send() {
                return new SmsSender(this.smsProperties);
        }
}
```

启用 starter 的自定义注解

```
package cn.netkiller.autoconfigure;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;

import org.springframework.context.annotation.Import;

@Target({ ElementType.TYPE })
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Import({ SmsAutoConfiguration.class })
public @interface EnableSms {

}
```

38.2. 引用 starter

Maven pom.xml 引入依赖

```
                <dependency>
                        <groupId>cn.netkiller</groupId>
                        <artifactId>spring-boot-starter-
customize</artifactId>
                        <version>0.0.1-SNAPSHOT</version>
                </dependency>
```

完整的 pom.xml 文件

```
<?xml version="1.0"?>
<project
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>spring-boot-starter-customize-
test</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>spring-boot-starter-customize-test</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>cn.netkiller</groupId>
                        <artifactId>spring-boot-starter-
```

```
customize</artifactId>
                        <version>0.0.1-SNAPSHOT</version>
                </dependency>
        </dependencies>
</project>
```

通过注解配置 starter

@EnableSms 启用自动配置短信发送模块

```
package cn.netkiller.starter.customize.test;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.context.ConfigurableApplicationContext;

import cn.netkiller.autoconfigure.EnableSms;
import cn.netkiller.sms.SmsSender;

@SpringBootApplication
@EnableSms
public class Application {
        public static void main(String[] args) {
                System.out.println("Hello World!");

                ConfigurableApplicationContext
applicationContext = SpringApplication.run(Application.class,
args);

                SmsSender smsSender =
applicationContext.getBean(SmsSender.class);
                smsSender.send("验证码发送成功! ");
        }
}
```

测试运行结果

```
Hello World!


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.2.RELEASE)

2020-08-02 20:51:54.564  INFO 43216 --- [           main]
c.n.starter.customize.test.Application   : Starting Application
on MacBook-Pro-Neo.local with PID 43216
(/Users/neo/git/springcloud/spring-boot-starter-customize-
test/target/classes started by neo in
/Users/neo/git/springcloud/spring-boot-starter-customize-test)
2020-08-02 20:51:54.567  INFO 43216 --- [           main]
c.n.starter.customize.test.Application   : No active profile
set, falling back to default profiles: default
2020-08-02 20:51:55.349  INFO 43216 --- [           main]
c.n.starter.customize.test.Application   : Started Application
in 1.539 seconds (JVM running for 1.942)
SmsProperties [url=https://sms.netkiller.cn/v1,
username=netkiller, password=passw0rd]
验证码发送成功!
```

39. SpringBoot Admin

39.1. 依赖

```
        <dependency>
            <groupId>de.codecentric</groupId>
            <artifactId>spring-boot-admin-starter-
server</artifactId>
            <version>2.1.6</version>
        </dependency>
```

39.2. 启用 Springboot Admin

```
@EnableAdminServer
public class SpringBootAdminApplication {
        public static void main(String[] args) {

SpringApplication.run(SpringBootAdminApplication.class,
args);
        }
}
```

39.3. Nginx 跨域

```
server {
    listen       192.168.30.11:80;
    listen       192.168.30.11:443 ssl http2;
    server_name  api.netkiller.cn;
```

```
    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_ciphers PROFILE=SYSTEM;
    ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/api.netkiller.cn.access.log;
    error_log /var/log/nginx/api.netkiller.cn.error.log;

    error_page 497 https://$host$uri?$args;

    if ($scheme = http) {
        return 301 https://$server_name$request_uri;
    }

    location / {
        add_header Content-Security-Policy "upgrade-insecure-
requests;connect-src *";
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header REMOTE-HOST $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_pass http://192.168.30.10:8088;
    }

    error_page 404 /404.html;
        location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
        location = /50x.html {
    }
}
```

40. Spring boot with Grafana

40.1. Springboot 集成 InfluxDB

Springboot 集成 InfluxDB 非常简单，先引入依赖即可，记得需要同时引入 actuator

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-influx</artifactId>
</dependency>
```

配置文件 application.yaml 如下

```
spring:
  application:
    name: springboot-with-influxdb
server:
  port: 8080
management:
  metrics:
    export:
      influx:
        enabled: true
        db: springboot
        uri: http://localhost:8086
        user-name:
        password:
        connect-timeout: 1s
        read-timeout: 10s
```

```
        auto-create-db: true
        step: 1m
        num-threads: 2
        consistency: one
        compressed: true
        batch-size: 1000
```

40.2. InfluxDB

配置好 Springboot 后，启动应用，稍后Springboot 就会将数据源源不断地写入到 InfluxDB 中。

```
> show measurements
name: measurements
name
----
jvm_buffer_count
jvm_buffer_memory_used
jvm_buffer_total_capacity
jvm_classes_loaded
jvm_classes_unloaded
jvm_gc_live_data_size
jvm_gc_max_data_size
jvm_gc_memory_allocated
jvm_gc_memory_promoted
jvm_gc_pause
jvm_memory_committed
jvm_memory_max
jvm_memory_used
jvm_threads_daemon
jvm_threads_live
jvm_threads_peak
jvm_threads_states
logback_events
process_cpu_usage
process_files_max
process_files_open
process_start_time
process_uptime
```

```
system_cpu_count
system_cpu_usage
system_load_average_1m
tomcat_sessions_active_current
tomcat_sessions_active_max
tomcat_sessions_alive_max
tomcat_sessions_created
tomcat_sessions_expired
tomcat_sessions_rejected
visits
```

查看数据

```
select * from process_cpu_usage
```

41. Spring Boot with Prometheus

41.1. Maven 依赖

```
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
webflux</artifactId>
                </dependency>
                <dependency>
                        <groupId>io.micrometer</groupId>
                        <artifactId>micrometer-registry-
prometheus</artifactId>
                </dependency>
        </dependencies>
```

41.2. application.properties 配置文件

开启 metrics

```
spring.application.name=springboot-with-prometheus
#management.endpoints.web.exposure.include=*
management.endpoints.web.exposure.include=prometheus
management.metrics.tags.application=${spring.application.name}
```

41.3. 启动类

```
```

```
package cn.netkiller.welcome;

import java.net.InetAddress;
import java.net.UnknownHostException;

import org.reactivestreams.Publisher;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.actuate.autoconfigure.metrics.MeterR
egistryCustomizer;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import
org.springframework.web.bind.annotation.RestController;

import io.micrometer.core.instrument.MeterRegistry;
import reactor.core.publisher.Mono;

@SpringBootApplication
@RestController
public class Application {

        @GetMapping("/")
        @ResponseBody
        public Publisher<String> index() {
                return Mono.just("Hello world! \r\n");
        }

        @GetMapping("/address")
        @ResponseBody
        public Publisher<String> address() throws
UnknownHostException {
                InetAddress addr =
InetAddress.getLocalHost();
                return Mono.just(String.format("Address %s,
Hostname %s \r\n", addr.getHostAddress(),
addr.getHostName()));
        }

        @Bean
        MeterRegistryCustomizer<MeterRegistry>
```

```
configurer(@Value("${spring.application.name}") String
applicationName) {
                return (registry) ->
registry.config().commonTags("application", applicationName);
        }

        public static void main(String[] args) {
                System.out.println("Welcome!");
                SpringApplication.run(Application.class,
args);
        }
}
```

41.4. 测试

启动 Springboot

```
Welcome!

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.3.4.RELEASE)

2020-10-28 21:57:54.110  INFO 64079 --- [           main]
cn.netkiller.welcome.Application         : Starting Application
on MacBook-Pro-Neo.local with PID 64079
(/Users/neo/workspace/microservice/welcome/target/classes
started by neo in /Users/neo/workspace/microservice/welcome)
2020-10-28 21:57:54.114  INFO 64079 --- [           main]
cn.netkiller.welcome.Application         : No active profile
set, falling back to default profiles: default
2020-10-28 21:57:55.877  INFO 64079 --- [           main]
o.s.b.a.e.web.EndpointLinksResolver      : Exposing 14
endpoint(s) beneath base path '/actuator'
2020-10-28 21:57:56.364  INFO 64079 --- [           main]
```

```
o.s.b.web.embedded.netty.NettyWebServer  : Netty started on
port(s): 8080
2020-10-28 21:57:56.380  INFO 64079 --- [           main]
cn.netkiller.welcome.Application         : Started Application
in 2.773 seconds (JVM running for 3.439)
```

获取监控数据

```
neo@MacBook-Pro-Neo ~ % curl
http://localhost:8080/actuator/prometheus
# HELP jvm_threads_states_threads The current number of threads
having NEW state
# TYPE jvm_threads_states_threads gauge
jvm_threads_states_threads{application="springboot-with-
prometheus",state="terminated",} 0.0
jvm_threads_states_threads{application="springboot-with-
prometheus",state="blocked",} 0.0
jvm_threads_states_threads{application="springboot-with-
prometheus",state="waiting",} 2.0
jvm_threads_states_threads{application="springboot-with-
prometheus",state="timed-waiting",} 2.0
jvm_threads_states_threads{application="springboot-with-
prometheus",state="runnable",} 7.0
jvm_threads_states_threads{application="springboot-with-
prometheus",state="new",} 0.0
# HELP jvm_gc_memory_allocated_bytes_total Incremented for an
increase in the size of the young generation memory pool after
one GC to before the next
# TYPE jvm_gc_memory_allocated_bytes_total counter
jvm_gc_memory_allocated_bytes_total{application="springboot-
with-prometheus",} 1.9922944E7
# HELP system_cpu_usage The "recent cpu usage" for the whole
system
# TYPE system_cpu_usage gauge
system_cpu_usage{application="springboot-with-prometheus",} 0.0
# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Old Gen",} 1.1322368E7
jvm_memory_used_bytes{application="springboot-with-
```

```
prometheus",area="heap",id="G1 Eden Space",} 1.6777216E7
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="nonheap",id="Metaspace",} 3.1712968E7
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Survivor Space",} 1487328.0
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'non-nmethods'",}
1277184.0
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'non-profiled
nmethods'",} 1413760.0
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="nonheap",id="Compressed Class Space",}
4253200.0
jvm_memory_used_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'profiled nmethods'",}
7536256.0
# HELP jvm_memory_committed_bytes The amount of memory in bytes
that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Old Gen",} 2.5165824E7
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Eden Space",} 2.8311552E7
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="nonheap",id="Metaspace",} 3.3161216E7
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Survivor Space",} 2097152.0
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'non-nmethods'",}
2555904.0
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'non-profiled
nmethods'",} 2555904.0
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="nonheap",id="Compressed Class Space",}
4849664.0
jvm_memory_committed_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'profiled nmethods'",}
7602176.0
# HELP system_load_average_1m The sum of the number of runnable
entities queued to available processors and the number of
runnable entities running on the available processors averaged
over a period of time
# TYPE system_load_average_1m gauge
```

```
system_load_average_1m{application="springboot-with-
prometheus",} 2.263671875
# HELP process_files_open_files The open file descriptor count
# TYPE process_files_open_files gauge
process_files_open_files{application="springboot-with-
prometheus",} 89.0
# HELP jvm_classes_unloaded_classes_total The total number of
classes unloaded since the Java virtual machine has started
execution
# TYPE jvm_classes_unloaded_classes_total counter
jvm_classes_unloaded_classes_total{application="springboot-
with-prometheus",} 0.0
# HELP jvm_buffer_total_capacity_bytes An estimate of the total
capacity of the buffers in this pool
# TYPE jvm_buffer_total_capacity_bytes gauge
jvm_buffer_total_capacity_bytes{application="springboot-with-
prometheus",id="direct",} 1.6777223E7
jvm_buffer_total_capacity_bytes{application="springboot-with-
prometheus",id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_total_capacity_bytes{application="springboot-with-
prometheus",id="mapped",} 0.0
# HELP jvm_gc_live_data_size_bytes Size of old generation
memory pool after a full GC
# TYPE jvm_gc_live_data_size_bytes gauge
jvm_gc_live_data_size_bytes{application="springboot-with-
prometheus",} 0.0
# HELP jvm_gc_pause_seconds Time spent in GC pause
# TYPE jvm_gc_pause_seconds summary
jvm_gc_pause_seconds_count{action="end of minor
GC",application="springboot-with-prometheus",cause="G1
Evacuation Pause",} 1.0
jvm_gc_pause_seconds_sum{action="end of minor
GC",application="springboot-with-prometheus",cause="G1
Evacuation Pause",} 0.008
# HELP jvm_gc_pause_seconds_max Time spent in GC pause
# TYPE jvm_gc_pause_seconds_max gauge
jvm_gc_pause_seconds_max{action="end of minor
GC",application="springboot-with-prometheus",cause="G1
Evacuation Pause",} 0.008
# HELP process_files_max_files The maximum file descriptor
count
# TYPE process_files_max_files gauge
process_files_max_files{application="springboot-with-
prometheus",} 10240.0
# HELP jvm_threads_live_threads The current number of live
```

```
threads including both daemon and non-daemon threads
# TYPE jvm_threads_live_threads gauge
jvm_threads_live_threads{application="springboot-with-
prometheus",} 11.0
# HELP process_start_time_seconds Start time of the process
since unix epoch.
# TYPE process_start_time_seconds gauge
process_start_time_seconds{application="springboot-with-
prometheus",} 1.603893473057E9
# HELP jvm_classes_loaded_classes The number of classes that
are currently loaded in the Java virtual machine
# TYPE jvm_classes_loaded_classes gauge
jvm_classes_loaded_classes{application="springboot-with-
prometheus",} 6965.0
# HELP jvm_buffer_memory_used_bytes An estimate of the memory
that the Java virtual machine is using for this buffer pool
# TYPE jvm_buffer_memory_used_bytes gauge
jvm_buffer_memory_used_bytes{application="springboot-with-
prometheus",id="direct",} 1.6777224E7
jvm_buffer_memory_used_bytes{application="springboot-with-
prometheus",id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_memory_used_bytes{application="springboot-with-
prometheus",id="mapped",} 0.0
# HELP process_cpu_usage The "recent cpu usage" for the Java
Virtual Machine process
# TYPE process_cpu_usage gauge
process_cpu_usage{application="springboot-with-prometheus",}
0.0
# HELP jvm_buffer_count_buffers An estimate of the number of
buffers in the pool
# TYPE jvm_buffer_count_buffers gauge
jvm_buffer_count_buffers{application="springboot-with-
prometheus",id="direct",} 4.0
jvm_buffer_count_buffers{application="springboot-with-
prometheus",id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_count_buffers{application="springboot-with-
prometheus",id="mapped",} 0.0
# HELP jvm_gc_max_data_size_bytes Max size of old generation
memory pool
# TYPE jvm_gc_max_data_size_bytes gauge
jvm_gc_max_data_size_bytes{application="springboot-with-
prometheus",} 2.147483648E9
# HELP jvm_threads_peak_threads The peak live thread count
since the Java virtual machine started or peak was reset
# TYPE jvm_threads_peak_threads gauge
```

```
jvm_threads_peak_threads{application="springboot-with-
prometheus",} 11.0
# HELP logback_events_total Number of error level events that
made it to the logs
# TYPE logback_events_total counter
logback_events_total{application="springboot-with-
prometheus",level="debug",} 0.0
logback_events_total{application="springboot-with-
prometheus",level="trace",} 0.0
logback_events_total{application="springboot-with-
prometheus",level="info",} 3.0
logback_events_total{application="springboot-with-
prometheus",level="error",} 0.0
logback_events_total{application="springboot-with-
prometheus",level="warn",} 0.0
# HELP jvm_gc_memory_promoted_bytes_total Count of positive
increases in the size of the old generation memory pool before
GC to after GC
# TYPE jvm_gc_memory_promoted_bytes_total counter
jvm_gc_memory_promoted_bytes_total{application="springboot-
with-prometheus",} 1924096.0
# HELP jvm_threads_daemon_threads The current number of live
daemon threads
# TYPE jvm_threads_daemon_threads gauge
jvm_threads_daemon_threads{application="springboot-with-
prometheus",} 9.0
# HELP process_uptime_seconds The uptime of the Java virtual
machine
# TYPE process_uptime_seconds gauge
process_uptime_seconds{application="springboot-with-
prometheus",} 35.375
# HELP jvm_memory_max_bytes The maximum amount of memory in
bytes that can be used for memory management
# TYPE jvm_memory_max_bytes gauge
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Old Gen",} 2.147483648E9
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Eden Space",} -1.0
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="nonheap",id="Metaspace",} -1.0
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="heap",id="G1 Survivor Space",} -1.0
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'non-nmethods'",}
5840896.0
```

```
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'non-profiled
nmethods'",} 1.22908672E8
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="nonheap",id="Compressed Class Space",}
1.073741824E9
jvm_memory_max_bytes{application="springboot-with-
prometheus",area="nonheap",id="CodeHeap 'profiled nmethods'",}
1.22908672E8
# HELP system_cpu_count The number of processors available to
the Java virtual machine
# TYPE system_cpu_count gauge
system_cpu_count{application="springboot-with-prometheus",} 8.0
```

41.5. 控制器监控

```
@RestController
@RequestMapping("/app")
public class AppController {
  private final Counter counter;

  public AppController(final MeterRegistry registry) {
    this.counter = registry.counter("greeting");
  }

  @RequestMapping("/greeting")
  public String greeting() {
    this.counter.increment();
    return "hello world #" + this.counter.count();
  }
}
```

41.6. 自定义埋点监控

prometheus 监控指标有如下几种类型

- Counter 类型代表数据递增的指标，即只增不减，除非监控系统重置
- Guage 类型代表数据可以任意变化的指标，即可增可减
- Histogram 由bucket{le=""}，bucket{le="+Inf"},sum，count 组成，用于一段时间范围内对数据进行采样，并能够对其指定区间以及总数进行统计，通常它采集的数据展示为直方图。
- Summary 由{quantile="<φ>"}，sum，count 组成，用于一段时间内数据采样结果（通常是请求持续时间或响应大小），它直接存储了 quantile 数据，而不是根据统计区间计算出来的。

拦截器

```
package cn.netkiller.welcome.config;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.servlet.HandlerInterceptor;

import cn.netkiller.welcome.component.RestfulApiCounter;

public class PrometheusInterceptor implements
HandlerInterceptor {

        @Autowired
        private RestfulApiCounter restfulApiCounter;

        public PrometheusInterceptor() {

        }

        @Override
        public void afterCompletion(HttpServletRequest
request, HttpServletResponse response, Object handler,
Exception ex) throws Exception {

                restfulApiCounter.increment();
```

```
        }

}
```

计数器元件

```
package cn.netkiller.welcome.component;

import org.springframework.stereotype.Component;

import io.micrometer.core.instrument.Counter;
import io.micrometer.core.instrument.MeterRegistry;

@Component
public class RestfulApiCounter {
        private final Counter counter;

        public RestfulApiCounter(MeterRegistry registry) {
                this.counter =
registry.counter("restful_api_requests_total");
        }

        public void increment() {
                this.counter.increment();
        }
}
```

配置类

```
package cn.netkiller.welcome.config;

import org.springframework.context.annotation.Bean;
```

```
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.Interceptor
Registry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;

@Configuration
public class IntercepterConfiguration implements
WebMvcConfigurer {

        @Bean
        public PrometheusInterceptor prometheusInterceptor()
{

                return new PrometheusInterceptor();
        }

        @Override
        public void addInterceptors(InterceptorRegistry
registry) {

registry.addInterceptor(prometheusInterceptor()).addPathPatte
rns("/**");

        }
}
```

测试埋点效果

```
neo@MacBook-Pro-Neo ~ % curl -s
http://localhost:8080/actuator/prometheus | grep restful
# HELP restful_api_requests_total
# TYPE restful_api_requests_total counter
restful_api_requests_total{application="springboot-with-
prometheus",} 0.0
```

```
neo@MacBook-Pro-Neo ~ % curl http://localhost:8080/
Hello world!

neo@MacBook-Pro-Neo ~ % curl http://localhost:8080/address
Address 127.0.0.1, Hostname MacBook-Pro-Neo.local

neo@MacBook-Pro-Neo ~ % curl -s
http://localhost:8080/actuator/prometheus | grep restful
# HELP restful_api_requests_total
# TYPE restful_api_requests_total counter
restful_api_requests_total{application="springboot-with-
prometheus",} 2.0
```

42. Spring boot with Git version

Spring boot 每次升级打包发给运维操作，常常运维操作不当致使升级失败，开发怎样确认线上的jar/war包与升级包一致呢?

请看下面的解决方案

42.1. CommonRestController 公共控制器

所有 RestController将会集成 CommonRestController

```
package cn.netkiller.api.rest;

import org.springframework.http.HttpStatus;
import
org.springframework.security.core.annotation.AuthenticationPr
incipal;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.ResponseStatus;

public class CommonRestController {

@RequestMapping("ping")
@ResponseStatus(HttpStatus.OK)
public String welcome() {
    return "PONG";
}

@RequestMapping("commit")
public String commit() {
    return "$Id$";
}

@RequestMapping("auth")
```

```
@ResponseStatus(HttpStatus.OK)
public String auth(@AuthenticationPrincipal final UserDetails
user) {
    return String.format("%s: %s %s", user.getUsername(),
user.getPassword(), user.getAuthorities());
}
}
```

42.2. VersionRestController 测试控制器

我们创建一个RestController并继承CommonRestController用来测试

```
package cn.netkiller.api.rest;

@RestController
@RequestMapping("/public/version")
public class VersionRestController extends
CommonRestController {
private static final Logger logger =
LoggerFactory.getLogger(VersionRestController.class);

public VersionRestController() {
    // TODO Auto-generated constructor stub
}

@RequestMapping("welcome")
@ResponseStatus(HttpStatus.OK)
public String welcome() {
    return "Welcome to RestTemplate version 1.0.";
}

}
```

42.3. 创建 .gitattributes 文件

```
# vim .gitattributes
src/main/java/cn/netkiller/api/rest/CommonRestController.java
ident
```

使用curl命令调用commit接口可以显示当前war/jar最后一次提交的版本号码（你同样可以使用IE浏览器）

```
curl https://api.netkiller.cn/public/version/commit.json
$Id: 929bc9e4c90b4d68c25dc693618f23b33fd6ba0f $
```

43. Spring boot with Session share

43.1. Redis

Maven

增加下面代码到pom.xml

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-
redis</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.session</groupId>
        <artifactId>spring-session-data-redis</artifactId>
    </dependency>
```

pom.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>cn.netkiller</groupId>
<artifactId>deploy</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>deploy.netkiller.cn</name>
```

```
<description>Deploy project for Spring Boot</description>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.1.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
</parent>

<properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
</properties>

<dependencies>
    <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency> -->
    <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency> -->
    <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency> -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-
redis</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-
redis</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.session</groupId>
        <artifactId>spring-session-data-redis</artifactId>
    </dependency>
    <!-- <dependency>
```

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jdbc</artifactId>
</dependency> -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-
websocket</artifactId>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>webjars-locator</artifactId>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>sockjs-client</artifactId>
        <version>1.0.2</version>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>stomp-websocket</artifactId>
        <version>2.3.3</version>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>bootstrap</artifactId>
        <version>3.3.7</version>
    </dependency>
    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>jquery</artifactId>
        <version>3.1.0</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
```

```
    </dependency>
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>

</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

<repositories>
    <repository>
        <id>spring-snapshots</id>
        <name>Spring Snapshots</name>
        <url>https://repo.spring.io/snapshot</url>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
    <repository>
        <id>spring-milestones</id>
        <name>Spring Milestones</name>
        <url>https://repo.spring.io/milestone</url>
        <snapshots>
```

```
            <enabled>false</enabled>
        </snapshots>
    </repository>
</repositories>
<pluginRepositories>
    <pluginRepository>
        <id>spring-snapshots</id>
        <name>Spring Snapshots</name>
        <url>https://repo.spring.io/snapshot</url>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </pluginRepository>
    <pluginRepository>
        <id>spring-milestones</id>
        <name>Spring Milestones</name>
        <url>https://repo.spring.io/milestone</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </pluginRepository>
</pluginRepositories>

</project>
```

application.properties

spring.session.store-type=redis 将Session 存储在Redis中

```
spring.redis.database=0
spring.redis.host=192.168.4.1
spring.redis.port=6379
#spring.redis.password=
spring.redis.pool.max-active=8
spring.redis.pool.max-wait=30
spring.redis.pool.max-idle=8
spring.redis.pool.min-idle=0
```

```
spring.redis.timeout=10

spring.session.store-type=redis
```

Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;
import
org.springframework.session.data.redis.config.annotation.web.
http.EnableRedisHttpSession;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
@EnableMongoRepositories
@EnableJpaRepositories
@EnableScheduling
public class Application {

public static void main(String[] args) {
    SpringApplication.run(Application.class, args);

}
```

```
}
```

RedisHttpSessionConfig.java

```
package cn.netkiller.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.session.data.redis.config.annotation.web.
http.EnableRedisHttpSession;

@Configuration
@EnableRedisHttpSession
public class RedisHttpSessionConfig {

public RedisHttpSessionConfig() {
    // TODO Auto-generated constructor stub
}

}
```

43.2. 测试 Session

```
package cn.netkiller.web;

import java.util.Date;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
@Controller
public class TestController {

public TestController() {
    // TODO Auto-generated constructor stub
}

@RequestMapping("/session/set")
@ResponseBody
public String set(HttpSession session) {
    String key = "test";
    session.setAttribute(key, new Date());
    return key;
}

@RequestMapping("/session/get")
@ResponseBody
public String get(HttpSession session) {
    String value = (String)
session.getAttribute("test").toString();
    return value;
}

}
```

keys spring:session:* 查看 Session Key

```
$ telnet 192.168.4.1 6379
Connecting to 192.168.4.1:6379...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
keys spring:session:*
*7
$68
spring:session:sessions:expires:a510f46f-0a2f-4649-af05-
34bd750562c1
$40
spring:session:expirations:1476100200000
$40
spring:session:expirations:1476098400000
```

```
$60
spring:session:sessions:f6494a2f-591e-42ba-b381-ce2596f4046d
$60
spring:session:sessions:a510f46f-0a2f-4649-af05-34bd750562c1
$112
spring:session:index:org.springframework.session.FindByIndexNam
eSessionRepository.PRINCIPAL_NAME_INDEX_NAME:user
$60
spring:session:sessions:627018c8-243e-43ac-87b9-fc07f130c899
```

43.3. JDBC

```
spring.session.store-type=jdbc
spring.session.jdbc.table-name=SESSIONS
```

43.4. Springboot 2.1

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-redis</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-data-redis</artifactId>
</dependency>
```

开启Redis共享SESSION @EnableRedisHttpSession

```
package cn.netkiller.oauth2;
```

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.session.data.redis.config.annotation.web.ht
tp.EnableRedisHttpSession;

@SpringBootApplication
@EnableAutoConfiguration
@EnableRedisHttpSession
public class Application {
public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
}
}
```

application.properties中配置redis服务器

```
spring.redis.host=localhost
spring.redis.port=6379
```

44. Spring boot with Caching

https://docs.spring.io/spring-boot/docs/current/reference/html/io.html#io.caching.provider.redis

44.1. maven

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-cache</artifactId>
    </dependency>
```

Redis

使用 Redis

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
cache</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-
redis</artifactId>
        </dependency>
```

```
spring.data.redis.host=cch.netkiller.cn
```

```
spring.data.redis.port=6379
spring.data.redis.password=passw0rd
spring.data.redis.database=10
spring.data.redis.timeout=30000
spring.data.redis.lettuce.pool.max-active=8
spring.data.redis.lettuce.pool.max-wait=-1
spring.data.redis.lettuce.pool.max-idle=8
spring.data.redis.lettuce.pool.min-idle=0

spring.cache.type=redis
spring.cache.redis.time-to-live=3600000
spring.cache.redis.cache-null-values=true
```

44.2. 启用 Cache

添加 @EnableCaching

```
package hello;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cache.annotation.EnableCaching;

@SpringBootApplication
@EnableCaching
public class Application {

        public static void main(String[] args) {
            SpringApplication.run(Application.class, args);
        }

}
```

44.3. @Cacheable 的用法

```
@Cacheable(value="users", key="#id")
public User find(Integer id) {

        return null;

}
```

引用对象

```
@Cacheable(value="users", key="#user.id")
public User find(User user) {

return null;

}
```

条件判断

```
@Cacheable(value="messagecache", key="#id", condition="id <
10")
public String getMessage(int id){

return "hello"+id;

}

@Cacheable(value="test",condition="#userName.length()>2")
@Cacheable(value={"users"}, key="#user.id",
condition="#user.id%2==0")
```

#p0 参数索引，p0表示第一个参数

```
@Cacheable(value="users", key="#p0")
public User find(Integer id) {

return null;

}

@Cacheable(value="users", key="#p0.id")
public User find(User user) {

return null;

}
```

@Cacheable 如果没有任何参数将会自动生成 key ， 前提是必须设置 @CacheConfig(cacheNames = "test")

```
@GetMapping("/cache/auto")
@Cacheable()
public Attribute auto() {
    Attribute attribute = new Attribute();
    attribute.setName("sdfsdf");
    return attribute;
}
```

```
127.0.0.1:6379> keys *
1) "test::SimpleKey []"
```

SpEL表达式

```
@GetMapping("/cache/expire")
@Cacheable("test1#${select.cache.timeout:1000}")
public String expire() {
    return "Test";
}

@GetMapping("/cache/expire")
@Cacheable("test1#${select.cache.timeout:1000}#${select.cache
.refresh:600}")
public String expire() {
    return "Test";
}
```

排除 null 结果

使用 unless 排除 null 结果

```
    @Cacheable(value = "translate", key = "#chinese",
unless="#result == null")
    public String translate(String chinese) {
    }
```

通过配置文件设置spring.cache.redis.cache-null-values

```
spring.cache.redis.cache-null-values=false
```

44.4. @CachePut 用法

@CachePut 每次都会执行方法，都会将结果存入指定key的缓存中，@CachePut 不会判断是否 key 已经存在，二是始终覆盖。

```
@CachePut("users")
public User find(Integer id) {

return null;

}
```

44.5. 清空缓存

缓存返回结果

```
@Cacheable("cacheable")
@RequestMapping("/test/cacheable")
@ResponseBody
public String cacheable() {
    Date date = new Date();
    String message = date.toString();
    return message;
}
```

5秒钟清楚一次缓存

```
@Scheduled(fixedDelay = 5000)
@CacheEvict(allEntries = true, value = "cacheable")
public void expire() {
```

```
    Date date = new Date();
    String message = date.toString();
    System.out.println(message);
}
```

44.6. @Caching

```
    @Caching(
            cacheable = {
                    @Cacheable(value = "emp",key = "#p0"),
                    ...
            },
            put = {
                    @CachePut(value = "emp",key = "#p0"),
                    ...
            },
            evict = {
                    @CacheEvict(value = "emp",key = "#p0"),
                    ....
            }
    )
    public User save(User user) {
        ....
    }
```

44.7. 解决Expire 和 TTL 过期时间

Springboot 1.x

```
@Bean
public CacheManager cacheManager(RedisTemplate redisTemplate)
{
```

```
    RedisCacheManager cacheManager = new
RedisCacheManager(redisTemplate);
    cacheManager.setDefaultExpiration(60);      //缓存默认 60
秒
    Map<String, Long> expiresMap = new HashMap<>();
    expiresMap.put("Product", 5L);  //设置 key = Product 时 5秒
缓存。你可以添加很多规则。
    cacheManager.setExpires(expiresMap);
    return cacheManager;
}
```

Springboot 2.x

```
package api.config;

import java.time.Duration;
import java.util.HashMap;
import java.util.Map;

import org.springframework.cache.CacheManager;
import org.springframework.cache.interceptor.KeyGenerator;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.redis.cache.RedisCacheConfiguration;
import
org.springframework.data.redis.cache.RedisCacheManager;
import org.springframework.data.redis.cache.RedisCacheWriter;
import
org.springframework.data.redis.connection.RedisConnectionFact
ory;
import
org.springframework.data.redis.serializer.Jackson2JsonRedisSe
rializer;
import
org.springframework.data.redis.serializer.RedisSerializationC
ontext;

import com.fasterxml.jackson.annotation.JsonAutoDetect;
```

```
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.databind.ObjectMapper;

@Configuration
public class CachingConfigurer {

public CachingConfigurer() {
    // TODO Auto-generated constructor stub
}

@Bean
public KeyGenerator simpleKeyGenerator() {
    return (o, method, objects) -> {
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append(o.getClass().getSimpleName());
        stringBuilder.append(".");
        stringBuilder.append(method.getName());
        stringBuilder.append("[");
        for (Object obj : objects) {
            stringBuilder.append(obj.toString());
        }
        stringBuilder.append("]");

        return stringBuilder.toString();
    };
}

@Bean
public CacheManager cacheManager(RedisConnectionFactory
redisConnectionFactory) {
    return new
RedisCacheManager(RedisCacheWriter.nonLockingRedisCacheWriter
(redisConnectionFactory),
    this.redisCacheConfiguration(600),  // 默认配置
    this.initialCacheConfigurations()); // 指定key过期时间配置
}

private Map<String, RedisCacheConfiguration>
initialCacheConfigurations() {
    Map<String, RedisCacheConfiguration>
redisCacheConfigurationMap = new HashMap<>();
    redisCacheConfigurationMap.put("UserInfoList",
this.redisCacheConfiguration(3000));
    redisCacheConfigurationMap.put("UserInfoListAnother",
this.redisCacheConfiguration(18000));
```

```
    return redisCacheConfigurationMap;
}

private RedisCacheConfiguration
redisCacheConfiguration(Integer seconds) {
    Jackson2JsonRedisSerializer<Object>
jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer<>(Object.class);
    ObjectMapper om = new ObjectMapper();
    om.setVisibility(PropertyAccessor.ALL,
JsonAutoDetect.Visibility.ANY);

om.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
    jackson2JsonRedisSerializer.setObjectMapper(om);

    RedisCacheConfiguration redisCacheConfiguration =
RedisCacheConfiguration.defaultCacheConfig();
    redisCacheConfiguration =
redisCacheConfiguration.serializeValuesWith(RedisSerializatio
nContext.SerializationPair.fromSerializer(jackson2JsonRedisSe
rializer)).entryTtl(Duration.ofSeconds(seconds));

    return redisCacheConfiguration;
}

}
```

```
@Cacheable(value = "DefaultKey", keyGenerator =
"simpleKeyGenerator") // 600秒，使用默认策略
@Cacheable(value = "UserInfoList", keyGenerator =
"simpleKeyGenerator") // 3000秒
@Cacheable(value = "UserInfoListAnother", keyGenerator =
"simpleKeyGenerator") // 18000秒
```

```
127.0.0.1:6379> keys *
1) "test2::SimpleKey []"

127.0.0.1:6379> ttl "test2::SimpleKey []"
(integer) 584
```

45. Spring boot with Email

45.1. Maven

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

例 5.7. Spring boot with Email (pom.xml)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>netkiller.cn</groupId>
<artifactId>api.netkiller.cn</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>api.netkiller.cn</name>
<url>http://maven.apache.org</url>

<properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
</properties>

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
```

```
    <version>2.3.1.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <!-- <dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency> -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-
redis</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-amqp</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>

    <dependency>
```

```
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-mongodb</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-oracle</artifactId>
        <version>1.0.0.RELEASE</version>
    </dependency>

    <dependency>
        <groupId>com.oracle</groupId>
        <artifactId>ojdbc6</artifactId>
        <!-- <version>12.1.0.1</version> -->
        <version>11.2.0.3</version>
        <scope>system</scope>
        <systemPath>${basedir}/lib/ojdbc6.jar</systemPath>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-mail</artifactId>
    </dependency>

    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
        <plugin>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.3</version>
            <configuration>
                <source />
                <target />
            </configuration>
        </plugin>
        <plugin>
            <artifactId>maven-war-plugin</artifactId>
            <version>2.6</version>
            <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>
```

45.2. Resource

application.properties

Postfix / Exam4 / Sendmail 邮件服务器配置

```
spring.mail.host=smtp.163.com
```

SMTP 配置

```
spring.mail.host=smtp.163.com
spring.mail.username=openunix@163.com
spring.mail.password=your_password
spring.mail.properties.mail.smtp.auth=true
#spring.mail.properties.mail.smtp.starttls.enable=true
#spring.mail.properties.mail.smtp.starttls.required=true
```

45.3. POJO

```
package api.pojo;

public class Email {
public String from;
public String to;
public String subject;
public String text;
public boolean status;

public String getFrom() {
    return from;
}
public void setFrom(String from) {
    this.from = from;
}
public String getTo() {
    return to;
}
public void setTo(String to) {
    this.to = to;
}
public String getSubject() {
    return subject;
}
public void setSubject(String subject) {
```

```
    this.subject = subject;
}
public String getText() {
    return text;
}
public void setText(String text) {
    this.text = text;
}

public boolean isStatus() {
    return status;
}
public void setStatus(boolean status) {
    this.status = status;
}

@Override
public String toString() {
    return "Email [from=" + from + ", to=" + to + ",
subject=" + subject + ", text=" + text + "]";
}
public Email() {

}
public Email(String from, String to, String subject, String
text) {
    super();
    this.from = from;
    this.to = to;
    this.subject = subject;
    this.text = text;
}

}
```

45.4. RestController

```
package api.rest;
```

```
import java.io.File;

import javax.mail.internet.MimeMessage;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.FileSystemResource;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.ResponseStatus;
import
org.springframework.web.bind.annotation.RestController;

import api.pojo.Email;

@RestController
@RequestMapping("/v1/email")
public class EmailRestController extends CommonRestController
{

@Autowired
private JavaMailSender javaMailSender;

@RequestMapping("version")
@ResponseStatus(HttpStatus.OK)
public String version() {
    return "[OK] Welcome to withdraw Restful version 1.0";
}

@RequestMapping(value = "send", method = RequestMethod.POST,
produces = { "application/xml", "application/json" })
public ResponseEntity<Email> sendSimpleMail(@RequestBody
Email email) {
    SimpleMailMessage message = new SimpleMailMessage();
    message.setFrom(email.getFrom());
    message.setTo(email.getTo());
```

```
    message.setSubject(email.getSubject());
    message.setText(email.getText());
    javaMailSender.send(message);
    email.setStatus(true);

    return new ResponseEntity<Email>(email, HttpStatus.OK);
}

@RequestMapping(value = "attachments", method =
RequestMethod.POST, produces = { "application/xml",
"application/json" })
public ResponseEntity<Email> attachments(@RequestBody Email
email) throws Exception {

    MimeMessage mimeMessage =
javaMailSender.createMimeMessage();

    MimeMessageHelper mimeMessageHelper = new
MimeMessageHelper(mimeMessage, true);
    mimeMessageHelper.setFrom(email.getFrom());
    mimeMessageHelper.setTo(email.getTo());
    mimeMessageHelper.setSubject(email.getSubject());
    mimeMessageHelper.setText("<html><body><img
src=\"cid:banner\" >" + email.getText() + "</body></html>",
true);

    FileSystemResource file = new FileSystemResource(new
File("banner.jpg"));
    mimeMessageHelper.addInline("banner", file);

    FileSystemResource fileSystemResource = new
FileSystemResource(new File("Attachment.jpg"));
    mimeMessageHelper.addAttachment("Attachment.jpg",
fileSystemResource);

    javaMailSender.send(mimeMessage);
    email.setStatus(true);

    return new ResponseEntity<Email>(email, HttpStatus.OK);
}

// 如果你不想使用 application.properties 中的 spring.mail.host 配
置，想自行配置SMTP主机可以参考下面例子
@RequestMapping(value = "sendmail", method =
RequestMethod.POST, produces = { "application/xml",
```

```
"application/json" })
public ResponseEntity<Email> sendmail(@RequestBody Email
email) {
    JavaMailSenderImpl javaMailSender = new
JavaMailSenderImpl();
    javaMailSender.setHost(email.getHost());
    SimpleMailMessage message = new SimpleMailMessage();
    message.setFrom(email.getFrom());
    message.setTo(email.getTo());
    message.setSubject(email.getSubject());
    message.setText(email.getText());
    try{
        javaMailSender.send(message);
        email.setStatus(true);
    }catch(Exception e){
        email.setText(e.getMessage());
        email.setStatus(false);
    }

    return new ResponseEntity<Email>(email, HttpStatus.OK);
}
}
```

45.5. Test

```
$ curl -i -H "Accept: application/json" -H "Content-Type:
application/json" -X POST -d '{"from":"root@netkiller.cn",
"to":"21214094@qq.com","subject":"Hello","text":"Hello
world!!!"}' http://172.16.0.20:8080/v1/email/send.json
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Wed, 10 Aug 2016 06:38:00 GMT

{"from":"root@netkiller.cn","to":"21214094@qq.com","subject":"H
ello","text":"Hello world!!!","status":true}
```

46. Spring boot with Hessian

46.1. Maven

```
    <dependency>
        <groupId>com.caucho</groupId>
        <artifactId>hessian</artifactId>
        <version>4.0.38</version>
    </dependency>
```

46.2. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
//import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
//import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan
// @EnableMongoRepositories
```

```
// @EnableJpaRepositories
@EnableScheduling
public class Application {

public static void main(String[] args) {
    SpringApplication.run(Application.class, args);

}
}
```

46.3. HessianServiceExporter

```
package cn.netkiller.config;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.remoting.caucho.HessianProxyFactoryBean;
//import
org.springframework.remoting.caucho.HessianProxyFactoryBean;
import
org.springframework.remoting.caucho.HessianServiceExporter;

import cn.netkiller.service.HelloWorldService;

@Configuration
public class HessionConfig {
@Autowired
private HelloWorldService helloWorldService;

@Bean(name = "/HelloWorldService")
public HessianServiceExporter hessianServiceExporter() {
    HessianServiceExporter exporter = new
HessianServiceExporter();
    exporter.setService(helloWorldService);
    exporter.setServiceInterface(HelloWorldService.class);
    return exporter;
```

```
}

@Bean
public HessianProxyFactoryBean helloClient() {
    HessianProxyFactoryBean factory = new
HessianProxyFactoryBean();

factory.setServiceUrl("http://localhost:7000/HelloWorldServic
e");
    factory.setServiceInterface(HelloWorldService.class);
    return factory;
}
}
```

46.4. Service

```
package cn.netkiller.service;

public interface HelloWorldService {
String sayHello(String name);
}
```

```
package cn.netkiller.service.impl;

import org.springframework.stereotype.Component;

import cn.netkiller.service.HelloWorldService;

@Component
public class HelloWorldServiceImpl implements
HelloWorldService {
@Override
public String sayHello(String name) {
    return "Hello World! " + name;
```

```
}
}
```

46.5. RestController

```
package cn.netkiller.rest.hession;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import cn.netkiller.service.HelloWorldService;

@RestController
@RequestMapping("/public/hession")
public class TestRestController {
@Autowired
HelloWorldService helloWorldService;

@RequestMapping("/hello")
public String test() {
    return helloWorldService.sayHello("Spring boot with
Hessian.");
}
}
```

47. Spring boot with Async

47.1. Callable 实现异步

```
@GetMapping("/email")
public Callable<String> order() {
    System.out.println("主线程开始: " +
Thread.currentThread().getName());
    Callable<String> result = () -> {
        System.out.println("副线程开始: " +
Thread.currentThread().getName());
        Thread.sleep(1000);
        System.out.println("副线程返回: " +
Thread.currentThread().getName());
        return "success";
    };

    System.out.println("主线程返回: " +
Thread.currentThread().getName());
    return result;
}
```

47.2. WebAsyncTask 实现异步

```
@GetMapping("/webAsyncTask")
public WebAsyncTask<String> webAsyncTask() {
    log.info("外部线程: " + Thread.currentThread().getName());
    WebAsyncTask<String> result = new WebAsyncTask<>(60 *
1000L, new Callable<String>() {
        @Override
        public String call() {
            log.info("内部线程: " +
Thread.currentThread().getName());
            return "success";
```

```
        }
    });
    result.onTimeout(new Callable<String>() {
        @Override
        public String call() {
            log.info("timeout callback");
            return "timeout callback";
        }
    });
    result.onCompletion(new Runnable() {
        @Override
        public void run() {
            log.info("finish callback");
        }
    });
    return result;
}
```

47.3. DeferredResult 实现异步访问

```
        private DeferredResult<String> deferredResult = new
DeferredResult<String>();

        @ResponseBody
    @GetMapping("/receive")
    public DeferredResult<String> receive() throws Exception
{
        return deferredResult;
    }

    @ResponseBody
    @GetMapping("/send")
    public void send() throws Exception {
        deferredResult.setResult("Helloworld!!!");
    }
```

```
        private final List<DeferredResult<String>>
deferredResultList = new ArrayList<DeferredResult<String>>();

    @ResponseBody
    @GetMapping("/receive")
    public DeferredResult<String> receive() throws Exception
{
        DeferredResult<String> deferredResult = new
DeferredResult<>();

        //先存起来，等待触发
        deferredResultList.add(deferredResult);
        return deferredResult;
    }

    @ResponseBody
    @GetMapping("/send")
    public void send() throws Exception {
        // 让所有hold住的请求给与响应
        deferredResultList.forEach(d -> d.setResult("say
hello to all"));
    }
```

DeferredResult 与 Callback 配合使用，用来获取 Callback 返回值

```
    @GetMapping("/tts")
    @Operation(summary = "音频合成")
    @ResponseBody
    public DeferredResult<ResponseJson>
test(@RequestParam("text") String text,
@RequestParam("filename") String filename) {
        DeferredResult<ResponseJson> deferredResult = new
DeferredResult<ResponseJson>();
        speechSynthesizerService.tts(text, new
XfyunCallback() {
            @Override
            public void onCallback(String sid, String text) {
```

```
                String audio =
aliyunService.uploadMp3FromBase64(text,
filename.concat(".mp3"));
                ResponseJson response = new
ResponseJson(true, ResponseJson.Code.SUCCESS, "", audio);
                deferredResult.setResult(response);
            }
        });
        return deferredResult;
    }
```

47.4. SimpleAsyncTaskExecutor

启用异步执行 @EnableAsync

```
@EnableAsync
@SpringBootApplication
public class ThreadPoolApplication {

    public static void main(String[] args) {
        SpringApplication.run(ThreadPoolApplication.class,
args);
    }

}
```

编写异步执行代码

```
@Component
@Slf4j
public class AsyncTask {
    @Async
    public void  asyncRun() throws InterruptedException {
```

```
        Thread.sleep(10);
        log.info(Thread.currentThread().getName()+":处理完成");
    }
}
```

配置线程池

默认线程池的配置很简单，配置参数如下：

```
spring.task.execution.pool.core-size：线程池创建时的初始化线程数，默认为8
spring.task.execution.pool.max-size：线程池的最大线程数，默认为int最大值
spring.task.execution.pool.queue-capacity：用来缓冲执行任务的队列，默认为int最大值
spring.task.execution.pool.keep-alive：线程终止前允许保持空闲的时间
spring.task.execution.pool.allow-core-thread-timeout：是否允许核心线程超时
spring.task.execution.shutdown.await-termination：是否等待剩余任务完成后才关闭应用
spring.task.execution.shutdown.await-termination-period：等待剩余任务完成的最大时间
spring.task.execution.thread-name-prefix：线程名的前缀，设置好了之后可以方便我们在日志中查看处理任务所在的线程池
```

具体配置含义如下：

```
spring.task.execution.pool.core-size=8
spring.task.execution.pool.max-size=20
spring.task.execution.pool.queue-capacity=10
spring.task.execution.pool.keep-alive=60s
spring.task.execution.pool.allow-core-thread-timeout=true
spring.task.execution.shutdown.await-termination=true
spring.task.execution.shutdown.await-termination-period=60
```

```
spring.task.execution.thread-name-prefix=task-
```

```
spring:
  task:
    execution:
      thread-name-prefix: task- # 线程池的线程名的前缀。默认为 task-，建议根据自己应用来设置
      pool: # 线程池相关
        core-size: 8 # 核心线程数，线程池创建时候初始化的线程数。默认为 8 。
        max-size: 20 # 最大线程数，线程池最大的线程数，只有在缓冲队列满了之后，才会申请超过核心线程数的线程。默认为 Integer.MAX_VALUE
        keep-alive: 60s # 允许线程的空闲时间，当超过了核心线程之外的线程，在空闲时间到达之后会被销毁。默认为 60 秒
        queue-capacity: 200 # 缓冲队列大小，用来缓冲执行任务的队列的大小。默认为 Integer.MAX_VALUE 。
        allow-core-thread-timeout: true # 是否允许核心线程超时，即开启线程池的动态增长和缩小。默认为 true 。
      shutdown:
        await-termination: true # 应用关闭时，是否等待定时任务执行完成。默认为 false ，建议设置为 true
        await-termination-period: 60 # 等待任务完成的最大时长，单位为秒。默认为 0 ，根据自己应用来设置
```

47.5. ThreadPoolTaskExecutor 自定义线程池

Bean 注入配置线程池

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.scheduling.annotation.EnableAsync;
import
org.springframework.scheduling.concurrent.ThreadPoolTaskExecu
```

```
tor;

import java.util.concurrent.Executor;

@SpringBootApplication
@EnableAsync
public class Application {

        public static void main(String[] args) {
            // close the application context to shut down the
custom ExecutorService
            SpringApplication.run(Application.class,
args).close();
        }

        @Bean
        public Executor asyncExecutor() {
            ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
            executor.setCorePoolSize(2);
            executor.setMaxPoolSize(2);
            executor.setQueueCapacity(500);
            executor.setThreadNamePrefix("Netkiller -");
            executor.initialize();
            return executor;
        }

        @Bean("thread")
    public Executor taskExecutor() {
        ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
        // 设置核心线程数
        executor.setCorePoolSize(5);
        // 设置最大线程数
        executor.setMaxPoolSize(10);
        // 设置队列容量
        executor.setQueueCapacity(20);
        // 设置线程活跃时间（秒）
        executor.setKeepAliveSeconds(60);
        // 设置线程名称
        executor.setThreadNamePrefix("hello-");
        // 设置拒绝策略
        executor.setRejectedExecutionHandler(new
ThreadPoolExecutor.CallerRunsPolicy());
        // 等待所有任务结束后再关闭线程池
```

```
        executor.setWaitForTasksToCompleteOnShutdown(true);

        return executor;
    }

}
```

设置线程池参数

最简单的配置

```
@SpringBootApplication
@EnableAsync
public class Application {
public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
}
}
```

```
@Component
public class Task {

        @Async
        public void doTaskOne() throws Exception {
            // 业务逻辑
        }

        @Async
        public void doTaskTwo() throws Exception {
            // 业务逻辑
        }

        @Async("asyncExecutor")
```

```
        public void doTaskThree() throws Exception {
            // 业务逻辑
        }

}
```

队列

线程池能接受多少队列?

下面配置是 executor.setQueueCapacity(10); 也就是 10个，但是实测结果跟你想的不同

```
package cn.netkiller.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableAsync;
import
org.springframework.scheduling.concurrent.ThreadPoolTaskExecu
tor;

import java.util.concurrent.ThreadPoolExecutor;

@Configuration
@EnableAsync
public class ThreadPoolTaskExecutorConfiguration {
    @Bean("asyncExecutor")
    public ThreadPoolTaskExecutor executor() {
        ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
        executor.setThreadGroupName("job");
        executor.setThreadNamePrefix("async-job-");
        executor.setCorePoolSize(5);
        executor.setMaxPoolSize(10);
        executor.setQueueCapacity(10);
        executor.setKeepAliveSeconds(60);
        executor.setRejectedExecutionHandler(new
```

```
ThreadPoolExecutor.AbortPolicy());
        executor.setAwaitTerminationSeconds(60);
        executor.setWaitForTasksToCompleteOnShutdown(true);
        executor.initialize();
        return executor;
    }
}
```

实测结果是，首次执行可以容纳 20 个线程，20个线程执行完毕之后，再添加任务，就只接受 10 个，超过的部分会跑出异常

```
 Executor
[java.util.concurrent.ThreadPoolExecutor@7e729046[Running, pool
size = 10, active threads = 10, queued tasks = 10, completed
tasks = 0]] did not accept task:
org.springframework.aop.interceptor.AsyncExecutionInterceptor$$
Lambda$1775/0x0000000801b6afb0@20eaccc2
```

这是因为线程池可以容纳 10 个任务，队列还能排队 10 个任务。

47.6. 定义多个线程池

```
package cn.netkiller.wallet.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableAsync;
```

```
@Configuration
@EnableAsync
public class ExecutorConfiguration {
        /** Set the ThreadPoolExecutor's core pool size. */
        private int corePoolSize = 10;
        /** Set the ThreadPoolExecutor's maximum pool size.
*/
        private int maxPoolSize = 200;
        /** Set the capacity for the ThreadPoolExecutor's
BlockingQueue. */
        private int queueCapacity = 10;

        @Bean
        public Executor OneAsync() {
            ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
            executor.setCorePoolSize(corePoolSize);
            executor.setMaxPoolSize(maxPoolSize);
            executor.setQueueCapacity(queueCapacity);
            executor.setThreadNamePrefix("MySimpleExecutor-
");
            executor.initialize();
            return executor;
        }

        @Bean
        public Executor TwoAsync() {
            ThreadPoolTaskExecutor executor = new
ThreadPoolTaskExecutor();
            executor.setCorePoolSize(corePoolSize);
            executor.setMaxPoolSize(maxPoolSize);
            executor.setQueueCapacity(queueCapacity);
            executor.setThreadNamePrefix("MyExecutor-");

            // rejection-policy：当pool已经达到max size的时候，如
何处理新任务
            // CALLER_RUNS：不在新线程中执行任务，而是有调用者所在的
线程来执行
            executor.setRejectedExecutionHandler(new
ThreadPoolExecutor.CallerRunsPolicy());
            executor.initialize();
            return executor;
        }

}
```

```
@Service
public class DemoAsyncServiceImpl implements DemoAsyncService
{

        public static Random random =new Random();

        @Async("OneAsync")
        public Future<String> doTaskOne() throws Exception {
            System.out.println("开始做任务一");
            long start = System.currentTimeMillis();
            Thread.sleep(random.nextInt(10000));
            long end = System.currentTimeMillis();
            System.out.println("完成任务一，耗时：" + (end -
start) + "毫秒");
            return new AsyncResult<>("任务一完成");
        }

        @Async("TwoAsync")
        public Future<String> doTaskTwo() throws Exception {
            System.out.println("开始做任务二");
            long start = System.currentTimeMillis();
            Thread.sleep(random.nextInt(10000));
            long end = System.currentTimeMillis();
            System.out.println("完成任务二，耗时：" + (end -
start) + "毫秒");
            return new AsyncResult<>("任务二完成");
        }

        @Async
        public Future<String> doTaskThree() throws Exception
{
            System.out.println("开始做任务三");
            long start = System.currentTimeMillis();
            Thread.sleep(random.nextInt(10000));
            long end = System.currentTimeMillis();
            System.out.println("完成任务三，耗时：" + (end -
start) + "毫秒");
            return new AsyncResult<>("任务三完成");
```

```
        }

}
```

47.7. 自定义线程池

自定义线程池

ThreadPoolExecutor

```
    @Bean("queueThreadPool")
    public ThreadPoolExecutor queueThreadPool() {
        ThreadPoolExecutor threadPoolExecutor = new
ThreadPoolExecutor(
                5,
                10,
                60,
                TimeUnit.SECONDS,
                new LinkedBlockingDeque<>(10),
                Executors.defaultThreadFactory(),
                new ThreadPoolExecutor.AbortPolicy());
        return threadPoolExecutor;
    }
```

注入自定义线程池bean

```
         // 注入自定义线程池bean
    @Autowired
    private ThreadPoolExecutor threadPoolExecutor;

    threadPoolExecutor.execute(new Runnable() {
        @Override
```

```
        public void run() {
            System.out.println("=======");

        }
    });
```

47.8. 设置线程名称

```
public static Random random = new Random();

    @Async("jobExecutor")
    public void doAsyncTask() throws InterruptedException {
        Thread.currentThread().setName("测试线程-" +
random.nextInt(1000));
        System.out.println("开始做任务一");
        long start = System.currentTimeMillis();
        Thread.sleep(random.nextInt(100000));
        long end = System.currentTimeMillis();
        System.out.println("完成任务一，耗时：" + (end - start)
+ "毫秒");
    }
```

47.9. 线程池监控

监控指标

```
neo@MacBook-Pro-M2 ~> curl -s
http://www.netkiller.cn:8080/actuator/metrics | jq | grep
executor
    "executor.active",
    "executor.completed",
    "executor.pool.core",
```

```
    "executor.pool.max",
    "executor.pool.size",
    "executor.queue.remaining",
    "executor.queued",
```

获取指标

```
neo@MacBook-Pro-M2 ~> curl -s
http://www.netkiller.cn:8080/actuator/metrics/executor.active |
jq
{
  "name": "executor.active",
  "description": "The approximate number of threads that are
actively executing tasks",
  "baseUnit": "threads",
  "measurements": [
    {
      "statistic": "VALUE",
      "value": 0
    }
  ],
  "availableTags": [
    {
      "tag": "name",
      "values": [
        "asyncExecutor"
      ]
    }
  ]
}
```

```
    @Autowired
    ThreadPoolTaskExecutor threadPoolTaskExecutor;

    @GetMapping("/pool")
    public String pool() {
```

```
        int activeCount =
threadPoolTaskExecutor.getActiveCount();
        long completedTaskCount =
threadPoolTaskExecutor.getThreadPoolExecutor().getCompletedTa
skCount();
        long taskCount =
threadPoolTaskExecutor.getThreadPoolExecutor().getTaskCount()
;
        int queue =
threadPoolTaskExecutor.getThreadPoolExecutor().getQueue().siz
e();
        String monitor = String.format("Task: %d, Queue: %d,
Active: %d, Completed: %d\n", taskCount, queue, activeCount,
completedTaskCount);
        log.info(monitor);
        return monitor;
    }
```

48. Springboot with Ethereum (web3j)

48.1. Maven

```
    <dependency>
        <groupId>org.web3j</groupId>
        <artifactId>web3j-spring-boot-starter</artifactId>
        <version>1.6.0</version>
    </dependency>
```

48.2. application.properties

```
web3j.client-
address=https://ropsten.infura.io/CsS9shwaAab0z7B4LP2d
web3j.admin-client=true
```

48.3. TestRestController

```
package cn.netkiller.wallet.restful;

import java.io.IOException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;
import org.web3j.protocol.Web3j;
```

```
import
org.web3j.protocol.core.methods.response.Web3ClientVersion;

@RestController
public class TestRestController {
private static final Logger logger =
LoggerFactory.getLogger(TestRestController.class);

@Autowired
private Web3j web3j;

public TestRestController() {
    // TODO Auto-generated constructor stub
}

@GetMapping("/version")
public String version() throws IOException {
    Web3ClientVersion web3ClientVersion =
web3j.web3ClientVersion().send();
    String clientVersion =
web3ClientVersion.getWeb3ClientVersion();
    logger.info(clientVersion);
    return clientVersion;
}

}
```

48.4. 测试

```
neo@MacBook-Pro ~ % curl http://localhost:8080/version
Geth/v1.8.3-stable/linux-amd64/go1.10
```

49. Java Record 新特性

49.1. Record 替代 POJO 类

POJO类：

```
@Data
public class Point {
    private String x;
    private String y;
}
```

record 实现方式

```
public record Point(String x, String y) {
}
```

49.2. Record 作为 Properties

```
@ConfigurationProperties(prefix = "user")
public record UserProperties(String firstName, String
lastName) {
}
```

49.3. Record 作为实体类

```
package cn.netkiller.record;

import jakarta.persistence.*;

@Entity
@Table(name = "member")
public record Member(
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false,
insertable = true, updatable = false)
        Long id,
        String name

) {
}
```

49.4. Record 作为 Service

```
@Service
public record PersonService(PersonRepository
personRepository){

        //保存person
    public Person save(Person person){
        Person person = new Person(person.firstName(),
person.lastName(), person.age());
        return personRepository.save(person);
    }

        //按照lastName查询people，返回值只有firstName和lastName
    public List<PersonOnlyWithName> findByLastName(String
lastName){
        return personRepository.findByLastName(lastName);
    }
}
```

```
public interface PersonRepository extends
JpaRepository<Person, Long> {
    List<PersonOnlyWithName> findByLastName(String lastName);
}
```

49.5. Record 作为 Controller

```
@RestController
@RequestMapping("/people")
public record PersonController(PersonService personService) {

    @PostMapping
    public ResponseEntity<Person> save(@RequestBody Person
person){
        return ResponseEntity.ok(personService.save(person));
    }

    @GetMapping("/findByLastName")
    public ResponseEntity<List<PersonOnlyWithName>>
findByLastName(String lastName){
        return
ResponseEntity.ok(personService.findByLastName(lastName));
    }
}
```

50. Springboot 接入阿里云

50.1. 阿里云 OSS - STS进行临时授权访问获取 HTTPS 地址

默认获取URL是 HTTP，我们的需求是需要获取HTTPS地址，当然你可以使用字符串替换操作，将 http 替换成 https，但这并不是最有解决方案。

```
package cn.netkiller.aliyun;

import java.net.URL;
import java.util.Date;

import com.aliyun.oss.ClientBuilderConfiguration;
import com.aliyun.oss.OSS;
import com.aliyun.oss.OSSClientBuilder;
import com.aliyun.oss.common.comm.Protocol;

public class App {
        public static void main(String[] args) {
                System.out.println("Hello World!");

                // yourEndpoint填写Bucket所在地域对应的Endpoint。以华东1（杭州）为例，Endpoint填写为https://oss-cn-hangzhou.aliyuncs.com。
                String endpoint = "oss-cn-shanghai.aliyuncs.com";
                // 从STS服务获取的临时访问密钥（AccessKey ID和AccessKey Secret）。
                String accessKeyId = "DYmJeLTAI5tm1ZaCEB9nUxAP";
                String accessKeySecret = "QkXusBiLMoMIsW3JJhG0D5NOFBEh5a";
                // 从STS服务获取的安全令牌（SecurityToken）。
                String securityToken = "yourSecurityToken";
                // 填写Bucket名称，例如examplebucket。
                String bucketName = "production";
                // 填写Object完整路径，例如exampleobject.txt。
```

```
Object完整路径中不能包含Bucket名称。
                String objectName = "exampleobject.txt";

                ClientBuilderConfiguration
clientBuilderConfiguration = new
ClientBuilderConfiguration();

clientBuilderConfiguration.setProtocol(Protocol.HTTPS);
                // 创建OSSClient实例。
                OSS ossClient = new
OSSClientBuilder().build(endpoint, accessKeyId,
accessKeySecret, securityToken, clientBuilderConfiguration);
                // 设置签名URL过期时间为3600秒（1小时）。
                Date expiration = new Date(new
Date().getTime() + 3600 * 1000);
                // 生成以GET方法访问的签名URL，访客可以直接通过浏览器
访问相关内容。
                URL url =
ossClient.generatePresignedUrl(bucketName, objectName,
expiration);
                System.out.println(url);

System.out.println(url.toString().replace(bucketName + "." +
endpoint, "oss.netkiller.cn"));
                // 关闭OSSClient。
                ossClient.shutdown();
        }
}
```

第 6 章 Spring MVC

Spring MVC 有两种启动模式，一种是传统Tomcat，需要配置很多XML文件。另一种方式是采用 Spring Boot 需要些一个Java程序，不需要写xml文件，这个程序会帮助你处理启动所需的一切，并且采用嵌入方式启动 Tomcat 或者 Jetty.

两种方式各有优缺点，Tomcat 方式配置繁琐，但是可以使用虚拟机，同一个IP地址使用不同域名访问，出现不同的内容。而Spring Boot一个应用一个容器一个端口，比不得不通过端口来区分应用。

1. @EnableWebMvc

```
package cn.netkiller.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.DefaultServ
letHandlerConfigurer;
import
org.springframework.web.servlet.config.annotation.EnableWebMv
c;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;
import
org.springframework.web.servlet.view.InternalResourceViewReso
lver;

@Configuration
@EnableWebMvc
public class WebMvcConfig extends WebMvcConfigurerAdapter {

        @Override
        public void
configureDefaultServletHandling(DefaultServletHandlerConfigur
```

```
er configurer) {
                configurer.enable();
        }

        @Bean
        public InternalResourceViewResolver viewResolver() {
                InternalResourceViewResolver resolver = new
InternalResourceViewResolver();
                resolver.setPrefix("WEB-INF/jsp/");
                resolver.setSuffix(".jsp");
                return resolver;
        }

}
```

1.1. CORS 跨域请求

```
@Configuration
public class CorsConfiguration
{
    @Bean
    public WebMvcConfigurer corsConfigurer()
    {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry
registry) {
                registry.addMapping("/**");
            }
        };
    }
}
```

```
        @Bean
```

```
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry
registry) {

registry.addMapping("/**").allowedOrigins("*");
            }
        };
    }
```

1.2. Spring MVC CORS with WebMvcConfigurerAdapter

```
@Configuration
@EnableWebMvc
public class CorsConfiguration extends
WebMvcConfigurerAdapter
{
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**").allowedMethods("GET",
"POST");
    }
}
```

```
@Configuration
@EnableWebMvc
public class AppConfig extends WebMvcConfigurerAdapter {
        @Override
        public void addCorsMappings(CorsRegistry registry) {
          registry.addMapping("/info/**")
                  .allowedOrigins("http://localhost:8080",
"http://localhost:8000")
                  .allowedMethods("POST", "GET",  "PUT",
```

```
"OPTIONS", "DELETE")
                    .allowedHeaders("X-Auth-Token", "Content-
Type")
                    .exposedHeaders("custom-header1", "custom-
header2")
                    .allowCredentials(false)
                    .maxAge(4800);
        }
}
```

2. @Controller

```
package cn.netkiller.controller;

import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class Welcome {

        @RequestMapping("/welcome")
        public ModelAndView helloWorld() {
                String message = "Helloworld!!!";
                return new ModelAndView("welcome", "message",
message);
        }
}
```

2.1. @RequestMapping

```
@RequestMapping("/welcome")
```

```
@RequestMapping(value = "/list", method =
RequestMethod.GET)
```

@RequestMapping("/")

```
package com.cf88.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/")
public class HelloController {

        @RequestMapping(value = "/{name}", method =
RequestMethod.GET)
        public String getMovie(@PathVariable String name,
ModelMap model) {
                model.addAttribute("name", name);
                return "hello";
        }

}
```

同时支持多种操作方法

```
        @RequestMapping(value = "/name", method = {
RequestMethod.GET, RequestMethod.POST })
```

映射多个URL

```
@RequestMapping({ "/news/zh-cn", "/news/zh-tw" })
@ResponseBody
public String getNewsByPath() {
        return "Hello";
}
```

匹配通配符

```
@Controller
@RequestMapping("/test/*")

public class TestController {

        @RequestMapping
        public String default() {
                return "OK";
        }
}
```

headers

```
@RequestMapping(value = "/news/json", method = GET, headers =
"Accept=application/json")
@ResponseBody
public String getFoosAsJsonFromBrowser() {
    return "{...}";
}
```

```
curl -H "Accept:application/json,text/html"
http://localhost:8080/spring/news/json.html
```

2.2. @GetMapping

@GetMapping 等效与 @RequestMapping

```
@RequestMapping(value = "/news/list", method = GET)
```

范例

```
import org.springframework.web.bind.annotation.GetMapping;

        @GetMapping("/finance/list")
        public String financeList() {
                return financeService.financeList();
        }
```

```
@GetMapping(value = "/user",produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
```

2.3. @PostMapping

@GetMapping 等效与 @RequestMapping

```
@RequestMapping(value = "/news/list", method =
RequestMethod.POST)
```

范例

```
import org.springframework.web.bind.annotation.PostMapping;

        @PostMapping("/finance/list")
        public String financeList() {
                return financeService.financeList();
        }
```

Content-Type: multipart/form-data

```
@PostMapping(path = "/upload", consumes =
MediaType.MULTIPART_FORM_DATA_VALUE)
```

2.4. @RequestBody

原始数据

处理 raw 原始数据，例如提交的时 application/json, application/xml 等

```
@RequestMapping(value = "/something", method =
RequestMethod.PUT)
public void handle(@RequestBody String body, Writer writer)
throws IOException {
        writer.write(body);
}
```

@RequestBody 传递 List

```
package cn.netkiller.api.restful;

import java.util.List;

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class TestRestController {

        @RequestMapping(value = "/test/list/{siteId}", method
= RequestMethod.POST)
        public List<String> ping(@PathVariable("siteId") int
siteId, @RequestBody List<String> tags) {
                System.out.println(String.format("%d, %s",
siteId, tags));
                return (tags);
        }

}
```

```
$ curl -H "Content-Type: application/json" -X POST -d
'["Neo","Netkiller"]' http://localhost:8440/test/list/22.json

["Neo","Netkiller"]
```

传递 Map 数据

```
        @PostMapping("/finance/list")
        public String financeList(@RequestBody
Map<String,String> map) {
                return financeService.financeList(map);
        }
```

```
% curl -H "Content-Type: application/json" -X POST -d
'{"date":"2017-11-08"}'
http://localhost:8440/finance/list.json
```

获取 JSONObject 数据

```
    @PostMapping(value = "/{device}/post")
    public Mono<String> post(@PathVariable String device,
@RequestBody JSONObject jsonObject) {
        log.info(jsonObject.toString());
        return Mono.just(jsonObject.toString());
    }
```

2.5. RequestMapping with Request Parameters - @RequestParam

@RequestParam 用来处理 HTTP GET/POST 请求的变量

```
import

org.springframework.web.bind.annotation.RequestParam;
```

HTTP GET

```
        @RequestMapping("/request/param")
        @ResponseBody
        public String
getBarBySimplePathWithRequestParam(@RequestParam("id") long
id) {
            return "Get a specific Bar with id=" + id;
        }
```

```
http://localhost:8080/Spring/request/param.html?id=100
```

HTTP POST

```
package cn.netkiller.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class Http {

        @RequestMapping("/http/form")
        public ModelAndView createCustomer(){
                ModelMap model = new ModelMap();

                model.addAttribute("email",
"netkiller@msn.com");
                model.addAttribute("phone", "13113668890");

            return new ModelAndView("http/form", model);
        }

        @RequestMapping(value= "/http/post", method =
RequestMethod.POST)
        public ModelAndView saveCustomer(HttpServletRequest
request,
                @RequestParam(value="Email", required=false)
String email,
                @RequestParam(value="Password",
required=false) String password,
                @RequestParam(value="Phone", required=false)
String phone){

                ModelMap model = new ModelMap();

                model.addAttribute("email", email);
                model.addAttribute("password", password);
                model.addAttribute("phone", phone);
```

```
            return new ModelAndView("http/post", model);
        }

}
```

http/form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

        <form method="POST"

action="http://localhost:8080/Spring/http/post.html"
id="Register"
                name="Register">
                Email: <input class="register" type="text"
id="Email" name="Email" value="${email}" /> <br />
                Password:  <input class="register"
type="password" id="Password" name="Password" value="" /><br
/>
                Phone:  <input class="register" type="text"
id="Phone" name="Phone" value="${phone}" /> <br />
                <input type="submit" id="btnRegister"
name="btnRegister" value="Register" style="cursor: pointer"
/>
        </form>

</body>
</html>
```

```
```

http/post.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
        ${email}<br>
        ${password}     <br>
        ${phone} <br>
</body>
</html>
```

@RequestParam 传递特殊字符串

URL 中 “+” 有特殊意义，表示空格。

如果 @RequestParam 传递参数含有空格可以这样处理。

```
        @RequestMapping("/RequestParam")
        @ResponseBody
        public String query(@RequestParam("code") String
code) {

                return code.replace(" ", "+");
```

```
        }
```

传递日期参数

```
        @RequestMapping("/range")
        public ModelAndView range(@RequestParam("beginDate")
@DateTimeFormat(pattern = "yyyy-MM-dd") Date beginDate,
@RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-
dd") Date endDate) {
                log.info("===== Begin ===== {}", beginDate);

                // 你的逻辑

                log.info("===== End ===== {}", endDate);
                return new ModelAndView("activity/index",
"message", "操作成功");
        }
```

上传文件

```
package cn.netkiller.restful;

import java.io.IOException;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import
org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.multipart.MultipartFile;

@RestController
@RequestMapping("/upload")
public class UploadRestController {

        private static final Logger logger =
LoggerFactory.getLogger(UploadRestController.class);

        public UploadRestController() {
                // TODO Auto-generated constructor stub
        }

        @PostMapping("/add")
        public String fileUpload(@RequestParam("file")
MultipartFile multipartFile) throws IOException {

                String name =
multipartFile.getOriginalFilename();
                System.out.println("File name: " + name);
                // todo save to a file via
multipartFile.getInputStream()
                byte[] bytes = multipartFile.getBytes();
                System.out.println("File uploaded content:\n"
+ new String(bytes));
                return "file uploaded";
        }

}
```

操作演示，首先创建一个文件

```
echo "Helloworld!!!" > hello.txt
```

上传该文件

```
neo@MacBook-Pro /tmp % curl "http://localhost:8080/upload/add"
\
    -X POST \
    -H "Content-Type: multipart/form-data" \
    -F file=@"hello.txt"

file uploaded
```

@RequestParam - POST 数组

HTTP 头

```
picture[]: gather/293a93baa02cb18a840631bac1f9eeb20b7d436f.jpeg
picture[]: gather/be7572e4df527b4389d605766ea65aafcf2d822a.jpg
```

```
        @PostMapping("/save")
        public String save(@RequestParam(value = "picture[]",
 required = true) String[] picture) {
                return String.join(",", picture);
        }
```

默认值

```
@RequestParam(name = "name", defaultValue = "xxx") String
```

```
name
```

是否非必须

使用 required = false 标注参数是非必须的

```
@RequestParam(name = "age", required = false) Integer age
```

用 Map 接收 From 数据

```
    @PostMapping("/token")
    @ResponseBody
    public String token(@RequestParam Map<String, String>
params) {
        log.debug(params.toString());
        String token =
jwtTokeComponent.getTestToken(params.get("appId"),
params.get("appKey"), params.get("subject"),
params.get("audience"));
        return token;
    }
```

2.6. @RequestHeader - 获取 HTTP Header 信息

```
@RequestMapping("/displayHeaderInfo")
public void displayHeaderInfo(@RequestHeader("Accept-
Encoding") String encoding,
```

```
                                    @RequestHeader("Keep-Alive")
long keepAlive)  {

  //...

}
```

获取用户当前语言

```
        @GetMapping("/lang")
        public String language(@RequestHeader("Accept-
Language") String locale ) {
                System.out.println(locale);
                return locale;
        }
```

下面代码可以获得相同效果

```
        @GetMapping("/lang")
        public String language(Locale locale) {
                System.out.println(locale);
                return locale;
        }

        @GetMapping("/lang")
        public String language() {
                String locale =
LocaleContextHolder.getLocale().toString();
                System.out.println(locale);
                return locale;
        }
```

@RequestHeader 从 Http 头中获取变量

```
        @PostMapping(value = "/token")
        public TokenResponse token(@RequestParam String
symbol, @RequestHeader String token) {

                        TokenResponse tokenResponse =
walletService.getTokenBySymbol(symbol);

                        return tokenResponse;
        }
```

2.7. RequestMapping with Path Variables - @PathVariable

PATHINFO 变量可通过 @Pathvariable注解绑定它传过来的值到方法的参数上。

URL 参数传递

需求，我们需要通过URL传递参数，所传递的值是分类ID与文章ID，例如 /news/1.html, /news/1/1.html。

```
package cn.netkiller.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class Pathinfo {
        @RequestMapping("/pathinfo/{id}")
```

```
        public ModelAndView urlTestId(@PathVariable String
id) {

                return new ModelAndView("pathinfo/param",
"id", id);
        }

        @RequestMapping("/pathinfo/{cid}/{id}")
        public ModelAndView urlTestId(@PathVariable String
cid, @PathVariable String id) {

                ModelMap model = new ModelMap();

                model.addAttribute("cid", cid);
                model.addAttribute("id", id);

                return new ModelAndView("pathinfo/param",
model);
        }
}
```

jsp测试文件

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
${ cid } <br>
${ id } <br>
</body>
</html>
```

```

```

默认值

required 设置参数不是必须的

```
@PathVariable(required = false) String id
```

设置多个映射

```
@RequestMapping(value = {"/organization/{pageNumber}",
"/organization"} , method = RequestMethod.GET)
public String list(@PathVariable(required = false) Integer
pageNumber, ModelMap modelMap){
...
}
```

URL 传递 Date 类型

http://localhost:7000/history/2016-09-28%2000:00:00/

```
        @RequestMapping("/history/{datetime}")
        public String history(@PathVariable
@DateTimeFormat(pattern="yyyy-MM-dd HH:mm:ss") Date datetime)
throws Exception {

                System.out.println(datetime)
```

```
                return null;
        }
```

处理特殊字符

http://www.netkiller.cn/release/1.0.1

```
@RequestMapping(value = "/release/{version:[a-zA-Z0-9\\.]+}",
method = RequestMethod.GET)
public @ResponseBody
        String release(@PathVariable String version) {
        log.debug("version: ", version);
        return version;
}
```

http://www.netkiller.cn/release/1.0.1/other

```
@RequestMapping(value="/release/{version:.+}",method=RequestM
ethod.GET)
public void download(HttpSession
        session,@PathVariable("version")String version){
        return version;
}
```

@PathVariable 注意事项

@PathVariable 参数传统需要注意，参数中不能携带 “/”,斜杠会被视为目录。

```
        @RequestMapping("/PathVariable/{code}.html")
        @ResponseBody
        public String urlTestId(@PathVariable String code) {
                return code;
        }
```

2.8. @ModelAttribute

@ModelAttribute 处理 HTML FORM POST 提交

```
package cn.netkiller.pojo;

import java.util.List;

public class Deploy {

        private String group;
        private String envionment;
        private String project;
        private List<String> arguments;
        public Deploy() {
                // TODO Auto-generated constructor stub
        }
        // Getters & Setters
}
```

```
        @RequestMapping(value="/deploy/post",  method =
RequestMethod.POST)
        public ModelAndView
post(@ModelAttribute("deploy")Deploy deploy, BindingResult
result) {
```

```
                if (result.hasErrors()) {

System.out.println(result.toString());
        }
                System.out.println(deploy.toString());
                return new
ModelAndView("output").addObject("output",
deploy.toString());
        }
```

2.9. @ResponseBody

```
import org.springframework.web.bind.annotation.ResponseBody;
```

直接返回HTML

```
package cn.netkiller.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class Pathinfo {

        @RequestMapping(value = "/news/shenzhen/{numericId:
[\\d]+}")
        @ResponseBody
        public String getNewsWithPathVariable(@PathVariable
```

```
final long numericId) {
                return "Get a specific Bar with id=" +
numericId;
        }

}
```

2.10. @ResponseStatus 设置 HTTP 状态

```
        @RequestMapping(value = "/", method =
RequestMethod.POST)
    @ResponseStatus(HttpStatus.CREATED)
    public String create(@RequestBody MultiValueMap<String,
String> map) {
        return "OK";
    }
```

2.11. @CrossOrigin

```
        @CrossOrigin(origins = "http://localhost:9000")
        @GetMapping("/greeting")
        public Greeting
greeting(@RequestParam(required=false, defaultValue="World")
String name) {
                System.out.println("==== in greeting ====");
                return new
Greeting(counter.incrementAndGet(),
String.format(template,name));
        }
```

```
@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class HomeController
{
    @GetMapping(path="/")
    public String home() {
        return "home";
    }
}
```

全局放行所有轻松，方法权限单独控制

```
@RestController
@CrossOrigin(origins = "*", allowedHeaders = "*")
public class HomeController
{
    @CrossOrigin(origins = "http://example.com")
    @GetMapping(path="/")
    public String home() {
        return "home";
    }
}
```

maxAge

```
@CrossOrigin(origins = {"http://localhost:8585"}, maxAge =
4800, allowCredentials = "false")
@RestController
@RequestMapping("/info")
public class PersonController {
        @Autowired
```

```
        private PersonService service;
        @CrossOrigin(origins = {"http://localhost:8080"},
maxAge = 6000)
        @RequestMapping("home")
        public List<Person> show() {
                List<Person> list = service.getAllPerson();
                return list;
        }
}
```

2.12. @CookieValue - 获取 Cookie 值

```
@RequestMapping("/sessionInfo")
public void sessionInfo(@CookieValue("JSESSIONID") String
cookie)  {

  //...

}
```

2.13. @SessionAttributes

@SessionAttributes: 该注解用来绑定HttpSession中的attribute对象的值，便于在方法中的参数里使用。 该注解有value、types两个属性，可以通过名字和类型指定要使用的attribute 对象；

```
@Controller
@RequestMapping("/editProfile")
@SessionAttributes("profile")
public class ProfileForm {
    // ...
}
```

```
@Controller
@SessionAttributes("myRequestObject")
public class MyController {
  ...
}
```

2.14. ModelAndView

变量传递

```
@RequestMapping("/testString")
public ModelAndView     helloWorld() {
        String message = "Helloworld!!!";
        return new
        ModelAndView("welcome", "message", message);
}
```

```
public ModelAndView handleRequestInternal() {

        ModelAndView mav = new ModelAndView("test");//
        实例化一个VIew的ModelAndView实例
        mav.addObject("variable", "Hello World!");//
        添加一个带名的model对象
        return mav;
}
```

ModelMap 传递多个变量

传递多个字符串

```
@RequestMapping("/testModelMap")
public ModelAndView     testModelMap() {
        ModelMap model = new ModelMap();

        model.addAttribute("username", "Neo");
        model.addAttribute("password", "Netkiller");

        return new ModelAndView("test/modelmap", model);
}
```

推荐使用ModelMap

```
        @RequestMapping("/testMapString")
        public ModelAndView testMapString() {

                Map<String,String> data = new
HashMap<String,String>();
                data.put("username","Neo");
                data.put("password","Netkiller");
            return new ModelAndView("test/modelmap",data);

        }
```

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
```

```
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
${username}<br>
${password}<br>
</body>
</html>
```

redirect

```
        @RequestMapping("/testRedirect")
        public ModelAndView testRedirect(){
                    RedirectView view = new
RedirectView("testMapString.html");
                    return new ModelAndView(view);
        }
```

ArrayList

```
        @RequestMapping(value = "testList")
        public ModelAndView testList() {
                ModelAndView mav = new ModelAndView();
                mav.setViewName("/test/list");

                // List
                List<String> list = new ArrayList<String>();
                list.add("java");
                list.add("c++");
```

```
                list.add("oracle");
                mav.addObject("bookList", list);

                return mav;
        }
```

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        ${bookList}
        <br>

        <c:forEach items="${bookList}" var="node">
                <c:out value="${node}"></c:out><br>
        </c:forEach>

</body>
</html>
```

HashMap

```
        @RequestMapping("/testMap")
        public ModelAndView testMap() {
```

```
                ModelAndView mav = new ModelAndView();
                mav.setViewName("test/map"); // 返回的文件名

                // Map
                Map<String, String> map = new HashMap<String,
String>();
                map.put("Java",
"http://www.netkiller.cn/java");
                map.put("PHP",
"http://www.netkiller.cn/php");
                map.put("Home", "http://www.netkiller.cn");
                mav.addObject("channel", map);

                return mav;
        }
```

```
<%@ page language="java" contentType="text/html; charset=utf-
8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <c:forEach items="${channel}" var="node">
                <a href="<c:out value="${node.value}">
</c:out>"><c:out value="${node.key}"></c:out></a>
        <br/>
        </c:forEach>
</body>
</html>
```

传递对象

```
        @RequestMapping("/testObject")
        public ModelAndView testObject() {
                ModelMap model = new ModelMap();

                User user = new User("neo", "passw0rd");
                model.addAttribute("user", user);
                return new ModelAndView("test/object",
model);

        }
```

```
package cn.netkiller;

public class User {
        public String username;
        public String password;
        public User(String username, String password){
                this.username = username;
                this.password = password;
        }
        public String getUsername(){
                return this.username;
        }
        public String getPassword(){
                return this.password;
        }
}
```

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
```

```
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
Username: ${user.username}<br>
Password: ${user.password}<br>
</body>
</html>
```

2.15. HttpServletRequest / HttpServletResponse

HttpServletResponse

HttpServletResponse 实例

```
package cn.netkiller.api.rest;

import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import api.util.MatrixToImageWriter;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

```
import java.util.Hashtable;

@Controller
@RequestMapping("/public/QRCode")
public class QRCodeController {
    private static final Logger log =
LoggerFactory.getLogger(QRCodeController.class);

    @RequestMapping("/create/{code}" )
    @ResponseBody
    public void create(@PathVariable String code,
HttpServletResponse httpServletResponse) throws
WriterException, IOException {
        log.info(code);
        if (code != null && !"".equals(code)){
            ServletOutputStream stream = null;
            try {
                String text = code;     // 二维码内容
                int width = 300;                // 二维码图片宽
度
                int height = 300;               // 二维码图片高
度
                String format = "gif";  // 二维码的图片格式

                Hashtable<EncodeHintType, String> hints = new
Hashtable<EncodeHintType, String>();
                hints.put(EncodeHintType.CHARACTER_SET, "utf-
8");   // 内容所使用字符集编码

                BitMatrix bitMatrix = new
MultiFormatWriter().encode(text, BarcodeFormat.QR_CODE,
width, height, hints);
                // 生成二维码
                stream =
httpServletResponse.getOutputStream();
                MatrixToImageWriter.writeToStream(bitMatrix,
format, stream);

            }catch (WriterException e) {
                e.printStackTrace();
            } finally {
                if (stream != null) {
                    stream.flush();
                    stream.close();
                }
```

```
            }
        }
    }

    @RequestMapping("show")
    @ResponseBody
    public ModelAndView show(){

        return new ModelAndView("/qrcode/qrcode");
    }
}
```

HttpServletRequest

```
package com.example.demo.controller;

import java.io.IOException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/test")
public class TestController {

        public TestController() {
                // TODO Auto-generated constructor stub
        }
```

```
        @GetMapping("/get")
        public String get(@RequestHeader String lang) throws
IOException {
                System.out.println(lang);
                return lang;
        }

        @PostMapping("/post")
        public String post(@RequestHeader String lang) throws
IOException {
                System.out.println(lang);
                return lang;
        }

        @GetMapping("/list")
        public Map<String, String> x(HttpServletRequest
request) throws IOException {

                return getHeadersInfo(request);
        }

        private Map<String, String>
getHeadersInfo(HttpServletRequest request) {

                Map<String, String> map = new HashMap<String,
String>();

                Enumeration<?> headerNames =
request.getHeaderNames();
                while (headerNames.hasMoreElements()) {
                        String key = (String)
headerNames.nextElement();
                        String value =
request.getHeader(key);
                        map.put(key, value);
                }

                return map;
        }

}
```

3. @RestController

3.1. 上传文件

```
    @PostMapping("/{device}/question/voice")
    public String questionVoice(@PathVariable String device,
@RequestParam("file") MultipartFile file,
@RequestParam("session") String session) {

        if (file.isEmpty()) {
            logger.error("上传失败，请选择文件");
        }

        String fileName = file.getOriginalFilename();
        String filePath =
"/tmp/".concat(session.concat(".mp3"));
        File saveAs = new File(filePath);
        try {
            file.transferTo(saveAs);
            logger.info("上传成功 " + fileName);
        } catch (IOException e) {
            logger.error(e.toString(), e);
            return "上传失败";
        }
        return "上传成功";
    }
```

3.2. 返回实体

```
        @RequestMapping("/get/{id}")
        public Member getStatistics(@PathVariable long id) {
                Member statistics =
memberRepostitory.findOne(id);
                if (statistics == null) {
```

```
                        statistics = new Member();
                }
                return statistics;
        }
```

3.3. JSON

MediaType.APPLICATION_JSON_VALUE 执行结果反馈json数据

```
@RestController
@RequestMapping("/api/persons")
public class MainController {

    @RequestMapping(
            value = "/detail/{id}",
            method = RequestMethod.GET,
            produces = MediaType.APPLICATION_JSON_VALUE
            )
    public ResponseEntity<Persons>
getUserDetail(@PathVariable Long id) {

        Persons user = personsRepository.findById(id);

        return new ResponseEntity<>(user, HttpStatus.OK);
    }

}
```

3.4. 处理原始 RAW JSON 数据

```
        @RequestMapping(value = "/create", method =
RequestMethod.POST, produces = { "application/xml",
"application/json" })
```

```
        public ResponseEntity<Member> create(@RequestBody
Member member) {
                memberRepository.save(member);
                return new ResponseEntity<Member>(member,
HttpStatus.OK);
        }
```

3.5. 返回 JSON 对象 NULL 专为 "" 字符串

```
package api.config;

import java.io.IOException;

import
org.springframework.boot.autoconfigure.condition.ConditionalO
nMissingBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import
org.springframework.http.converter.json.Jackson2ObjectMapperB
uilder;

import com.fasterxml.jackson.core.JsonGenerator;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonSerializer;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializerProvider;

@Configuration
public class JacksonConfig {
    @Bean
    @Primary
    @ConditionalOnMissingBean(ObjectMapper.class)
    public ObjectMapper
jacksonObjectMapper(Jackson2ObjectMapperBuilder builder) {
        ObjectMapper objectMapper =
builder.createXmlMapper(false).build();
```

```
objectMapper.getSerializerProvider().setNullValueSerializer(n
ew JsonSerializer<Object>() {
            @Override
            public void serialize(Object o, JsonGenerator
jsonGenerator, SerializerProvider serializerProvider) throws
IOException, JsonProcessingException {
                jsonGenerator.writeString("");
            }
        });
        return objectMapper;
    }
}
```

3.6. XML

restful 将同时支持 json 和 xml 数据传递

```
package com.example.api.restful;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.PageRequest;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;

import com.example.api.domain.RecentRead;
import com.example.api.repository.RecentReadRepostitory;

@RestController
@RequestMapping("/restful/article")
```

```
public class ArticleRestController {

        @Autowired
        private RecentReadRepostitory recentReadRepostitory;

        @RequestMapping(value =
"/recent/read/add/{memberId}/{articleId}", method =
RequestMethod.GET, produces = { "application/xml",
"application/json" })
        public ResponseEntity<RecentRead>
recentAdd(@PathVariable long memberId, @PathVariable long
articleId) {
                RecentRead recentRead = new RecentRead();
                recentRead.setMemberId(memberId);
                recentRead.setArticleId(articleId);
                recentReadRepostitory.save(recentRead);
                return new ResponseEntity<RecentRead>
(recentRead, HttpStatus.OK);
        }

        @RequestMapping(value="/recent/read/list/{id}",
produces = { "application/xml", "application/json" })
        public List<RecentRead> recentList(@PathVariable long
id) {
                int page = 0;
                int limit = 20;
                List<RecentRead> recentRead =
recentReadRepostitory.findByMemberId(id, new
PageRequest(page, limit));
                return recentRead;
        }
}
```

3.7. 兼容传统 json 接口

开发中发现很多人不适应新的接口方式，有时候只能妥协，这些顽固不化的人需要这样的数据库格式

```
```

```
{
   "status":true,
   "reason":"登录成功",
   "code":1,
   "data":{
      "id":2,
      "name":null,
      "sex":null,
      "age":0,
      "wechat":null,
      "mobile":"13113668890",
      "picture":null,
      "ipAddress":"0:0:0:0:0:0:0:1"
   }
}
```

返回数据必须放在 data 字典中, 而我通常是采用 http status code 来返回状态，返回结果是对象。实现上面的需求我们需要加入一个data成员变量，因为我们不清楚最终要返回什么对象。所以声明为java.lang.Object

```
package com.example.api.pojo;

import java.io.Serializable;

public class RestfulResponse implements Serializable {
        /**
         *
         */
        private static final long serialVersionUID =
-4045645995352698349L;

        private boolean status;
        private String reason;
        private int code;
        private Object data;

        public RestfulResponse(boolean status, int code,
String reason, Object data) {
```

```
                this.status = status;
                this.code = code;
                this.reason = reason;
                this.data = data;

        }

        public boolean isStatus() {
                return status;
        }

        public void setStatus(boolean status) {
                this.status = status;
        }

        public String getReason() {
                return reason;
        }

        public void setReason(String reason) {
                this.reason = reason;
        }

        public int getCode() {
                return code;
        }

        public void setCode(int code) {
                this.code = code;
        }

        public Object getData() {
                return data;
        }

        public void setData(Object data) {
                this.data = data;
        }

        @Override
        public String toString() {
                return "RestfulResponse [status=" + status +
", reason=" + reason + ", code=" + code + ", data=" + data +
"]";
        }
```

```
}
```

Service

```
        public RestfulResponse bindWechat(String mobile,
String wechat) {
                Member member =
memberRepository.findByMobile(mobile);
                member.setWechat(wechat);
                memberRepository.save(member);
                return new RestfulResponse(true, 1, "微信绑定成
功", member);
        }
```

Controller

```
        @RequestMapping("/login/sms/{mobile}/{code}")
        public RestfulResponse sms(@PathVariable String
mobile, @PathVariable String wechat) {
                return memberService.bindWechat(mobile,
wechat);
        }
```

3.8. 上传文件

```
spring.servlet.multipart.max-file-size=128KB
spring.servlet.multipart.max-request-size=128KB
spring.http.multipart.enabled=false
```

RestController

```
package api.restful;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;

import org.springframework.web.bind.annotation.PostMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;
import
org.springframework.web.servlet.mvc.support.RedirectAttribute
s;

import api.pojo.RestfulResponse;

@RestController
@RequestMapping("/upload")
public class UploadRestController {

        private final static String FOLDER = "/tmp";

        public UploadRestController() {

        }

        @PostMapping("/single")
        public RestfulResponse upload(@RequestParam("file")
MultipartFile file, RedirectAttributes redirectAttributes) {
```

```
                if (file.isEmpty()) {
                        return new RestfulResponse(false, 0,
"Please select a file to upload", "");
                }
                try {
                        byte[] bytes = file.getBytes();
                        Path path = Paths.get(FOLDER + "/" +
file.getOriginalFilename());
                        Files.write(path, bytes);

                        return new RestfulResponse(true, 0,
"", path.toString());
                } catch (Exception e) {
                        return new RestfulResponse(false, 0,
e.getMessage(), null);
                }
        }

        @PostMapping(value = "/group")
        public RestfulResponse group(@RequestParam("files")
MultipartFile[] files) {
                List<String> filelist = new ArrayList<String>
();
                try {

                        for (MultipartFile file : files) {
                                File tmpfile = new
File(FOLDER + "/" + file.getOriginalFilename());
                                file.transferTo(tmpfile);

filelist.add(tmpfile.getPath());
                        }
                        return new RestfulResponse(true, 0,
null, filelist);
                } catch (Exception e) {
                        return new RestfulResponse(false, 0,
e.getMessage(), null);
                }
        }
}
```

由于上传文件名可能存在空格等特殊字符，这里使用UUID替代文件名

```
        @PostMapping(value = "/file")
        public RestfulResponse file(@RequestParam("file")
MultipartFile[] files) {
                List<Object> filelist = new ArrayList<Object>
();
                try {

                        for (MultipartFile file : files) {

                                UUID uuid =
UUID.randomUUID();
                                String filename =
String.format("%s/%s.%s", folder, uuid.toString(),
this.getExtensionName(filename));

                                File tmpfile = new
File(filename);
                                String filepath =
tmpfile.getPath();
                                System.out.println(filepath);
                                file.transferTo(tmpfile);

filelist.add(tmpfile.toString());
                        }
                        return new RestfulResponse(true, 0,
null, filelist);
                } catch (Exception e) {
                        return new RestfulResponse(false, 0,
e.getMessage(), null);
                }
        }

        private String getExtensionName(String filename) {
                if ((filename != null) && (filename.length()
> 0)) {
                        int dot = filename.lastIndexOf('.');
                        if ((dot > -1) && (dot <
(filename.length() - 1))) {
                                return filename.substring(dot
```

```
+ 1);
                        }
                }
                return filename;
        }
```

获取文件名及后缀信息

```
MultipartFile file = new MultipartFile();
String file = file.getOriginalFilename()
```

获取文件名

```
MultipartFile file = new MultipartFile();
String fileName =
file.getOriginalFilename().substring(0,file.getOriginalFilena
me().lastIndexOf("."))
```

获取文件后缀

```
MultipartFile file = new MultipartFile();
String fileSuffix =
file.getOriginalFilename().substring(file.getOriginalFilename
().lastIndexOf("."))
```

获取文件类型

```
MultipartFile file = new MultipartFile();
String fileType = file.getContentType()
```

获取文件大小

```
MultipartFile file = new MultipartFile();
String fileSize = file.getSize()
```

3.9. Spring boot with csv

下面是一个导出 CSV 文件的例子

```
        @GetMapping("/export")
        public void export(HttpServletResponse response)
throws IOException {
                response.setContentType("application/csv");
                //
response.setContentType("application/csv;charset=gb18030");
                response.setHeader("Content-Disposition",
"attachment; filename=\"file.csv\"");

                BufferedWriter writer = new
BufferedWriter(response.getWriter());

                // 需要写入 utf8bom 头否则会出现中文乱码
                // byte[] uft8bom = { (byte) 0xef, (byte)
0xbb, (byte) 0xbf };
                String bom = new String(new byte[] { (byte)
0xEF, (byte) 0xBB, (byte) 0xBF });
                writer.write(bom);
                writer.write("A,B,C");
                writer.newLine();
                tableRepository.findAll().forEach(table -> {
                        try {
                                String tmp =
String.format("%s,%s,%s", table.getId(), table.getMethod(),
table.getMoney());
                                writer.write(tmp);
                                writer.newLine();
                        } catch (IOException e) {
                                // TODO Auto-generated catch
block
                                e.printStackTrace();
                        }

                });
```

```
                writer.flush();
                writer.close();
        }
```

3.10. Problem Details [RFC 7807]

HTTP RFC 7807 规范：https://www.rfc-editor.org/rfc/rfc7807。这个规范里定义了HTTP API的“问题细节”（Problem Details）内容。用它来携带HTTP错误返回信息，避免自定义新的错误返回格式。

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
        "status": 403,
        "type": "https://bankname.com/common-problems/low-
balance",
        "title": "You not have enough balance",
        "detail": "Your current balance is 30 and you are
transterring 50",
        "instance": "/account-transfer-service"
}
```

```
type:           问题的类型；
title:          问题类型描述；
status:         HTTP状态码；
detail:         问题实例描述；
instance:       URI的内容应该用来描述问题实例，但不是必须的。
```

```
    @GetMapping("/ProblemDetail/v1/{id}")
    public ResponseEntity config(@PathVariable("id") Long id)
{
        if (id < 100) {
            return ResponseEntity.ok(new Member(id,
"netkiller"));
        } else {
            ProblemDetail pd =
ProblemDetail.forStatusAndDetail(HttpStatus.NOT_FOUND,
"Member id '" + id + "' does no exist");

pd.setType(URI.create("https://www.netkiller.cn/errors/not-
found"));
            pd.setTitle("Record Not Found");
            pd.setProperty("hostname", "www.netkiller.cn");
            return ResponseEntity.status(404).body(pd);
        }
    }
```

```
    @GetMapping(path = "/ProblemDetail/v2/{id}")
    public ResponseEntity
getEmployeeById_V3(@PathVariable("id") Long id) {

        try {
            //somthing threw this exception
            throw new NullPointerException("Something was
expected but it was null");
        } catch (NullPointerException npe) {
            ProblemDetail pd = ProblemDetail

.forStatusAndDetail(HttpStatus.INTERNAL_SERVER_ERROR,
                            "Null Pointer Exception");

pd.setType(URI.create("https://www.netkiller.cn/errors/npe"))
;
            pd.setTitle("Null Pointer Exception");
            pd.setProperty("hostname", "www.netkiller.cn");
```

```
            throw new
ErrorResponseException(HttpStatus.NOT_FOUND, pd, npe);
        }
    }
```

ResponseEntity

```
@PostMapping(path = "/foo", params = {"id", "name=John"})
public ResponseEntity<String> handlePostRequest() {
    // 处理请求
    return ResponseEntity.ok("Success");
}
```

status

```
return ResponseEntity
                  .status(HttpStatus.CREATED)
                  .header("Location", locationUri)
                  .body("Employee created successfully.
Location: " + locationUri);
```

3.11. Jackson

Jackson 相关配置

```
#序列化时间格式
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
```

```
spring.mvc.date-format=yyyy-MM-dd HH:mm:ss
#mvc序列化时候时区选择
spring.jackson.time-zone=GMT+8
```

@JsonIgnore 返回json是不含有该字段

```
        @JsonIgnore
        private String entityName =
this.getClass().getSimpleName();
```

@JsonFormat 格式化 json 时间格式

日期格式化

默认 json 中的时间格式是这样的

```
"createDate":"2018-09-11T07:34:20.106+0000","updateDate":"2018-
09-11T07:34:20.106+0000"
```

@JsonFormat 可以格式化 json 返回的时间格式。

```
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
```

格式化后

```
"createDate":"2018-09-11 07:42:44","updateDate":"2018-09-11
07:42:44"
```

解决时区问题，MongoDb 默认使用UTC,显示时间相差8小时

```
@JsonFormat(timezone = "GMT+8", pattern = "yyyy-MM-dd
HH:mm:ss")
private Date createdDate = new Date();
```

时区

```
public class Test {
   @JsonFormat(shape=JsonFormat.Shape.STRING, pattern="yyyy-
MMM-dd HH:mm:ss z", timezone="EST")
   @JsonProperty("pubDate")
   private Date recentBookPubDate;
}
```

枚举

```
public class Test {
   @JsonFormat(shape=JsonFormat.Shape.NUMBER)
   @JsonProperty("birthDate")
   private Date birthDate;
}
```

```
{
  "birthDate" : 1528702883858
}
```

枚举

```
package cn.netkiller;
import com.fasterxml.jackson.annotation.JsonFormat;

@JsonFormat(shape=JsonFormat.Shape.NUMBER)
enum Code {
    BLOCKING,
    CRITICAL,
    MEDIUM,
    LOW;
}

@JsonFormat(shape=JsonFormat.Shape.STRING)
enum Lang {
        Java,
        PHP,
        Python
}
```

@JsonComponent

```
package cn.netkiller.json;
```

```
public class Member {
	private String name;

	public Member() {
		// TODO Auto-generated constructor stub
	}

	public Member(String name) {
		// TODO Auto-generated constructor stub
		this.name = name;

	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	@Override
	public String toString() {
		return "Member [name=" + name + "]";
	}

}
```

```
package cn.netkiller.json;

import java.io.IOException;

import org.springframework.boot.jackson.JsonComponent;

import com.fasterxml.jackson.core.JsonGenerator;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.TreeNode;
import com.fasterxml.jackson.databind.DeserializationContext;
```

```
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonSerializer;
import com.fasterxml.jackson.databind.SerializerProvider;
import com.fasterxml.jackson.databind.node.TextNode;

@JsonComponent
public class Json {

	public Json() {
		// TODO Auto-generated constructor stub
	}

	public static class MemberJsonSerializer extends
JsonSerializer<Member> {

		@Override
		public void serialize(Member value,
JsonGenerator gen, SerializerProvider serializers) throws
IOException {
			// TODO Auto-generated method stub
			gen.writeStartObject();
			gen.writeStringField("member",
value.toString());
			gen.writeEndObject();

		}
	}

	public static class MemberJsonDeserializer extends
JsonDeserializer<Member> {

		@Override
		public Member deserialize(JsonParser p,
DeserializationContext ctxt) throws IOException,
JsonProcessingException {
			// TODO Auto-generated method stub
			TreeNode treeNode =
p.getCodec().readTree(p);
			TextNode member = (TextNode)
treeNode.get("member");
			return new Member(member.asText());
		}
	}
}
```

```
package cn.netkiller.json.controller;

import cn.netkiller.json.Member;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 *
 * @author neo
 */
@RestController
public class SimpleController {

        @Autowired
        public ObjectMapper objectMapper;

        @GetMapping("/")
        public String home() throws JsonMappingException,
JsonProcessingException {
                String json = "{\"name\":\"netkiller\"}";
                Member member = objectMapper.readValue(json,
Member.class);
                System.out.println(member.getName());
                return member.getName();
        }

}
```

Object to Json

```
ObjectMapper mapper = new ObjectMapper();
User user = new User();

//Object to JSON in file
mapper.writeValue(new File("c:\\user.json"), user);

//Object to JSON in String
String jsonInString = mapper.writeValueAsString(user);

//Convert object to JSON string and pretty print
String jsonInString =
mapper.writerWithDefaultPrettyPrinter().writeValueAsString(us
er);
```

```
        Notification notification = new Notification(status,
time, summary + info);
        ObjectMapper objectMapper = new ObjectMapper();
        String json =
objectMapper.writeValueAsString(notification);
```

Json To Object

```
ObjectMapper mapper = new ObjectMapper();
String jsonInString = "{'name' : 'mkyong'}";

//JSON from file to Object
User user = mapper.readValue(new File("c:\\user.json"),
User.class);

//JSON from String to Object
User user = mapper.readValue(jsonInString, User.class);
```

3.12. synchronized

避免接口无序执行，被同时多次执行，同一时间只能有一个请求，请求完毕之后才能进行下一次请求。

```
   @GetMapping("/lock/{id}")
    public String lock1(@PathVariable("id") String id) throws
InterruptedException {
        synchronized (id.intern()) {
            log.info(Thread.currentThread().getName() + " 上
锁");
            Thread.sleep(10000);
            log.info(Thread.currentThread().getName() + " 解
锁");
        }
        return Thread.currentThread().getName();
    }
```

使用 ConcurrentHashMap 数据共享

```
        private final Map<String, Object> share = new
ConcurrentHashMap<>();

    @GetMapping("/share/{id}")
    public Map<String, Object> shareTest(@PathVariable("id")
String id) throws InterruptedException {

//        share.computeIfAbsent(id, k -> new Object());
        share.computeIfAbsent(id, key -> {
            return new Date();
        });
```

```
        synchronized (share) {
            log.info(Thread.currentThread().getName() + " 上
锁");
            Thread.sleep(1000);
            log.info(Thread.currentThread().getName() + " 解
锁");
        }
        return share;
    }
```

3.13. SSE

```
@GetMapping("/mvc/sse")
public SseEmitter streamSseMvc() {
    SseEmitter emitter = new SseEmitter();
    ExecutorService sseMvcExecutor =
Executors.newSingleThreadExecutor();
    sseMvcExecutor.execute(() -> {
        try {
            for (int i = 0; true; i++) {
                SseEventBuilder event = SseEmitter.event()
                  .data("SSE MVC - " +
LocalTime.now().toString())
                  .id(String.valueOf(i))
                  .name("sse event - mvc");
                emitter.send(event);
                Thread.sleep(1000);
            }
        } catch (Exception ex) {
            emitter.completeWithError(ex);
        }
    });
    return emitter;
}
```

4. View

4.1. 配置静态文件目录

```
#静态资源访问路径
spring.mvc.static-path-pattern=/**

#静态资源映射路径
spring.resources.static-locations=classpath:/
```

4.2. 添加静态文件目录

```
package cn.netkiller.demo.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.ResourceHan
dlerRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;

@Configuration
public class MyWebMvcConfigurer implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry
registry) {

registry.addResourceHandler("/images/**").addResourceLocation
s("classpath:/images/");
    }
}
```

4.3. Using Spring’s form tag library

css

```

```

cssClass

cssClass 使用该属性指定表单元素CSS样式名，相当于HTML元素的class属性

```
<form:input path="userName" cssClass="inputStyle"/>
```

cssStyle

cssStyle 直接通过该属性指定样式，相当于HTML元素的style属性

```
<form:input path="userName" cssStyle="width:100px"/>
```

cssErrorClass

cssError Class表示表单元素发生错误时对应的样式

```

```

```
<form:input path="userName" cssClass="userNameClass"
cssErrorClass= "userNameClassError"/>
```

cssClass

```
```

4.4. Thymeleaf

http://thymeleaf.org/

Maven pom.xml

```
                        <dependency>

<groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-
thymeleaf</artifactId>
                </dependency>
```

Spring 配置

```
        <!--
************************************************************
*** -->
        <!-- THYMELEAF-SPECIFIC ARTIFACTS -->
```

```
        <!-- TemplateResolver <- TemplateEngine <-
ViewResolver -->
        <!--
**************************************************************
*** -->

        <bean id="templateResolver"

class="org.thymeleaf.templateresolver.ServletContextTemplateR
esolver">
                <property name="prefix" value="/WEB-
INF/templates/" />
                <property name="suffix" value=".html" />
                <property name="templateMode" value="HTML5"
/>
        </bean>

        <bean id="templateEngine"
class="org.thymeleaf.spring4.SpringTemplateEngine">
                <property name="templateResolver"
ref="templateResolver" />
        </bean>

        <bean
class="org.thymeleaf.spring4.view.ThymeleafViewResolver">
                <property name="templateEngine"
ref="templateEngine" />
        </bean>
```

controller

```
package cn.netkiller.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
```

```
@Controller
@RequestMapping("/")
public class HelloController {

	@RequestMapping(value = "/{name}", method =
RequestMethod.GET)
	public String getMovie(@PathVariable String name,
ModelMap model) {
		model.addAttribute("name", name);
		return "hello";
	}

}
```

HTML5 Template

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Spring MVC + Thymeleaf Example</title>
</head>
<body>
	Hello, <span th:text="${name}" />!
</body>
</html>
```

thymeleaf 渲染表格

```
	@RequestMapping("/list")
	public ModelAndView list() {
```

```
                Iterable<User> users =
userRepository.findAll();

                ModelAndView mv = new ModelAndView();
                mv.addObject("users", users);
                mv.setViewName("table");
                return mv;
        }
```

模板文件

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<title>用户登记</title>
</head>
<body>
        <h1>Welcome to Thymeleaf</h1>
        <table border="1" width="100%">
                <tr>
                        <td>ID</td>
                        <td>姓名</td>
                        <td>联系方式</td>
                        <td>详细地址</td>
                        <td>图片</td>
                </tr>
                <tr th:each="user : ${users}">
                        <td th:text="${user.id}"></td>
                        <td th:text="${user.name}"></td>
                        <td th:text="${user.tel}"></td>
                        <td th:text="${user.address}"></td>
                        <td th:text="${user.picture}"></td>
                </tr>
        </table>
</body>
</html>
```

URL 链接

```
        <span th:text="${number+1}"></span> /
        <span th:text="${totalPages}"></span>

        <a href="#"
                th:href="@{/api/user/browse?
sort=id,desc&size=10(page=${number-1})}">上一页</a>
        <a href="#"
                th:href="@{/api/user/browse?
sort=id,desc&size=10(page=${number+1})}">下一页</a>
```

拼接 URL 的方法

```
<img src="#" th:src="${'https://img.netkiller.cn/' + pic}"
height="128" th:target="_blank" />
```

拆分字符串

pictures 是一个以逗号分割得字符串。我们需要拆分并逐条显示。

```
        <div th:unless="${picture == null}">
                <a th:each="pic :
${#strings.arraySplit(pictures, ',')}" href="#"
th:href="${pic}"> <img src="#" th:src="${pic}" height="64" />
</a>
        </div>
```

日期格式化

```
        <span th:text="${#dates.format(createDate, 'yyyy-MM-
dd HH:mm')}"></span
```

```
// java.util.Date 处理

${#dates.day(date)}
${#dates.month(date)}
${#dates.monthName(date)}
${#dates.monthNameShort(date)}
${#dates.year(date)}
${#dates.dayOfWeek(date)}
${#dates.dayOfWeekName(date)}
${#dates.dayOfWeekNameShort(date)}
${#dates.hour(date)}
${#dates.minute(date)}
${#dates.second(date)}
${#dates.millisecond(date)}


// java.time 时间处理
${#temporals.day(date)}
${#temporals.month(date)}
${#temporals.monthName(date)}
${#temporals.monthNameShort(date)}
${#temporals.year(date)}
${#temporals.dayOfWeek(date)}
${#temporals.dayOfWeekName(date)}
${#temporals.dayOfWeekNameShort(date)}
${#temporals.hour(date)}
${#temporals.minute(date)}
${#temporals.second(date)}
${#temporals.millisecond(date)}

// 处理天实例
```

```
<p th:text="${#dates.day(standardDate)}"></p>
<p th:text="${#temporals.day(localDateTime)}"></p>
<p th:text="${#temporals.day(localDate)}"></p>

// 处理周实例

<p th:text="${#dates.dayOfWeekName(standardDate)}"></p>
<p th:text="${#temporals.dayOfWeekName(localDateTime)}"></p>
<p th:text="${#temporals.dayOfWeekName(localDate)}"></p>

// 处理秒实例

<p th:text="${#dates.second(standardDate)}"></p>
<p th:text="${#temporals.second(localDateTime)}"></p>
```

4.5. FreeMarker

http://freemarker.org/

5. Service

5.1. Application

@ComponentScan({ "web", "rest","service" }) 一定要包含 Service 目录。否则无法实现 @Autowired自动装配。你可以直接 @ComponentScan扫描所有目录。

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoCon
figuration;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.authentication.UserCredentials;
import org.springframework.data.mongodb.MongoDbFactory;
import org.springframework.data.mongodb.core.MongoTemplate;
import
org.springframework.data.mongodb.core.SimpleMongoDbFactory;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;

import com.mongodb.Mongo;

import pojo.ApplicationConfiguration;

@Configuration
```

```
@SpringBootApplication
@EnableConfigurationProperties(ApplicationConfiguration.class
)
@EnableAutoConfiguration(exclude = {
DataSourceAutoConfiguration.class })
@ComponentScan({ "web", "rest","service" })
@EnableMongoRepositories
public class Application {

        @SuppressWarnings("deprecation")
        public @Bean MongoDbFactory mongoDbFactory() throws
Exception {
                UserCredentials userCredentials = new
UserCredentials("finance", "your_password");
                return new SimpleMongoDbFactory(new
Mongo("mdb.netkiller.cn"), "finance", userCredentials);
        }

        public @Bean MongoTemplate mongoTemplate() throws
Exception {
                return new MongoTemplate(mongoDbFactory());
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

5.2. 定义接口

TestService 接口

```


package service;

public interface TestService {

        public String getName();
```

```
        public String toString();
        public String helloUser(String user);
}
```

5.3. 实现接口

实现 TestService 接口

```
package service.impl;

import org.springframework.stereotype.Component;

import service.TestService;

@Component
public class TestServiceImpl implements TestService {

        public String name = "Test";

        public void TestService() {

        }

        @Override
        public String helloUser(String user) {
                return "hello " + user;
        }

        public String getName() {
                return this.name;
        }

        @Override
        public String toString() {
                return "TestServiceImpl [config=" + this.name
+ "]";
        }
```

```
}
```

5.4. 调用 Service

控制器中调用 Service

```
package web;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import domain.City;
import pojo.ApplicationConfiguration;
import repository.CityRepository;
import service.TestService;

@Controller
public class IndexController {

        @Autowired
    private TestService testService;

        @RequestMapping("/service")
        @ResponseBody
        public String service() {
                return testService.helloUser("Neo");
        }

}
```

5.5. context.getBean 调用 Service

```
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        ConfigurableApplicationContext context =
SpringApplication.run(DemoApplication.class, args);
        TestService bean =
context.getBean(TestService.class);
        bean.test1();
        bean.test2("xsx");
        bean.test3("xsx`", 1);
        bean.test4("xsx2", 1, 2, 3, 4);
    }
}
```

5.6. AopContext

```
@Servcie
public class UserService {

   public void save(User user) {
         ((UserService)AopContext.currentProxy()).save(user);
   }

   @Transactional(rollbackFor=Exception.class)
   public void save(User user) {
                ...
                ...
    }
 }
```

5.7. 变量共享

Service 的变量是共享的，这是与 new Object 的区别。

```
package cn.netkiller.service;

import cn.netkiller.domain.Chat;
import cn.netkiller.repository.ChatRepository;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Service;

@Service
@Slf4j
public class ChatService {

    private String test;

    @Async
    public void test1() {
        this.test = "Test 1";
    }

    @Async
    public void test2() {
        this.test = "Test 2";
    }

    @Async
    public void test() {
        log.info(this.test);
    }
}
```

```
    @Autowired
    private ChatService chatService;
```

```
    @GetMapping("test")
    private Mono<String> test() {
        chatService.test();
        return Mono.just("OK");
    }

    @GetMapping("/test1")
    public Mono<String> test1() {
        String test = "测试";
        chatService.test1();
        return Mono.just(test);
    }

    @GetMapping("/test2")
    public Mono<String> test2() {
        chatService.test2();
        return Mono.just("OK");
    }
```

```
2024-01-01T14:09:10.022+08:00  INFO 59782 --- [watch-
development] [         task-1]
cn.netkiller.service.ChatService           : null
2024-01-01T14:09:24.694+08:00  INFO 59782 --- [watch-
development] [         task-3]
cn.netkiller.service.ChatService           : Test 1
2024-01-01T14:10:04.394+08:00  INFO 59782 --- [watch-
development] [         task-8]
cn.netkiller.service.ChatService           : Test 2
```

6. i18n 国际化

6.1. 在 appliction.properties 中配置启用 i18n

```
spring.messages.basename=message
spring.messages.encoding=UTF-8
```

6.2. 创建语言包文件

创建默认语言包文件 message.properties，当匹配不到语言时使用默认配置

```
member.name=Name
```

message_en_US.properties

```
member.name=Name
```

message_zh_CN.properties

```
member.name=姓名
```

注意：Eclipse 需要安装 properties 编辑工具，否则中文会自动转换成UTF8编码，无法直接阅读。

6.3. 控制器重引用语言包

RestController

```
@RestController
public class HomeController {
        @Autowired
        private MessageSource messageSource;

        @GetMapping("/lang")
        public String language() {
                String message =
messageSource.getMessage("member.name", null,
LocaleContextHolder.getLocale());
                return message;
        }
}
```

Controller

```
package cn.netkiller.controller;

import org.springframework.stereotype.Controller;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.context.MessageSource;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.ui.Model;
import java.util.Locale;
```

```
@Controller
public class HomeController {

    @Autowired
    private MessageSource messageSource;

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String index(Locale locale, Model model){

        // add parametrized message from controller
        String welcome =
messageSource.getMessage("welcome.message", new Object[]{"Neo
Chan"}, locale);
        model.addAttribute("message", welcome);

        // obtain locale from LocaleContextHolder
        Locale currentLocale =
LocaleContextHolder.getLocale();
        model.addAttribute("locale", currentLocale);
        model.addAttribute("startMeeting", "10:30");

        return "index";
    }

}
```

6.4. 参数传递

有时定义语言包会出现一种情况，一个句子中可能存在变量。例如:

恭喜你 XXXX 您已成为我们的会员

这样的需求，如果丁一两个key处理起来会非常麻烦。这里可以定义一个变量，通过参数传递来修改一句话中间的部分。

```
welcome=Welcom to {0}
```

```
        @GetMapping("/lang/args")
        public String welcome() {
                String[] args = { "China" };
                String message =
messageSource.getMessage("welcome", args,
LocaleContextHolder.getLocale());

                return message;
        }
```

参数以此类推 {0}, {1} ..... {n}

```
String welcome = messageSource.getMessage("welcome.message",
new Object[]{"Neo chen"}, locale);
```

7. 校验器(Validator)

常见的校验注解

```
@Null 被注释的元素必须为 null
@NotNull 被注释的元素必须不为 null
@AssertTrue 被注释的元素必须为 true
@AssertFalse 被注释的元素必须为 false
@Min(value) 被注释的元素必须是一个数字，其值必须大于等于指定的最小值
@Max(value) 被注释的元素必须是一个数字，其值必须小于等于指定的最大值
@DecimalMin(value) 被注释的元素必须是一个数字，其值必须大于等于指定的最小值
@DecimalMax(value) 被注释的元素必须是一个数字，其值必须小于等于指定的最大值
@Size(max=, min=) 被注释的元素的大小必须在指定的范围内
@Digits (integer, fraction) 被注释的元素必须是一个数字，其值必须在可接受的范围内
@Past 被注释的元素必须是一个过去的日期
@Future 被注释的元素必须是一个将来的日期
@Pattern(regex=,flag=) 被注释的元素必须符合指定的正则表达式

Hibernate Validator提供的校验注解：
@NotBlank(message =) 验证字符串非null，且长度必须大于0
@Email 被注释的元素必须是电子邮箱地址
@Length(min=,max=) 被注释的字符串的大小必须在指定的范围内
@NotEmpty 被注释的字符串的必须非空
@Range(min=,max=,message=) 被注释的元素必须在合适的范围内
```

7.1. 常规用法

定义校验器

```
package web.domain;
```

```
import javax.validation.constraints.Email;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

public class User {

	private Long id;

	@NotNull(message = "用户账号不能为空")
	@Size(min = 6, max = 11, message = "账号长度必须是6-11个
字符")
	private String username;

	@NotNull(message = "用户密码不能为空")
	@Size(min = 6, max = 8, message = "密码长度必须是6-8个字
符")
	private String password;

	@NotNull(message = "用户邮箱不能为空")
	@Email(message = "邮箱格式不正确")
	private String email;

	// 不允许为空，并且年龄的最小值为18
	@NotNull
	@Min(18)
	private Integer age;

	public User() {
		// TODO Auto-generated constructor stub
	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
```

```
                this.username = username;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }

        public Integer getAge() {
                return age;
        }

        public void setAge(Integer age) {
                this.age = age;
        }

        @Override
        public String toString() {
                return "User [id=" + id + ", username=" +
username + ", password=" + password + ", email=" + email + ",
age=" + age + "]";
        }

}
```

获取 **BindingResult** 结果

```
```

```
package web.restful;

import javax.validation.Valid;

import org.springframework.validation.BindingResult;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import web.domain.User;

@RestController
@RequestMapping("/restful")
public class TestRestController {

        @RequestMapping("/test")
        public String home() {
                return "OK";
        }

        @PostMapping("/validation")
        public String addUser(@RequestBody @Valid User user,
BindingResult bindingResult) {
                // 如果有参数校验失败，返回错误信息
                if (bindingResult.hasErrors()) {
                        System.out.println(user.toString());

System.out.println(bindingResult.getErrorCount());

System.out.println(bindingResult.getAllErrors());
                }

                for (ObjectError error :
bindingResult.getAllErrors()) {
                        return error.getDefaultMessage();
                }
                return user.toString();
        }

}
```

测试校验效果

```
neo@MacBook-Pro-Neo ~/workspace/Management % curl -H "Content-
Type: application/json" -d '{"id":100000,
"username":"netkiller", "password":"123456",
"email":"netkillermsn.com"}' curl
http://localhost:8080/restful/validation
邮箱格式不正确

neo@MacBook-Pro-Neo ~/workspace/Management % curl -H "Content-
Type: application/json" -d '{"id":100000,
"username":"netkiller", "password":"123456",
"email":"netkiller@msn.com"}' curl
http://localhost:8080/restful/validation
must not be null

neo@MacBook-Pro-Neo ~/workspace/Management % curl -H "Content-
Type: application/json" -d '{"id":100000,
"username":"netkiller", "password":"123456",
"email":"netkiller@msn.com", "age":20}' curl
http://localhost:8080/restful/validation
User [id=100000, username=netkiller, password=123456,
email=netkiller@msn.com, age=20]
```

7.2. 自定义注解

下面实现一个手机号码检查的注解。

```
@Retention ： 用来说明该注解类的生命周期。它有以下三个参数：
RetentionPolicy.SOURCE : 注解只保留在源文件中
RetentionPolicy.CLASS : 注解保留在class文件中，在加载到JVM虚拟机时丢
弃
RetentionPolicy.RUNTIME : 注解保留在程序运行期间，此时可以通过反射获得
```

定义在某个类上的所有注解。

@Target ： 用来说明该注解可以被声明在那些元素之前。
ElementType.TYPE：说明该注解只能被声明在一个类前。
ElementType.FIELD：说明该注解只能被声明在一个类的字段前。
ElementType.METHOD：说明该注解只能被声明在一个类的方法前。
ElementType.PARAMETER：说明该注解只能被声明在一个方法参数前。
ElementType.CONSTRUCTOR：说明该注解只能声明在一个类的构造方法前。
ElementType.LOCAL_VARIABLE：说明该注解只能声明在一个局部变量前。
ElementType.ANNOTATION_TYPE：说明该注解只能声明在一个注解类型前。
ElementType.PACKAGE：说明该注解只能声明在一个包名前。

@Constraint来限定自定义注解的方法

定义校验器注解接口

```
package cn.netkiller.web.annotation;

import java.lang.annotation.Documented;
import java.lang.annotation.Retention;
import java.lang.annotation.Target;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.ElementType;

import javax.validation.Constraint;
import javax.validation.Payload;

import cn.netkiller.web.annotation.impl.MobileValidator;

@Target({ ElementType.METHOD, ElementType.FIELD,
ElementType.ANNOTATION_TYPE, ElementType.CONSTRUCTOR,
ElementType.PARAMETER })
@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = MobileValidator.class)
@Documented
// 注解的实现类。
public @interface Mobile {
        // 校验错误的默认信息
        String message() default "手机号码格式不正确！";
```

```
        // 是否强制校验
        boolean isRequired() default true;

        Class<?>[] groups() default {};

        Class<? extends Payload>[] payload() default {};
}
```

实现接口

```
package cn.netkiller.web.annotation.impl;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

import org.springframework.util.StringUtils;

import cn.netkiller.web.annotation.Mobile;

public class MobileValidator implements
ConstraintValidator<Mobile, String> {

        public MobileValidator() {
                // TODO Auto-generated constructor stub
        }

        private boolean required = false;

        @Override
        public void initialize(Mobile constraintAnnotation) {
                required = constraintAnnotation.isRequired();
        }

        @Override
        public boolean isValid(String phone,
```

```
ConstraintValidatorContext constraintValidatorContext) {
                Pattern mobile_pattern =
Pattern.compile("1\\d{10}");
                // System.out.println(phone);
                // 是否为手机号的实现
                if (required) {
                        if (StringUtils.isEmpty(phone)) {
                                return false;
                        }
                        Matcher m =
mobile_pattern.matcher(phone);
                        return m.matches();

                } else {
                        return StringUtils.isEmpty(phone);
                }
        }
}
```

注解用法

```
package cn.netkiller.web.domain;

import java.util.Date;

import javax.validation.constraints.Email;
import javax.validation.constraints.Future;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import cn.netkiller.web.annotation.Mobile;

public class User {

        private Long id;

        @NotNull(message = "用户账号不能为空")
        @Size(min = 6, max = 11, message = "账号长度必须是6-11个
```

```
字符")
        private String username;

        @NotNull(message = "用户密码不能为空")
        @Size(min = 6, max = 8, message = "密码长度必须是6-8个字
符")
        private String password;

        @NotNull(message = "用户邮箱不能为空")
        @Email(message = "邮箱格式不正确")
        private String email;

        // 这里是新添加的注解奥
        @Mobile(message = "手机号码格式错误！！！")
        private String phone;

        // 不允许为空，并且年龄的最小值为18
        @NotNull
        @Min(18)
        private Integer age;

        @Future
        private Date createTime;

        public User() {
                // TODO Auto-generated constructor stub
        }

        public Long getId() {
                return id;
        }

        public void setId(Long id) {
                this.id = id;
        }

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public String getPassword() {
```

```
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }

        public Integer getAge() {
                return age;
        }

        public void setAge(Integer age) {
                this.age = age;
        }

        public String getPhone() {
                return phone;
        }

        public void setPhone(String phone) {
                this.phone = phone;
        }

        @Override
        public String toString() {
                return "User [id=" + id + ", username=" +
username + ", password=" + password + ", email=" + email + ",
phone=" + phone + ", age=" + age + "]";
        }

}
```

测试注解

```
neo@MacBook-Pro-Neo ~ % curl -H "Content-Type:
application/json" -d '{"id":100000, "username":"netkiller",
"password":"123456", "email":"netkiller@msn.com", "age":20,
"phone":"BB"}' curl http://localhost:8080/restful/validation
手机号码格式错误!!!

neo@MacBook-Pro-Neo ~ % curl -H "Content-Type:
application/json" -d '{"id":100000, "username":"netkiller",
"password":"123456", "email":"netkiller@msn.com", "age":20,
"phone":"2433"}' curl http://localhost:8080/restful/validation
手机号码格式错误!!!

neo@MacBook-Pro-Neo ~ % curl -H "Content-Type:
application/json" -d '{"id":100000, "username":"netkiller",
"password":"123456", "email":"netkiller@msn.com", "age":20,
"phone":"130"}' curl http://localhost:8080/restful/validation
手机号码格式错误!!! %


neo@MacBook-Pro-Neo ~ % curl -H "Content-Type:
application/json" -d '{"id":100000, "username":"netkiller",
"password":"123456", "email":"netkiller@msn.com", "age":20,
"phone":"13022223333"}' curl
http://localhost:8080/restful/validation
User [id=100000, username=netkiller, password=123456,
email=netkiller@msn.com, phone=13022223333, age=20]%
```

8. Interceptor

8.1. Session 拦截

WebMvcConfigurerAdapter

```
package mis.config;

import mis.interceptor.SpringMVCInterceptor;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.Interceptor
Registry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;

@Configuration
public class WebAppConfig extends WebMvcConfigurerAdapter {
    /**
     * 配置拦截器
     */
    @Override
    public void addInterceptors(InterceptorRegistry registry)
{
        registry.addInterceptor(new
SpringMVCInterceptor()).addPathPatterns("/**");
    }
}
```

HandlerInterceptor

```
package mis.interceptor;

import org.springframework.util.StringUtils;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SpringMVCInterceptor implements
HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object o) throws Exception {

        if(request.getServletPath().startsWith("/index") ||
request.getServletPath().startsWith("/login")) {
            return true;
        }

        String username =
(String)request.getSession().getAttribute("userName");
        if (StringUtils.isEmpty(username)){
            response.sendRedirect(request.getContextPath() +
"/index");
            return false;
        }
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest
httpServletRequest, HttpServletResponse httpServletResponse,
Object o, ModelAndView modelAndView) throws Exception {

    }

    @Override
    public void afterCompletion(HttpServletRequest
httpServletRequest, HttpServletResponse httpServletResponse,
Object o, Exception e) throws Exception {
```

```
    }
}
```

8.2. Token 拦截

```
package cn.netkiller.annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target({ElementType.METHOD, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface TokenPass {
    boolean required() default true;
}

package cn.netkiller.annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target({ElementType.METHOD, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface TokenVerification {
    boolean required() default true;
}
```

```
package cn.netkiller.component;
```

```
import cn.netkiller.annotation.TokenPass;
import cn.netkiller.annotation.TokenVerification;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;
import org.springframework.web.method.HandlerMethod;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import java.lang.reflect.Method;

@Slf4j
@Component
public class TokenHandlerInterceptor implements
HandlerInterceptor {

    // 返回 true 表示继续向下执行，返回 false 表示中断后续操作
    @Override
    public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler) throws
Exception {
        String token = request.getHeader("token");// 从 http
请求头中取出 token
        // 如果不是映射到方法直接通过
        if (!(handler instanceof HandlerMethod
handlerMethod)) {
            return true;
        }
        Method method = handlerMethod.getMethod();

        //检查方法是否有TokenPass注解，有则跳过认证，直接通过
        if (method.isAnnotationPresent(TokenPass.class)) {
            TokenPass tokenPass =
method.getAnnotation(TokenPass.class);
            if (tokenPass.required()) {
                return true;
            }
        }
        //检查 TokenVerification 需要用户权限的注解
        if
(method.isAnnotationPresent(TokenVerification.class)) {
            TokenVerification tokenVerification =
method.getAnnotation(TokenVerification.class);
            if (tokenVerification.required()) {
```

```
                // 执行认证
                if (token == null) {
                    throw new RuntimeException("无 token, 请重新登录");
                }

                // token 校验逻辑写在这里

            }
        }
        throw new RuntimeException("没有权限");
    }


    // 目标方法执行后, 该方法在控制器处理请求方法调用之后、解析视图之前执行, 可以通过此方法对请求域中的模型和视图做进一步修改
    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {
        log.debug("postHandle");

    }

    // 页面渲染后, 该方法在视图渲染结束后执行, 可以通过此方法实现资源清理、记录日志信息等工作
    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {
        log.debug("afterCompletion");

    }

}
```

```
package cn.netkiller.config;

import cn.netkiller.component.TokenHandlerInterceptor;
```

```
import jakarta.annotation.Resource;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.Interceptor
Registry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;


@Configuration
public class TokenWebMvcConfigurer implements
WebMvcConfigurer {
    @Resource
    private TokenHandlerInterceptor tokenHandlerInterceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry)
{
        //注册拦截器,并设置拦截的请求路径
        //addPathPatterns为拦截此请求路径的请求
        //excludePathPatterns为不拦截此路径的请求
        registry.addInterceptor(tokenHandlerInterceptor)
                .addPathPatterns("/mock/*")
                .excludePathPatterns("/device/*")
                .excludePathPatterns("/callback/*");
    }
}
```

```
package cn.netkiller.controller;

import
org.springframework.web.bind.annotation.ControllerAdvice;
import
org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.Map;
```

```
@ControllerAdvice
public class GloablExceptionHandler {
    @ResponseBody
    @ExceptionHandler(Exception.class)
    public Object handleException(Exception e) {
        String msg = e.getMessage();
        if (msg == null || msg.equals("")) {
            msg = "服务器出错";
        }

        return Map.of("message", msg, "status", 500);
    }
}
```

```
package cn.netkiller.controller;

import lombok.extern.slf4j.Slf4j;
import org.reactivestreams.Publisher;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import
org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

@RestController
@RequestMapping("/mock")
@Slf4j
public class HomeController {

        //无需 token 可以访问
    @TokenPass
    @GetMapping("/neo")
    public String neo() {
        return "https://www.netkiller.cn";
    }

    // 需要 Token 才能访问
```

```
    @TokenVerification
    @GetMapping("/netkiller")
    public String netkiller() {
        return "https://www.netkiller.cn";
    }
}
```

```
neo@MacBook-Pro-M2 ~> curl -X 'GET'
'http://localhost:8080/mock/neo'
https://www.netkiller.cn⏎

neo@MacBook-Pro-M2 ~> curl -X 'GET'
'http://localhost:8080/mock/netkiller'
{"message":"无 token，请重新登录","status":500}⏎

neo@MacBook-Pro-M2 ~> curl -X 'GET'
'http://localhost:8080/mock/flux' -H 'token: 8A8691CF-DC81-
4477-84D8-DC5CDDF98568'
https://www.netkiller.cn⏎
```

8.3. 拦截器获取PathVariable变量

```
    Map<String, String> pathVars = (Map<String, String>)
request.getAttribute(HandlerMapping.URI_TEMPLATE_VARIABLES_AT
TRIBUTE);
    log.info(pathVars.toString());
```

9. FAQ

9.1. o.s.web.servlet.PageNotFound

解决方法，加入下面代码到 dispatcher-servlet.xml 文件中

```
<mvc:annotation-driven />
```

dispatcher-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-
beans.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-
mvc.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-
context.xsd">

       <context:component-scan base-
package="cn.netkiller.controller" />
       <mvc:annotation-driven />
       <bean id="viewResolver"

class="org.springframework.web.servlet.view.UrlBasedViewResolve
r">
              <property name="viewClass"
```

```
value="org.springframework.web.servlet.view.JstlView" />
                <property name="prefix" value="/WEB-INF/jsp/"
/>
                <property name="suffix" value=".jsp" />
        </bean>

</beans>
```

9.2. HTTP Status 500 - Handler processing failed; nested exception is java.lang.NoClassDefFoundError: javax/servlet/jsp/jstl/core/Config

pom.xml 文件中加入依赖包

```
                <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>jstl</artifactId>
                        <version>1.2</version>
                </dependency>
```

9.3. 同时使用 Thymeleaf 与 JSP

Using both Thymeleaf and JSP

```
        <bean

class="org.springframework.web.servlet.view.InternalResourceV
iewResolver">
                <property name="viewClass"

value="org.springframework.web.servlet.view.JstlView" />
                <property name="prefix" value="/WEB-INF/jsp/"
```

```
/>
                <!-- <property name="suffix" value=".jsp" />
-->
                <property name="viewNames" value="*.jsp" />
        </bean>

        <bean id="templateResolver"

class="org.thymeleaf.templateresolver.ServletContextTemplateR
esolver">
                <property name="prefix" value="/WEB-
INF/templates/" />
                <!-- <property name="suffix" value=".html" />
-->
                <property name="templateMode" value="HTML5"
/>
        </bean>

        <bean id="templateEngine"
class="org.thymeleaf.spring4.SpringTemplateEngine">
                <property name="templateResolver"
ref="templateResolver" />
        </bean>

        <bean
class="org.thymeleaf.spring4.view.ThymeleafViewResolver">
                <property name="templateEngine"
ref="templateEngine" />
                <property name="viewNames" value="*.html" />
        </bean>
```

```
@RequestMapping("/thymeleaf")
public String thymeleafView(){
    return "thymeleaf.html";
}

@RequestMapping("/jsp")
public String jspView(){
    return "jstl.jsp";
}
```

```
                        <property name="viewNames"
value="*thymeleaf/*" />

@RequestMapping(value="/test")
public ModelAndView dboxPrint(Model model){
    ModelAndView modelAndView = new
ModelAndView("thymeleaf/test");
    return modelAndView;
}
```

9.4. 排除静态内容

方法一，排除静态内容如 images, css, js 等等

```
        <servlet>
        <servlet-name>springframework</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
        <servlet-mapping>
            <servlet-name>default</servlet-name>
            <url-pattern>/images/*</url-pattern>
            <url-pattern>*.css</url-pattern>
            <url-pattern>/js/*.js</url-pattern>
        </servlet-mapping>
    <servlet-mapping>
        <servlet-name>springframework</servlet-name>
        <url-pattern>/welcome.jsp</url-pattern>
        <url-pattern>/welcome.html</url-pattern>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>
```

```
```

方法二

```
        <mvc:resources location="/images/"
mapping="/images/**" />
        <mvc:resources location="/css/" mapping="/css/**" />
        <mvc:resources location="/js/" mapping="/js/**" />
```

9.5. HTTP Status 406

配置 url-pattern 增加需要传递给Spring的扩展名

```
    <servlet>
    <servlet-name>springframework</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>springframework</servlet-name>
    <url-pattern>/welcome.jsp</url-pattern>
    <url-pattern>/welcome.html</url-pattern>
    <url-pattern>*.json</url-pattern>
    <url-pattern>*.xml</url-pattern>
    <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

9.6. Caused by: java.lang.IllegalArgumentException: Not a managed type: class common.domain.Article

背景描述：Springboot 入口文件 Application.java 的包是 package api; 为了让 domain,pojo 共用，于是将 domain 放到Maven module下命令为 common。启动后出现这个故障。

解决方案增加 @EntityScan("common.domain") 即可。

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.domain.EntityScan;
import
org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import
org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication
@EnableScheduling
@EnableEurekaClient
@EntityScan("common.domain")
public class Application {

        public static void main(String[] args) {
                System.out.println( "Service Api Starting..."
);
                SpringApplication.run(Application.class,
args);
        }
}
```

9.7. {"error":"unauthorized","error_description":"Full authentication is required to access this resource"}

Oauth @RestController 一切正常， @Controller 提示如下

```
{"error":"unauthorized","error_description":"Full
authentication is required to access this resource"}
```

程序如下

```
package api.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/")
public class IndexController {

        public IndexController() {
                // TODO Auto-generated constructor stub
        }

        @GetMapping("/")
        public String index() {
                return "Helloworld!!!";

        }

        @GetMapping("/about")
        public String test() {
                return "Helloworld!!!";

        }
}
```

分析 @Controller 不允许直接返回字符串，必须使用 @ResponseBody 或者 ModelAndView，下改后问题解决。

```
package api.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("/web")
public class IndexController {

        public IndexController() {
                // TODO Auto-generated constructor stub
        }

        @GetMapping("/")
        @ResponseBody
        public String index() {
                return "Helloworld!!!";

        }

        @GetMapping("/about")
        @ResponseBody
        public String test() {
                return "Helloworld!!!";

        }
}
```

同时 @EnableWebSecurity 需要忽略 @Controller 的映射 URL

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfiguration extends
```

```
WebSecurityConfigurerAdapter {

        @Override
        public void configure(WebSecurity web) throws
Exception {

web.ignoring().antMatchers("/**/json").antMatchers("/about",
"/", "/css/**");
        }
}
```

10. RestClient

10.1. 创建 RestClient

Bean 注入方式

```
@Value("${REMOTE_BASE_URI:http://localhost:3000}")
String baseURI;

@Bean
RestClient restClient() {
  return RestClient.create(baseURI);
}
```

```
RestClient defaultClient = RestClient.create();

RestClient customClient = RestClient.builder()
  .requestFactory(new
HttpComponentsClientHttpRequestFactory())
  .messageConverters(converters -> converters.add(new
MyCustomMessageConverter()))
  .baseUrl("https://example.com")
  .defaultUriVariables(Map.of("variable", "foo"))
  .defaultHeader("My-Header", "Foo")
  .requestInterceptor(myCustomInterceptor)
  .requestInitializer(myCustomInitializer)
  .build();
```

10.2. Get 操作

```
restClient.get()
        .uri("/employees")
        //...

restClient.get()
        .uri("/employees/{id}", id)
        //...
```

```
List<Employee> employeeList = restClient.get()
    .uri("/employees")
    .accept(MediaType.APPLICATION_JSON)
    .retrieve()
    .body(List.class);

ResponseEntity<List> responseEntity = restClient.get()
    .uri("/employees")
    .accept(MediaType.APPLICATION_JSON)
    .retrieve()
    .toEntity(List.class);
```

10.3. Post Json

```
    @Cacheable(value = "translate", key = "#chinese", unless
= "#result == null")
    public String translate(String chinese) {
        String english = null;
        RestClient restClient =
RestClient.builder().baseUrl(url).build();
        String accessToken = this.getAccessToken();
        HashMap<String, String> data = new
LinkedHashMap<String, String>() {{
```

```
            put("q", chinese);
            put("from", "zh");
            put("to", "en");
        }};
        ResponseEntity<Translate> response =
restClient.post()
                .uri("/rpc/2.0/mt/texttrans/v1-?access_token=
{access_token}", Map.of("access_token", accessToken))
                .contentType(APPLICATION_JSON)
                .body(data)
                .retrieve()
                .toEntity(Translate.class);

        if (response.getStatusCode() == HttpStatus.OK) {
            Translate translate = response.getBody();
            if (translate.getResult() != null) {
                english =
translate.getResult().getTrans_result().get(0).get("dst");
            }
            log.info("Translate english: {}", english);
        } else {
            log.info("Translate: " + response);
        }
        return english;
    }
```

10.4. HTTP Authorization Basic

```
        @GetMapping("/{device}/test")
    public String get(@PathVariable String device) throws
InterruptedException {

        String username = System.getProperty("username",
"admin");
        String password = System.getProperty("password",
"uPQKFe98IwZCzgVGjbWIQRyRyyecb2Ha");

        Base64.Encoder encoder = Base64.getEncoder();
        String authorization =
```

```
encoder.encodeToString((username + ":" +
password).getBytes(StandardCharsets.UTF_8));
        RestClient restClient = RestClient.builder()
                .baseUrl("http://gpt.netkiller.cn:8080")
                .defaultHeader("Authorization", "Basic " +
authorization)
                .build();

        String question = "test";

        String result = restClient.get().uri(uriBuilder ->
uriBuilder
                .path("/ask/cache_chatgpt")
                .queryParam("question", question)
                .build()).retrieve().body(String.class);

        return result;
    }
```

10.5.

```
                ResponseEntity<JsonObject> responseEntity =
restClient.get()
                .uri(uriBuilder -> uriBuilder
                        .path("/articles/1.html")
                        .queryParam("question",
URLEncoder.encode(question, StandardCharsets.UTF_8))
                        .build())
                .retrieve()
                .onStatus(status -> status.value() == 404,
(request, response) -> {
                        throw new
ArticleNotFoundException(response)
                }).toEntity(JsonObject.class);
```

第 7 章 WebFlux framework

1. Getting Started

1.1. Maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.1.1.RELEASE</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>webflux</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>webflux</name>
        <description>Demo webflux project for Spring
Boot</description>

        <properties>
                <java.version>11</java.version>
        </properties>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
webflux</artifactId>
                </dependency>
```

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>io.projectreactor</groupId>
                        <artifactId>reactor-test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.restdocs</groupId>
                        <artifactId>spring-restdocs-
mockmvc</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```

1.2. Application

```
package cn.netkiller.webflux;
```

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class WebfluxApplication {

        public static void main(String[] args) {

SpringApplication.run(WebfluxApplication.class, args);
        }

}
```

1.3. RestController

```
package cn.netkiller.webflux;

import org.reactivestreams.Publisher;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import
org.springframework.web.bind.annotation.RestController;

import reactor.core.publisher.Mono;

@RestController
public class TestController {

        public TestController() {

        }

        @GetMapping("/")
        @ResponseBody
        public Publisher<String> index() {
                return Mono.just("Hello world!");
        }
```

```
}
```

1.4. 测试

```
neo@MacBook-Pro ~/webflux % mvn spring-boot:run
```

```
neo@MacBook-Pro ~ % curl http://localhost:8080
Hello world!%
```

2. WebFlux 与 SprintMVC 有什么不同?

2.1. 实验程序

```
package cn.netkiller.controller;

import java.util.concurrent.TimeUnit;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

import lombok.extern.slf4j.Slf4j;
import reactor.core.publisher.Mono;

@RestController
@Slf4j
public class WebFluxController {
        private static final Logger logger =
LoggerFactory.getLogger(WebFluxController.class);

        public WebFluxController() {
        }

        // 阻塞5秒钟
        private String job() {
                try {
                        TimeUnit.SECONDS.sleep(5);
                } catch (InterruptedException e) {
                }
                return "Hellowoard!!!";
        }

        // SpringMVC 方式
        @GetMapping("/SpringMVC")
        private String springmvc() {
                logger.info("start");
```

```
                String result = job();
                logger.info("done");
                return result;
        }

        // WebFlux 方式
        @GetMapping("/WebFlux")
        private Mono<String> webflux() {
                logger.info("start");
                Mono<String> result = Mono.fromSupplier(() ->
job());
                logger.info("done");
                return result;
        }
}
```

2.2. 实验结果

```
neo@MacBook-Pro-Neo ~> time curl
http://localhost:8080/SpringMVC
Hellowoard!!!
________________________________________________________
Executed in    5.02 secs      fish           external
   usr time    4.98 millis  242.00 micros    4.74 millis
   sys time    5.48 millis  993.00 micros    4.49 millis
```

```
2023-02-24T14:13:07.063+08:00 TRACE 1552 --- [  XNIO-1 task-2]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
cn.netkiller.controller.WebFluxController#springmvc()
2023-02-24T14:13:07.077+08:00  INFO 1552 --- [  XNIO-1 task-2]
c.n.controller.WebFluxController         : start
2023-02-24T14:13:12.082+08:00  INFO 1552 --- [  XNIO-1 task-2]
c.n.controller.WebFluxController         : done
```

从省输出日志可以看到 start 2023-02-24T14:13:07, done 2023-02-24T14:13:12 程序运行被阻塞了 5秒钟

```
neo@MacBook-Pro-Neo ~> time curl http://localhost:8080/WebFlux
Helloword!!!
________________________________________________________
Executed in    5.02 secs      fish           external
   usr time    5.19 millis  228.00 micros    4.96 millis
   sys time    6.05 millis  854.00 micros    5.20 millis
```

```
2023-02-24T14:14:54.720+08:00 TRACE 1583 --- [  XNIO-1 task-2]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
cn.netkiller.controller.WebFluxController#webflux()
2023-02-24T14:14:54.729+08:00  INFO 1583 --- [  XNIO-1 task-2]
c.n.controller.WebFluxController         : start
2023-02-24T14:14:54.731+08:00  INFO 1583 --- [  XNIO-1 task-2]
c.n.controller.WebFluxController         : done
2023-02-24T14:14:59.753+08:00 TRACE 1583 --- [  XNIO-1 task-3]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped to
cn.netkiller.controller.WebFluxController#webflux()
```

再看 webflux 的表现，start 2023-02-24T14:14:54, done 2023-02-24T14:14:54 执行时间不到一秒钟。

3. WebFlux Router

3.1. Component 原件

```
package cn.netkiller.webflux.component;

import org.springframework.http.MediaType;
import org.springframework.stereotype.Component;
import
org.springframework.web.reactive.function.BodyInserters;
import
org.springframework.web.reactive.function.server.ServerReques
t;
import
org.springframework.web.reactive.function.server.ServerRespon
se;

import reactor.core.publisher.Mono;

@Component
public class HelloWorldHandler {

        public HelloWorldHandler() {
        }

        public Mono<ServerResponse> helloWorld(ServerRequest
request) {
                return
ServerResponse.ok().contentType(MediaType.TEXT_PLAIN).body(Bo
dyInserters.fromObject("Hello World!!!"));
        }
}
```

3.2. 路由配置

```
package cn.netkiller.webflux.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.MediaType;
import
org.springframework.web.reactive.function.server.RequestPredi
cates;
import
org.springframework.web.reactive.function.server.RouterFuncti
on;
import
org.springframework.web.reactive.function.server.RouterFuncti
ons;
import
org.springframework.web.reactive.function.server.ServerRespon
se;

import cn.netkiller.webflux.component.HelloWorldHandler;

@Configuration
public class WebFluxRouter {

        public WebFluxRouter() {
        }

        @Bean
        public RouterFunction<ServerResponse>
routeHelloWorld(HelloWorldHandler helloWorldHandler) {

                return
RouterFunctions.route(RequestPredicates.GET("/hello").and(Req
uestPredicates.accept(MediaType.TEXT_PLAIN)),
helloWorldHandler::helloWorld);
        }
}
```

3.3. Thymeleaf

模板引擎 Thymeleaf 依赖

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

application.properties 相关的配置

```
spring.thymeleaf.cache=true # Enable template caching.
spring.thymeleaf.check-template=true # Check that the template
exists before rendering it.
spring.thymeleaf.check-template-location=true # Check that the
templates location exists.
spring.thymeleaf.enabled=true # Enable Thymeleaf view
resolution for Web frameworks.
spring.thymeleaf.encoding=UTF-8 # Template files encoding.
spring.thymeleaf.excluded-view-names= # Comma-separated list of
view names that should be excluded from resolution.
spring.thymeleaf.mode=HTML5 # Template mode to be applied to
templates. See also StandardTemplateModeHandlers.
spring.thymeleaf.prefix=classpath:/templates/ # Prefix that
gets prepended to view names when building a URL.
spring.thymeleaf.reactive.max-chunk-size= # Maximum size of
data buffers used for writing to the response, in bytes.
spring.thymeleaf.reactive.media-types= # Media types supported
by the view technology.
spring.thymeleaf.servlet.content-type=text/html # Content-Type
value written to HTTP responses.
spring.thymeleaf.suffix=.html # Suffix that gets appended to
view names when building a URL.
spring.thymeleaf.template-resolver-order= # Order of the
template resolver in the chain.
spring.thymeleaf.view-names= # Comma-separated list of view
names that can be resolved.
```

Webflux 控制器

```
        @GetMapping("/welcome")
    public Mono<String> hello(final Model model) {
        model.addAttribute("name", "Neo");
        model.addAttribute("city", "深圳");

        String path = "hello";
        return Mono.create(monoSink ->
monoSink.success(path));
    }

    @GetMapping("/list")
    public String listPage(final Model model) {
        final Flux<City> citys = cityService.findAllCity();
        model.addAttribute("cityLists", citys);
        return "cityList";
    }
```

Tymeleaf 视图

welcome.html

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8"/>
    <title>欢迎页面</title>
</head>

<body>

<h1 >你好，欢迎来自<p th:text="${city}"></p>的<p
th:text="${name}"></p></h1>
```

```
</body>
</html>
```

cityList.html

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8"/>
    <title>城市列表</title>
</head>

<body>

<div>


    <table>
        <legend>
            <strong>城市列表</strong>
        </legend>
        <thead>
        <tr>
            <th>城市编号</th>
            <th>省份编号</th>
            <th>名称</th>
            <th>描述</th>
        </tr>
        </thead>
        <tbody>
        <tr th:each="city : ${cityLists}">
            <td th:text="${city.id}"></td>
            <td th:text="${city.provinceId}"></td>
            <td th:text="${city.name}"></td>
            <td th:text="${city.description}"></td>
        </tr>
        </tbody>
    </table>
```

```
</div>

</body>
</html>
```

3.4. Webflux Redis

Maven Redis 依赖

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis-reactive</artifactId>
                </dependency>
```

Redis 配置

```
server:
  port: 8080
spring:
  application:
    name: webflux
  redis:
    host: 127.0.0.1
    port: 6379
    password: pwd2020
    timeout: 5000
    lettuce:
      pool:
        max-active: 200
        max-idle: 20
        min-idle: 5
```

```
        max-wait: 1000
```

Config

```
        @Bean
        public ReactiveRedisTemplate<String, String>
reactiveRedisTemplate(ReactiveRedisConnectionFactory factory)
{
                ReactiveRedisTemplate<String, String>
reactiveRedisTemplate = new ReactiveRedisTemplate<>
(factory,RedisSerializationContext.string());
                return reactiveRedisTemplate;
        }
```

Service

```
@Service
public class RedisServiceImpl implements RedisService {

        @Autowired
        private ReactiveRedisTemplate<String, String>
redisTemplate;

        @Override
        public Mono<String> get(String key) {

                ReactiveValueOperations<String, String>
operations = redisTemplate.opsForValue();
                return operations.get(key);
        }

        @Override
        public Mono<String> set(String key,User user) {
```

```
                ReactiveValueOperations<String, String>
operations = redisTemplate.opsForValue();
                return operations.getAndSet(key,
JSON.toJSONString(user));
        }

        @Override
        public Mono<Boolean> delete(String key) {

                ReactiveValueOperations<String, String>
operations = redisTemplate.opsForValue();
                return operations.delete(key);
        }

        @Override
        public Mono<String> update(String key,User user) {

                ReactiveValueOperations<String, String>
operations = redisTemplate.opsForValue();
                return operations.getAndSet(key,
JSON.toJSONString(user));
        }

        @Override
        public Flux<String> all(String key) {
                ReactiveListOperations<String, String>
operations = redisTemplate.opsForList();
                return operations.range(key, 0, -1);
        }


        @Override
        public Mono<Long> push(String key,List<String> list)
{

                ReactiveListOperations<String, String>
operations = redisTemplate.opsForList();
                return operations.leftPushAll(key, list);
        }

        @Override
        public Flux<String> find(String key) {
                ReactiveValueOperations<String, String>
operations = redisTemplate.opsForValue();
```

```
                return redisTemplate.keys(key).flatMap(keyId
->operations.get(keyId));
        }
}
```

```
@RestController
@RequestMapping("/user")
public class UserController {

        public final static String USER_KEY="user";

        @Autowired
        private RedisService redisService;

        @GetMapping("/get/{key}")
        public Mono<String>
getUserByKey(@PathVariable("id")String key){
                return redisService.get(key);
        }

        @GetMapping("/add")
        public Mono<String> add(User user){
                user = new User();
                user.setAccount("neo");
                user.setPassword("123456");
                user.setNickname("netkiller");
                user.setEmail("netkiller@msn.com");
                user.setPhone("");
                user.setGender(true);
                user.setBirthday("1980-01-30");
                user.setProvince("广东省");
                user.setCity("深圳市");
                user.setCounty("南山区");
                user.setAddress("");
                user.setState("Enabled");

                System.out.println(JSON.toJSONString(user));
                return redisService.set("neo",user);
        }
```

```
        @GetMapping("/addlist")
        public Mono<Long> addlist(){
                List<String> list=new ArrayList<String>();
                User user = new User();
                user.setAccount("neo");
                user.setPassword("123456");
                user.setNickname("netkiller");
                user.setEmail("netkiller@msn.com");
                user.setPhone("");
                user.setGender(true);
                user.setBirthday("1980-01-30");
                user.setProvince("广东省");
                user.setCity("深圳市");
                user.setCounty("南山区");
                user.setAddress("");
                user.setState("Enabled");

                //添加第一条数据
                list.add(JSON.toJSONString(user));
                //添加第二条数据
                list.add(JSON.toJSONString(user));
                //添加第三条数据
                list.add(JSON.toJSONString(user));

                return redisService.addlist("list", list);
        }

        @GetMapping(value="/findAll",produces =
MediaType.APPLICATION_STREAM_JSON_VALUE)
        public Flux<String> findAll(){
                return
redisService.all("list").delayElements(Duration.ofSeconds(2))
;
        }

        @GetMapping("/getUsers")
        public Flux<String> findUsers() {
                return
redisService.find("*").delayElements(Duration.ofSeconds(2));
        }
}
```

3.5. Webflux Mongdb

Maven 依赖

```
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-mongodb-
reactive</artifactId>
        </dependency>
```

Repository

```
import
org.springframework.data.mongodb.repository.ReactiveMongoRepo
sitory;

import cn.netkiller.entity.User;

public interface UserRepository extends
ReactiveMongoRepository<User, Long>{

}
```

Service

```
@Service
public class MongoServiceImpl implements MongoService {

        @Autowired
        private UserRepository userRepository;
```

```
        @Override
        public Mono<User> getById(Long id) {
                return userRepository.findById(id);
        }

        @Override
        public Mono<User> addUser(User user) {
                return userRepository.save(user);
        }

        @Override
        public Mono<Boolean> deleteById(Long id) {
                 userRepository.deleteById(id);
                 return Mono.create(userMonoSink ->
userMonoSink.success());
        }

        @Override
        public Mono<User> updateById(User user) {
                return userRepository.save(user);
        }

        @Override
        public Flux<User> findAllUser() {
                return userRepository.findAll();
        }
}
```

控制器

```
@RestController
@RequestMapping("/usermg")
public class UserMongoController {

        @Autowired
        private MongoService mongoService;

        @GetMapping("/add")
```

```
        public Mono<User> add(User user) {
                user = new User();
                User user = new User();
                user.setAccount("neo");
                user.setPassword("123456");
                user.setNickname("netkiller");
                user.setEmail("netkiller@msn.com");
                user.setPhone("");
                user.setGender(true);
                user.setBirthday("1980-01-30");
                user.setProvince("广东省");
                user.setCity("深圳市");
                user.setCounty("南山区");
                user.setAddress("");
                user.setState("Enabled");


                System.out.println(JSON.toJSONString(user));
                return mongoService.addUser(user);

        }

        /**
         *      注意这里 produces =
MediaType.APPLICATION_STREAM_JSON_VALUE 必须这样设置
         */
        @GetMapping(value="/findAll",produces =
MediaType.APPLICATION_STREAM_JSON_VALUE)
        public Flux<User> findAll(){
                return
mongoService.findAllUser().delayElements(Duration.ofSeconds(1
));
        }
}
```

produces 如果不是application/stream+json则调用端无法滚动得到结果，将一直阻塞等待数据流结束或超时。

3.6. Mono

Mono(返回0或1个元素)/Flux(返回0-n个元素)

```
    @GetMapping("mono")
    public Mono<Object> mono() {
        return Mono.create(monoSink -> {
                    log.info("创建 Mono");
                    monoSink.success("hello webflux");
                })
                .doOnSubscribe(subscription -> { //当订阅者去订
阅发布者的时候，该方法会调用
                    log.info("doOnSubscribe={}",
subscription);
                }).doOnNext(next -> { //当订阅者收到数据时，改方法
会调用
                    log.info("doOnNext={}", next);
                });
    }
```

从 Supplier 创建 Mono

```
    @GetMapping("/get")
    private Mono<String> get() {
        log.info("start");
        Mono<String> result = Mono.fromSupplier(() -> {
            try {
                TimeUnit.SECONDS.sleep(5);
            } catch (InterruptedException e) {
            }
            return "netkiller";
        });
        log.info("end");
        return result;
    }
```

3.7. Flux 返回多条数据

返回 List

```
    @GetMapping("flux")
    public Flux<Picture> flux() {
        List<Picture> list = new ArrayList<Picture>();
        IntStream.range(1, 10).forEach(i -> {
            Picture picture = new Picture();
            picture.setId(Long.valueOf(i));

picture.setImage("https://www.netkiller.cn/images/" + i +
".png");
            list.add(picture);
        });
        return Flux.fromIterable(list);
    }
```

返回 Map

```
    @GetMapping("map")
    public Flux<Map.Entry<String, String>> map() {
        Map<String, String> map = new HashMap<>();
        IntStream.range(1, 10).forEach(i -> {
            map.put("key" + i, "value" + i);
        });

        return Flux.fromIterable(map.entrySet());
    }
```

从 Stream 返回 Flux

```
    @GetMapping(path = "/sse", produces =
```

```
MediaType.TEXT_EVENT_STREAM_VALUE)
    private Flux<String> getWords() {
        Stream<String> items = Arrays.asList("alpha",
"bravo", "charlie").stream();
        return Flux.fromStream(items);
    }
```

3.8. SSE

一次性事件

```
    @GetMapping(path = "/sse", produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
    public Flux<String> createConnectionAndSendEvents() {
        return Flux.just("Alpha", "Omega");
    }
```

curl 访问 SSE 需要设置HTTP头 -H "Accept: text/event-stream"

```
neo@MacBook-Pro-M2 ~ % curl -H "Accept: text/event-stream" -X
'GET' 'http://localhost:8080/mock/sse'
data:Alpha

data:Omega
```

提示

Safari 浏览器不支持 SSE推送，微软的 Egde 支持。

周期性事件

每间隔一秒发送一次数据

```
    @GetMapping(path = "/sse", produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
    private Flux<String> getWords() {
        String[] WORDS = "The quick brown fox jumps over the
lazy dog.".split(" ");
        return Flux
                .zip(Flux.just(WORDS),
Flux.interval(Duration.ofSeconds(1)))
                .map(Tuple2::getT1);
    }

        @GetMapping("/random")
        public Flux<ServerSentEvent<Integer>> randomNumbers()
{
                return
Flux.interval(Duration.ofSeconds(1)).map(seq ->
Tuples.of(seq,
ThreadLocalRandom.current().nextInt())).map(data ->
ServerSentEvent.
<Integer>builder().event("random").id(Long.toString(data.getT
1())).data(data.getT2()).build());
        }

        @GetMapping(path = "/stream-flux", produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
        public Flux<String> streamFlux() {
            return Flux.interval(Duration.ofSeconds(1))
              .map(sequence -> "Flux - " +
LocalTime.now().toString());
        }
```

SSE 完整的例子

```
package cn.netkiller.webflux.controller;
```

```
import java.time.Duration;
import java.util.concurrent.ThreadLocalRandom;

import org.springframework.http.MediaType;
import org.springframework.http.codec.ServerSentEvent;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import reactor.core.publisher.Flux;
import reactor.util.function.Tuples;

@RestController
@RequestMapping("/sse")
public class SseController {
        private int count_down = 10;

        public SseController() {

        }

        @GetMapping(value = "/launch", produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
        public Flux<ServerSentEvent<Object>> countDown() {

                return
Flux.interval(Duration.ofSeconds(1)).map(seq ->
Tuples.of(seq, getCountDownSec())).map(data ->
ServerSentEvent.
<Object>builder().event("launch").id(Long.toString(data.getT1
()())).data(data.getT2().toString()).build());
        }

        private String getCountDownSec() {
                if (count_down > 0) {
                        count_down--;
                        return "倒计时: " + count_down;
                }
                return "发射";
        }
```

```
        @GetMapping("/range")
        public Flux<Object> range() {
                return Flux.range(10, 1).map(seq ->
Tuples.of(seq, getCountDownSec())).map(data ->
ServerSentEvent.
<Object>builder().event("launch").id(Long.toString(data.getT1
()())).data(data.getT2().toString()).build());
        }

        // WebFlux 服务器推送(SSE - >Server Send Event)
        @GetMapping(value = "/sse", produces =
MediaType.TEXT_EVENT_STREAM_VALUE)
        private Flux<String> flux() {
                Flux<String> result =
Flux.fromStream(IntStream.range(1, 10).mapToObj(i -> {
                        try {
                                TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                        }
                        logger.info("sse " + i);
                        return "flux data -- " + i;
                }));
                return result;
        }
}
```

运行结果

```
id:0
event:launch
data:倒计时：9

id:1
event:launch
data:倒计时：8

id:2
event:launch
```

```
data:倒计时：7

id:3
event:launch
data:倒计时：6

id:4
event:launch
data:倒计时：5

id:5
event:launch
data:倒计时：4

id:6
event:launch
data:倒计时：3

id:7
event:launch
data:倒计时：2

id:8
event:launch
data:倒计时：1

id:9
event:launch
data:倒计时：0

id:10
event:launch
data:发射
```

SSE Client 订阅实例

```
        @GetMapping("/server")
    public Flux<ServerSentEvent<String>> streamEvents() {
```

```
        return Flux.interval(Duration.ofSeconds(1))
                .map(sequence -> ServerSentEvent.
<String>builder()
                        .id(String.valueOf(sequence))
                        .event("test-event")
                        .data("LocalTime: " +
LocalTime.now())
                        .build());
    }

    @GetMapping("/client")
    public void consumeServerSentEvent() {
        WebClient client =
WebClient.create("http://localhost:8080");
        ParameterizedTypeReference<ServerSentEvent<String>>
type
                = new
ParameterizedTypeReference<ServerSentEvent<String>>() {
        };

        Flux<ServerSentEvent<String>> eventStream =
client.get()
                .uri("mock/server")
                .retrieve()
                .bodyToFlux(type);

        eventStream.subscribe(
                content -> log.info("Time: {} - event:
name[{}], id [{}], content[{}] ",
                        LocalTime.now(), content.event(),
content.id(), content.data()),
                error -> log.error("Error receiving SSE: {}",
error),
                () -> log.info("Completed!!!"));
    }
```

3.9. WebClient

配置 WebClient

```
@Configuration
public class WebConfig {

  @Bean
  public WebClient webClient() {

    WebClient webClient = WebClient.builder()
      .baseUrl("http://localhost:8080")
      .defaultCookie("cookie-name", "cookie-value")
      .defaultHeader(HttpHeaders.CONTENT_TYPE,
MediaType.APPLICATION_JSON_VALUE)
      .build();
  }
}
```

@Controller/@RestController 实例

```
    @GetMapping("webclient")
    public Mono<String> webclient() {
        WebClient webClient =
WebClient.create("http://localhost:8080");
        Mono<String> response = webClient
                .get().uri("/mock/mono")
                .retrieve()
                .bodyToMono(String.class);
        response.subscribe(System.out::println);
        return response;
    }
```

会返结果

```
```

```
public Mono<ResponseEntity<Employee>> createEmployee(Employee
newEmployee) {

  return webClient.post()
    .uri("/employees")
    .contentType(MediaType.APPLICATION_JSON)
    .bodyValue(newEmployee)
    .retrieve()
    .toEntity(Employee.class);
}



@PostMapping("/create")
public Mono<ResponseEntity<?>> createEmployee(@RequestBody
Employee newEmployee) {

  return employeeService.createEmployee(newEmployee)
    .map(responseEntity -> {
      if (responseEntity.getStatusCode().is2xxSuccessful()) {
        return ResponseEntity.ok(responseEntity.getBody());
      } else {
        return
ResponseEntity.status(responseEntity.getStatusCode())
          .body("Failed to create employee");
      }
    })
    .onErrorResume(exception -> {
      return
Mono.just(ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ER
ROR)
        .body("Internal Server Error: " +
exception.getMessage()));
    });
}
```

Get 请求实例

```
WebClient.create("http://localhost:8080")
```

```
    .get()
    .uri("/students")
    .retrieve()
    .bodyToFlux(Student.class);
```

URI 参数

字符串拼接方式

```
WebClient.create("http://localhost:8080")
    .get()
    .uri("/students/" + studentId)
    .retrieve()
    .bodyToMono(Student.class);
```

通过 uriBuilder 组装 Uri 参数

```
String endpoint = "/employees";

UriComponentsBuilder uriBuilder =
UriComponentsBuilder.fromPath(endpoint)
    .queryParam("param1", "value1")
    .queryParam("param2", "value2");

webClient.post()
        .uri(uriBuilder.build().toUri())
        .bodyValue(new Employee(...))
        .retrieve()
        .bodyToMono(Employee.class);
```

```
WebClient.create("http://localhost:8080")
    .get()
    .uri(uriBuilder -> uriBuilder
        .path("/student/{studentId}")
        .build(studentId))
    .retrieve()
    .bodyToMono(Student.class);

WebClient.create("http://localhost:8080")
    .get()
    .uri(uriBuilder -> uriBuilder

.path("/student/{studentId}/assignments/{assignmentId}")
        .build(studentId, assignmentId))
    .retrieve()
    .bodyToMono(Student.class);
```

uriTemplate 组装 Uri 参数

```
UriTemplate uriTemplate = new UriTemplate(
    "/student/{studentId}/assignments/{assignmentId}");

WebClient.create("http://localhost:8080")
    .get()
    .uri(uriTemplate.expand(studentId, assignmentId))
    .retrieve()
    .bodyToMono(Student.class);
```

查询参数

http://localhost:8080/students?firstName=Jon&year=1996

```
String firstName = "Jon";
String year = "1996";

WebClient.create("http://localhost:8080")
    .get()
    .uri(uriBuilder -> uriBuilder.path("/students")
        .queryParam("firstName", firstName)
        .queryParam("year", year)
        .build())
    .retrieve()
    .bodyToMono(Student.class);
```

http://localhost:8080/students?year=1995,1996,1997

```
WebClient.create("http://localhost:8080")
    .get()
    .uri(uriBuilder -> uriBuilder.path("/students")
        .queryParam("year", String.join(",", "1995", "1996",
"1997"))
        .build())
    .retrieve()
    .bodyToMono(Student.class);
```

"/products/?category=Phones&category=Tablets"

```
webClient.get()
  .uri(uriBuilder - > uriBuilder
    .path("/products/")
    .queryParam("category", "Phones", "Tablets")
    .build())
  .retrieve()
  .bodyToMono(String.class)
```

```
  .onErrorResume(e -> Mono.empty())
  .block();
```

Post 操作演示

```
Employee newEmployee = ...;  //Create a new employee object

webClient.post()
  .uri("/employees")
  .bodyValue(BodyInserters.fromValue(newEmployee))
  .retrieve()
  .toEntity(Employee.class)   //Change here
  .subscribe(
    responseEntity -> {
      // Handle success response here
      HttpStatusCode status = responseEntity.getStatusCode();
      URI location =
responseEntity.getHeaders().getLocation();
      Employee createdEmployee = responseEntity.getBody();
// Response body
      // handle response as necessary
    },
    error -> {
      // Handle the error here
      if (error instanceof WebClientResponseException) {
        WebClientResponseException ex =
(WebClientResponseException) error;
        HttpStatusCode status = ex.getStatusCode();
        System.out.println("Error Status Code: " +
status.value());
        //...
      } else {
        // Handle other types of errors
        System.err.println("An unexpected error occurred: " +
error.getMessage());
      }
    }
  );
```

Post 表单数据

```
@Service
public class EmployeeService {

  private final WebClient webClient;

  @Autowired
  public EmployeeService(WebClient webClient) {
      this.webClient = webClient;
  }

  public Mono<Employee> createEmployee(Map<String, String>
formParams) {
    return webClient.post()
      .uri("/employees")
      .body(BodyInserters.fromFormData("id",
formParams.get("id"))
        .with("name", formParams.get("name"))
        .with("status", formParams.get("status")))
      .retrieve()
      .onStatus(HttpStatus::is4xxClientError, clientResponse
-> {
          // Handle 4xx client errors here
      })
      .onStatus(HttpStatus::is5xxServerError, clientResponse
-> {
          // Handle 5xx server errors here
      })
      .toEntity(Employee.class)
      .flatMap(responseEntity ->
Mono.justOrEmpty(responseEntity.getBody()));
  }
}
```

```
        WebClient client =
WebClient.create("https://www.netkiller.cn");
        FormInserter formInserter =
fromMultipartData("name","neo")
                .with("age",19)
                .with("map",ImmutableMap.of("sex","F"))
                .with("file",new File("/tmp/netkiler.doc"));
        Mono<String> result = client.post()
                .uri("/article/index/{id}.html", 256)
                .contentType(MediaType.APPLICATION_JSON)
                .body(formInserter)
                //.bodyValue(ImmutableMap.of("name","neo"))
                .retrieve()
                .bodyToMono(String.class);
        result.subscribe(System.err::println);
```

上传文件

```
MultipartBodyBuilder builder = new MultipartBodyBuilder();

builder.part("file", new
FileSystemResource("/tmp/file.txt"));
builder.part("id", "190001", MediaType.TEXT_PLAIN);
builder.part("name", "Lokesh", MediaType.TEXT_PLAIN);
builder.part("status", "active", MediaType.TEXT_PLAIN);

Then we can submit the multipart form data by using the
method BodyInserters.fromMultipartData(builder.build()) and
send a normal request as in the previous examples.

webClient.post()
    .uri("/employees")
    .contentType(MediaType.MULTIPART_FORM_DATA)
    .body(BodyInserters.fromMultipartData(builder.build()))
    .retrieve()
    .toEntity(Employee.class)
```

```
    .doOnError(WriteTimeoutException.class, ex -> {
      System.err.println("WriteTimeout");
    })
    .subscribe(responseEntity -> {
      System.out.println("Status: " +
responseEntity.getStatusCode().value());
      System.out.println("Location URI: " +
responseEntity.getHeaders().getLocation().toString());
      System.out.println("Created New Employee : " +
responseEntity.getBody());
    });
```

设置 HTTP 头

```
webClient.get()
    .uri("/employees")
    .bodyValue(new Employee(...))
    .header("Authorization", "Bearer auth-token")
    .header("User-Agent", "Mobile App 1.0")
    .retrieve()
```

```
        WebClient.builder()
        .defaultCookie("session","f1d83210-0fc9-4689-82ab-
05df70da3367")
        .defaultUriVariables(ImmutableMap.of("name","kl"))
        .defaultHeader("header","neo")
        .defaultHeaders(httpHeaders -> {
            httpHeaders.add("header1","neo");
            httpHeaders.add("header2","chen");
        })
        .defaultCookies(cookie ->{
            cookie.add("cookie1","neo");
            cookie.add("cookie2","netkiller");
        })
```

```
        .baseUrl("https://www.netkiller.cn")
        .build();
```

websocket

```
WebSocketClient client = new ReactorNettyWebSocketClient();
URI url = new URI("ws://localhost:8080/path");
client.execute(url, session ->
        session.receive()
                .doOnNext(System.out::println)
                .then());
```

同步阻塞等待结果

```
 WebClient client =
WebClient.create("http://www.kailing.pub");
      String result = client .get()
                .uri("/article/index/arcid/{id}.html", 256)
                .retrieve()
                .bodyToMono(String.class)
                .block();
        System.err.println(result);
```

避免单独阻塞每个同步响应

```
      WebClient client =
WebClient.create("http://www.kailing.pub");
```

```
        Mono<String> result1Mono = client .get()
                .uri("/article/index/arcid/{id}.html", 255)
                .retrieve()
                .bodyToMono(String.class);
        Mono<String> result2Mono = client .get()
                .uri("/article/index/arcid/{id}.html", 254)
                .retrieve()
                .bodyToMono(String.class);
        Map<String,String>  map = Mono.zip(result1Mono,
result2Mono, (result1, result2) -> {
            Map<String, String> arrayList = new HashMap<>();
            arrayList.put("result1", result1);
            arrayList.put("result2", result2);
            return arrayList;
        }).block();
        System.err.println(map.toString());
```

4. Webflux 安全

4.1. Token 拦截器

```
package cn.netkiller.config;


import lombok.extern.slf4j.Slf4j;
import org.springframework.core.annotation.Order;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import
org.springframework.http.server.reactive.ServerHttpResponse;
import org.springframework.stereotype.Component;
import org.springframework.web.server.ServerWebExchange;
import org.springframework.web.server.WebFilter;
import org.springframework.web.server.WebFilterChain;
import reactor.core.publisher.Mono;


@Component
@Order
@Slf4j
public class TokenWebFilter implements WebFilter {
    @Override
    public Mono<Void> filter(ServerWebExchange exchange,
WebFilterChain chain) {
        log.info(exchange.getRequest().getURI().toString());
        if
(!exchange.getRequest().getHeaders().containsKey("token")) {
            ServerHttpResponse response =
exchange.getResponse();
            response.setStatusCode(HttpStatus.FORBIDDEN);

response.getHeaders().setContentType(MediaType.APPLICATION_JS
ON);
            return
response.writeWith(Mono.just(response.bufferFactory().wrap("
{\"msg\":\"no token\"}".getBytes())));
```

```
        } else {
            exchange.getAttributes().put("auth", "true");
            return chain.filter(exchange).doFinally(s -> {
                log.info("request after, url:{}, statusCode:
{}", exchange.getRequest().getURI(),
exchange.getResponse().getStatusCode());
            });
        }
    }
}
```

4.2. JWT

```
iss (issuer)：签发人
exp (expiration time)：过期时间
sub (subject)：主题
aud (audience)：受众
nbf (Not Before)：生效时间
iat (Issued At)：签发时间
jti (JWT ID)：编号
```

```
package cn.netkiller.config;


import cn.netkiller.component.JwtTokeComponent;
import cn.netkiller.utils.ResponseJson;
import com.google.gson.Gson;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.annotation.Order;
import org.springframework.core.io.buffer.DataBuffer;
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.MediaType;
import
org.springframework.http.server.reactive.ServerHttpResponse;
import org.springframework.stereotype.Component;
import org.springframework.util.AntPathMatcher;
import org.springframework.web.server.ServerWebExchange;
import org.springframework.web.server.WebFilter;
import org.springframework.web.server.WebFilterChain;
import reactor.core.publisher.Mono;

@Component
@Order
@Slf4j
public class TokenWebFilter implements WebFilter {

    private static final String[] patterns = {"/token",
"/verifier", "/mock/*", "/swagger/*", "/badges/**"};
    private final AntPathMatcher pathMatcher = new
AntPathMatcher();
    @Autowired
    private JwtTokeComponent jwtTokeComponent;

    public TokenWebFilter() {

    }

    @Override
    public Mono<Void> filter(ServerWebExchange exchange,
WebFilterChain chain) {


exchange.getFormData().subscribe(System.out::println);
        exchange.getFormData().doOnNext(n -> {
            n.forEach((k, v) -> {
                log.info("K: {}, V: {}", k, v);
            });
        });
        log.info("No Token");

        String path =
exchange.getRequest().getPath().toString();
        for (String pattern : patterns) {
            if (pathMatcher.match(pattern, path)) {
                log.info("Permit Pattern '" + pattern + "'
matches path '" + path + "'");
```

```
                return chain.filter(exchange);
            }
        }


        if
(!exchange.getRequest().getHeaders().containsKey("token")) {
            ServerHttpResponse response =
exchange.getResponse();
            response.setStatusCode(HttpStatus.FORBIDDEN);

response.getHeaders().setContentType(MediaType.APPLICATION_JS
ON);
            Mono<DataBuffer> message =
Mono.just(response.bufferFactory().wrap(new
ResponseJson(false, ResponseJson.Code.TokenException, "请提供
Token", null).toString().getBytes()));
            return response.writeWith(message);
        } else {
            String token =
exchange.getRequest().getHeaders().getFirst("token");
            log.info("token: " + token);
            ResponseJson jwt =
jwtTokeComponent.verifier(token);
            log.info("jwt: " + jwt.isStatus());
            if (jwt.isStatus()) {
                return chain.filter(exchange);
            } else {
                ServerHttpResponse response =
exchange.getResponse();
                response.setStatusCode(HttpStatus.FORBIDDEN);

response.getHeaders().setContentType(MediaType.APPLICATION_JS
ON);
                Gson gson = new Gson();
                String jsonString = gson.toJson(jwt);
                Mono<DataBuffer> message =
Mono.just(response.bufferFactory().wrap(jsonString.getBytes()
));
                return response.writeWith(message);
            }
        }
    }
}
```

```
package cn.netkiller.component;

import cn.netkiller.utils.ResponseJson;
import com.auth0.jwt.JWT;
import com.auth0.jwt.algorithms.Algorithm;
import com.auth0.jwt.exceptions.*;
import com.auth0.jwt.interfaces.DecodedJWT;
import com.auth0.jwt.interfaces.JWTVerifier;
import lombok.NoArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import reactor.core.publisher.Mono;

import java.util.Calendar;
import java.util.Date;
import java.util.List;

@Slf4j
@Component
@NoArgsConstructor
public class JwtTokeComponent {
    @Value("${app.expires}")
    private int expires;
    @Value("${app.audience}")
    private String audience;
    @Value("${app.id}")
    private String appId;
    @Value("${app.key}")
    private String appKey;
    @Value("${app.secret}")
    private String secret;
    @Value("${app.subject}")
    private String subject;
    @Value("#{'${app.role}'.split(',')}")
    private List role;

    public ResponseJson verifier(String token) {
        ResponseJson response;
```

```
        try {
            DecodedJWT jwt = this.verify(token);
            response = new ResponseJson(true,
ResponseJson.Code.SUCCESS, "Token 校验成功", jwt);
        } catch (SignatureVerificationException e) {
            response = new ResponseJson(false,
ResponseJson.Code.TokenException, e.getMessage(), "Token 签名
失败");
        } catch (TokenExpiredException e) {
            response = new ResponseJson(false,
ResponseJson.Code.TokenExpiredException, e.getMessage(),
"Token 过期");
        } catch (AlgorithmMismatchException e) {
            response = new ResponseJson(false,
ResponseJson.Code.AlgorithmMismatchException, e.getMessage(),
"Token 签名算法异常");
        } catch (JWTVerificationException e) {
            response = new ResponseJson(false,
ResponseJson.Code.TokenException, e.getMessage(), "Token 校验
失败");
        } catch (Exception e) {
            response = new ResponseJson(false,
ResponseJson.Code.Exception, e.getMessage(), "Token 异常");
        }
        log.error(response.toString());
        return response;
    }

    public Mono<String> getToken(String appId, String appKey)
{


        Calendar instance = Calendar.getInstance();
        instance.add(Calendar.DATE, expires);
//        instance.add(Calendar.SECOND, 30);
        try {
//            Algorithm algorithm =
Algorithm.RSA256(rsaPublicKey, rsaPrivateKey);
            Algorithm algorithm = Algorithm.HMAC256(secret);

            String token = JWT.create()
                    .withJWTId(appKey)
                    .withIssuer(appId)
                    .withIssuedAt(new Date())
                    .withSubject(subject)
```

```
                    .withKeyId(appKey)
                    .withAudience(audience)
                    .withClaim("role", role)
                    .withExpiresAt(instance.getTime())
                    .sign(algorithm);
            return Mono.just(token);
        } catch (JWTCreationException exception) {
            log.error(exception.getMessage());
        }
        return Mono.empty();
    }

    public DecodedJWT verify(String token) {
        Algorithm algorithm = Algorithm.HMAC256(secret);
        JWTVerifier verifier = JWT.require(algorithm)
                // specify an specific claim validations
                .withIssuer(appId)
                .withJWTId(appKey)
                .withSubject(subject)
                .withAudience(audience)
                // reusable verifier instance
                .build();

        DecodedJWT decodedJWT = verifier.verify(token);
        log.info(decodedJWT.getClaims().toString());

        return decodedJWT;
    }

}
```

4.3. spring-boot-starter-security

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-
security</artifactId>
        </dependency>
```

默认用户名是 user，密码会打印到终端

```
2024-01-03T18:52:01.027+08:00  WARN 33537 --- [watch-
development] [           main]
.s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: 823fcdcc-e2dd-4967-84b4-
546db9175357

This generated password is for development use only. Your
security configuration must be updated before running your
application in production.

2024-01-03T18:52:01.081+08:00  INFO 33537 --- [watch-
development] [           main]
o.s.b.a.e.web.EndpointLinksResolver      : Exposing 13
endpoint(s) beneath base path '/actuator'
```

访问方式

```
neo@MacBook-Pro-M2 ~> curl -X 'GET' 'http://user:823fcdcc-e2dd-
4967-84b4-546db9175357@localhost:8080/mock/mono'
hello webflux⏎
```

修改密码，在配置文件中增加

```
#
spring.security.user.name=user
spring.security.user.password=123456
#
```

```
package cn.netkiller.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import
org.springframework.security.config.annotation.method.configu
ration.EnableReactiveMethodSecurity;
import
org.springframework.security.config.annotation.web.reactive.E
nableWebFluxSecurity;
import
org.springframework.security.config.web.server.ServerHttpSecu
rity;
import
org.springframework.security.core.userdetails.MapReactiveUser
DetailsService;
import org.springframework.security.core.userdetails.User;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.server.SecurityWebFilterChai
n;

@Configuration
@EnableWebFluxSecurity
@EnableReactiveMethodSecurity
public class SecurityConfig {
//    @Autowired
//    private TokenWebFilter tokenWebFilter;

    @Bean
    public MapReactiveUserDetailsService userDetailsService()
{
        UserDetails user = User.withDefaultPasswordEncoder()
                .username("user")
                .password("user")
                .roles("USER")
```

```
                .build();
        return new MapReactiveUserDetailsService(user);
    }

    @Order(Ordered.HIGHEST_PRECEDENCE)
    @Bean
    SecurityWebFilterChain filterChain(ServerHttpSecurity
httpSecurity) throws Exception {
        httpSecurity.csrf().disable()
                .authorizeExchange(exchanges -> exchanges
                        .pathMatchers("/").permitAll()
                        .pathMatchers("/token",
"/verifier").permitAll()
                        .pathMatchers("/mock/*").permitAll()
                        .pathMatchers("/ping",
"/version").permitAll()
                        .pathMatchers("/badges/**",
"/chat/**", "/status/**", "/picture/**").permitAll()
                        .anyExchange().authenticated()
                );
//                .addFilterBefore(tokenWebFilter,
SecurityWebFiltersOrder.FIRST);
//                .httpBasic(withDefaults())
//                .formLogin(withDefaults());
        return httpSecurity.build();
    }

}
```

5. 常见问题

5.1. The Java/XML config for Spring MVC and Spring WebFlux cannot both be enabled, e.g. via @EnableWebMvc and @EnableWebFlux, in the same application.

是用 @EnableWebFlux 注解是，出现错误。这是因为 Mvc 与 Webflux 不能同时启用，通过下面配置可以解决。

```
spring.main.web-application-type=reactive
```

5.2. @EnableWebFluxSecurity 与 @EnableReactiveMethodSecurity 不生效

Mvc 不能与 Webflux 同时存在，默认系统是 Mvc 导致 Security 在 Webflux 下工作不正常。

去掉 Mvc 依赖

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
```

删除 @EnableWebSecurity 注解

```
@EnableWebMvc
```

```
@EnableWebSecurity
```

增加配置

```
spring.main.web-application-type=reactive
```

```
package cn.netkiller.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import
org.springframework.security.config.annotation.method.configu
ration.EnableReactiveMethodSecurity;
import
org.springframework.security.config.annotation.web.reactive.E
nableWebFluxSecurity;
import
org.springframework.security.config.web.server.ServerHttpSecu
rity;
import
org.springframework.security.core.userdetails.MapReactiveUser
DetailsService;
import org.springframework.security.core.userdetails.User;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.server.SecurityWebFilterChai
n;

import static
org.springframework.security.config.Customizer.withDefaults;

@Configuration
```

```
@EnableWebFluxSecurity
@EnableReactiveMethodSecurity
public class SecurityConfig {

    @Bean
    public MapReactiveUserDetailsService userDetailsService()
{
        UserDetails user = User.withDefaultPasswordEncoder()
                .username("user")
                .password("user")
                .roles("USER")
                .build();
        return new MapReactiveUserDetailsService(user);
    }

    @Order(Ordered.HIGHEST_PRECEDENCE)
    @Bean
    SecurityWebFilterChain filterChain(ServerHttpSecurity
httpSecurity) throws Exception {
        httpSecurity
                .authorizeExchange(exchanges -> exchanges
                        .anyExchange().authenticated()
                )
                .httpBasic(withDefaults())
                .formLogin(withDefaults());
        return httpSecurity.build();
    }

}
```

5.3. webflux netty 不支持 Content-Type: application/x-www-form-urlencoded

Post 数据无法使用 @RequestParam 读取，只能用于读取 URL 上的 GET 数据

```
[UnsupportedMediaTypeStatusException: 415
UNSUPPORTED_MEDIA_TYPE "Content type 'application/x-www-form-
```

```
urlencoded' not supported"]
```

第 8 章 Spring Data

1. Spring Data with Redis

1.1. 集成 Redis XML 方式

pom.xml

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
redis</artifactId>
                </dependency>
```

springframework-servlet.xml

```
        <!-- Redis Connection Factory -->
        <bean id="jedisConnFactory"

class="org.springframework.data.redis.connection.jedis.JedisCon
nectionFactory"
                p:host-name="192.168.2.1" p:port="6379" p:use-
pool="true" />

        <!-- redis redisTemplate definition -->
        <bean id="redisTemplate"
class="org.springframework.data.redis.core.RedisTemplate"
                p:connection-factory-ref="jedisConnFactory" />
```

例 8.1. Spring Data Redis Example

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-
beans.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-
mvc.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-
context.xsd">

       <mvc:resources location="/images/" mapping="/images/**"
/>
       <mvc:resources location="/css/" mapping="/css/**" />

       <context:component-scan base-
package="cn.netkiller.controller" />

       <mvc:annotation-driven />

       <bean

class="org.springframework.web.servlet.view.InternalResourceVie
wResolver">
              <property name="viewClass"

value="org.springframework.web.servlet.view.JstlView" />
              <property name="prefix" value="/WEB-INF/jsp/"
/>
              <property name="suffix" value=".jsp" />
              <!-- <property name="viewNames" value="*.jsp"
/> -->
       </bean>

       <bean id="configuracion"
```

```
class="org.springframework.beans.factory.config.PropertyPlaceho
lderConfigurer">
                <property name="location"
value="classpath:resources/development.properties" />
        </bean>

        <bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataSou
rce">
                <property name="driverClassName"
value="${jdbc.driverClassName}" />
                <property name="url" value="${jdbc.url}" />
                <property name="username"
value="${jdbc.username}" />
                <property name="password"
value="${jdbc.password}" />
        </bean>

        <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
                <property name="dataSource" ref="dataSource" />
        </bean>
        <bean
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
                <property name="basePackage"
value="cn.netkiller.mapper" />
        </bean>

        <bean id="userService"
class="cn.netkiller.service.UserService">
        </bean>

        <!-- Redis Connection Factory -->
        <bean id="jedisConnFactory"

class="org.springframework.data.redis.connection.jedis.JedisCon
nectionFactory"
                p:host-name="192.168.2.1" p:port="6379" p:use-
pool="true" />

        <!-- redis redisTemplate definition -->
        <bean id="redisTemplate"
class="org.springframework.data.redis.core.RedisTemplate"
                p:connection-factory-ref="jedisConnFactory" />
```

```
</beans>
```

Controller

```
package cn.netkiller.controller;

import javax.annotation.Resource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.ListOperations;
import org.springframework.data.redis.core.RedisTemplate;
import
org.springframework.data.redis.serializer.StringRedisSerializer
;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import cn.netkiller.model.User;

@Controller
public class CacheController {

        // inject the actual redisTemplate
        @Autowired
        private RedisTemplate<String, String> redisTemplate;

        // inject the redisTemplate as ListOperations
        @Resource(name = "redisTemplate")
        private ListOperations<String, String> listOps;

        @RequestMapping("/cache")
        public ModelAndView cache() {

                String message = "";

                User user = new User();
                user.setId("1");
                user.setName("Neo");
```

```
                user.setAge(30);

                String key = "user";
                listOps.leftPush(key, user.toString());
                message = listOps.leftPop(key);

                redisTemplate.setKeySerializer(new
StringRedisSerializer());
                redisTemplate.setValueSerializer(new
StringRedisSerializer());
                redisTemplate.opsForValue().set("key",
user.toString());

                return new ModelAndView("index/index",
"variable", message);
        }
}
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<br>
        <div style="text-align:center">
                <h2>
                        ${variable}
                </h2>
        </div>
</body>
```

```
</html>
```

测试

请求URL http://your.domain.com/your.html

```
[root@master ~]# redis-cli
redis 127.0.0.1:6379> keys *
1) "\xac\xed\x00\x05t\x00\x04user"
2) "key"

redis 127.0.0.1:6379> get key
"\xac\xed\x00\x05t\x00\x1dUser [id=1, name=Neo, age=30]"
```

提示

Spring Redis 默认使用 Byte数据类型存储Key，在redis-cli中会看到 "\xac\xed\x00\x05t\x00\x04" 前缀不方便get操作，所以我们会设置使用字符串，通过 redisTemplate.setKeySerializer(new StringRedisSerializer()); 实现

1.2. RedisTemplate

stringRedisTemplate 基本用法

```
stringRedisTemplate.opsForValue().set("test",
"100",60*10,TimeUnit.SECONDS);  //向redis里存入数据和设置缓存时间
stringRedisTemplate.opsForValue().get("test")
//根据key获取缓存中的val
stringRedisTemplate.getExpire("test")
//根据key获取过期时间
stringRedisTemplate.getExpire("test",TimeUnit.SECONDS)
//根据key获取过期时间并换算成指定单位
```

```
stringRedisTemplate.delete("test");
//根据key删除缓存
stringRedisTemplate.hasKey("546545");
//检查key是否存在，返回boolean值
stringRedisTemplate.expire("test",1000 ,
TimeUnit.MILLISECONDS);                              //设置过期时间
```

设置缓存时间

例子：设置 name 缓存 10 秒

```
redisTemplate.opsForValue().set("name","neo",10,
TimeUnit.SECONDS);
redisTemplate.opsForValue().get("name")
```

结果：由于设置的是10秒失效，十秒之内查询有结果，十秒之后返回为null

字符串截取

设置：redisTemplate.opsForValue().set("hello","Helloworld");
代码：

```
System.out.println(redisTemplate.opsForValue().get("hello",0,
5));
```

结果：Hellow
代码：

```
System.out.println(redisTemplate.opsForValue().get("hello",0,
-1));
```

结果：Helloworld
代码：

```
System.out.println(redisTemplate.opsForValue().get("hello",-3
,-1));
```

结果：rld

追加字符串

```
redisTemplate.opsForValue().append("hello","Hello");
System.out.println(redisTemplate.opsForValue().get("hello"));

redisTemplate.opsForValue().append("hello","world");
System.out.println(redisTemplate.opsForValue().get("hello"));
// 结果：Helloworld
```

设置键的字符串值并返回其旧值

```
        redisTemplate.opsForValue().set("name","neo");

System.out.println(redisTemplate.opsForValue().getAndSet("nam
e","Jerry"));
        // 结果 neo
```

increment

```
stringRedisTemplate.opsForValue().set("test", "100");
//向redis里存入数据
stringRedisTemplate.boundValueOps("test").increment(-50);
//val做-60操作
stringRedisTemplate.boundValueOps("test").increment(100);
//val +100
stringRedisTemplate.opsForValue().get("test")
```

```
//根据key获取缓存中的val
```

删除 **key**

```
        private void cleanNewToday() {
                long begin = System.currentTimeMillis();

                redisTemplate.delete("news:today");

        long end = System.currentTimeMillis();
                logger.info("Schedule clean redis {} 耗时 {}
秒", "cleanNewFlash()", (end-begin) / 1000 );
        }
```

返回字符串长度

```
redisTemplate.opsForValue().set("key","hello world");
System.out.println(redisTemplate.opsForValue().size("key"));
```

如果**key**不存便缓存。

```
System.out.println(redisTemplate.opsForValue().setIfAbsent("n
ame","neo"));   // name 之前已经存在 false
System.out.println(redisTemplate.opsForValue().setIfAbsent("a
ge","11"));             // age 之前不存在 true
```

setIfAbsent 实现分布式锁

```
boolean static =
Boolean.TRUE.equals(redisTemplate.opsForValue().setIfAbsent("
lock:order", 1, 1, TimeUnit.DAYS));
```

缓存多个值 /获取多个值 multiSet / multiGet

```
Map<String,String> maps = new HashMap<String, String>();
        maps.put("multi1","multi1");
        maps.put("multi2","multi2");
        maps.put("multi3","multi3");

redisTemplate.opsForValue().multiSet(maps);

List<String> keys = new ArrayList<String>();
        keys.add("multi1");
        keys.add("multi2");
        keys.add("multi3");

System.out.println(redisTemplate.opsForValue().multiGet(keys)
);
```

输出结果

```
[multi1, multi2, multi3]
```

为多个键分别设置它们的值，如果存在则返回false，不存在返回true

```
Map<String,String> maps = new HashMap<String, String>();
        maps.put("multi11","multi11");
        maps.put("multi22","multi22");
        maps.put("multi33","multi33");
Map<String,String> maps2 = new HashMap<String, String>();
        maps2.put("multi1","multi1");
        maps2.put("multi2","multi2");
        maps2.put("multi3","multi3");

System.out.println(redisTemplate.opsForValue().multiSetIfAbse
nt(maps));      // 返回 true
System.out.println(redisTemplate.opsForValue().multiSetIfAbse
nt(maps2));     // 返回 false
```

List

rightPush

```
ListOperations<String, Object> list =
redisTemplate.opsForList();
list.rightPush("books", "Linux");
list.rightPush("books", "Java");
System.out.println(list.range("books", 0, 1));

System.out.println(redisTemplate.opsForList().size("list"));
```

rightPushAll

```
                String[] stringarrays = new String[]
{"1","2","3"};

redisTemplate.opsForList().rightPushAll("listarrayright",stri
ngarrays);

System.out.println(redisTemplate.opsForList().range("listarra
yright",0,-1));
```

```
                List<Object> strings = new ArrayList<Object>
();
        strings.add("1");
        strings.add("2");
        strings.add("3");

redisTemplate.opsForList().rightPushAll("listcollectionright"
, strings);

System.out.println(redisTemplate.opsForList().range("listcoll
ectionright",0,-1));
```

rightPushIfPresent

```
                System.out.println("========== KEY 不存在
==========");

System.out.println(redisTemplate.opsForList().rightPushIfPres
ent("rightPushIfPresent","aa"));

System.out.println(redisTemplate.opsForList().rightPushIfPres
ent("rightPushIfPresent","bb"));
        System.out.println("========== KEY 已经存在
```

```
==========");

System.out.println(redisTemplate.opsForList().rightPushIfPres
ent("rightPushIfPresent","aa"));

System.out.println(redisTemplate.opsForList().rightPushIfPres
ent("rightPushIfPresent","bb"));
```

leftPush

```
redisTemplate.opsForList().leftPush("list","java");
redisTemplate.opsForList().leftPush("list","python");
redisTemplate.opsForList().leftPush("list","c++");
```

leftPushAll

批量把一个数组插入到列表中

```
        String[] stringarrays = new String[]{"1","2","3"};

redisTemplate.opsForList().leftPushAll("listarray",stringarra
ys);
```

批量把一个集合插入到列表中

```
使用：List<Object> strings = new ArrayList<Object>();
        strings.add("1");
        strings.add("2");
```

```
        strings.add("3");

redisTemplate.opsForList().leftPushAll("listcollection",
strings);

System.out.println(redisTemplate.opsForList().range("listcoll
ection",0,-1));
结果:[3, 2, 1]
```

range

```
System.out.println(redisTemplate.opsForList().range("listarra
y",0,-1));
        // 结果:[3, 2, 1]
```

SET 数据类型

Redis的Set是无序集合并且集合成员是唯一的，这就意味着集合中不能出现重复的数据。

```
stringRedisTemplate.opsForSet().add("test", "1","2","3");
//向指定key中存放set集合
stringRedisTemplate.opsForSet().isMember("test", "1")
//根据key查看集合中是否存在指定数据
stringRedisTemplate.opsForSet().members("test");
//根据key获取set集合
```

```
//添加 一个 set 集合
```

```
SetOperations<String, Object> set =
redisTemplate.opsForSet();
set.add("Member", "neo");
set.add("Member", "36");
set.add("Member", "178cm");
//输出 set 集合
System.out.println(set.members("Member"));
```

```
package cn.netkiller.api.restful;

import java.util.Set;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import common.pojo.ResponseRestful;

@RestController
@RequestMapping("/news")
public class NewsRestController {

        @Autowired
        private RedisTemplate<String, String> redisTemplate;

        @RequestMapping(value = "/flash/{count}")
        public ResponseRestful flash(@PathVariable("count")
long count) {
                if(count == 0L) {
                        count=10L;
                }
                Set<String> news =
this.redisTemplate.opsForZSet().reverseRange("news:flash", 0,
count);
                if (news == null) {
```

```
                        return new ResponseRestful(false, 10,
"没有查询到结果", news);
                }
                return new ResponseRestful(true, 0, "返回数据:
" + news.size() + " 条", news);
        }

        public void addRecentUser(long userId, String name) {
            String key =
RedisKeyGenerator.genRecentBrowsingPositionsKey(String.valueO
f(userId));
            // 获取已缓存的最近浏览的职位
            ZSetOperations<String, String> zSetOperations =
redisTempalte.opsForZSet();
        //zset内部是按分数来排序的，这里用当前时间做分数
            zSetOperations.add(key, name,
System.currentTimeMillis());
            zSetOperations.removeRange(key, 0, -6);
        }

}
```

返回集合中的所有成员

```
System.out.println(redisTemplate.opsForSet().members("setTest
"));
```

取出一个成员

```
System.out.println(redisTemplate.opsForSet().pop("setTest"));
```

随机获取无序集合中的一个元素

```
System.out.println("Random member: " +
redisTemplate.opsForSet().randomMember("setTest"));
```

随机获取 **n** 个成员（存在重复数据）

```
System.out.println("Random member: " +
redisTemplate.opsForSet().randomMembers("setTest",5));
// 结果 Random member: [ccc, ddd, ddd, ddd, aaa]
```

随机获取 **n** 个不重复成员

```
System.out.println("Random members: " +
redisTemplate.opsForSet().distinctRandomMembers("setTest",5))
;
//结果 Random members: [aaa, bbb, ddd, ccc]
```

在两个 **SET** 间移动数据

```
        redisTemplate.opsForSet().move("key1","aaa","key2");

System.out.println(redisTemplate.opsForSet().members("key1"))
;
```

```
System.out.println(redisTemplate.opsForSet().members("key2"))
;
```

成员删除

```
String[] arrays = new String[]{"Java","PHP"};
System.out.println(redisTemplate.opsForSet().remove("setTest"
,arrays));
```

返回集合数量

```
System.out.println(redisTemplate.opsForSet().size("setTest"))
;
```

判断元素是否在集合成员中

```
ystem.out.println(redisTemplate.opsForSet().isMember("setTest
","Linux"));
```

对比两个集合求交集

```
System.out.println(redisTemplate.opsForSet().members("key"));
```

```
System.out.println(redisTemplate.opsForSet().members("otherKe
y"));

System.out.println(redisTemplate.opsForSet().intersect("key",
"otherKey"));
```

```
        List<String> library2 = new ArrayList<String>();
        library2.add("Linux");
        library2.add("FreeBSD");

System.out.println(redisTemplate.opsForSet().intersect("libra
ry1",library2));
```

对比两个集合求交集，然后存储到新的 **key** 中

```
System.out.println(redisTemplate.opsForSet().intersectAndStor
e("key","otherKey","destKey"));
```

```
                List<String> otherKey = new ArrayList<String>
();
        otherKey.add("《Netkiller Java 手札》");
        otherKey.add("《Netkiller Spring Cloud 手札》");

System.out.println(redisTemplate.opsForSet().intersectAndStor
e("key",otherKey,"destKey"));
```

合并两个集合，并去处重复数据

```
System.out.println(redisTemplate.opsForSet().union("setTest1"
,"setTest2"));

        List<String> otherKey = new ArrayList<String>();
    otherKey.add("《Netkiller Java 手札》");
    otherKey.add("《Netkiller Spring Cloud 手札》");

System.out.println(redisTemplate.opsForSet().union("setTest",
otherKey));
```

合并两个集合去重复后保存到新的 **key** 中

```
System.out.println(redisTemplate.opsForSet().unionAndStore("k
ey","otherKey","destKey"));

System.out.println(redisTemplate.opsForSet().unionAndStore("k
ey",otherKey,"destKey"));
```

计算两个合集的差集

```
System.out.println(redisTemplate.opsForSet().difference("key"
,"otherKey"));

        List<String> otherKey = new ArrayList<String>();
    otherKey.add("setTest2");
    otherKey.add("setTest3");
```

```
System.out.println(redisTemplate.opsForSet().difference("key"
,otherKey));
```

计算两个合集的差集，然后保存到新的 **key** 中

```
System.out.println(redisTemplate.opsForSet().differenceAndSto
re("key","otherKey","destKey"));
```

遍历 **SET** 集合

```
        Cursor<Object> curosr =
redisTemplate.opsForSet().scan("setTest", ScanOptions.NONE);
    while(curosr.hasNext()){
        System.out.println(curosr.next());
    }
```

有序的 set 集合

```
//添加有序的 set 集合
ZSetOperations<String, Object> zset =
redisTemplate.opsForZSet();
zset.add("zMember", "neo", 0);
zset.add("zMember", "36", 1);
zset.add("zMember", "178cm", 2);
//输出有序 set 集合
System.out.println(zset.rangeByScore("zMember", 0, 2));
```

```

```

Hash

put

```
redisTemplate.opsForHash().put("redisHash","name","neo");
    redisTemplate.opsForHash().put("redisHash","age",30);

redisTemplate.opsForHash().put("redisHash","nickname","netkil
ler");
```

putAll

```
HashOperations<String, Object, Object>  hash =
redisTemplate.opsForHash();
Map<String,Object> map = new HashMap<String,Object>();
map.put("name", "neo");
map.put("age", "36");
hash.putAll("member", map);

System.out.println(hash.entries("member"));
```

从键中的哈希获取给定**hashKey**的值

```
System.out.println(redisTemplate.opsForHash().get("redisHash"
```

```
,"age"));
```

delete

删除指定的哈希 hashKeys

```
System.out.println(redisTemplate.opsForHash().delete("redisHa
sh","name"));
```

确定哈希hashKey是否存在

确定哈希hashKey是否存在

```
System.out.println(redisTemplate.opsForHash().hasKey("redisHa
sh","age"));
```

从哈希中获取指定的多个 hashKey 的值

```
        List<Object> keys = new ArrayList<Object>();
    keys.add("name");
    keys.add("age");

System.out.println(redisTemplate.opsForHash().multiGet("redis
Hash",keys))
```

只有**hashKey**不存在时才能添加值

```
System.out.println(redisTemplate.opsForHash().putIfAbsent("redisHash","age",30));
```

获取整个**Hash**

```
System.out.println(redisTemplate.opsForHash().entries("redisHash"));
```

获取所有**key**

```
System.out.println(redisTemplate.opsForHash().keys("redisHash1"));
```

通过 **hashKey** 获取所有值

```
System.out.println(redisTemplate.opsForHash().values("redisHash"));
```

值加法操作

```
System.out.println(redisTemplate.opsForHash().increment("redi
sHash","age",1)
```

遍历 Hash 表

```
        Cursor<Map.Entry<Object, Object>> curosr =
redisTemplate.opsForHash().scan("redisHash",
ScanOptions.ScanOptions.NONE);
        while(curosr.hasNext()){
            Map.Entry<Object, Object> entry = curosr.next();

System.out.println(entry.getKey()+":"+entry.getValue());
        }
```

过期时间未执行

Spring Redis 中设置过期时间方法如下

```
设置 key
redisTemplate.opsForValue().setIfAbsent("key", "value");
设置过期时间
redisTemplate.expire("key", 30000, TimeUnit.MILLISECONDS);
释放 key
redisTemplate.delete("key");
```

这样存在一个问题，当程序运行一半被强行终止，可能导致 setIfAbsent运行完成，但是expire未被执行，这样 key 便永远不会释

放。解决方案如下，使用RedisCallback执行原生 Redis 命令。

```
String result = redisTemplate.execute(new
RedisCallback<String>() {
        @Override
        public String doInRedis(RedisConnection connection)
throws DataAccessException {
                JedisCommands commands = (JedisCommands)
connection.getNativeConnection();
                return commands.set(key, value, "NX", "PX",
expire);
        }
});
```

setBit / getBit 二进制位操作

```
setBit Boolean setBit(K key, long offset, boolean value);

offset 二进制位置(从左向右数)
value 位 ture 表示 0, false 表示 1
```

```
    // 'a' 的ASCII码是 97 转换为二进制是: 01100001
    // 'b' 的ASCII码是 98 转换为二进制是: 01100010
    // 'c' 的ASCII码是 99 转换为二进制是: 01100011

        redisTemplate.opsForValue().set("bitTest","a");

    redisTemplate.opsForValue().setBit("bitTest",7, false);
// 01100011
    redisTemplate.opsForValue().setBit("bitTest",8, true);
// 01100010
```

```
System.out.println(redisTemplate.opsForValue().get("bitTest")
);
    redisTemplate.opsForValue().setBit("bitTest",8, false);
// 01100011

System.out.println(redisTemplate.opsForValue().get("bitTest")
);
```

getBit Boolean getBit(K key, long offset); 获取键对应值的ascii码的在offset处位值

```
System.out.println(redisTemplate.opsForValue().getBit("bitTes
t",7));
```

存储 Json 对象

集成 RedisTemplate 定义新类 JsonRedisTemplate

```
package cn.netkiller.wallet.redis;

import
org.springframework.data.redis.connection.RedisConnectionFact
ory;
import org.springframework.data.redis.core.RedisTemplate;
import
org.springframework.data.redis.serializer.Jackson2JsonRedisSe
rializer;
import
org.springframework.data.redis.serializer.RedisSerializer;
import
org.springframework.data.redis.serializer.StringRedisSerializ
er;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;

public class JsonRedisTemplate extends RedisTemplate<String,
Object> {

        public JsonRedisTemplate(RedisConnectionFactory
connectionFactory, ObjectMapper objectMapper, Class<?>
valueType) {
                RedisSerializer<String> stringSerializer =
new StringRedisSerializer();
                super.setKeySerializer(stringSerializer);
                super.setHashKeySerializer(stringSerializer);

super.setHashValueSerializer(stringSerializer);
                Jackson2JsonRedisSerializer<?>
jsonRedisSerializer = new Jackson2JsonRedisSerializer<>
(valueType);

jsonRedisSerializer.setObjectMapper(objectMapper);

super.setValueSerializer(jsonRedisSerializer);

super.setConnectionFactory(connectionFactory);
                super.afterPropertiesSet();
        }
}
```

配置 Redis

```
package cn.netkiller.wallet.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.data.redis.connection.RedisConnectionFact
ory;
import
org.springframework.data.redis.core.StringRedisTemplate;
```

```
import org.springframework.data.redis.listener.ChannelTopic;
import
org.springframework.data.redis.listener.RedisMessageListenerC
ontainer;
import
org.springframework.data.redis.listener.adapter.MessageListen
erAdapter;

import com.fasterxml.jackson.databind.ObjectMapper;

import cn.netkiller.wallet.redis.JsonRedisTemplate;
import cn.netkiller.wallet.redis.RedisMessageSubscriber;

@Configuration
public class RedisConfig {

        public RedisConfig() {
        }

        @Bean
        public StringRedisTemplate
stringRedisTemplate(RedisConnectionFactory connectionFactory)
{
                StringRedisTemplate redisTemplate = new
StringRedisTemplate();

redisTemplate.setConnectionFactory(connectionFactory);
                return redisTemplate;
        }

        @Bean
        public MessageListenerAdapter messageListener() {
                return new MessageListenerAdapter(new
RedisMessageSubscriber());
        }

        @Bean
        public ChannelTopic topic() {
                return new ChannelTopic("demo");
        }

        @Bean
        public RedisMessageListenerContainer
redisContainer(RedisConnectionFactory connectionFactory,
MessageListenerAdapter messageListener) {
```

```
                RedisMessageListenerContainer container = new
RedisMessageListenerContainer();


container.setConnectionFactory(connectionFactory);

container.addMessageListener(messageListener(), topic());

container.addMessageListener(messageListener(), new
ChannelTopic("test"));
                return container;
        }

        @Bean
        public ObjectMapper objectMapper() {
                return new ObjectMapper();
        }

        @Bean
        public JsonRedisTemplate
jsonRedisTemplate(RedisConnectionFactory connectionFactory,
ObjectMapper objectMapper) {
                return new
JsonRedisTemplate(connectionFactory, objectMapper,
Object.class);
        }

}
```

测试

```


package cn.netkiller.wallet.restful;

import java.io.IOException;
import java.util.UUID;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.listener.ChannelTopic;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

import cn.netkiller.wallet.pojo.RestfulResponse;
import cn.netkiller.wallet.redis.JsonRedisTemplate;
import cn.netkiller.wallet.redis.RedisMessagePublisher;

@RestController
public class TestRestController {
        private static final Logger logger =
LoggerFactory.getLogger(TestRestController.class);

        @Autowired
        private StringRedisTemplate stringRedisTemplate;

        @Autowired
        private JsonRedisTemplate jsonRedisTemplate;

        public TestRestController() {

        }

        @GetMapping("/version")
        public String version() throws IOException {
                Web3ClientVersion web3ClientVersion =
web3j.web3ClientVersion().send();
                String clientVersion =
web3ClientVersion.getWeb3ClientVersion();
                logger.info(clientVersion);
                return clientVersion;
        }

        @GetMapping("/pub/demo")
        public String pub() {

                RedisMessagePublisher publisher = new
RedisMessagePublisher(stringRedisTemplate, new
ChannelTopic("demo"));
```

```
                String message = "Message " +
UUID.randomUUID();
                publisher.publish(message);
                return message;
        }

        @GetMapping("/pub/test")
        public String pub(@RequestParam String message) {

                RedisMessagePublisher publisher = new
RedisMessagePublisher(stringRedisTemplate, new
ChannelTopic("test"));
                publisher.publish(message);
                return message;
        }

        @GetMapping("/pub/json")
        public RestfulResponse pubJson() {
                RestfulResponse restfulResponse = new
RestfulResponse(true, 0, null, null);
                jsonRedisTemplate.opsForValue().set("test",
restfulResponse);
                jsonRedisTemplate.convertAndSend("test",
restfulResponse);
                return restfulResponse;
        }
}
```

1.3. Spring Data Redis - Repository Examples

@EnableRedisRepositories 启动 Redis 仓库

```
package api.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.data.redis.repository.configuration.Enabl
eRedisRepositories;
```

```
@Configuration
@EnableRedisRepositories
public class CachingConfigurer {

}
```

定义 Domain 类

```
package api.domain;

import java.util.List;

import org.springframework.data.annotation.Id;
import org.springframework.data.annotation.Reference;
import org.springframework.data.redis.core.RedisHash;
import org.springframework.data.redis.core.index.Indexed;

@RedisHash("persons")
public class Person {

        public enum Gender {
                FEMALE, MALE
        }

        @Id
        private String id;

        @Indexed
        private String firstname;
        @Indexed
        private String lastname;

        private Gender gender;
        private Address address;

        @Reference
        private List<Person> children;
```

```
	public Person() {
		// TODO Auto-generated constructor stub
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getFirstname() {
		return firstname;
	}

	public void setFirstname(String firstname) {
		this.firstname = firstname;
	}

	public String getLastname() {
		return lastname;
	}

	public void setLastname(String lastname) {
		this.lastname = lastname;
	}

	public Gender getGender() {
		return gender;
	}

	public void setGender(Gender gender) {
		this.gender = gender;
	}

	public Address getAddress() {
		return address;
	}

	public void setAddress(Address address) {
		this.address = address;
	}

	public List<Person> getChildren() {
```

```
                return children;
        }

        public void setChildren(List<Person> children) {
                this.children = children;
        }

        @Override
        public String toString() {
                return "Person [id=" + id + ", firstname=" +
firstname + ", lastname=" + lastname + ", gender=" + gender +
", address=" + address + ", children=" + children + "]";
        }

}
```

```
package api.domain;

import org.springframework.data.geo.Point;
import org.springframework.data.redis.core.index.GeoIndexed;
import org.springframework.data.redis.core.index.Indexed;

public class Address {

        private @Indexed String city;
        private String country;
        private @GeoIndexed Point location;

        public Address(String city, String country, Point
location) {
                this.city = city;
                this.country = country;
                this.location = location;
        }

        public String getCity() {
                return city;
        }
```

```
        public void setCity(String city) {
                this.city = city;
        }

        public String getCountry() {
                return country;
        }

        public void setCountry(String country) {
                this.country = country;
        }

        public Point getLocation() {
                return location;
        }

        public void setLocation(Point location) {
                this.location = location;
        }

}
```

Repository 接口

```
package api.repository;

import java.util.List;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.geo.Circle;
import org.springframework.data.repository.CrudRepository;

import api.domain.Person;

public interface PersonRepository extends
CrudRepository<Person, String> {
        List<Person> findByLastname(String lastname);
```

```
        Page<Person> findPersonByLastname(String lastname,
Pageable page);

        List<Person> findByFirstnameAndLastname(String
firstname, String lastname);

        List<Person> findByFirstnameOrLastname(String
firstname, String lastname);

        List<Person> findByAddress_City(String city);

        List<Person> findByAddress_LocationWithin(Circle
circle);
}
```

测试代码

```
package api.restful;

import java.util.Arrays;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.geo.Point;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import api.domain.Person;
import api.domain.Address;

import api.repository.PersonRepository;
```

```
@RestController
@RequestMapping("/test")
public class TestRestController {
        private static final Logger logger =
LoggerFactory.getLogger(TestRestController.class);

        @Autowired
        private PersonRepository personRepository;

        public TestRestController() {

        }

        @GetMapping("/redis")
        public Person redis() {

                Person children = new Person();
                children.setFirstname("Lisa");
                children.setLastname("Chen");
                children.setGender(Person.Gender.FEMALE);

                Person person = new Person();
                person.setFirstname("Neo");
                person.setLastname("Chen");
                person.setGender(Person.Gender.MALE);

                // List<Person> childrens = new
ArrayList<Person>();

                person.setChildren(Arrays.asList(children));

                Point point = new
Point(Double.valueOf("28.352734"),
Double.valueOf("32.807382"));
                Address address = new Address("Shenzhen",
"China", point);
                person.setAddress(address);
                personRepository.save(person);
                return person;
        }

}
```

2. Spring Data with MongoDB

https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/

2.1. Example Spring Data MongoDB

pom.xml

注意Spring4 与 1.9.1.RELEASE有兼容性问题，日志提示 Error creating bean with name 'mongoTemplate' defined in ServletContext resource

```
                <dependency>
<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
mongodb</artifactId>
                        <version>1.8.1.RELEASE</version>
                </dependency>
```

springframework-servlet.xml

```
        <mongo:db-factory id="mongoDbFactory"
host="${mongo.host}" port="${mongo.port}"
dbname="${mongo.database}" />
        <!-- username="${mongo.username}"
password="${mongo.password}" -->

        <bean id="mongoTemplate"
class="org.springframework.data.mongodb.core.MongoTemplate">
        <constructor-arg name="mongoDbFactory"
```

```
ref="mongoDbFactory"/>
    </bean>

    <mongo:mapping-converter id="converter" db-factory-
ref="mongoDbFactory"/>
    <bean id="gridFsTemplate"
class="org.springframework.data.mongodb.gridfs.GridFsTemplate">
      <constructor-arg ref="mongoDbFactory"/>
      <constructor-arg ref="converter"/>
    </bean>
```

例 8.2. Spring Data MongoDB - springframework-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:p="http://www.springframework.org/schema/p"

xmlns:mongo="http://www.springframework.org/schema/data/mongo"
        xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-
beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-
mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context.xsd

http://www.springframework.org/schema/data/mongo

http://www.springframework.org/schema/data/mongo/spring-mongo-
1.5.xsd
        ">

        <mvc:resources location="/images/" mapping="/images/**"
```

```
/>
        <mvc:resources location="/css/" mapping="/css/**" />
        <mvc:resources location="/js/" mapping="/js/**" />
        <mvc:resources location="/zt/" mapping="/zt/**" />
        <mvc:resources location="/sm/" mapping="/sm/**" />
        <mvc:resources location="/module/" mapping="/module/**"
/>

        <context:component-scan base-
package="cn.netkiller.controller" />
        <!-- <context:property-placeholder
location="classpath:resources/development.properties" /> -->
        <mvc:annotation-driven />

        <bean id="viewResolver"
class="org.springframework.web.servlet.view.UrlBasedViewResolve
r">
                <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView" />
                <property name="prefix" value="/WEB-INF/jsp/"
/>
                <property name="suffix" value=".jsp" />
        </bean>

        <bean id="configuracion"
class="org.springframework.beans.factory.config.PropertyPlaceho
lderConfigurer">
                <property name="location"
value="classpath:resources/development.properties" />
        </bean>

        <!-- MongoDB Connection Factory -->
        <mongo:db-factory id="mongoDbFactory"
host="${mongo.host}" port="${mongo.port}"
dbname="${mongo.database}" />
        <!-- username="${mongo.username}"
password="${mongo.password}" -->

        <!-- MongoDB template definition -->
        <bean id="mongoTemplate"
class="org.springframework.data.mongodb.core.MongoTemplate">
        <constructor-arg name="mongoDbFactory"
ref="mongoDbFactory"/>
    </bean>
```

```
    <!-- MongoDB GridFS template definition -->
    <mongo:mapping-converter id="converter" db-factory-
ref="mongoDbFactory"/>
    <bean id="gridFsTemplate"
class="org.springframework.data.mongodb.gridfs.GridFsTemplate">
      <constructor-arg ref="mongoDbFactory"/>
      <constructor-arg ref="converter"/>
    </bean>

        <!-- Redis Connection Factory -->
        <bean id="jedisConnFactory"
class="org.springframework.data.redis.connection.jedis.JedisCon
nectionFactory" p:host-name="192.168.2.1" p:port="6379" p:use-
pool="true" />

        <!-- redis template definition -->
        <bean id="redisTemplate"
class="org.springframework.data.redis.core.RedisTemplate"
p:connection-factory-ref="jedisConnFactory" />

</beans>
```

development.properties 配置内容

```
mongo.host=192.168.4.1
mongo.port=27017
mongo.username=test
mongo.password=passw0rd
mongo.database=website
```

POJO

```
package cn.netkiller.pojo;

import org.springframework.data.annotation.Id;
```

```
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "tracker")
public class Tracker {
	@Id
	private String id;
	private String name;
	private String unique;
	private String hostname;
	private String referrer;
	private String href;

	public Tracker() {
		// TODO Auto-generated constructor stub
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getUnique() {
		return unique;
	}

	public void setUnique(String unique) {
		this.unique = unique;
	}

	public String getHostname() {
		return hostname;
	}

	public void setHostname(String hostname) {
		this.hostname = hostname;
	}

	public String getReferrer() {
		return referrer;
	}

	public void setReferrer(String referrer) {
```

```
                this.referrer = referrer;
        }

        public String getHref() {
                return href;
        }

        public void setHref(String href) {
                this.href = href;
        }

        @Override
        public String toString() {
                return "Tracker [id=" + id + ", name=" + name +
", unique=" + unique + ", hostname=" + hostname + ", referrer="
+ referrer + ", href=" + href + "]";
        }

}
```

Controller

```
package cn.netkiller.controller;

import cn.netkiller.pojo.Tracker;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
public class TrackerController {

        @Autowired
        private MongoTemplate mongoTemplate;

        public TrackerController() {
```

```
        }

        @RequestMapping("/tracker/test")
        @ResponseBody
        String hello() {
                return "Hello World!";
        }

        @RequestMapping("/tracker")
        @ResponseBody
        String execute() {
                Tracker tracker = new Tracker();
                tracker.setName("test");
                tracker.setUnique("111223456");
                tracker.setHostname("www.example.com");

tracker.setHref("http://example.com/test.html");
                tracker.setReferrer("http://example.com/");
                this.mongoTemplate.insert(tracker);

                return tracker.toString();
        }

}
```

查看测试结果

```
> db.tracker.find();
{ "_id" : ObjectId("5757c0b92c526a6bda5eea3a"), "_class" :
"cn.netkiller.repositories.Tracker", "name" : "test", "unique"
: "111223456", "hostname" : "www.example.com", "referrer" :
"http://example.com/", "href" : "http://example.com/test.html"
}
```

条件查询

```
        @RequestMapping("/read/name/{name}")
        public ArrayList<Tracker> sort(@PathVariable String
name) {

                Query query = new
Query(Criteria.where("name").is(name));

                ArrayList<Tracker> trackers =
(ArrayList<Tracker>) mongoTemplate.find(query, Tracker.class);
                return trackers;
        }
```

2.2. MongoDB 多数据源

Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
                </dependency>
```

Application 禁止自动配置 MongoDB

exclude = { MongoAutoConfiguration.class, MongoDataAutoConfiguration.class }

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
```

```
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.data.mongo.MongoDataAu
toConfiguration;
import
org.springframework.boot.autoconfigure.mongo.MongoAutoConfigu
ration;
import
org.springframework.boot.context.ApplicationPidFileWriter;

@SpringBootApplication(exclude = {
MongoAutoConfiguration.class,
MongoDataAutoConfiguration.class })
public class Application {

        public static void main(String[] args) {

                System.out.println("Starting...");
                SpringApplication springApplication = new
SpringApplication(Application.class);
                springApplication.addListeners(new
ApplicationPidFileWriter());
                springApplication.run(args);
        }
}
```

application.properties 新增配置项

```
mongodb.primary.uri=mongodb://netkiller:chen@192.168.30.10:27
017/news
mongodb.secondary.uri=mongodb://netkiller:chen@192.168.30.5:2
7017/member
```

MongoDB 配置类

```
package cn.netkiller.config;

import org.springframework.data.mongodb.MongoDatabaseFactory;
import
org.springframework.data.mongodb.core.SimpleMongoClientDataba
seFactory;
import com.mongodb.ConnectionString;

public abstract class AbstractMongoConfigure {

        public MongoDatabaseFactory
mongoDatabaseFactory(String uri) {
                ConnectionString connectionString = new
ConnectionString(uri);
                return new
SimpleMongoClientDatabaseFactory(connectionString);
        }
}
```

配置多数据源

```
package cn.netkiller.config;

import
org.springframework.beans.factory.annotation.Qualifier;
import
org.springframework.boot.autoconfigure.mongo.MongoProperties;
import
org.springframework.boot.context.properties.ConfigurationProp
erties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.data.mongodb.core.MongoTemplate;
```

```
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;

@Configuration
@ConfigurationProperties(prefix = "mongodb")
@EnableMongoRepositories(basePackages = {
"cn.netkiller.repository" }, mongoTemplateRef =
MultipleMongoConfigure.primaryMongoTemplate)
public class MultipleMongoConfigure extends
AbstractMongoConfigure {

        protected static final String primaryMongoTemplate =
"primaryMongoTemplate";
        protected static final String secondaryMongoTemplate
= "secondaryMongoTemplate";

        private MongoProperties primary = new
MongoProperties();
        private MongoProperties secondary = new
MongoProperties();

        public MongoProperties getPrimary() {
                return primary;
        }

        public void setPrimary(MongoProperties primary) {
                this.primary = primary;
        }

        public MongoProperties getSecondary() {
                return secondary;
        }

        public void setSecondary(MongoProperties secondary) {
                this.secondary = secondary;
        }

        public MultipleMongoConfigure() {
        }

        @Primary
        @Bean(name =
MultipleMongoConfigure.primaryMongoTemplate)
        @Qualifier(value =
```

```
MultipleMongoConfigure.primaryMongoTemplate)
        public MongoTemplate primaryMongoTemplate() throws
Exception {
                String uri = this.getPrimary().getUri();
                return new
MongoTemplate(mongoDatabaseFactory(uri));
        }

        @Bean(name = "secondaryMongoTemplate")
        @Qualifier("secondaryMongoTemplate")
        public MongoTemplate secondaryMongoTemplate() throws
Exception {
                String uri = this.getSecondary().getUri();
                return new
MongoTemplate(mongoDatabaseFactory(uri));
        }

}
```

创建 Document 关系映射类

```
package cn.netkiller.domain;

import java.io.Serializable;
import java.util.Date;

import
org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.mongodb.core.mapping.MongoId;

import com.fasterxml.jackson.annotation.JsonFormat;

@Document
public class User implements Serializable {
        private static final long serialVersionUID =
-3258839839160856613L;
        // private Long id;
        @MongoId
        private String id;
```

```
        private String useruame;
        private String password;
        private String name;
        private String sex;
        private Integer age;
        @JsonFormat(pattern = "yyyy-MM-dd", timezone =
"GMT+8")
        private Date birthday;

        public String getId() {
                return id;
        }

        public void setId(String id) {
                this.id = id;
        }

        public String getUseruame() {
                return useruame;
        }

        public void setUseruame(String useruame) {
                this.useruame = useruame;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getSex() {
                return sex;
        }
```

```
        public void setSex(String sex) {
                this.sex = sex;
        }

        public Integer getAge() {
                return age;
        }

        public void setAge(Integer age) {
                this.age = age;
        }

        public Date getBirthday() {
                return birthday;
        }

        public void setBirthday(Date birthday) {
                this.birthday = birthday;
        }

}
```

测试控制器

```
package cn.netkiller.controller;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

import cn.netkiller.domain.User;
import cn.netkiller.service.UserService;

@RestController
```

```
public class TestMongoController {

	@Autowired
	@Qualifier(value = "primaryMongoTemplate")
	private MongoTemplate primaryMongoTemplate;

	@Autowired
	@Qualifier(value = "secondaryMongoTemplate")
	private MongoTemplate secondaryMongoTemplate;

	public TestMongoController() {
		// TODO Auto-generated constructor stub
	}

	@GetMapping("/mongo/primary/save")
	public String primarysave() {
		User user = new User();
		user.setUseruame("netkiller");
		user.setPassword("123456");
		primaryMongoTemplate.save(user);
		return "Success\r\n";
	}

	@GetMapping("/mongo/secondary/save")
	public String secondaryMongoTemplate() {
		User user = new User();
		user.setUseruame("netkiller");
		user.setPassword("123456");
		secondaryMongoTemplate.save(user);
		return "Success\r\n";
	}
}
```

测试

启动 Springboot 可以看到下面👇日志，两个MongoDB都链接成功。

```
```

```
2021-10-14 19:29:50.037  INFO 93698 --- [           main]
org.mongodb.driver.cluster               : Cluster created with
settings {hosts=[192.168.30.10:27017], mode=SINGLE,
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
2021-10-14 19:29:50.156  INFO 93698 --- [168.30.10:27017]
org.mongodb.driver.connection            : Opened connection
[connectionId{localValue:1, serverValue:126}] to
192.168.30.10:27017
2021-10-14 19:29:50.157  INFO 93698 --- [168.30.10:27017]
org.mongodb.driver.cluster               : Monitor thread
successfully connected to server with description
ServerDescription{address=192.168.30.10:27017, type=STANDALONE,
state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=13,
maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=31466638}
2021-10-14 19:29:50.157  INFO 93698 --- [168.30.10:27017]
org.mongodb.driver.connection            : Opened connection
[connectionId{localValue:2, serverValue:127}] to
192.168.30.10:27017
2021-10-14 19:29:50.266  INFO 93698 --- [           main]
org.mongodb.driver.cluster               : Cluster created with
settings {hosts=[192.168.30.5:27017], mode=SINGLE,
requiredClusterType=UNKNOWN, serverSelectionTimeout='30000 ms'}
2021-10-14 19:29:50.272  INFO 93698 --- [.168.30.5:27017]
org.mongodb.driver.connection            : Opened connection
[connectionId{localValue:3, serverValue:969}] to
192.168.30.5:27017
2021-10-14 19:29:50.272  INFO 93698 --- [.168.30.5:27017]
org.mongodb.driver.connection            : Opened connection
[connectionId{localValue:4, serverValue:968}] to
192.168.30.5:27017
2021-10-14 19:29:50.272  INFO 93698 --- [.168.30.5:27017]
org.mongodb.driver.cluster               : Monitor thread
successfully connected to server with description
ServerDescription{address=192.168.30.5:27017, type=STANDALONE,
state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=13,
maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30,
roundTripTimeNanos=2345376}
```

```
neo@MacBook-Pro-Neo ~ % curl
```

```
http://localhost:8080/mongo/primary/save
Success

neo@MacBook-Pro-Neo ~ % curl
http://localhost:8080/mongo/secondary/save
Success
```

现在去两个 MongoDB 数据查看输入是否保存成功。

在使用 curl 调用的时候，日志会显示链接两个 MongoDB 的状态。

```
2021-10-14 19:34:27.795  INFO 93698 --- [  XNIO-1 task-1]
org.mongodb.driver.connection            : Opened connection
[connectionId{localValue:5, serverValue:970}] to
192.168.30.5:27017
2021-10-14 19:34:31.096  INFO 93698 --- [  XNIO-1 task-1]
org.mongodb.driver.connection            : Opened connection
[connectionId{localValue:6, serverValue:130}] to
192.168.30.10:27017
```

2.3. @Document

复杂的 @Document 数据类型定义

```
package cn.netkiller.domain;

import java.util.Date;
import java.util.List;
import java.util.Map;

import org.springframework.data.annotation.Id;
import
org.springframework.data.mongodb.core.mapping.Document;
```

```
@Document
public class MultilevelDirectSellingTradingRebate {

	public enum Type {
		POINT, CASH, GIFT
	}

	public enum Rebate {
		DIRECT, INDIRECT
	}

	public enum Status {
		New, Rejected, Approved
	}

	@Id
	private String id;
	public String name;
	public Date beginDate;
	public Date endDate;
	public double lowAmount;
	public double highAmount;
	public Type type;
	public Status status = Status.New;
	public List<Map<String, Map<?, ?>>> product;

	@Override
	public String toString() {
		return "MultilevelDirectSellingTradingRebate
[id=" + id + ", name=" + name + ", beginDate=" + beginDate
				+ ", endDate=" + endDate + ",
lowAmount=" + lowAmount + ", highAmount=" + highAmount + ",
type=" + type
				+ ", status=" + status + ",
product=" + product + "]";
	}

}
```

指定表名

默认使用 class 作为表名

```
@Document
public class Multilevel {
        ...
        ...
}
```

指定特别表名

```
@Document(collection = "author")
```

@Id

```
        @Id
        private String id;
```

@Version

```
        @Version
        private Long version;
```

@Field 定义字段名

```
@Field("url")
private String link;
```

@Indexed

https://docs.spring.io/spring-data/mongodb/docs/current/api/org/springframework/data/mongodb/core/index/Indexed.html

索引

普通索引

```
@Indexed
```

唯一索引

```
@Indexed(unique=true)
```

索引排序方式

```

```

```
@Indexed(name = "first_name_index", direction =
IndexDirection.DESCENDING)
```

稀疏索引

稀疏索引允许唯一索引存在多个 null 值

```
        @Indexed(unique = true, sparse = true)
        private String uuid;

        @Indexed(unique = true, sparse = true)
        private String transactionId = null;
```

索引过期时间设置

```
@Indexed(name = "expire_after_seconds_index",
expireAfterSeconds = 10)
private LocalDateTime updateDate;
```

@CompoundIndex 复合索引

普通复合索引

```
@Document
@CompoundIndexes({
    @CompoundIndex(name = "email_age", def = "{'email.id' :
1, 'age': 1}")
```

```
})
public class User {
    //
}
```

```
@Document
@CompoundIndexes({
  @CompoundIndex(def = "{'firstName':1, 'salary':-1}", name =
"compound_index_1"),
  @CompoundIndex(def = "{'secondName':1, 'profession':1}",
name = "compound_index_2")
})
public class Person {
  @Id private String id;
  private String firstName;
  private String secondName;
  private LocalDateTime dateOfBirth;
  private Address address;
  private String profession;
  private int salary;
  // constructor
  // getters and setters
}
```

唯一复合索引

唯一复合索引：楼层和房号不能相同，不然就是同一个房间了

```
@CompoundIndexes({
    @CompoundIndex(name = "floor_num", def = "{'floor' : 1,
'num': 1}",unique=true)
})
```

不允许同名

```
@CompoundIndexes({ @CompoundIndex(name = "username", def = "
{'firstname' : 1, 'lastname': 1}", unique = true) })
```

@TextIndexed

```
@Document(language = "spanish")
class SomeEntity {

    @TextIndexed String foo;

    @Language String lang;

    Nested nested;
}

class Nested {

    @TextIndexed(weight=5) String bar;
    String roo;
}
```

@GeoSpatialIndex 地理位置索引

点数据索引

```
        @GeoSpatialIndexed
        private GeoJsonPoint location; // GPS 定位信息
```

```

```

2D 数据索引

```
@GeoSpatialIndexed(type = GeoSpatialIndexType.GEO_2DSPHERE)
```

@Transient 丢弃数据，不存到 mongodb

```
public class User {

    @Transient
    private Integer age;

    // standard getter and setter

}
```

@DBRef 做外外键引用

Article 类

```
package cn.netkiller.api.domain;

import java.util.List;

import org.springframework.data.mongodb.core.mapping.DBRef;
import
org.springframework.data.mongodb.core.mapping.Document;
```

```
@Document
public class Article {

	private String title; // 名称
	private String description; // 描述
	private String tag; // 类型
	@DBRef
	private List<Hypermedia> hypermedia; // 图片, 视频

	public Article() {
		// TODO Auto-generated constructor stub
	}

	public String getTitle() {
		return title;
	}

	public void setTitle(String title) {
		this.title = title;
	}

	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description;
	}

	public String getTag() {
		return tag;
	}

	public void setTag(String tag) {
		this.tag = tag;
	}

	public List<Hypermedia> getHypermedia() {
		return hypermedia;
	}

	public void setHypermedia(List<Hypermedia>
hypermedia) {
		this.hypermedia = hypermedia;
```

```
        }

        @Override
        public String toString() {
                return "Article [title=" + title + ",
description=" + description + ", tag=" + tag + ",
hypermedia=" + hypermedia + "]";
        }

}
```

Hypermedia 类

```
package api.domain;

import org.springframework.data.annotation.Id;
import
org.springframework.data.mongodb.core.mapping.Document;

@Document
public class Hypermedia {

        @Id
        private String id;
        private String hash;
        private String name;
        private String size;

        public Hypermedia() {
                // TODO Auto-generated constructor stub
        }

        public Hypermedia(String hash, String name, String
size) {
                this.hash = hash;
                this.name = name;
                this.size = size;
        }
```

```
        public String getId() {
                return id;
        }

        public void setId(String id) {
                this.id = id;
        }

        public String getHash() {
                return hash;
        }

        public void setHash(String hash) {
                this.hash = hash;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getSize() {
                return size;
        }

        public void setSize(String size) {
                this.size = size;
        }

        @Override
        public String toString() {
                return "Hypermedia [id=" + id + ", hash=" +
hash + ", name=" + name + ", size=" + size + "]";
        }

}
```

如果你只查询 Article 表，不会单独查询 Hypermedia，返回结果可以掩藏 Id ，不写 get/set 方法即可。

```
package cn.netkiller.api.domain;

import org.springframework.data.annotation.Id;
import
org.springframework.data.mongodb.core.mapping.Document;

@Document
public class Hypermedia {

        @Id
        private String id;
        private String hash;
        private String name;
        private String size;

        public Hypermedia() {
                // TODO Auto-generated constructor stub
        }

        public Hypermedia(String hash, String name, String
size) {
                this.hash = hash;
                this.name = name;
                this.size = size;
        }

        public String getHash() {
                return hash;
        }

        public void setHash(String hash) {
                this.hash = hash;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
```

```
                this.name = name;
        }

        public String getSize() {
                return size;
        }

        public void setSize(String size) {
                this.size = size;
        }

        @Override
        public String toString() {
                return "Hypermedia [hash=" + hash + ", name="
+ name + ", size=" + size + "]";
        }

}
```

MongoRepository

```
package cn.netkiller.api.repository;

import
org.springframework.data.mongodb.repository.MongoRepository;

import api.domain.Article;

public interface ArticleRepository extends
MongoRepository<Article, String> {

}
```

```
package cn.netkiller.api.repository;

import
org.springframework.data.mongodb.repository.MongoRepository;

import api.domain.Hypermedia;

public interface HypermediaRepository extends
MongoRepository<Hypermedia, String> {

}
```

RestController

```
package cn.netkiller.api.restful;

import java.util.ArrayList;
import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import api.domain.Article;
import api.domain.Hypermedia;
import api.repository.ArticleRepository;
import api.repository.HypermediaRepository;

@RestController
@RequestMapping("/article")
public class ArticleRestController {

        @Autowired
```

```
        private ArticleRepository articleRepository;

        @Autowired
        private HypermediaRepository hypermediaRepository;

        public ArticleRestController() {
                // TODO Auto-generated constructor stub
        }

        @GetMapping("/save")
        public Article save() {

                Article article = new Article();
                article.setTitle("标题");
                article.setDescription("摘要");
                article.setTag("标签");

                Hypermedia hypermedia = new Hypermedia("AAA",
"BBB", "CCC");
                hypermediaRepository.save(hypermedia);

                List<Hypermedia> hypermedias = new
ArrayList<Hypermedia>();
                hypermedias.add(hypermedia);

                article.setHypermedia(hypermedias);

                articleRepository.save(article);

                System.out.println(article);

                return article;
        }

}
```

运行结果



```
neo@MacBook-Pro ~ % curl -s -H "Accept: application/json" -H
"Content-Type: application/json" -H "Authorization: Bearer
${TOKEN}" -X GET ${URL}/article/save | jq
{
  "title": "标题",
  "description": "摘要",
  "tag": "标签",
  "hypermedia": [
    {
      "hash": "AAA",
      "name": "BBB",
      "size": "CCC"
    }
  ]
}
```

MongoDB 结果

db.getCollection('article').find({})

```
/* 1 */
{
    "_id" : ObjectId("5bab66f8c92782395817cb05"),
    "title" : "标题",
    "description" : "摘要",
    "tag" : "标签",
    "hypermedia" : [
        {
            "$ref" : "hypermedia",
            "$id" : ObjectId("5bab66f8c92782395817cb04")
        }
    ],
    "_class" : "cn.netkiller.api.domain.Article"
}
```

db.getCollection('hypermedia').find({})

```
/* 1 */
{
    "_id" : ObjectId("5bab66b9c927823951f4f5fe"),
    "hash" : "AAA",
    "name" : "BBB",
    "size" : "CCC",
    "_class" : "api.domain.Hypermedia"
}
```

@DateTimeFormat

```
@DateTimeFormat( pattern = "yyyy-MM-dd" )
private Date birthday

@DateTimeFormat(iso = DateTimeFormat.ISO.NONE)
private final Calendar datetime;

@DateTimeFormat(pattern="yyyy-MM-dd HH:mm:ss")
private Date date;

@DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME)
private Date createdDate = new Date();
```

@NumberFormat

```
@NumberFormat(style=Style.CURRENCY)
                                private double money;
```

在 @Document 中使用 Enum 类型

```
        public enum Type {
                POINT, CASH, GIFT
        }

        public enum Rebate {
                DIRECT, INDIRECT
        }

        public enum Status {
                New, Rejected, Approved
        }
```

枚举类型的赋值方法

```
                MultilevelDirectSellingTradingRebate
multilevelDirectSellingTradingRebate = new
MultilevelDirectSellingTradingRebate();
                multilevelDirectSellingTradingRebate.name =
"TEST";

multilevelDirectSellingTradingRebate.beginDate = new Date();
                multilevelDirectSellingTradingRebate.endDate
= new Date();

multilevelDirectSellingTradingRebate.lowAmount = 1.5d;

multilevelDirectSellingTradingRebate.highAmount = 100d;
                multilevelDirectSellingTradingRebate.type =
Type.CASH;
```

在 @Document 中定义数据结构 List/Map

```
        public List<Map<String, Map<?, ?>>> product;
```

下面是数据集结构的赋值例子

```
        Map<Enum<Rebate>, Double> rebate = new
HashMap<Enum<Rebate>, Double>();

        rebate.put(Rebate.DIRECT, 10.05d);
        rebate.put(Rebate.INDIRECT, 6.05d);

        Map<String, Map<?, ?>> prod1 = new HashMap<String,
Map<?, ?>>();
        prod1.put("USDRMB", rebate);

        List<Map<String, Map<?, ?>>> products = new
ArrayList<Map<String, Map<?, ?>>>();
        products.add(prod1);
        multilevelDirectSellingTradingRebate.product =
products;
```

GeoJson 数据类型

```
        @GeoSpatialIndexed
        private GeoJsonPoint location; // GPS 地址位置
```

```
location = new GeoJsonPoint(Double.valueOf(longitude),
```

```
Double.valueOf(latitude));
```

2.4. MongoRepository

扫描仓库接口

默认不需要设置，除非你的包不在当前包下，或者命令不是repository。

```
@EnableMongoRepositories(basePackages =
"cn.netkiller.repository")
```

findAll()

```
        @RequestMapping(value = "read", method =
RequestMethod.GET, produces = { "application/xml",
"application/json" })
        @ResponseStatus(HttpStatus.OK)
        public List<Withdraw> read() {
                return repository.findAll();
        }
```

deleteAll()

```
repository.deleteAll();
```

```

```

save()

```
repository.save(new City("Shenzhen", "China"));
```

count()

```
        @RequestMapping("count")
        public long count() {
                return repository.count();
        }
```

exists() 判断是否存在

```
boolean isExists = userRepository.exists(user.getId());
```

existsById()

```
memberRepository.existsById(id);
```

```

```

findByXXXX

```
List<User> findByName(String name);

List<User> users = userRepository.findByName("Eric");
```

findAll with OrderBy

order by boolean 布尔型数据排序

因为 boolean 数据 true = 1, false = 0 所以 ASC false 会排列在前面。所有很多时候而我们需要 DESC 排序

```
List<ShippingAddress> shippingAddress =
shippingAddressRepository.findAllByMemberIdOrderByDefaultsDes
c(memberId);
```

findAll with Sort

```
List<User> users = userRepository.findAll(new
Sort(Sort.Direction.ASC, "name"));
```

FindAll with Pageable

```
Pageable pageable = PageRequest.of(0, 1);
Page<User> page = userRepository.findAll(pageable);
List<User> users = pages.getContent();
```

PageRequest - springboot 1.x 旧版本

```
Page<User> findByLastname(String lastname, Pageable
pageable);
```

```
        @RequestMapping(value = "read/{size}/{page}", method
= RequestMethod.GET, produces = { "application/xml",
"application/json" })
        @ResponseStatus(HttpStatus.OK)
        public List<Withdraw> readPage(@PathVariable int
size, @PathVariable int page){
                PageRequest pageRequest = new
PageRequest(page-1,size);
                return
repository.findAll(pageRequest).getContent();
        }
```

URL翻页参数，每次返回10条记录

```
                                                第一页
http://localhost:8080/v1/withdraw/read/10/1.json
```

```
                                                第二页
http://localhost:8080/v1/withdraw/read/10/2.json
                                                ...
                                                第五页
http://localhost:8080/v1/withdraw/read/10/5.json
```

StartingWith 和 EndingWith

```
List<User> findByNameStartingWith(String regexp);
List<User> findByNameEndingWith(String regexp);

List<User> users =
userRepository.findByNameStartingWith("N");
List<User> users = userRepository.findByNameEndingWith("o");
```

Between

数值范围

```
List<User> findByAgeBetween(int ageGT, int ageLT);

List<User> users = userRepository.findByAgeBetween(20, 50);
```

日期范围，取值 e.g. 2018-07-04 00:00:00 and 2018-07-04 23:59:59

```
List<Member> findByCreatedDateBetween(DateTime start,
DateTime end);
```

```
List<Member> findByCreatedDate(@Temporal(TemporalType.DATE)
Date date);
```

Before / After

```
List<Assets> findAllByUpdateDateBefore(Date yesterday);
List<Assets> findAllByUpdateDateBeforeAndStatus(Date
yesterday, String status);

List<Assets> findAllByUpdateDateAfter(Date yesterday);
```

@Query

```
public interface PersonRepository extends
MongoRepository<Person, String> {
        @Query("{ 'name' : ?0 }")
        List<Person> findWithQuery(String userId);
}

        @Query(value = "{'statusHistories':{$elemMatch:
{'status':{$in:['PROCESSABLE']}}},'created' : { '$gt' : {
'$date' : ':?0' } , '$lt' : { '$date' : ':?1'}}}", count =
true)
        Long countMe(@Param("dateFrom") Date datefrom,
@Param("dateTo") Date dateTo);
```

2.5. mongoTemplate

导入与模板相关的包

```
import org.springframework.data.mongodb.core.MongoTemplate;
import org.springframework.data.mongodb.core.query.Criteria;
import org.springframework.data.mongodb.core.query.Query;
import org.springframework.data.mongodb.core.query.Update;
```

注入 MongoTemplate 对象

```
        @Autowired
        private MongoTemplate mongoTemplate;
```

Save 保存

```
User user = new User();
user.setName("Netkiller");
mongoTemplate.save(user, "user");
```

更新数据

```
user =
mongoTemplate.findOne(Query.query(Criteria.where("name").is("
Jam")), User.class);
user.setName("Neo");
mongoTemplate.save(user, "user");
```

Insert

```
User user = new User();
user.setName("Neo");
mongoTemplate.insert(user, "user");
```

```
BSONObject personBsonObj = BasicDBObjectBuilder.start()
                .add("name","Neo Chen")
                .add("age",27)
                .add("address",null).get();

mongoTemplate.insert(personBsonObj,"personCollection");
```

document in the db:

```
db.personCollection.findOne().pretty();
{"age":21,"name":"John Doe";"address":null}*
```

updateFirst 修改符合条件第一条记录

updateFirst 修改符合条件第一条记录

```
Query query = new Query();
query.addCriteria(Criteria.where("name").is("Neo"));
Update update = new Update();
update.set("name", "Netkiller");
```

```
mongoTemplate.updateFirst(query, update, User.class);
```

updateMulti 修改符合条件的所有

更新所有数据

```
Query query = new Query();
query.addCriteria(Criteria.where("name").is("Neo"));
Update update = new Update();
update.set("name", "Jerry");
mongoTemplate.updateMulti(query, update, User.class);
```

查找并保存

```
Query query = new Query();
query.addCriteria(Criteria.where("name").is("Luck"));
Update update = new Update();
update.set("name", "Lisa");
User user = mongoTemplate.findAndModify(query, update,
User.class);
```

upsert - 修改符合条件时如果不存在则添加

```
Query query = new Query();
query.addCriteria(Criteria.where("name").is("Green"));
Update update = new Update();
```

```
update.set("name", "Tom");
mongoTemplate.upsert(query, update, User.class);
```

```
mongoTemplate.upsert(new
Query(Criteria.where("age").is("18")), new
Update().set("name", "neo"), collectionName);
```

删除

```
User user = new User();
user.setId("5bbf091efd9557069c4a25c5")
mongoTemplate.remove(user, "user");
```

查找一条数据

```
public Person findOneByName(String name) {
   Query query = new Query();
   query.addCriteria(Criteria.where("name").is(name));
   return mongoTemplate.findOne(query, Person.class);
}
```

查找所有数据

```
public List<Person> findByName(String name) {
   Query query = new Query();
   query.addCriteria(Criteria.where("name").is(name));
   return mongoTemplate.find(query, Person.class);
}
```

Query

翻页

```
public List<Person> getAllPersonPaginated(int pageNumber, int
pageSize) {
   Query query = new Query();
   query.skip(pageNumber * pageSize);
   query.limit(pageSize);
   return mongoTemplate.find(query, Person.class);
}
```

between

实现一个区间条件 new Criteria("createdDate").gte(beginDate).lte(endDate)

```
        public boolean AccountDeposit(Date beginDate, Date
endDate) {

                MatchOperation matchOperation = match(new
Criteria("createdDate").gte(beginDate).lte(endDate));
                GroupOperation groupOperation =
group("loginname").sum("amount").as("amount");
```

```
                SortOperation sortOperation = sort(new
Sort(Direction.ASC, "loginname"));

                Aggregation aggregation =
newAggregation(matchOperation, groupOperation,
sortOperation);
                AggregationResults<AccountSettlementDetails>
results = mongoTemplate.aggregate(aggregation,
AccountSettlementDetails.class,
AccountSettlementDetails.class);

                if (results.getMappedResults() != null) {

log.info(results.getRawResults().get("result").toString());
                        for (AccountSettlementDetails
settlementDetails : results.getMappedResults()) {

                                log.info("{}",
settlementDetails.toString());

                        }
                }
                return true;
        }
```

Criteria

is

```
Query query = new Query();
query.addCriteria(Criteria.where("name").is("Neo"));
List<User> users = mongoTemplate.find(query, User.class);
```

Regex 正则表达式搜索

查询以N开头的名字

```
Query query = new Query();
query.addCriteria(Criteria.where("name").regex("^N"));
List<User> users = mongoTemplate.find(query,User.class);
```

查询以o结尾的名字

```
Query query = new Query();
query.addCriteria(Criteria.where("name").regex("o$"));
List<User> users = mongoTemplate.find(query, User.class);
```

lt 和 gt

查询年龄小于 < 30 并 > 20 的用户

```
Query query = new Query();
query.addCriteria(Criteria.where("age").lt(30).gt(20));
List<User> users = mongoTemplate.find(query,User.class);
```

查找日期范围

```
Date start = DateUtil.convertStringToDateTime("2014-02-10
20:38:44");
Date end = DateUtil.convertStringToDateTime("2014-02-10
```

```
20:38:50");

Query query = new Query();
Criteria criteria = Criteria.where("delflag").is(false);
criteria.and("modifyDate").gte(start).lte(end);
query.addCriteria(criteria);
query.limit(10);
```

exists()

```
Query query = new Query();
query.addCriteria(
    new Criteria().andOperator(
        Criteria.where("field1").exists(true),
        Criteria.where("field1").ne(false)
    )
);

List<Foo> result = mongoTemplate.find(query, Foo.class);
System.out.println("query - " + query.toString());

for (Foo foo : result) {
    System.out.println("result - " + foo);
}
```

包含

```
public List<Person> findByFavoriteBooks(String favoriteBook)
{
   Query query = new Query();

query.addCriteria(Criteria.where("favoriteBooks").in(favorite
Book));
```

```
    return mongoTemplate.find(query, Person.class);
}
```

Update

set

```
Update update = new Update();
update.set("name", "Netkiller");
```

追加数据

```
        Query query =
Query.query(Criteria.where("id").is("5bbf091efd9557069c4a25c5
"));
        Update update = new Update().push("author", new
Author("neo", "chen"));
        mongoTemplate.updateFirst(query, update,
Article.class);
```

更新数据

```
        Query query =
Query.query(Criteria.where("classId").is("1").and("Students.s
tudentId").is("1"));
    Update update = Update.update("Students.$.name", "lisa");
    mongoTemplate.upsert(query, update, "class");
```

```
```

删除数据

```
        Query query =
Query.query(Criteria.where("classId").is("1").and("Students.s
tudentId").is("3"));
        Update update = new Update();
        update.unset("Students.$");
        mongoTemplate.updateFirst(query, update, "class");
```

inc

```
public void updateMultiplePersonAge() {
   Query query = new Query();
   Update update = new Update().inc("age", 1);
   mongoTemplate.findAndModify(query, update, Person.class);;
}
```

update.addToSet

```
Query query = Query.query(Criteria.where("classId").is("1"));
Student student = new Student("1", "lisa", 3, "girl");
Update update = new Update();
update.addToSet("Students", student);
mongoTemplate.upsert(query, update, "class");
```

BasicUpdate

BasicUpdate 是底层更新可操作，需要手动实现$set等语句

```
BasicDBObject basicDBObject = new BasicDBObject();
basicDBObject.put("$set", new BasicDBObject("date","2018-09-
09"));
Update update = new BasicUpdate(basicDBObject);
mongoTemplate.updateFirst(new
Query(Criteria.where("nickname").is("netkiller")),
update,collectionName);
```

Sort

按照年龄排序

```
Query query = new Query();
query.with(new Sort(Sort.Direction.ASC, "age"));
List<User> users = mongoTemplate.find(query,User.class);
```

Query + PageRequest

```
final Pageable pageableRequest = new PageRequest(0, 2);
Query query = new Query();
query.with(pageableRequest);
```

newAggregation

```
MultilevelDirectSellingAccountRewardsSettlementDetails
multilevelDirectSellingAccountRewardsSettlementDetails = new
MultilevelDirectSellingAccountRewardsSettlementDetails();

multilevelDirectSellingAccountRewardsSettlementDetails.setLog
inname("111");

multilevelDirectSellingAccountRewardsSettlementDetails.setPho
ne("111");

multilevelDirectSellingAccountRewardsSettlementDetails.setRec
ommenderLoginname("111");

multilevelDirectSellingAccountRewardsSettlementDetails.setRec
ommenderPhone("111");

multilevelDirectSellingAccountRewardsSettlementDetails.setRec
ommenderName("Neo");

multilevelDirectSellingAccountRewardsSettlementDetails.setRec
ommenderType("客户");

multilevelDirectSellingAccountRewardsSettlementDetails.setAmo
unt(5.02);

multilevelDirectSellingAccountRewardsSettlementDetails.setCre
atedDate(new Date());

multilevelDirectSellingAccountRewardsSettlementDetailsReposit
ory.save(multilevelDirectSellingAccountRewardsSettlementDetai
ls);

                Date beginDate = this.getToday("00:00:00");
                Date endDate = this.getToday("23:59:59");
                log.info(beginDate.toString() + " ~ " +
endDate.toString());

                GroupOperation groupOperation =
```

```
group("loginname").sum("amount").as("amount");
                MatchOperation matchOperation = match(new
Criteria("createdDate").gte(beginDate).lte(endDate));
                SortOperation sortOperation = sort(new
Sort(Direction.ASC, "loginname"));

                Aggregation aggregation =
newAggregation(matchOperation, groupOperation,
sortOperation);

AggregationResults<MultilevelDirectSellingAccountRewardsSettl
ementDetails> results = mongoTemplate.aggregate(aggregation,
MultilevelDirectSellingAccountRewardsSettlementDetails.class,
MultilevelDirectSellingAccountRewardsSettlementDetails.class)
;

System.out.println(results.getRawResults().get("result").toSt
ring());
```

创建索引

```
mongoOps.indexOps(User.class).ensureIndex(new
Index().on("name", Direction.ASC));
```

子对象操作

List 类型

```
package cn.netkiller.api.domain;

import java.util.List;
```

```
import javax.persistence.Id;
import
org.springframework.data.mongodb.core.mapping.Document;

@Document
public class Article {

        @Id
        private String id;
        private String title;
        private String description;

        List<Author> author;
        public static class Author {
                private String id;
                private String firstname;
                private String lastname;

                public Author(String firstname, String
lastname) {
                        this.firstname = firstname;
                        this.lastname = lastname;
                }
        }
}
```

更新

```
db.getCollection('foo').update({"author.firstname":"neo"},
{"$set":{"author.$.firstname":"netkiller"}})
```

更新数据

```
        Query query =
Query.query(Criteria.where("author.firstname").is("neo"));
```

```
        Update update = new
Update().set("author.$.firstname", "netkiller");
        mongoTemplate.updateFirst(query, update,
Article.class);
```

追加数据

```
        Query query =
Query.query(Criteria.where("id").is("5bbf091efd9557069c4a25c5
"));
        Update update = new Update().push("author", new
Author("neo", "chen"));
        mongoTemplate.updateFirst(query, update,
Article.class);
```

删除数据

```
                Query query =
Query.query(Criteria.where("id").is("5bbf091efd9557069c4a25c5
"));
                Update update = new Update().pull("author",
new Author("jerry", "lee"));
                mongoTemplate.updateFirst(query, update,
Article.class);
```

2.6. GeoJson 反序列化

正常情况下是不需要做反序列化操作的。如花你想测试，打印一些信息可以这样做。

```
package cn.netkiller.api.config;

import java.io.IOException;

import
org.springframework.data.mongodb.core.geo.GeoJsonPoint;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonNode;

public class GeoJsonDeserializer extends
JsonDeserializer<GeoJsonPoint> {

        @Override
        public GeoJsonPoint deserialize(JsonParser
jsonParser, DeserializationContext deserializationContext)
throws IOException {

                final JsonNode tree =
jsonParser.getCodec().readTree(jsonParser);
                final String type =
tree.get("type").asText();
                final JsonNode coordsNode =
tree.get("coordinates");

                System.out.println(tree.toString());
                System.out.println(type);
                System.out.println(coordsNode.toString());

                double x = 0;
                double y = 0;
                if ("Point".equalsIgnoreCase(type)) {
                        x = coordsNode.get(0).asDouble();
                        y = coordsNode.get(1).asDouble();
                } else {
                        System.out.println(String.format("No
logic present to deserialize %s ", tree.asText()));
                }

                final GeoJsonPoint point = new
```

```
GeoJsonPoint(x, y);

                return point;
        }
}
```

使用 @JsonDeserialize 指定反序列化 Class

```
@Document
public class Address {
        @Id
        private String id;
        // @GeoSpatialIndexed
        @JsonDeserialize(using = GeoJsonDeserializer.class)
        private GeoJsonPoint location; // GPS 定位信息
}
```

2.7. FAQ

location object expected, location array not in correct format; nested exception is com.mongodb.MongoWriteException: location object expected, location array not in correct format

GeoJsonPoint 可能设置了索引，且有些数据部正确。

3. Spring Data with MySQL

3.1. 选择数据库表引擎

正常创建表会使用数据库默认引擎，有时数据库默认引擎并不是我们需要的，通过下面配置可以指定表引擎

```
# Spring boot 1.x.x
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect

# Spring boot 2.0.2
spring.jpa.hibernate.use-new-id-generator-mappings=true
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
```

3.2. 声明实体

@Entity 声明实体

声明 Class 即是数据库表

```
@Entity
@Table
public class Your_table {
        ...
        ...
}
```

@Table 定义表名

catalog

```
@Table(name="CUSTOMERS",catalog="hibernate")
```

schema

配置Schema

```
@Table(name="tabname", schema="public")
```

uniqueConstraints

唯一索引

```
@Table(name="CUSTOMERS",uniqueConstraints=
{@UniqueConstraint(columnNames={"name","email"})})
```

定义多组唯一索引

```
uniqueConstraints={@UniqueConstraint(columnNames=
{"name","email"}),@UniqueConstraint(columnNames=
{"name","age"})}
```

@Id 定义主键

ID 字段，数据库中的主键。

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id", unique = true, nullable = false,
insertable = true, updatable = false)
private int id;
```

@GeneratedValue 主键生成策略：

```
@GeneratedValue(strategy= GenerationType.IDENTITY)      该注解由数据库自动生成，AUTO_INCREMENT 自增主键，在 mysql 数据库中使用最频繁，oracle 不支持。
@GeneratedValue(strategy= GenerationType.AUTO)          主键由程序控制，默认的主键生成策略，oracle 默认是序列化的方式，mysql 默认是主键自增的方式。
@GeneratedValue(strategy= GenerationType.SEQUENCE)      根据底层数据库的序列来生成主键，条件是数据库支持序列，Oracle支持，Mysql不支持。
@GeneratedValue(strategy= GenerationType.TABLE)         使用一个特定的数据库表格来保存主键，较少使用。
```

Long = bigint

```
package cn.netkiller.domain;

import jakarta.persistence.*;
import lombok.Data;

import java.io.Serial;
import java.io.Serializable;
import java.util.Date;

@Entity
```

```
@Table
@Data
public class Picture implements Serializable {
    @Serial
    public static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", unique = true, nullable = false,
insertable = true, updatable = false)
    private Long id;
    private String device;
    private String model;
    private String session;
    private String prompt;
    private String thumbnail;
    private String image;
    private String story;
    private boolean share;
    private int likes;
    private int favorites;
    private int forward;
    private Date ctime;
    private Date mtime;

}
```

```
CREATE TABLE `picture` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `ctime` datetime(6) DEFAULT NULL,
  `favorites` int NOT NULL,
  `image` varchar(255) DEFAULT NULL,
  `likes` int NOT NULL,
  `mtime` datetime(6) DEFAULT NULL,
  `prompt` varchar(255) DEFAULT NULL,
  `session` varchar(255) DEFAULT NULL,
  `share` bit(1) NOT NULL,
  `story` varchar(255) DEFAULT NULL,
  `thumbnail` varchar(255) DEFAULT NULL,
```

```
  `device` varchar(255) DEFAULT NULL,
  `model` varchar(255) DEFAULT NULL,
  `forward` int NOT NULL
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

字符串做主键

```
package api.domain;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table()
public class TransactionsPostion {

        @Id
        private String address;
        private String startblock;
        private String endblock;

        public TransactionsPostion() {
                // TODO Auto-generated constructor stub
        }

        public String getAddress() {
                return address;
        }

        public void setAddress(String address) {
                this.address = address;
        }

        public String getStartblock() {
                return startblock;
        }
```

```
        public void setStartblock(String startblock) {
                this.startblock = startblock;
        }

        public String getEndblock() {
                return endblock;
        }

        public void setEndblock(String endblock) {
                this.endblock = endblock;
        }

}
```

对应数据库表

```
CREATE TABLE "transactions_postion" (
  "address" varchar(255) NOT NULL,
  "endblock" varchar(255) DEFAULT NULL,
  "startblock" varchar(255) DEFAULT NULL,
  PRIMARY KEY ("address")
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

@Column 定义字段：

```
unique          属性表示该字段是否为唯一标识，默认为false。如果表中有一个字段需要唯一标识，则既可以使用该标记，也可以使用@Table标记中的@UniqueConstraint。
nullable        属性表示该字段是否可以为null值，默认为true。
insertable      属性表示在使用“INSERT”脚本插入数据时，是否需要插入该字段的值。
updatable       属性表示在使用“UPDATE”脚本插入数据时，是否需要更新该字段的值。insertable和updatable属性一般多用于只读的属性，例如主键和外键等。这些字段的值通常是自动生成的。
```

columnDefinition属性表示创建表时，该字段创建的SQL语句，一般用于通过Entity生成表定义时使用。
table　　属性表示当映射多个表时，指定表的表中的字段。默认值为主表的表名。
length　　属性表示字段的长度，当字段的类型为varchar时，该属性才有效，默认为255个字符。
precision　　属性和scale属性表示精度，当字段类型为double时，precision表示数值的总长度，scale表示小数点所占的位数。
scale　　int　　列的精度，仅对十进制数值有效，表示小数位的总位数。默认为0。

字段长度

字段长度定义

```
@Column(name="name", length=80, nullable=true)
```

浮点型

```
        @Column(precision=18, scale=5)
    private BigDecimal principal;

    @Column(name="Price", columnDefinition="Decimal(10,2) default '100.00'")
```

创建于更新控制

```
        @Column(name = "ctime", nullable = false, insertable = false, updatable = false)
```

TEXT 类型

```
        private String subject;
        @Column(columnDefinition = "TEXT")
        private String content;
```

整形数据类型

无符号整形

```
package com.example.api.domain.elasticsearch;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class Member {
        @Id
        private int id;

        @Column(columnDefinition = "INT(10) UNSIGNED NOT
NULL")
        private int age;

        @Column(insertable = false, updatable = false,
columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
        private Date ctime;
```

```
        @Column(nullable = true, insertable = false,
updatable = false, columnDefinition = "TIMESTAMP NULL DEFAULT
NULL ON UPDATE CURRENT_TIMESTAMP")
        private Date mtime;

        @Column(columnDefinition = "enum('Y','N') DEFAULT
'N'")
        private boolean status;
}
```

```
CREATE TABLE `member` (
  `id` int(11) NOT NULL,
  `age` int(10) unsigned NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP,
  `status` enum('Y','N') DEFAULT 'N',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

非数据库字段

@Transient 该注解标注的字段不会被应射到数据库当中

@Lob 注解属性将被持久化为 Blog 或 Clob 类型

Clob（Character Large Ojects）类型是长字符串类型，具体的 java.sql.Clob, Character[], char[] 和 java.lang.String 将被持久化为 Clob 类型。
Blob（Binary Large Objects）类型是字节类型，具体的 java.sql.Blob, Byte[], byte[] 和 serializable type 将被持久化为 Blob 类

型。
@Lob 持久化为Blob或者Clob类型,根据get方法的返回值不同,自动进行Clob和Blob的转换。
因为这两种类型的数据一般占用的内存空间比较大，所以通常使用延迟加载的方式，与@Basic标记同时使用，设置加载方式为FetchType.LAZY。

```
@Lob
@Basic(fetch = FetchType.LAZY)
@Column(name="content", columnDefinition="CLOB",
nullable=true)
public String getContent() {
    return content;
}
```

@NotNull 不能为空声明

```
        @NotNull
        public String username;
```

@Temporal 日期定义

```
@Entity
public class Article {

    @Id
    @GeneratedValue
    Integer id;
```

```
    @Temporal(TemporalType.DATE)
    Date publicationDate;

    @Temporal(TemporalType.TIME)
    Date publicationTime;

    @Temporal(TemporalType.TIMESTAMP)
    Date creationDateTime;
}
```

创建日期

```
   @Column(name = "create_at")
   @CreatedDate
   private Timestamp create_date;
```

CreatedDate

Spring 提供了 import org.springframework.data.annotation.CreatedDate;

但是这些只能作用于实体类。

```
       @CreatedDate
   private Date createdDateTime;
```

与时间日期有关的 hibernate 注解

设置默认时间

```
@Column(insertable = false)
@org.hibernate.annotations.ColumnDefault("1.00")
@org.hibernate.annotations.Generated(
org.hibernate.annotations.GenerationTime.INSERT
)
protected Date lastModified;
```

创建时间

```
@Temporal(TemporalType.TIMESTAMP)
@Column(updatable = false)
@org.hibernate.annotations.CreationTimestamp
protected Date createdDate;
```

更新时间

```
@Column(name="update_time")
@org.hibernate.annotations.UpdateTimestamp
@Temporal(TemporalType.TIMESTAMP)
private Date updateTime;
```

```
@Temporal(TemporalType.TIMESTAMP)
@Column(insertable = false, updatable = false)
@org.hibernate.annotations.Generated(
org.hibernate.annotations.GenerationTime.ALWAYS
)
```

数据库级别的默认创建日期时间定义

```
package cn.netkiller.api.domain.elasticsearch;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class ElasticsearchTrash {
        @Id
        private int id;

        @Column(columnDefinition = "TIMESTAMP DEFAULT
CURRENT_TIMESTAMP")
        private Date ctime;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public Date getCtime() {
                return ctime;
        }

        public void setCtime(Date ctime) {
                this.ctime = ctime;
        }

}
```

```

```

对应数据库DDL

```
CREATE TABLE `elasticsearch_trash` (
  `id` int(11) NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

数据库级别的默认创建日期与更新时间定义

需求是这样的:

1. 创建时间与更新时间只能由数据库产生，不允许在实体类中产生，因为每个节点的时间/时区不一定一直。另外防止人为插入自定义时间时间。

2. 插入记录的时候创建默认时间，创建时间不能为空，时间一旦插入不允许日后在实体类中修改。

3. 记录创建后更新日志字段为默认为 null 表示该记录没有被修改过。一旦数据被修改，修改日期字段将记录下最后的修改时间。

4. 甚至你可以通过触发器实现一个history 表，用来记录数据的历史修改，详细请参考作者另一部电子书《Netkiller Architect 手札》数据库设计相关章节。

```
package cn.netkiller.api.domain.elasticsearch;

import java.util.Date;
```

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.Null;

@Entity
@Table
public class ElasticsearchTrash {
	@Id
	private int id;

	// 创建时间
	@Column(insertable = false, updatable = false,
columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
	private Date ctime;

	// 修改时间
	@Column(nullable = true, insertable = false,
updatable = false, columnDefinition = "TIMESTAMP NULL DEFAULT
NULL ON UPDATE CURRENT_TIMESTAMP")
	private Date mtime;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public Date getCtime() {
		return ctime;
	}

	public void setCtime(Date ctime) {
		this.ctime = ctime;
	}

	public Date getMtime() {
		return mtime;
	}

	public void setMtime(Date mtime) {
		this.mtime = mtime;
```

```
        }

}
```

对应数据库DDL

```
CREATE TABLE `elasticsearch_trash` (
  `id` int(11) NOT NULL,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

最后修改时间

需求：记录最后一次修改时间

```
package cn.netkiller.api.domain.elasticsearch;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class ElasticsearchTrash {
        @Id
        private int id;
```

```
        @Column(columnDefinition = "TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP")
        private Date lastModified;

}
```

产生DDL语句如下

```
CREATE TABLE `elasticsearch_trash` (
  `id` int(11) NOT NULL,
  `last_modified` timestamp NOT NULL DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

@DateTimeFormat 处理日期时间格式

```
        public java.sql.Date createdate; 创建日期 YYYY-MM-DD
        public java.util.Date finisheddate; 创建日期时间 YYYY-
MM-DD HH:MM:SS
```

Json默认为 yyyy-MM-ddTHH:mm:ss 注意日期与时间中间的T，修改日期格式将T去掉

```
      @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
      @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
      private Date createDate;
```

```
    /**
     * 日期 DATE  YYYY-MM-DD
     */
    @Column(name = "create_date")
    @JsonFormat(shape= JsonFormat.Shape.STRING,pattern="yyyy-
MM-dd HH:mm:ss",timezone="GMT+8")
    private Date date;
```

Enum 枚举数据类型

Enum 枚举数据类型 MySQL 特殊数据类型

实体中处理 enum 类型，存储字符串

@Enumerated(EnumType.STRING) 注解可以使其成功字符串类型。

```
        public enum StatisticsType {
                LIKE, COMMENT, BROWSE;
        }

        @Enumerated(EnumType.STRING)
        private StatisticsType type;
```

SQL

```
CREATE TABLE `statistics_history` (
```

```
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `member_id` bigint(20) NOT NULL,
  `statistics_id` bigint(20) NOT NULL,
  `type` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

实体中处理 **enum** 类型，存储序号

@Enumerated(value = EnumType.ORDINAL) //ORDINAL序数

在实体中处理枚举类型适用于所有数据库，Spring data 将枚举视为 String 类型。

```
package cn.netkiller.api.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "statistics_history")
public class StatisticsHistory implements Serializable {

        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id",unique = true, nullable = false,
insertable = true, updatable = false)
        private long id;
        private long memberId;
        private long statisticsId;
```

```
        public enum StatisticsType {
                LIKE, COMMENT, BROWSE;
        }

        private StatisticsType type;

        public Long getId() {
                return id;
        }

        public void setId(Long id) {
                this.id = id;
        }

        public long getMemberId() {
                return memberId;
        }

        public void setMemberId(long memberId) {
                this.memberId = memberId;
        }

        public long getStatisticsId() {
                return statisticsId;
        }

        public void setStatisticsId(long statisticsId) {
                this.statisticsId = statisticsId;
        }

        public StatisticsType getType() {
                return type;
        }

        public void setType(StatisticsType type) {
                this.type = type;
        }

}
```

默认 enum 类型创建数据库等效 int(11)

```
CREATE TABLE `statistics_history` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `member_id` bigint(20) NOT NULL,
  `statistics_id` bigint(20) NOT NULL,
  `type` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
SELECT * FROM test.statistics;
```

数据库枚举类型

在枚举中处理类型虽然可以适用于所有数据库，但有时我们希望适用数据库的枚举类型（例如MySQL），数据库中得枚举类型要比字符串效率更高

```
package cn.netkiller.api.domain.elasticsearch;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class NetkillerTrash {
        @Id
        private int id;

        @Column(columnDefinition = "enum('Y','N') DEFAULT
'N'")
        private boolean status;

        public int getId() {
                return id;
```

```
        }

        public void setId(int id) {
                this.id = id;
        }

        public boolean isStatus() {
                return status;
        }

        public void setStatus(boolean status) {
                this.status = status;
        }

}
```

实际对应的数据库DLL

```
CREATE TABLE `netkiller_trash` (
  `id` int(11) NOT NULL,
  `status` enum('Y','N') DEFAULT 'N',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

自定义枚举value属性

```
public enum Gender {
    MALE("男士"),
    FEMALE("女士");

    private final String value;

    private Gender(String value) {
        this.value = value;
```

```
    }

    // value 转枚举
    public static Gender fromValue(String value) {
        for (Gender gender : values()) {
            if (gender.toValue().equals(value)) {
                return gender;
            }
        }
        return null;
    }

    // 枚举转 value
    public String toValue() {
        return value;
    }
}
```

创建 Gender 的自定义转换器

```
// 实现 AttributeConverter 接口
java复制代码public class GenderConverter implements
AttributeConverter<Gender, String> {
    @Override
    public String convertToDatabaseColumn(Gender gender) {
        return gender.toValue();
    }

    @Override
    public Gender convertToEntityAttribute(String value) {
        return Gender.fromValue(value);
    }
}
```

在实体中，枚举字段加 @Convert 注解

```
@Convert(converter = GenderConverter.class)
@Column(name = "gender")
private Gender gender;
```

SET 数据结构

```
package common.domain;

import java.util.Date;
import java.util.Map;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Convert;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

import org.springframework.format.annotation.DateTimeFormat;
import com.fasterxml.jackson.annotation.JsonFormat;

import common.type.OptionConverter;

@Entity
public class ItemPool {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false,
insertable = false, updatable = false)
        public int id;

        @ManyToOne(cascade = { CascadeType.PERSIST,
CascadeType.REMOVE })
```

```
        @JoinColumn(name = "site_id", referencedColumnName =
"id")
        private Site site;

        public String question;

        @Column(columnDefinition = "json DEFAULT NULL")
        @Convert(converter = OptionConverter.class)
        public Map<String, String> options;

        @Column(columnDefinition =
"SET('A','B','C','D','E','F','G') DEFAULT NULL COMMENT '答
案'")
        public String answer;

        @ManyToOne(cascade = { CascadeType.PERSIST,
CascadeType.REMOVE })
        @JoinColumn(name = "category_id",
referencedColumnName = "id")
        private Category category;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "GMT+8")
        @Column(columnDefinition = "TIMESTAMP DEFAULT
CURRENT_TIMESTAMP COMMENT '创建时间'")
        public Date ctime;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "GMT+8")
        @Column(columnDefinition = "TIMESTAMP NULL DEFAULT
NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '更改时间'")
        public Date mtime;
}
```

定义 SET 如下，在JAVA中 SET被映射为逗号分隔的字符串（String），所以操作起来并无不同。使用字符串"A,B,C"存储即可，取出也同样是字符串。

```
@Column(columnDefinition = "SET('A','B','C','D','E','F','G')
DEFAULT NULL COMMENT '答案'")
```

接入后查看

```
mysql> select answer from item_pool;
+--------+
| answer |
+--------+
| A,B,C  |
+--------+
1 row in set (0.00 sec)
```

完美实现

JSON 数据类型

MySQL 5.7 中增加了 json 数据类型，下面是一个例子：

```
CREATE TABLE `test` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `your` json DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
```

我们需要在 Java 实体中定义 json 数据库结构，我搜索遍了整个互联网（Google,Bing,Baidu.....），没有找到解决方案，功夫不负有心

人，反复尝试后终于成功。记住我是第一个这样用的：）。

```
package common.domain;

import java.util.Date;
import java.util.Map;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Convert;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

import org.springframework.format.annotation.DateTimeFormat;
import com.fasterxml.jackson.annotation.JsonFormat;

import common.type.OptionConverter;

@Entity
public class ItemPool {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false,
insertable = false, updatable = false)
        public int id;

        @ManyToOne(cascade = { CascadeType.PERSIST,
CascadeType.REMOVE })
        @JoinColumn(name = "site_id", referencedColumnName =
"id")
        private Site site;

        public String name;

        @Column(columnDefinition = "json DEFAULT NULL")
        @Convert(converter = OptionConverter.class)
        public Map<String, String> options;
```

```
        @ManyToOne(cascade = { CascadeType.PERSIST,
CascadeType.REMOVE })
        @JoinColumn(name = "category_id",
referencedColumnName = "id")
        private Category category;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "GMT+8")
        @Column(columnDefinition = "TIMESTAMP DEFAULT
CURRENT_TIMESTAMP COMMENT '创建时间'")
        public Date ctime;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "GMT+8")
        @Column(columnDefinition = "TIMESTAMP NULL DEFAULT
NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '更改时间'")
        public Date mtime;
}
```

类型转换 Class

```
package common.type;

import java.util.Map;
import javax.persistence.AttributeConverter;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

public class OptionConverter implements
AttributeConverter<Map<String, String>, String> {

        Gson json = new Gson();

        @Override
        public String convertToDatabaseColumn(Map<String,
```

```
String> items) {
                return json.toJson(items, new
TypeToken<Map<String, String>>() {
                }.getType());
        }

        @Override
        public Map<String, String>
convertToEntityAttribute(String str) {
                return json.fromJson(str, new
TypeToken<Map<String, String>>() {
                }.getType());
        }
}
```

通过 @Column(columnDefinition = "json DEFAULT NULL") 定义数据库为 JSON 数据类型

数据存储与取出通过 @Convert(converter = OptionConverter.class) 做转换

这里我需要使用 Map 数据结构 public Map<String, String> options;， 你可以根据你的实际需要定义数据类型 Class

启动 Spring 项目后创建 Schema 如下:

```
CREATE TABLE `item_pool` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `ctime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
COMMENT '????',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP COMMENT '????',
  `name` varchar(255) DEFAULT NULL,
  `category_id` int(11) DEFAULT NULL,
  `site_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `FKgwuxedi20fxclobkk2po053hj` (`category_id`),
  KEY `FKiujumwssofow95st51ukklpgv` (`site_id`),
```

```
  CONSTRAINT `FKgwuxedi20fxclobkk2po053hj` FOREIGN KEY
(`category_id`) REFERENCES `category` (`id`),
  CONSTRAINT `FKiujumwssofow95st51ukklpgv` FOREIGN KEY
(`site_id`) REFERENCES `site` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8
```

我们做个简单的测试,创建仓库。

```
package common.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import common.domain.ItemPool;

@Repository
public interface ItemPoolRepository extends
CrudRepository<ItemPool, Integer> {

}
```

```
package cn.netkiller.api.restful;

import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
```

```
import org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;

import common.domain.ItemPool;
import common.repository.ItemPoolRepository;

@RestController
public class TestRestController {

        private static final Logger logger =
LoggerFactory.getLogger(TestRestController.class);
        @Autowired
        private ItemPoolRepository itemPoolRepository;

        @GetMapping("/test/json/data/type")
        public void jsonType() {

                ItemPool itemPool = new ItemPool();
                itemPool.name = "Which is Operstion System?";
                Map<String, String> opt = new
LinkedHashMap<String, String>();
                opt.put("A", "Linux");
                opt.put("B", "Java");
                itemPool.options = opt;
                itemPoolRepository.save(itemPool);

                itemPool = null;
                itemPool = itemPoolRepository.findOne(1);
                System.out.println(itemPool.toString());
        }

}
```

只能用完美来形容

```
mysql> select options from item_pool;
+-----------------------------+
| options                     |
+-----------------------------+
| {"A": "Linux", "B": "Java"} |
+-----------------------------+
1 row in set (0.00 sec)
```

索引

普通索引

```
@Table(indexes = { @Index(name = "name", columnList = "name
DESC"), @Index(name = "path", columnList = "path") })
```

```
package common.domain;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Index;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;
```

```
import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(indexes = { @Index(name = "name", columnList = "name
DESC"), @Index(name = "path", columnList = "path") })
public class Category {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false,
insertable = true, updatable = false)
        public int id;
        public String name;
        public String description;
        public String path;

        @Column(columnDefinition =
"enum('Enabled','Disabled') DEFAULT 'Enabled' COMMENT '状态'")
        public String status;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "GMT+8")
        @Column(columnDefinition = "TIMESTAMP DEFAULT
CURRENT_TIMESTAMP COMMENT '创建时间'")
        public Date ctime;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "GMT+8")
        @Column(columnDefinition = "TIMESTAMP NULL DEFAULT
NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '更改时间'")
        public Date mtime;

        @ManyToOne(cascade = { CascadeType.PERSIST,
CascadeType.REMOVE })
        @JoinColumn(name = "pid", referencedColumnName =
"id")
        private Category categorys;

        @JsonIgnore
        @OneToMany(cascade = CascadeType.ALL, mappedBy =
```

```
"category", fetch = FetchType.EAGER)
        private Set<Category> category;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getDescription() {
                return description;
        }

        public void setDescription(String description) {
                this.description = description;
        }

        public String getPath() {
                return path;
        }

        public void setPath(String path) {
                this.path = path;
        }

        public String getStatus() {
                return status;
        }

        public void setStatus(String status) {
                this.status = status;
        }

        public Date getCtime() {
                return ctime;
```

```
	}

	public void setCtime(Date ctime) {
		this.ctime = ctime;
	}

	public Date getMtime() {
		return mtime;
	}

	public void setMtime(Date mtime) {
		this.mtime = mtime;
	}

	public Category getCategorys() {
		return categorys;
	}

	public void setCategorys(Category categorys) {
		this.categorys = categorys;
	}

	public Set<Category> getCategory() {
		return category;
	}

	public void setCategory(Set<Category> category) {
		this.category = category;
	}

	@Override
	public String toString() {
		return "Category [id=" + id + ", name=" +
name + ", description=" + description + ", path=" + path + ",
status="
				+ status + ", ctime=" + ctime
+ ", mtime=" + mtime + ", categorys=" + categorys + ",
category="
				+ category + "]";
	}

}
```

唯一索引

针对字段做唯一索引

```
@Column(unique = true)
```

复合索引

创建由多个字段组成的复合索引

```
package cn.netkiller.api.model;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.UniqueConstraint;

import com.fasterxml.jackson.annotation.JsonFormat;

@Entity
@Table(name = "comment", uniqueConstraints = {
@UniqueConstraint(columnNames = { "member_id", "articleId" })
```

```
})
public class Comment implements Serializable {
	/**
	 *
	 */
	private static final long serialVersionUID =
-1484408775034277681L;
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	@Column(name = "id", unique = true, nullable = false,
insertable = true, updatable = false)
	private int id;

	@ManyToOne(cascade = { CascadeType.ALL })
	@JoinColumn(name = "member_id")
	private Member member;

	private int articleId;

	private String message;

	@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
	@Temporal(TemporalType.TIMESTAMP)
	@Column(updatable = false)
	@org.hibernate.annotations.CreationTimestamp
	protected Date createDate;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public Member getMember() {
		return member;
	}

	public void setMember(Member member) {
		this.member = member;
	}

	public int getArticleId() {
		return articleId;
```

```
        }

        public void setArticleId(int articleId) {
                this.articleId = articleId;
        }

        public String getMessage() {
                return message;
        }

        public void setMessage(String message) {
                this.message = message;
        }

        public Date getCreateDate() {
                return createDate;
        }

        public void setCreateDate(Date createDate) {
                this.createDate = createDate;
        }
}
```

```
CREATE TABLE `comment` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `article_id` int(11) NOT NULL,
  `create_date` datetime DEFAULT NULL,
  `message` varchar(255) DEFAULT NULL,
  `member_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK5qxfiu92nwlvgli7bl3evl11m`
(`member_id`,`article_id`),
  CONSTRAINT `FKmrrrpi513ssu63i2783jyiv9m` FOREIGN KEY
(`member_id`) REFERENCES `member` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

嵌入

@Embeddable / @Embedded

```
package cn.aigcsst.domain.demo;

import jakarta.persistence.Embeddable;

@Embeddable
public class Address {

    private String city;
    private String district;
    private String street;
    private String community;
}
```

```
package cn.aigcsst.domain.demo;

import jakarta.persistence.*;

@Entity
public class Company {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;

    @Embedded
    private Address address;
}
```

```
CREATE TABLE `company` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `city` varchar(255) DEFAULT NULL,
  `community` varchar(255) DEFAULT NULL,
  `district` varchar(255) DEFAULT NULL,
  `street` varchar(255) DEFAULT NULL,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

@AttributeOverrides 定义字段名称

```
@Data
@Embeddable
public class Address {
    private String province;
    private String city;
    private String street;
}
```

```
@Data
@Entity
@Table
public class Company {
    @Id
    @GeneratedValue
    private Long id;

    @Column
    private String name;

    // 公司地址
    private Address address;
```

```
    // 注册地址
    @AttributeOverrides({
        @AttributeOverride(name = "city", column =
@Column(name= "location_city")),
        @AttributeOverride(name = "province",
column=@Column(name="location_province")),
        @AttributeOverride(name = "street", column =
@Column(name="location_street"))
    })
    private Address locationAddress;

}
```

创建复合主键

定义实体

```
package cn.netkiller.wallet.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Embeddable;
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;

@Entity
public class UserToken {
        @EmbeddedId
        @Column(unique = true, nullable = false, insertable =
true, updatable = false)
        private UserTokenPrimaryKey primaryKey;

        private String name;
        private String symbol;
        private int decimals;

        public UserToken() {
```

```
                // TODO Auto-generated constructor stub
        }

        public UserTokenPrimaryKey getPrimaryKey() {
                return primaryKey;
        }

        public void setPrimaryKey(UserTokenPrimaryKey
primaryKey) {
                this.primaryKey = primaryKey;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getSymbol() {
                return symbol;
        }

        public void setSymbol(String symbol) {
                this.symbol = symbol;
        }

        public int getDecimals() {
                return decimals;
        }

        public void setDecimals(int decimals) {
                this.decimals = decimals;
        }

        @Override
        public String toString() {
                return "UserToken [primaryKey=" + primaryKey
+ ", name=" + name + ", symbol=" + symbol + ", decimals=" +
decimals + "]";
        }

        @Embeddable
        public static class UserTokenPrimaryKey implements
```

```
Serializable {

		private static final long serialVersionUID =
1242827922377178368L;
		private String address;
		private String contractAddress;

		public UserTokenPrimaryKey() {
		}

		public UserTokenPrimaryKey(String address,
String contractAddress) {
			this.address = address;
			this.contractAddress =
contractAddress;
		}

		public String getAddress() {
			return address;
		}

		public void setAddress(String address) {
			this.address = address;
		}

		public String getContractAddress() {
			return contractAddress;
		}

		public void setContractAddress(String
contractAddress) {
			this.contractAddress =
contractAddress;
		}

		@Override
		public String toString() {
			return "UserTokenPrimaryKey
[address=" + address + ", contractAddress=" + contractAddress
+ "]";
		}

	}

}
```

实际效果

```
CREATE TABLE "user_has_token" (
  "address" varchar(255) NOT NULL,
  "contract_address" varchar(255) NOT NULL,
  "decimals" int(11) NOT NULL,
  "name" varchar(255) DEFAULT NULL,
  "symbol" varchar(255) DEFAULT NULL,
  PRIMARY KEY ("address","contract_address")
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

```
package cn.netkiller.wallet.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import cn.netkiller.wallet.domain.UserToken;
import
cn.netkiller.wallet.domain.UserToken.UserTokenPrimaryKey;;

public interface UserTokenRepository extends
JpaRepository<UserToken, UserTokenPrimaryKey> {

        UserToken findOneByPrimaryKey(UserTokenPrimaryKey
primaryKey);

        @Query("select ut from UserToken ut where
ut.primaryKey.address=:address")
        List<UserToken> getByAddress(@Param("address") String
address);
```

```
        @Query("select ut from UserToken ut where
ut.primaryKey.address=:address and
ut.primaryKey.contractAddress=:contractAddress")
        List<UserToken> findByPrimaryKey(@Param("address")
String address, @Param("contractAddress") String
contractAddress);
}
```

外键

@JoinColumn

@JoinColumn与@Column注释类似，它的定义如下代码所示。

```
@Target({METHOD, FIELD}) @Retention(RUNTIME)

public @interface JoinColumn {

String name() default "";

String referencedColumnName() default "";

boolean unique() default false;

boolean nullable() default true;

boolean insertable() default true;

boolean updatable() default true;

String columnDefinition() default "";

String table() default "";

}
```

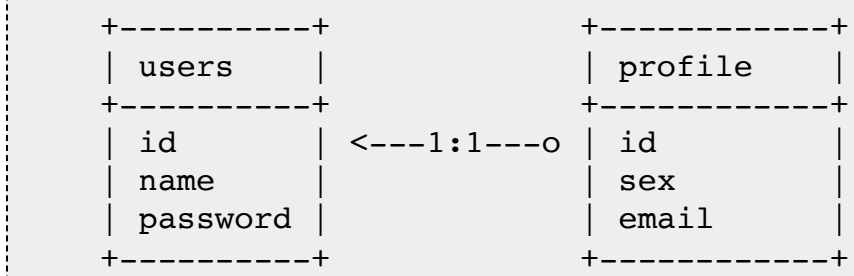
定义外键名称 @ForeignKey(name = "picture_id")

```
    @Id
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval =
true)
    @JoinColumn(name = "id", foreignKey = @ForeignKey(name =
"picture_id"))
    private Picture picture;
```

```
CREATE TABLE `picture_psychoanalysis` (
  `analysis` text COMMENT '心里分析',
  `ctime` timestamp NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创
建时间',
  `emotion` varchar(255) DEFAULT NULL COMMENT '感谢|愉快|抱怨|愤
怒|喜爱|厌恶|恐惧|悲伤',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP COMMENT '修改时间',
  `replies` text COMMENT '建议回复话术',
  `sentiment` varchar(255) DEFAULT NULL COMMENT '负向情绪|中性情
绪|正向情绪',
  `id` bigint unsigned NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `picture_id` FOREIGN KEY (`id`) REFERENCES
`picture` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='文生图心里分析'
```

@OneToOne

一对一表结构，如下面ER图所示，users表是用户表里面有登陆信息，profile 保存的时死人信息，这样的目的是我们尽量减少users表的字段，在频繁操作该表的时候性能比较好，另外一个目的是为了横向水平扩展。




```
package cn.netkiller.api.domain.test;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "users")
public class Users implements Serializable {
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private int id;
        private String name;
        private String password;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
```

```
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getPassword() {
                return password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

        @Override
        public String toString() {
                return "Users [id=" + id + ", name=" + name +
", password=" + password + "]";
        }
}
```

```
package cn.netkiller.api.domain.test;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "profile")
public class Profile implements Serializable {
        /**
         *
         */
        private static final long serialVersionUID =
-2500499458196257167L;
```

```
	@Id
	@OneToOne
	@JoinColumn(name = "id")
	private Users users;

	private int age;
	private String sex;
	private String email;

	public Users getUsers() {
		return users;
	}

	public void setUsers(Users users) {
		this.users = users;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public String getSex() {
		return sex;
	}

	public void setSex(String sex) {
		this.sex = sex;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	@Override
	public String toString() {
		return "Profile [users=" + users + ", age=" +
age + ", sex=" + sex + ", email=" + email + "]";
```

```
        }

}
```

```
CREATE TABLE `users` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(255) NULL DEFAULT NULL,
        `password` VARCHAR(255) NULL DEFAULT NULL,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

CREATE TABLE `profile` (
        `age` INT(11) NOT NULL,
        `email` VARCHAR(255) NULL DEFAULT NULL,
        `sex` VARCHAR(255) NULL DEFAULT NULL,
        `id` INT(11) NOT NULL,
        PRIMARY KEY (`id`),
        CONSTRAINT `FK6x079ilawxjrfsljwyyi5ujjq` FOREIGN KEY
(`id`) REFERENCES `users` (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

如果第二张表关联的并非主表的PK（主键）需要使用 referencedColumnName 指定。

```
@JoinColumn(name =
"member_id",referencedColumnName="member_id")
```

案例一

```
package cn.netkiller.domain.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;

}
```

```
package cn.netkiller.domain.demo;

import jakarta.persistence.*;

@Entity
public class BookDetail {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private long numberOfPages;

    @OneToOne
    private Book book;
}
```

```
CREATE TABLE `book` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `book_detail` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `number_of_pages` bigint NOT NULL,
  `book_id` bigint DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK_29qtqq9pgixv8kqlt0woj1hyp` (`book_id`),
  CONSTRAINT `FKl1hmgccsvfwcxhem3qw6l7gpm` FOREIGN KEY
(`book_id`) REFERENCES `book` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

```
package cn.netkiller.domain.demo;

import jakarta.persistence.*;

@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "book_detail")
    private BookDetail bookDetail;
}
```

```
package cn.netkiller.domain.demo;

import jakarta.persistence.*;

@Entity
public class BookDetail {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;


    private long numberOfPages;


    @OneToOne(cascade = CascadeType.ALL, mappedBy =
"bookDetail")
    private Book book;
}
```

```
CREATE TABLE `book` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `book_detail` bigint DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK_d8im4vqhlm2eo0mj9lwjvib94` (`book_detail`),
  CONSTRAINT `FKagqqxsh6783b9dd9197ow49a5` FOREIGN KEY
(`book_detail`) REFERENCES `book_detail` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `book_detail` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `number_of_pages` bigint NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

```
package cn.netkiller.domain.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Users {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;

}
```

```
package cn.netkiller.domain.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.OneToOne;

@Entity
public class Profile {
    @Id
    @OneToOne
    private Users users;

    private int age;


}
```

```
CREATE TABLE `users` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `profile` (
  `age` int NOT NULL,
  `users_id` bigint NOT NULL,
  PRIMARY KEY (`users_id`),
  CONSTRAINT `FKi6d1noonlpe6oyk6pnwc1q49e` FOREIGN KEY
(`users_id`) REFERENCES `users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

指定一对一字段名，默认是表名+PK，例如上面的例子 users_id，如果我们希望自定义字段名，可以使用 @JoinColumn(name = "id")

```
package cn.netkiller.domain.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.OneToOne;
import lombok.Data;

@Entity
@Data
public class Profile {
    @Id
    @OneToOne
    @JoinColumn(name = "id")
```

```
    private Users users;

    private int age;
    private boolean sex;
}
```

效果展示

```
CREATE TABLE `profile` (
  `age` int NOT NULL,
  `sex` bit(1) NOT NULL,
  `id` bigint NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `FK6x079ilawxjrfsljwyyi5ujjq` FOREIGN KEY (`id`)
REFERENCES `users` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

does not define an IdClass

如果一对一的关系中，我们希望两端都是用 id 字段，而 @OneToOne 一端是对象，必须定义一个 @Id，这时加入 @MapsId 可以解决 does not define an IdClass 错误，这时一段表中没有 @Id

```
package cn.netkiller.domain.demo;

import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.Comment;
import org.hibernate.annotations.DynamicInsert;
import org.hibernate.annotations.DynamicUpdate;
```

```
@Entity
@Table
@Data
@DynamicInsert
@DynamicUpdate
@Comment("用户表")
public class Users {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String username;
    private String password;
    private boolean status;

}
```

```
package cn.netkiller.domain.demo;

import jakarta.persistence.Table;
import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.*;

@Entity
@Table
@Data
@DynamicInsert
@DynamicUpdate
@Comment("用户信息表")
public class Profile {

    @Id
    @Column(name = "id")
    private Long id;

    @MapsId
    @OneToOne
    @JoinColumn(name = "id")
    @OnDelete(action = OnDeleteAction.CASCADE)
```

```
    private Users users;

    private int age;
    private boolean sex;
}
```

```
package cn.netkiller.repository.demo;

import cn.netkiller.domain.demo.Profile;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface TestRepository extends
JpaRepository<Profile, Long> {
}
```

```
CREATE TABLE `users` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `password` varchar(255) DEFAULT NULL,
  `status` bit(1) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='用户表'

CREATE TABLE `profile` (
  `id` bigint NOT NULL,
  `age` int NOT NULL,
  `sex` bit(1) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `FK6x079ilawxjrfsljwyyi5ujjq` FOREIGN KEY (`id`)
REFERENCES `users` (`id`) ON DELETE CASCADE
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='用户信息表'
```

共享主键

```
package cn.netkiller.domain.demo;

import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.Comment;
import org.hibernate.annotations.DynamicInsert;
import org.hibernate.annotations.DynamicUpdate;

@Entity
@Table
@Data
@DynamicInsert
@DynamicUpdate
@Comment("用户表")
public class Users {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String username;
    private String password;
    private boolean status;

    @OneToOne(mappedBy = "users", cascade = CascadeType.ALL)
    @PrimaryKeyJoinColumn
    private Profile profile;
}
```

```
package cn.netkiller.domain.demo;
```

```
import jakarta.persistence.Table;
import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.*;

@Entity
@Table
@Data
@DynamicInsert
@DynamicUpdate
@Comment("用户信息表")
public class Profile {

    @Id
    @Column(name = "id")
    private Long id;

    @MapsId
    @OneToOne
    @JoinColumn(name = "id")
    @OnDelete(action = OnDeleteAction.CASCADE)
    private Users users;

    private int age;
    private boolean sex;
}
```

```
CREATE TABLE `users` (
  `id` bigint NOT NULL AUTO_INCREMENT,
  `password` varchar(255) DEFAULT NULL,
  `status` bit(1) NOT NULL,
  `username` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='用户表'

CREATE TABLE `profile` (
  `id` bigint NOT NULL,
```

```
  `age` int NOT NULL,
  `sex` bit(1) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `FK6x079ilawxjrfsljwyyi5ujjq` FOREIGN KEY (`id`)
REFERENCES `users` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='用户信息表'
```

null identifier

@OneToOne 保存提示 null identifier，经过排查需要配置

```
package cn.aigcsst.domain;

import jakarta.persistence.CascadeType;
import jakarta.persistence.ForeignKey;
import jakarta.persistence.Table;
import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.*;

import java.io.Serial;
import java.io.Serializable;
import java.util.Date;

@Entity
@Table
@DynamicUpdate
@DynamicInsert
@Data
@Comment("文生图心里分析")

public class PicturePsychoanalysis implements Serializable {
    @Serial
    public static final long serialVersionUID = 1L;

    @Comment("Robert Plutchik 情感轮盘")
    @Column(columnDefinition = "json")
    public String plutchik;
```

```
    @Id
    @Column(name = "id")
    @Comment("Picture Id 一对一关系")
    private Long id;
    @MapsId
    @OneToOne(cascade = CascadeType.MERGE)
    @JoinColumn(name = "id", insertable = true, updatable =
false, columnDefinition = "bigint unsigned", foreignKey =
@ForeignKey(name = "picture_id"))
    @OnDelete(action = OnDeleteAction.CASCADE)
    private Picture picture;
    @Comment("负向情绪|中性情绪|正向情绪")
    private String sentiment;
    @Comment("感谢|愉快|抱怨|愤怒|喜爱|厌恶|恐惧|悲伤")
    private String emotion;
    @Lob
    @Basic(fetch = FetchType.LAZY)
    @Column(nullable = true, columnDefinition = "text")
    @Comment("建议回复话术")
    private String replies;
    @Lob
    @Basic(fetch = FetchType.LAZY)
    @Column(nullable = true, columnDefinition = "text")
    @Comment("心里分析")
    private String analysis;
    //    public JSONObject plutchik;
    @Column(insertable = false, updatable = false,
columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
    @Comment("创建时间")
    private Date ctime;

    @Column(nullable = true, insertable = false, updatable =
false, columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON
UPDATE CURRENT_TIMESTAMP")
    @Comment("修改时间")
    private Date mtime;
}
```

讲 @OneToOne 增加 @OneToOne(cascade = CascadeType.MERGE) 参数，如果 CascadeType.ALL 需要改为 CascadeType.MERGE

```
        @Id
    @Column(name = "id")
    @Comment("Picture Id 一对一关系")
    private Long id;
    @MapsId
    @OneToOne(cascade = CascadeType.MERGE)
    @JoinColumn(name = "id", insertable = true, updatable =
false, columnDefinition = "bigint unsigned", foreignKey =
@ForeignKey(name = "picture_id"))
    @OnDelete(action = OnDeleteAction.CASCADE)
    private Picture picture;
```

OneToMany 一对多

我们要实现一个一对多实体关系，ER 图如下

```
    +----------+          +------------+
    | Classes  |          | Student    |
    +----------+          +------------+
    | id       | <---+    | id         |
    | name     |     |    | name       |
    +----------+     +--o | classes_id |
                          +------------+
```

classes 表需要 OneToMany 注解，Student 表需要 ManyToOne 注解，这样就建立起了表与表之间的关系

```
package cn.netkiller.api.domain.test;

import java.io.Serializable;
```

```
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name="classes")
public class Classes implements Serializable{
        /**
         *
         */
        private static final long serialVersionUID =
-5422905745519948312L;
        @Id
        @GeneratedValue(strategy=GenerationType.AUTO)
        private int id;
        private String name;


@OneToMany(cascade=CascadeType.ALL,mappedBy="classes")
        private Set<Student> students;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public Set<Student> getStudents() {
                return students;
```

```
        }

        public void setStudents(Set<Student> students) {
                this.students = students;
        }

        @Override
        public String toString() {
                return "classes [id=" + id + ", name=" + name
+ ", students=" + students + "]";
        }
}
```

```
package cn.netkiller.api.domain.test;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "student")
public class Student implements Serializable{
        /**
         *
         */
        private static final long serialVersionUID =
6737037465677800326L;
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private int id;
        private String name;

        // 若有多个cascade，可以是：
```

```
{CascadeType.PERSIST,CascadeType.MERGE}
        @ManyToOne(cascade = { CascadeType.ALL })
        @JoinColumn(name = "classes_id")
        private Classes classes;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public Classes getClasses() {
                return classes;
        }

        public void setClasses(Classes classes) {
                this.classes = classes;
        }

        @Override
        public String toString() {
                return "Student [id=" + id + ", name=" + name
+ ", classes=" + classes + "]";
        }

}
```

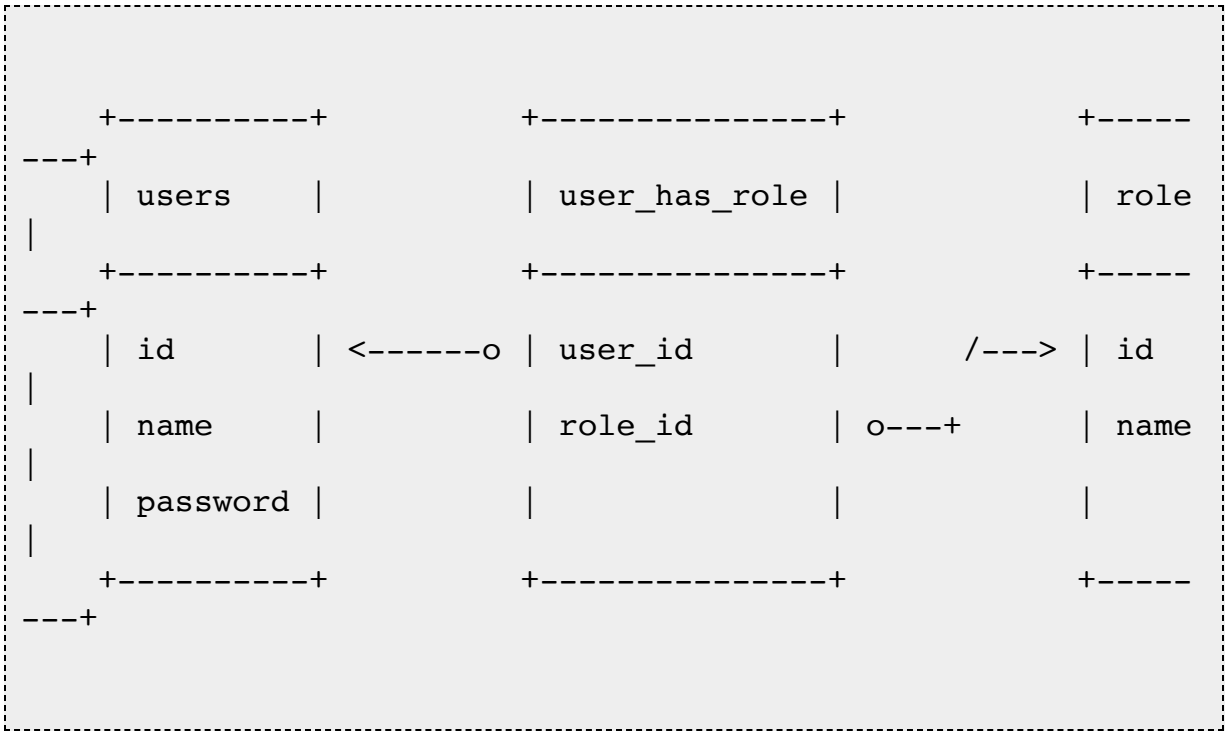
最终 SQL 表如下

```
CREATE TABLE `classes` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(255) NULL DEFAULT NULL,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

CREATE TABLE `student` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(255) NULL DEFAULT NULL,
        `class_id` INT(11) NULL DEFAULT NULL,
        PRIMARY KEY (`id`),
        INDEX `FKnsl7w2nw6o6eq53hqlxfcijpm` (`class_id`),
        CONSTRAINT `FKnsl7w2nw6o6eq53hqlxfcijpm` FOREIGN KEY
(`class_id`) REFERENCES `classes` (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

```
      Classes classes=new Classes();
      classes.setName("One");

      Student st1=new Student();
      st1.setSname("jason");
      st1.setClasses(classes);
      studentRepostitory.save(st1);

      Student st2=new Student();
      st2.setSname("neo");
      st2.setClasses(classes);
      studentRepostitory.save(st2);
```

ManyToMany 多对多

用户与角色就是一个多对多的关系，多对多是需要中间表做关联的。所以我方需要一个 user_has_role 表。



创建 User 表

```
package cn.netkiller.api.domain.test;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.JoinColumn;

@Entity
```

```
@Table(name = "users")
public class Users implements Serializable {
	/**
	 *
	 */
	private static final long serialVersionUID =
-2480194112597046349L;
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int id;
	private String name;
	private String password;

	@ManyToMany(fetch = FetchType.EAGER)
	@JoinTable(name = "user_has_role", joinColumns = {
@JoinColumn(name = "user_id", referencedColumnName = "id") },
inverseJoinColumns = { @JoinColumn(name = "role_id",
referencedColumnName = "id") })
	private Set<Roles> roles;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public Set<Roles> getRoles() {
```

```
                return roles;
        }

        public void setRoles(Set<Roles> roles) {
                this.roles = roles;
        }

        @Override
        public String toString() {
                return "Users [id=" + id + ", name=" + name +
", password=" + password + ", roles=" + roles + "]";
        }

}
```

创建 Role 表

```
package cn.netkiller.api.domain.test;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "roles")
public class Roles implements Serializable {
        private static final long serialVersionUID =
67370374656778003 26L;
        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private int id;
        private String name;
        @ManyToMany(mappedBy = "roles")
```

```
        private Set<Users> users;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public Set<Users> getUsers() {
                return users;
        }

        public void setUsers(Set<Users> users) {
                this.users = users;
        }

        @Override
        public String toString() {
                return "Roles [id=" + id + ", name=" + name +
", users=" + users + "]";
        }

}
```

最终产生数据库表如下

```
CREATE TABLE `users` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
```

```
        `name` VARCHAR(255) NULL DEFAULT NULL,
        `password` VARCHAR(255) NULL DEFAULT NULL,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

CREATE TABLE `roles` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(255) NULL DEFAULT NULL,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

CREATE TABLE `user_has_role` (
        `user_id` INT(11) NOT NULL,
        `role_id` INT(11) NOT NULL,
        PRIMARY KEY (`user_id`, `role_id`),
        INDEX `FKsvvq61v3koh04fycopbjx72hj` (`role_id`),
        CONSTRAINT `FK2dl1ftxlkldulcp934i3125qo` FOREIGN KEY
(`user_id`) REFERENCES `users` (`id`),
        CONSTRAINT `FKsvvq61v3koh04fycopbjx72hj` FOREIGN KEY
(`role_id`) REFERENCES `roles` (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

```
```

外键级联操作

cascade 属性： 指定级联操作的行为(可多选)

```
CascadeType.PERSIST：级联新增（又称级联保存）：对A对象保存时也会对B对
```

象进行保存。并且，只有A类新增时，会级联B对象新增。若B对象在数据库存在则抛异常。对应EntityManager的presist方法。
CascadeType.MERGE：级联合并（级联更新）：指A类新增或者变化，会级联B对象（新增或者变化）。对应EntityManager的merge方法。
CascadeType.REMOVE：级联删除：只有A类删除时，会级联删除B类,即在设置的那一端进行删除时，另一端才会级联删除。对应EntityManager的remove方法。
CascadeType.REFRESH：级联刷新：获取A对象时也重新获取最新的B对象。对EntityManager的refresh(object)方法。即会重新查询数据库里的最新数据（用的比较少）
CascadeType.DETACH：级联分离。
CascadeType.ALL：级联所有操作。

CascadeType.PERSIST

```
@OneToMany(mappedBy = "boss", cascade = CascadeType.PERSIST)
private List<Staff> staffList;
```

CascadeType.REMOVE

```
@OneToMany(mappedBy = "boss", cascade = CascadeType.REMOVE)
private List<Staff> staffList;
```

外键级联删除

orphanRemoval 是 JPA 定义，并不是数据库原生

```
    @Id
    @OneToOne(cascade = CascadeType.ALL, orphanRemoval =
```

```
true)
    @JoinColumn(name = "id", foreignKey = @ForeignKey(name =
"picture_id"))
    private Picture picture;
```

orphanRemoval = true 可以实现数据级联删除

```
package cn.netkiller.api.domain;

import java.io.Serializable;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
@Table(name = "member")
public class Member implements Serializable {
        /**
         *
         */
        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "id", unique = true, nullable = false,
insertable = true, updatable = false)
        private int id;

        private String name;
        private String sex;
```

```
        private int age;
        private String wechat;

        @Column(unique = true)
        private String mobile;
        private String picture;
        private String ipAddress;

        @JsonIgnore
        @OneToMany(cascade = CascadeType.ALL, orphanRemoval =
true, mappedBy = "member")
        private Set<Comment> comment;
        @JsonIgnore
        @OneToMany(cascade = CascadeType.ALL, orphanRemoval =
true, mappedBy = "member")
        private Set<StatisticsHistory> statisticsHistory;

        public Member() {
        }

        public Member(int id) {
                this.id = id;
        }

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public String getSex() {
                return sex;
        }

        public void setSex(String sex) {
```

```
                this.sex = sex;
        }

        public int getAge() {
                return age;
        }

        public void setAge(int age) {
                this.age = age;
        }

        public String getWechat() {
                return wechat;
        }

        public void setWechat(String wechat) {
                this.wechat = wechat;
        }

        public String getMobile() {
                return mobile;
        }

        public void setMobile(String mobile) {
                this.mobile = mobile;
        }

        public String getPicture() {
                return picture;
        }

        public void setPicture(String picture) {
                this.picture = picture;
        }

        public String getIpAddress() {
                return ipAddress;
        }

        public void setIpAddress(String ipAddress) {
                this.ipAddress = ipAddress;
        }

        @Override
        public String toString() {
```

```
                return "Member [id=" + id + ", name=" + name
+ ", sex=" + sex + ", age=" + age + ", wechat=" + wechat + ",
mobile=" + mobile + ", picture=" + picture + ", ipAddress=" +
ipAddress + "]";
        }

}
```

MySQL ON DELETE CASCADE

@OnDelete(action = OnDeleteAction.CASCADE)

```
package cn.netkiller.domain;

import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.Comment;
import org.hibernate.annotations.DynamicUpdate;
import org.hibernate.annotations.OnDelete;
import org.hibernate.annotations.OnDeleteAction;

import java.io.Serial;
import java.io.Serializable;
import java.util.Date;

@Entity
@Table
@DynamicUpdate
@Data
@Comment("文生图心里分析")
public class PicturePsychoanalysis implements Serializable {
    @Serial
    public static final long serialVersionUID = 1L;

    @Id
    @OneToOne()
    @JoinColumn(name = "id", foreignKey = @ForeignKey(name =
"picture_id"))
    @OnDelete(action = OnDeleteAction.CASCADE)
```

```
    private Picture picture;

//    @Column(unique = true, nullable = false, insertable =
true, updatable = false)
//    @Comment("会话")
//    private String session;

    @Comment("负向情绪|中性情绪|正向情绪")
    private String sentiment;

    @Comment("感谢|愉快|抱怨|愤怒|喜爱|厌恶|恐惧|悲伤")
    private String emotion;

    @Lob
    @Basic(fetch = FetchType.LAZY)
    @Column(nullable = true, columnDefinition = "text")
    @Comment("建议回复话术")
    private String replies;

    @Lob
    @Basic(fetch = FetchType.LAZY)
    @Column(nullable = true, columnDefinition = "text")
    @Comment("心里分析")
    private String analysis;

    @Column(insertable = false, updatable = false,
columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
    @Comment("创建时间")
    private Date ctime;

    @Column(nullable = true, insertable = false, updatable =
false, columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON
UPDATE CURRENT_TIMESTAMP")
    @Comment("修改时间")
    private Date mtime;
}
```

```
CREATE TABLE `picture_psychoanalysis` (
  `analysis` text COMMENT '心里分析',
```

```
  `ctime` timestamp NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创
建时间',
  `emotion` varchar(255) DEFAULT NULL COMMENT '感谢|愉快|抱怨|愤
怒|喜爱|厌恶|恐惧|悲伤',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP COMMENT '修改时间',
  `replies` text COMMENT '建议回复话术',
  `sentiment` varchar(255) DEFAULT NULL COMMENT '负向情绪|中性情
绪|正向情绪',
  `id` bigint unsigned NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `picture_id` FOREIGN KEY (`id`) REFERENCES
`picture` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='文生图心里分析'
```

@JoinTable

```
@JoinTable(name = "table")
```

```
@OneToMany(cascade = CascadeType.ALL)
@JoinTable
private List<UserProfile> userProfile;
```

```
@JoinTable(name = "cust_user",
    joinColumns = @JoinColumn(name = "user_id",
referencedColumnName = "id")
)
```

```
@JoinTable(name = "cust_user",
    inverseJoinColumns = @JoinColumn(name = "user_ext_id",
referencedColumnName = "id")
)
```

```
@JoinTable(name = "cust_user",
    joinColumns = @JoinColumn(name = "user_id",
referencedColumnName = "id"),
    inverseJoinColumns = @JoinColumn(name = "user_ext_id",
referencedColumnName = "id"),
    uniqueConstraints = {
        @UniqueConstraint(name = "unique_user_id",
columnNames = {"user_id"}),
        @UniqueConstraint(name = "unique_user_ext_id",
columnNames = {"user_ext_id"})
    }
)
```

多对多实例

```
package cn.netkiller.domain;

import jakarta.persistence.*;
import lombok.Data;

import java.io.Serializable;

@Entity
@Table
@Data
public class Consumer implements Serializable {
    public static final long serialVersionUID =
```

```
79989034212655388O1L;
    public String firstName;
    public String lastName;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", unique = true, nullable = false,
insertable = false, updatable = false, columnDefinition =
"int unsigned")
    public Integer id;

    @OneToOne(cascade = CascadeType.ALL, orphanRemoval =
true)
    @JoinTable(name = "consumer_has_device",
            joinColumns =
                    {@JoinColumn(name = "consumer_id",
referencedColumnName = "id")},
            inverseJoinColumns =
                    {@JoinColumn(name = "device_id",
referencedColumnName = "id")})
    private Device device;

    public Consumer() {
    }

}
```

```
package cn.netkiller.domain;

import jakarta.persistence.*;
import lombok.Data;
import org.hibernate.annotations.Comment;
import org.hibernate.annotations.DynamicUpdate;

import java.io.Serial;
import java.io.Serializable;
import java.util.Date;

@Entity
@Table
```

```
@DynamicUpdate
@Data
@Comment("设备表")
public class Device implements Serializable {
    @Serial
    public static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", unique = true, nullable = false,
insertable = false, updatable = false, columnDefinition =
"int unsigned")
    @Comment("主键")
    private Integer id;

    @Comment("设备名称")
    private String name;

    @Comment("型号")
    private String model;
    @Comment("版本")
    private String fireware;
    @Comment("版本")
    private String version;

    @Column(unique = true, nullable = false, insertable =
true, updatable = false)
    @Comment("序列号")
    private String sn;
    @Comment("ip")
    private String ip;

    @Comment("mac")
    private String mac;

    @Temporal(TemporalType.TIMESTAMP)
    @Comment("最后一次登陆时间")
    private Date lastTime;

    @Column(columnDefinition = "enum('Y','N') DEFAULT 'N'")
    @Comment("设备状态")
    private boolean status;

    @Column(insertable = false, updatable = false,
columnDefinition = "TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
```

```
    @Comment("创建时间")
    private Date ctime;

    @Column(nullable = true, insertable = false, updatable =
false, columnDefinition = "TIMESTAMP NULL DEFAULT NULL ON
UPDATE CURRENT_TIMESTAMP")
    @Comment("修改时间")
    private Date mtime;
}
```

```
CREATE TABLE `consumer` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(255) DEFAULT NULL,
  `last_name` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci

CREATE TABLE `device` (
  `id` int unsigned NOT NULL AUTO_INCREMENT COMMENT '主键',
  `ctime` timestamp NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创
建时间',
  `fireware` varchar(255) DEFAULT NULL COMMENT '版本',
  `ip` varchar(255) DEFAULT NULL COMMENT 'ip',
  `last_time` datetime(6) DEFAULT NULL COMMENT '最后一次登陆时
间',
  `mac` varchar(255) DEFAULT NULL COMMENT 'mac',
  `model` varchar(255) DEFAULT NULL COMMENT '型号',
  `mtime` timestamp NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP COMMENT '修改时间',
  `name` varchar(255) DEFAULT NULL COMMENT '设备名称',
  `sn` varchar(255) NOT NULL COMMENT '序列号',
  `status` enum('Y','N') DEFAULT 'N' COMMENT '设备状态',
  `version` varchar(255) DEFAULT NULL COMMENT '版本',
  PRIMARY KEY (`id`),
  UNIQUE KEY `UK_bg7pgyvfwv0q65tmquumxff3d` (`sn`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci COMMENT='设备表'
```

```
CREATE TABLE `consumer_has_device` (
  `device_id` int unsigned DEFAULT NULL,
  `consumer_id` int unsigned NOT NULL,
  PRIMARY KEY (`consumer_id`),
  UNIQUE KEY `UK_ibck20j1s6ch97lncg99uvpgh` (`device_id`),
  CONSTRAINT `FKcottusf6sx3bnouahp29vjdwk` FOREIGN KEY
(`device_id`) REFERENCES `device` (`id`),
  CONSTRAINT `FKq7u4eyw8pmfkwg4yymjljx8ra` FOREIGN KEY
(`consumer_id`) REFERENCES `consumer` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci
```

@OrderBy

```
// JPA 默认根据 Student 的 ID 主键对 studentList 集合数据进行递增排
序
@OneToMany(cascade = CascadeType.ALL)
@OrderBy
private List<Student> studentList;

// 手动指定 id 字段的排序方式，ASC 递增排序，DESC 递减排序
@OneToMany(cascade = CascadeType.ALL)
@OrderBy("id desc")
private List<Student> studentList;

// 手动指定按照 salary 属性进行递减排序
@OneToMany(cascade = CascadeType.ALL)
@OrderBy("salary desc")
private List<Student> studentList;

// 手动指定按照多个属性进行排序
// 下面将根据 sex 和 salay 进行递增排序
@OneToMany(cascade = CascadeType.ALL)
@OrderBy("sex,salary")
private List<Student> studentList;

// 下面将根据 sex 递增排序，salary 递增排序
@OneToMany(cascade = CascadeType.ALL)
@OrderBy("sex asc,salary asc")
```

```
private List<Student> studentList;

// 下面将根据 sex 递增排序，salary 递减排序
@OneToMany(cascade = CascadeType.ALL)
@OrderBy("sex asc,salary desc")
private List<Student> studentList;
```

@ElementCollection

```
import javax.persistence.*;
import java.util.*;
import javax.persistence.*;

@Entity
public class Employee {
     @Id
     @GeneratedValue(strategy = GenerationType.AUTO)
     private int id;
     private String name;

     @ElementCollection
     private Set<Address> address = new HashSet<Address>();
}
```

```
import javax.persistence.*;

@Embeddable
public class Address {
     private String province;
     private String city;
     private String state;
}
```

```
```

Set 集合

```
        @ElementCollection
    private final Set<String> address = new HashSet<String>
();
```

List 集合

```
        @ElementCollection(targetClass = String.class) //指定
集合中元素的类型
        @CollectionTable(name = "school_inf", joinColumns =
@JoinColumn(name="pid",nullable = false)) //表示外键不能为空
        @Column(name = "school_name")            //指定表中保存集
合元素的列名
        @OrderColumn(name = "list_order")        //索引列
        private List<String> schools = new ArrayList<String>
();
```

数组集合

```
@Entity
@Table
public class Student {

        @Id @Column
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Integer sid;
        private String name;
```

```
        private Integer age;
        @ElementCollection(targetClass=String.class) //集合中元
素的类型
        @CollectionTable(name = "school", joinColumns =
@JoinColumn(name="sid",nullable = false))//指定外键的名称为sid,
并且不能为空
        @Column(name = "school_name") //指定schools属性，在表中
的列名
        @OrderColumn(name = "array_order")
        private String[] schools = new String[3];

}
```

Map 集合

```
@Entity
@Table(name = "student")
public class Student {

        @Id @Column
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Integer sid;
        private String name;
        private Integer age;
        @ElementCollection(targetClass = Float.class) //对于
Map类型的属性：指定的是Value的类型
        @CollectionTable(name = "score_info", joinColumns =
@JoinColumn(name="sid", nullable = false))
        @MapKeyClass(String.class)              // 指定Map中
key的类型
        @MapKeyColumn(name="suject")    //指定索引列，也就是key的
列名
        @Column(name = "score") //映射保存Map, Value的列名
        private Map<String, Float> scores = new
HashMap<String, Float>(); //科目和成绩

}
```

外键名称

```
    @ElementCollection
    @CollectionTable(joinColumns = @JoinColumn(name = "pid",
nullable = false))
    private Set<Status> address = new HashSet<Status>();
```

@JsonIgnore

当尸体返回 Json 数据结构是，将不包含 @JsonIgnore 定义变量。

```
        @JsonIgnore
    @OneToMany(mappedBy = "owner")
    private List<Pet> pets;
```

@EnableJpaAuditing 开启 JPA 审计功能

```
@SpringBootApplication
@EnableJpaAuditing
public class Application {

    public static void main(String[] args) throws Exception {
        SpringApplication.run(Application .class, args);
    }
}
```

在需要审计实体中加入
@EntityListeners(AuditingEntityListener.class)

```
@EntityListeners(AuditingEntityListener.class)
public class Member  implements Serializable {

    private static final long serialVersionUID =
-6163675075289529459L;

    @JsonIgnore
    String entityName = this.getClass().getSimpleName();

    @CreatedBy
    String createdBy;

    @LastModifiedBy
    String modifiedBy;
    /**
     * 实体创建时间
     */
    @Temporal(TemporalType.TIMESTAMP)
    @CreatedDate
    protected Date dateCreated = new Date();

    /**
     * 实体修改时间
     */
    @Temporal(TemporalType.TIMESTAMP)
    @LastModifiedDate
    protected Date dateModified = new Date();

     #省略getter setter
}
```

注释 @Comment

```
    @Column(columnDefinition = "int unsigned NOT NULL DEFAULT
'0'")
    @Comment("点赞")
    private int likes;
    @Column(columnDefinition = "int unsigned NOT NULL DEFAULT
'0'")
    @Comment("收藏")
    private int favorites;
    @Column(columnDefinition = "int unsigned NOT NULL DEFAULT
'0'")
    @Comment("转发")
    private int forward;
```

@Pattern 数据匹配

```
    /**
     * 性别 CHAR(1) 0:女 1:男
     */
    @Pattern(regexp = "[01]")
    @Column(name = "gender",columnDefinition = "char(1)")
    private String gender;

    /**
     * 身份证号 CHAR(18)
     */
    @Pattern(regexp = "^([1-6][1-9]|50)\\d{4}(18|19|20)\\d{2}
((0[1-9])|10|11|12)(([0-2][1-9])|10|20|30|31)\\d{3}[0-9Xx]$")
    @Column(name = "identityCard",columnDefinition =
"char(18)")
    private String identityCard;

    /**
     * 所属部门 CHAR(2)  01：金融一部, 02：金融二部, 03：创新中心
     */
    @Pattern(regexp = "(01|02|03)")
    @Column(name = "department",columnDefinition = "char(2)")
```

```
    private String department;
```

实体继承

B、C 类继承 A 所有属性，并且主键均为数据库（auto_increment）

```
@MappedSuperclass
@(strategy = InheritanceType.TABLE_PER_CLASS)
public class A{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int id;
}
```

```
@Entity
@Table(name="b")
public class B extends A{

}
```

```
@Entity
@Table(name="c")
public class C extends A{

}
```

3.3. Repository

```
Repository： 仅仅是一个标识，没有任何方法，方便Spring自动扫描识别
CrudRepository： 继承Repository，实现了一组CRUD相关的方法
PagingAndSortingRepository： 继承CrudRepository，实现了一组分页排序相关的方法
JpaRepository： 继承PagingAndSortingRepository，实现一组JPA规范相关的方法
```

Spring Data JPA 为此提供了一些表达条件查询的关键字：

```
Keyword Sample  JPQL snippet
And                     findByLastnameAndFirstname      … where
x.lastname = ?1 and x.firstname = ?2
Or                      findByLastnameOrFirstname       … where
x.lastname = ?1 or x.firstname = ?2
Is,Equals       findByFirstnameIs,findByFirstnameEquals … where
x.firstname = ?1
Between         findByStartDateBetween  … where x.startDate
between ?1 and ?2
LessThan        findByAgeLessThan       … where x.age < ?1
LessThanEqual   findByAgeLessThanEqual  … where x.age ⇐ ?1
GreaterThan     findByAgeGreaterThan    … where x.age > ?1
GreaterThanEqual        findByAgeGreaterThanEqual       … where
x.age >= ?1
After           findByStartDateAfter    … where x.startDate > ?
1
Before          findByStartDateBefore   … where x.startDate < ?
1
IsNull          findByAgeIsNull … where x.age is null
IsNotNull,NotNull       findByAge(Is)NotNull    … where x.age
not null
Like            findByFirstnameLike     … where x.firstname
like ?1
NotLike         findByFirstnameNotLike  … where x.firstname not
like ?1
StartingWith    findByFirstnameStartingWith     … where
```

```
x.firstname like ?1 (parameter bound with appended %)
EndingWith      findByFirstnameEndingWith       … where
x.firstname like ?1 (parameter bound with prepended %)
Containing      findByFirstnameContaining       … where
x.firstname like ?1 (parameter bound wrapped in %)
OrderBy         findByAgeOrderByLastnameDesc    … where x.age =
?1 order by x.lastname desc
Not                     findByLastnameNot       … where
x.lastname <> ?1
In                      findByAgeIn(Collection ages)    … where
x.age in ?1
NotIn           findByAgeNotIn(Collection age)  … where x.age
not in ?1
TRUE            findByActiveTrue()      … where x.active = true
FALSE           findByActiveFalse()     … where x.active =
false
IgnoreCase      findByFirstnameIgnoreCase       … where
UPPER(x.firstame) = UPPER(?1)

常用如下:
And --- 等价于 SQL 中的 and 关键字, 比如
findByUsernameAndPassword(String user, Striang pwd)
Or --- 等价于 SQL 中的 or 关键字, 比如
findByUsernameOrAddress(String user, String addr)
Between --- 等价于 SQL 中的 between 关键字, 比如
findBySalaryBetween(int max, int min)
LessThan --- 等价于 SQL 中的 "<", 比如 findBySalaryLessThan(int
max)
GreaterThan --- 等价于 SQL 中的">", 比如
findBySalaryGreaterThan(int min)
IsNull --- 等价于 SQL 中的 "is null", 比如 findByUsernameIsNull()
IsNotNull --- 等价于 SQL 中的 "is not null", 比如
findByUsernameIsNotNull()
NotNull --- 与 IsNotNull 等价
Like --- 等价于 SQL 中的 "like", 比如 findByUsernameLike(String
user)
NotLike --- 等价于 SQL 中的 "not like", 比如
findByUsernameNotLike(String user)
OrderBy ---等价于 SQL 中的 "order by", 比如
findByUsernameOrderBySalaryAsc(String user)
Not --- 等价于 SQL 中的 "!  =", 比如 findByUsernameNot(String
user)
In --- 等价于 SQL 中的 "in", 比如
findByUsernameIn(Collection<String> userList) , 方法的参数可以是
Collection 类型, 也可以是数组或者不定长参数
```

```
NotIn --- 等价于 SQL 中的 "not in"，比如
findByUsernameNotIn(Collection<String> userList) ，方法的参数可以
是 Collection 类型，也可以是数组或者不定长
```

CrudRepository

CrudRepository 接口提供了最基本的对实体类的添删改查操作

```
T save(T entity);
//保存单个实体
Iterable<T> save(Iterable<? extends T> entities);//保存集合
T findOne(ID id);
//根据id查找实体
boolean exists(ID id);
//根据id判断实体是否存在
Iterable<T> findAll();
//查询所有实体,不用或慎用!
long count();
//查询实体数量
void delete(ID id);
//根据Id删除实体
void delete(T entity);
//删除一个实体
void delete(Iterable<? extends T> entities);             //删除一
个实体的集合
void deleteAll();
//删除所有实体,不用或慎用!
```

批量保存

```
        List<Book> books = new ArrayList<>();
        books.add(new Book("Book A", new BookDetail(1)));
        books.add(new Book("Book B", new BookDetail(2)));
        books.add(new Book("Book C", new BookDetail(3)));
```

```
        bookRepository.save(books);
```

JpaRepository

https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html

```
Modifier and Type       Method and Description
void    deleteAllInBatch()
Deletes all entities in a batch call.
void    deleteInBatch(Iterable<T> entities)
Deletes the given entities in a batch which means it will
create a single Query.
List<T> findAll()
<S extends T>
List<S> findAll(Example<S> example)
<S extends T>
List<S> findAll(Example<S> example, Sort sort)
List<T> findAll(Sort sort)
List<T> findAllById(Iterable<ID> ids)
void    flush()
Flushes all pending changes to the database.
T       getOne(ID id)
Returns a reference to the entity with the given identifier.
<S extends T>
List<S> saveAll(Iterable<S> entities)
<S extends T>
S       saveAndFlush(S entity)
Saves an entity and flushes changes instantly.
```

PagingAndSortingRepository

Pageable

接口实现 PagingAndSortingRepository

```
package api.repository.h5;

import
org.springframework.data.repository.PagingAndSortingRepositor
y;

import api.domain.User;

public interface GatherRepository extends
PagingAndSortingRepository<User, Integer> {

}
```

控制器添加 Pageable pageable 参数

```
        @RequestMapping("/browse")
        public ModelAndView browse(Pageable pageable) {
                Page<User> users =
userRepository.findAll(pageable);

                System.out.println(users.toString());
                ModelAndView mv = new ModelAndView();
                mv.addObject("users", users.getContent());
                mv.addObject("number", users.getNumber());
                mv.addObject("size", users.getSize());
                mv.addObject("totalPages",
users.getTotalPages());
                mv.setViewName("table");

                return mv;
        }
```

解决 PagingAndSortingRepository 没有 save 等方法的问题

如果 Repository 继承了 PagingAndSortingRepository 你会发 CrudRepository 中的 save 等方法不能使用了，我的解决方法是写两个 Repository

一个 CURD 的 ChatRepository 放在 cn.netkiller.repository

```
package cn.netkiller.repository;


import cn.netkiller.domain.Chat;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface ChatRepository extends CrudRepository<Chat,
String> {
    List<Chat> findAllBySession(String session);

    Chat findOneBySession(String session);

}
```

另一个分页的 PagingAndSortingRepository 放在 cn.netkiller.repository.pageable

```
package cn.netkiller.repository.pageable;


import cn.netkiller.domain.Chat;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
```

```
import
org.springframework.data.repository.PagingAndSortingRepositor
y;
import org.springframework.stereotype.Repository;

@Repository
public interface ChatPageableRepository extends
PagingAndSortingRepository<Chat, String> {
    Page<Chat> findAllByDevice(String device, Pageable
pageable);
}
```

@PageableDefault 分页

```
@RequestMapping(value = "/list", method=RequestMethod.GET)
public Page<Blog> getEntryByPageable1(@PageableDefault( sort
= { "id" }, direction = Sort.Direction.DESC)
    Pageable pageable) {
    return blogRepository.findAll(pageable);
}

@RequestMapping(value = "/blog", method=RequestMethod.GET)
public Page<Blog> getEntryByPageable(@PageableDefault(value =
15, sort = { "id" }, direction = Sort.Direction.DESC)
    Pageable pageable) {
    return blogRepository.findAll(pageable);
}

@RequestMapping(value = "/list", method=RequestMethod.GET)
public Page<Blog> getEntryByPageable2(@PageableDefault
Pageable pageable) {
    return blogRepository.findAll(pageable);
}

@ModelAttribute("users")
public Page<User> users(@PageableDefault(size = 5) Pageable
pageable) {
        return userManagement.findAll(pageable);
}
```

我们只需要在方法的参数中直接定义一个pageable类型的参数，当Spring发现这个参数时，Spring会自动的根据request的参数来组装该pageable对象，Spring支持的request参数如下：

page，第几页，从0开始，默认为第0页
size，每一页的大小，默认为20
sort，排序相关的信息，以property,property(,ASC|DESC)的方式组织，例如sort=firstname&sort=lastname,desc表示在按firstname正序排列基础上按lastname倒序排列
这样，我们就可以通过url的参数来进行多样化、个性化的查询，而不需要为每一种情况来写不同的方法了。

通过url来定制pageable很方便，但唯一的缺点是不太美观，因此我们需要为pageable设置一个默认配置，这样很多情况下我们都能够通过一个简洁的url来获取信息了。

Spring提供了@PageableDefault帮助我们个性化的设置pageable的默认配置。例如@PageableDefault(value = 15, sort = { "id" }, direction = Sort.Direction.DESC)表示默认情况下我们按照id倒序排列，每一页的大小为15。

findByXXX

```
        @Autowired
        private ArticleRepository articleRepository;

        @RequestMapping("/mysql")
        @ResponseBody
        public String mysql() {
                articleRepository.save(new Article("Neo",
"Chen"));
                for (Article article :
articleRepository.findAll()) {
```

```
                        System.out.println(article);
                }
                Article tmp =
articleRepository.findByTitle("Neo");
                return tmp.getTitle();
        }

        @RequestMapping("/search")
        @ResponseBody
        public String search() {


                for (Article article :
articleRepository.findBySearch(1)) {
System.out.println(article); }

                List<Article> tmp =
articleRepository.findBySearch(1L);

                tmp.forEach((temp) -> {
                        System.out.println(temp.toString());
                });

                return tmp.get(0).getTitle();
        }
```

传 **Boolean** 参数

```
package cn.netkiller.wallet.repository.fcoin;

import java.util.List;

import org.springframework.data.domain.Pageable;
import org.springframework.data.repository.CrudRepository;

import cn.netkiller.wallet.domain.fcoin.Fcoin;;

public interface FcoinRepository extends
CrudRepository<Fcoin, String> {
```

```
        Fcoin findOneByAddress(String address);

        int countByAirdropFalse();

        List<Fcoin> findByAirdrop(boolean airdrop, Pageable
pageable);

}
```

Eunm 传递枚举参数

```
package cn.netkiller.api.repository;

import org.springframework.data.repository.CrudRepository;

import cn.netkiller.api.domain.StatisticsHistory;

public interface StatisticsHistoryRepostitory extends
CrudRepository<StatisticsHistory, Long> {

        public StatisticsHistory
findByMemberIdAndStatisticsIdAndType(long member_id, long
statistics_id,
                        StatisticsHistory.StatisticsType
type);

}
```

```
        @Autowired
        private StatisticsHistoryRepostitory
statisticsHistoryRepostitory;
```

```
statisticsHistoryRepostitory.findByMemberIdAndStatisticsIdAnd
Type(uid, id, type);
```

count 操作

```
public interface UserRepository extends CrudRepository<User,
Long> {

    Long countByFirstName(String firstName);

}
```

delete 删除操作

```
    @Transactional
    Long deleteByFirstName(String firstName);

    @Transactional
    List<User> removeByFirstName(String firstName);
```

OrderBy

```
public List<StudentEntity> findAllByOrderByIdAsc();
public List<StudentEntity> findAllByOrderByIdDesc();
List<RecentRead> findByMemberIdOrderByIdDesc(int memberId,
Pageable pageable);
```

GreaterThan

```
package schedule.repository;

import java.util.Date;

import org.springframework.data.repository.CrudRepository;

import common.domain.CmsTrash;

public interface CmsTrashRepository extends
CrudRepository<CmsTrash, Integer> {

        Iterable<CmsTrash>
findBySiteIdAndTypeOrderByCtimeASC(int siteId, String
string);

        Iterable<CmsTrash>
findBySiteIdAndTypeAndCtimeGreaterThanOrderByCtimeASC(int
siteId, String string, Date date);

}
```

Sort 排序操作操作

```
List<UserModel> findByName(String name, Sort sort);
```

```
Sort sort = new Sort(Direction.DESC, "id");
repostitory.findByName("Neo", sort);
```

```
userRepository.findAll(Sort.by(Sort.Direction.ASC, "name"));
userRepository.findAll(Sort.by("LENGTH(name)"));
```

Pageable 翻页操作

Page 返回数据和页码等数据

```
PageRequest(int page, int size, Sort sort)  Deprecated.
use PageRequest.of(int, int, Sort) instead.
```

```
package cn.netkiller.repository;


import cn.netkiller.domain.Picture;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface PictureRepository extends
```

```
JpaRepository<Picture, Long> {
    Picture findAllBySession(String session);

    Optional<Picture> findOneBySession(String session);

    Page<Picture> findAll(Pageable pageable);
}
```

```
    public Page<Picture> page(Pageable pageable) {
        return pictureRepository.findAll(pageable);
    }
```

```
    @GetMapping("/{device}/page")
    public Mono<Page<Picture>> page(@PathVariable String
device, Pageable pageable) {

        return Mono.just(pictureService.page(pageable));
    }
```

```
排序 /picture/test/page?sort=id,desc
每页返回数量 /picture/test/page?size=10
返回第二页5条数据 /picture/test/page?size=5&page=1
返回第二页5条数据，ID倒序排序 /picture/test/page?
size=5&page=1&sort=id,desc
```

```
curl -X 'GET' \
  'http://localhost:8080/picture/test/page?
```

```
page=0&size=1&sort=id' \
  -H 'accept: */*'
```

PageRequest.of

```
package cn.netkiller.api.repository;

import java.util.List;

import org.springframework.data.domain.Pageable;
import org.springframework.data.repository.CrudRepository;

import cn.netkiller.api.domain.RecentRead;

public interface RecentReadRepostitory extends
CrudRepository<RecentRead, Long> {

        List<RecentRead> findByMemberId(long id, Pageable
pageable);

}
```

Top 10 实例

```
        @RequestMapping("/recent/read/list/{id}")
        public List<RecentRead> recentList(@PathVariable long
id) {
                int page = 0;
                int limit = 10;
                List<RecentRead> recentRead =
recentReadRepostitory.findByMemberId(id, new
PageRequest(page, limit));
                return recentRead;
        }
```

翻页返回数据可以选择 Iterable/List 或者 Page。

Iterable/List 只返回数据，不含页码等数据

注意 PageRequest(int page, int size) 在新版 Spring boot 2.x 中已经废弃请使用 PageRequest.of(page, size) 替代

```
List<Fcoin> fcoins = fcoinRepository.findByAirdrop(false,
PageRequest.of(0, size));
```

@DynamicInsert 与 @DynamicUpdate

@DynamicUpdate 只更新修改的字段

继承已存在的 Repository

```
public interface MemberRepository extends JpaRepository<User,
Integer>, UserRepository {
    ...
}
```

3.4. TransactionTemplate

```
        @Autowired
```

```
        private TransactionTemplate transactionTemplate;
        ...
        ...
        public void save(final User user) {
                transactionTemplate.execute((status) => {
                        doSomeThing(user);
                        doSomeThing1();
                        doSomeThing2();
                        doSomeThingN();
                        return Boolean.TRUE;
                })
        }
```

3.5. JPQL @Query

@Modifying 更新/删除

更新/删除操作需要加上 @Modifying 注解

```
@Modifying
@Query("update Money m set m.isDeleted=?2 where  m.money=?1")
void updateStateByMoney(Long money, Byte state);
```

```
  @Modifying(clearAutomatically=true, flushAutomatically =
true)
```

事务 @Transactional

下面介绍一下@Transactional注解的参数以及使用:

事物传播行为介绍:

@Transactional(propagation=Propagation.REQUIRED)： 如果有事务, 那么加入事务, 没有的话新建一个(默认情况下)
@Transactional(propagation=Propagation.NOT_SUPPORTED)： 容器不为这个方法开启事务
@Transactional(propagation=Propagation.REQUIRES_NEW)： 不管是否存在事务,都创建一个新的事务,原来的挂起,新的执行完毕,继续执行老的事务
@Transactional(propagation=Propagation.MANDATORY)： 必须在一个已有的事务中执行,否则抛出异常
@Transactional(propagation=Propagation.NEVER)： 必须在一个没有的事务中执行,否则抛出异常(与Propagation.MANDATORY相反)
@Transactional(propagation=Propagation.SUPPORTS)： 如果其他bean调用这个方法,在其他bean中声明事务,那就用事务.如果其他bean没有声明事务,那就不用事务.
事物超时设置:

@Transactional(timeout=30) //默认是30秒
事务隔离级别:

@Transactional(isolation = Isolation.READ_UNCOMMITTED)：读取未提交数据(会出现脏读, 不可重复读) 基本不使用
@Transactional(isolation = Isolation.READ_COMMITTED)：读取已提交数据(会出现不可重复读和幻读)
@Transactional(isolation = Isolation.REPEATABLE_READ)：可重复读(会出现幻读)
@Transactional(isolation = Isolation.SERIALIZABLE)：串行化
MYSQL: 默认为REPEATABLE_READ级别　SQLSERVER: 默认为READ_COMMITTED
@Transactional注解中常用参数说明

注意的几点:

@Transactional 只能被应用到public方法上, 对于其它非public的方法,如果标记了@Transactional也不会报错,但方法没有事务功能.
用 spring 事务管理器,由spring来负责数据库的打开,提交,回滚.默认遇到运行期例外(throw new RuntimeException("注释");)会回滚，即遇到不受检查（unchecked）的例外时回滚；而遇到需要捕获的例外(throw new Exception("注释");)不会回滚,即遇到受检查的例外（就是非运行时抛出的异常，编译器会检查到的异常叫受检查例外或说受检查异常）时，需我们指定方式来让事务回滚要想所有异常都回滚,要加上 @Transactional(rollbackFor={Exception.class,其它异常}) .如果让unchecked例外不回
滚： @Transactional(notRollbackFor=RunTimeException.class)
@Transactional 注解应该只被应用到 public 可见度的方法上。 如果你在 protected、private 或者 package-visible 的方法上使
用 @Transactional 注解，它也不会报错， 但是这个被注解的方法将不会展示已配置的事务设置。
@Transactional 注解可以被应用于接口定义和接口方法、类定义和类的 public 方法上。然而，请注意仅仅 @Transactional 注解的出现不足于开启事务行为，它仅仅 是一种元数据，能够被可以识
别 @Transactional 注解和上述的配置适当的具有事务行为的beans所使用。上面的例子中，其实正是 元素的出现 开启 了事务行为。
Spring团队的建议是你在具体的类（或类的方法）上使
用 @Transactional 注解，而不要使用在类所要实现的任何接口上。你当然可以在接口上使用 @Transactional 注解，但是这将只能当你设置了基于接口的代理时它才生效。因为注解是不能继承的，这就意味着如果你正在使用基于类的代理时，那么事务的设置将不能被基于类的代理所识别，而且对象也将不会被事务代理所包装（将被确认为严重的）。因此，请接受Spring团队的建议并且在具体的类上使
用 @Transactional 注解。

删除更新需要 @Transactional 注解

```
package cn.netkiller.api.repository;

import javax.transaction.Transactional;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import cn.netkiller.api.domain.RecentRead;

@Repository
public interface RecentReadRepostitory extends
CrudRepository<RecentRead, Integer> {

        Page<RecentRead> findByMemberIdOrderByIdDesc(int
memberId, Pageable pageable);

        int countByMemberId(int memberId);

        @Transactional
        @Modifying
        @Query("DELETE FROM RecentRead r WHERE r.memberId = ?
1 AND r.articleId = ?2")
        void deleteByMemberIdAndArticleId(int memberId, int
articleId);

        @Transactional
        @Modifying
        @Query("delete from RecentRead where member_id =
:member_id")
        public void deleteByMemberId(@Param("member_id") int
memberId);

        int countByMemberIdAndArticleId(int memberId, int
articleId);

}
```

回滚操作

```
        // 指定Exception回滚
        @Transactional(rollbackFor=Exception.class)
    public void methodName() {
       // 不会回滚
       throw new Exception("...");
    }

        //指定Exception回滚，但其他异常不回滚
        @Transactional(noRollbackFor=Exception.class)
    public ItimDaoImpl getItemDaoImpl() {
        // 会回滚
        throw new RuntimeException("注释");
    }
```

```
@Service
public class UserService {
    @Autowired
    private UserRepostitory userRepostitory;

    @Transactional
    public void add(User user) {
        try {
            userRepostitory.save(user);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        }
        // 不会回滚
    }
    @Transactional
    public void add(User user) throws Exception {
        try {
```

```
            userRepostitory.delete(user);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
            // 抛出异常才会回滚
            throw new Exception(e);
        }
    }
}
```

private、default、protected 和 final 不支持事物

@Transactional 必须与 public 一起使用，不能定义为 private、default、protected

```
@Service
public class UserService {
    @Transactional
    private void add(User user) {
         save(user);
    }
}
```

Service 注意事项

this 调用不支持事物

```
@Service
public class UserService {

    @Autowired
    private UserRepostitory userRepostitory;
```

```
    @Transactional
    public void add(User user) {
        userRepostitory.save(user);
        this.update(user);
    }

    @Transactional
    public void update(User user) {
        userRepostitory.update(user);
    }
}
```

解决方案

```
@Servcie
public class UserService {
   @Autowired
   prvate ProfileService profileService;

   public void save(User user) {
         profileService.save(user);
   }
}

 @Servcie
 public class ProfileService {

    @Transactional(rollbackFor=Exception.class)
    public void save(User user) {

    }

}
```

需要 **@Service** 注解配合使用

@Transactional 需要在 @Controller、@Service、@Component、@Repository 等注解下才能使用

```
// @Servcie
public class UserService {
        @Autowired
        prvate ProfileService profileService;

        @Transactional
        public void save(User user) {
         profileService.save(user);
        }
}
```

屏蔽 @Servcie 后观察 save 的 @Transactional 是不生效的。

```
@Service
public class UserService {

    @Autowired
    private UserRepostitory userRepostitory;
    @Autowired
    private RoleService roleService;

    @Transactional
    public void add(User user) throws Exception {
        userRepostitory.save(user);
        new Thread(() -> {
            roleService.doOtherThing();
        }).start();
    }
}

@Service
public class RoleService {
```

```
    @Transactional
    public void doOtherThing() {
        ...
        ...
    }
}
```

参数传递

```
package api.repository.oracle;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import api.domain.oracle.Member;

@Repository
public interface MemberRepository extends
CrudRepository<Member, Long> {
        public Page<Member> findAll(Pageable pageable);

        // public Member findByBillno(String billno);

        public Member findById(String id);

        @Query("SELECT m FROM Member m WHERE m.status = 'Y'
AND m.id = :id")
        public Member findFinishById(@Param("id") String id);

}
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface PersonRepository extends
JpaRepository<Person, Long> {
    @Query("SELECT p FROM Person p WHERE LOWER(p.lastName) =
LOWER(:lastName)")
    public List<Person> find(@Param("lastName") String
lastName);
}
```

原生 SQL

```
public interface UserRepository extends JpaRepository<User,
Long> {

        @Query(value = "SELECT * FROM USERS WHERE
EMAIL_ADDRESS = ?0", nativeQuery = true)
        User findByEmailAddress(String emailAddress);
}
```

insert ignore

```
        @Modifying
    @Query(value = "insert ignore into emp(create, modified,
user_id, user_name, user_nickname, user_mail) values(?1, ?2,
?3, ?4, ?5, ?6)", nativeQuery = true)
    void insertIgnoreEmployee(Timestamp create, Timestamp
modified, String userId, String name, String nickname, String
mail);
```

@Query 与 Pageagble

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#_native_queries

```
@Query(value = "SELECT u FROM User u ORDER BY id")
Page<User> findAllUsersWithPagination(Pageable pageable);
```

```
package api.domain;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Index;
import javax.persistence.Table;

@Entity
@Table(indexes = { @Index(name = "address", columnList =
"from_address,to_address"), @Index(name = "contractAddress",
columnList = "contractAddress") })

public class TransactionHistory implements Serializable {
        private static final long serialVersionUID =
6710992220657056861L;
        @Id
        @Column(name = "blockNumber", unique = true, nullable
= false, insertable = true, updatable = false)
        private int blockNumber;
        private String timeStamp;
```

```
	private String hash;
	@Column(name = "from_address")
	private String from;
	@Column(name = "to_address")
	private String to;
	private String value;
	private String gas;
	private String gasPrice;
	private String isError;
	private String contractAddress;
	private String gasUsed;
	private String symbol;

	public TransactionHistory() {
		// TODO Auto-generated constructor stub
	}

	public int getBlockNumber() {
		return blockNumber;
	}

	public void setBlockNumber(int blockNumber) {
		this.blockNumber = blockNumber;
	}

	public String getTimeStamp() {
		return timeStamp;
	}

	public void setTimeStamp(String timeStamp) {
		this.timeStamp = timeStamp;
	}

	public String getHash() {
		return hash;
	}

	public void setHash(String hash) {
		this.hash = hash;
	}

	public String getFrom() {
		return from;
	}
```

```
        public void setFrom(String from) {
                this.from = from;
        }

        public String getTo() {
                return to;
        }

        public void setTo(String to) {
                this.to = to;
        }

        public String getValue() {
                return value;
        }

        public void setValue(String value) {
                this.value = value;
        }

        public String getGas() {
                return gas;
        }

        public void setGas(String gas) {
                this.gas = gas;
        }

        public String getGasPrice() {
                return gasPrice;
        }

        public void setGasPrice(String gasPrice) {
                this.gasPrice = gasPrice;
        }

        public String getIsError() {
                return isError;
        }

        public void setIsError(String isError) {
                this.isError = isError;
        }

        public String getContractAddress() {
```

```
                return contractAddress;
        }

        public void setContractAddress(String
contractAddress) {
                this.contractAddress = contractAddress;
        }

        public String getGasUsed() {
                return gasUsed;
        }

        public void setGasUsed(String gasUsed) {
                this.gasUsed = gasUsed;
        }

        public static long getSerialversionuid() {
                return serialVersionUID;
        }

        public String getSymbol() {
                return symbol;
        }

        public void setSymbol(String symbol) {
                this.symbol = symbol;
        }

        @Override
        public String toString() {
                return "TransactionHistory [blockNumber=" +
blockNumber + ", timeStamp=" + timeStamp + ", hash=" + hash +
", from=" + from + ", to=" + to + ", value=" + value + ",
gas=" + gas + ", gasPrice=" + gasPrice + ", isError=" +
isError + ", contractAddress=" + contractAddress + ",
gasUsed=" + gasUsed + ", symbol=" + symbol + "]";
        }

}
```

```
package api.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import api.domain.TransactionHistory;

@Repository
public interface TransactionHistoryRepository extends
CrudRepository<TransactionHistory, Integer> {

        @Query(value = "SELECT * FROM transaction_history th
WHERE (th.from_address = :address or th.to_address =
:address) and contract_address is NULL",
                        countQuery = "SELEÇT count(*) FROM
transaction_history th WHERE (th.from_address = :address or
th.to_address = :address) and contract_address is NULL",
                        nativeQuery = true)
        public Page<TransactionHistory>
findEthByAddress(@Param("address") String address, Pageable
pageable);

}
```

返回指定字段

通过实体返回数据有时结果集非常庞大，可能会影响性能，这时我们只需要返回指定字段即可。

```
@Query(value = "select u.userName, ui.name, ui.gender,
ui.description from UserInfo ui, User u where u.id =
ui.userId")
```

```
public List<Object> getCustomField();
```

返回指定的模型

临时写一个新的模型

```
public class MyModel implements Serializable {

    private String userName;
    private String name;
    private String gender;
    private String description;

    public MyModel() {};

    public MyModel(String userName, String name, String
gender, String description) {
        this.userName = userName;
        this.name = name;
        this.gender = gender;
        this.description = description;
    }
}
```

使用构造方法赋值

```
@Query(value = "select new
cn.netkiller.model.MyModel(u.userName, ui.name, ui.gender,
ui.description) from UserInfo ui, User u where u.id =
ui.userId")
public List<MyModel> getAllRecord();
```

Collection

返回集合

```
@Query("SELECT u FROM User u WHERE u.status = 1")
Collection<User> findAllActiveUsers();
```

处理子查询 IN

```
@Query(value = "SELECT u FROM User u WHERE u.name IN :names")
List<User> findUserByNameList(@Param("names")
Collection<String> names);
```

原生SQL查询

这里的nativeQuery=true代表在执行这个方法的时候使用原生sql语句，直接写数据库中的实际表名和表中的字段名，而不是实体表名。

```
    @Modifying
    @Query(nativeQuery = true, value = "UPDATE project p,
(SELECT MIN(start) AS start, MAX(finish) AS finish FROM
project WHERE parent_id = :id) t SET p.start = t.start,
p.finish = t.finish WHERE p.id = :id")
    public void updateStartAndFinishById(@Param("id") Long
id);
```

在什么情况下使用呢？例如上面，同时操作两张表，做更新，如果不使用 nativeQuery = true 无法实现。

Sort

```
        @Query(value = "SELECT u FROM User u")
        List<User> findAllUsers(Sort sort);
```

锁 @Lock

```
interface UserRepository extends Repository<User, Long> {

  // Plain query method
  @Lock(LockModeType.READ)
  List<User> findByLastname(String lastname);
```

4. EntityManager

```
@Repository
@Transactional(readOnly = true)
class AccountServiceImpl implements AccountService {

  @PersistenceContext
  private EntityManager em;

  @Override
  @Transactional
  public Account save(Account account) {

    if (account.getId() == null) {
      em.persist(account);
      return account;
    } else {
      return em.merge(account);
    }
  }

  @Override
  public List<Account> findByCustomer(Customer customer) {

    TypedQuery query = em.createQuery("select a from Account
a where a.customer = ?1", Account.class);
    query.setParameter(1, customer);

    return query.getResultList();
  }

  @Override
  public List<Customer> findAll(int page, int pageSize) {

    TypedQuery query = em.createQuery("select c from Customer
c", Customer.class);

    query.setFirstResult(page * pageSize);
    query.setMaxResults(pageSize);
```

```
    return query.getResultList();
  }

}
```

5. Spring Data with JdbcTemplate

5.1. execute

```
jdbcTemplate.execute("CREATE TABLE USER (id integer, name
varchar(100))");
```

5.2. queryForInt

```
int count = jdbcTemplate.queryForInt("SELECT COUNT(*) FROM
USER");
```

5.3. queryForLong

```
lang count = jdbcTemplate.queryForLong("SELECT COUNT(*) FROM
USER");
```

5.4. queryForObject

返回整形与字符型

```
```

```
Integer age = queryForObject("select age from emp",
Integer.class);
String name = queryForObject("select name from
emp",String.class);
```

查询 **Double** 类型数据库

```
        private double getSumByMemberId(int memberId) {
                double result = 0.0d;
                String sql = "SELECT sum(o.price::NUMERIC) as
total FROM public.order o group by member_id ="+ memberId;
                try {
                        result =
jdbcTemplate.queryForObject(sql, Double.class);
                } catch
(org.springframework.dao.EmptyResultDataAccessException e) {
                        log.info("{} {}", MemberId,
e.toString());
                }
                return result;
        }
```

返回日期

注意 Date 是 java.util 不是 java.sql

```
        private static final Logger log =
LoggerFactory.getLogger(ScheduledTasks.class);
        private static final SimpleDateFormat dateFormat =
new SimpleDateFormat("yyyy-mm-dd HH:mm:ss");

        @Autowired
```

```
        private JdbcTemplate jdbcTemplate;

        @Scheduled(initialDelay = 1000, fixedRate = 60000)
        public void currentDate() {
                Date date =
jdbcTemplate.queryForObject("select sysdate from dual",
Date.class);
                log.info("The oracle sysdate is {}",
dateFormat.format(date));
        }
```

返回结果集

```
        @Autowired
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value = "/article")
        public @ResponseBody String dailyStats(@RequestParam
Integer id) {
                String query = "SELECT id, title, content
from article where id = " + id;

                return jdbcTemplate.queryForObject(query,
(resultSet, i) -> {

System.out.println(resultSet.getLong(1) + "," +
resultSet.getString(2) + "," + resultSet.getString(3));
                        return (resultSet.getLong(1) + "," +
resultSet.getString(2) + "," + resultSet.getString(3));
                });

        }
```

通过 "?" 向SQL传递参数

```
package com.example.api.restful;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;

import com.example.api.pojo.ResponseRestful;

@RestController
@RequestMapping("/restful/cms")
public class CmsRestController {
        @Autowired
        private JdbcTemplate jdbcTemplate;

        @RequestMapping(value =
"/article/update/count/{articleId}", method =
RequestMethod.GET, produces = { "application/xml",
"application/json" })
        public ResponseRestful updateCount(@PathVariable int
articleId) {
                String sql = "SELECT count(*) FROM
cms.article WHERE id > ?";
                int count = jdbcTemplate.queryForObject(sql,
new Object[] { articleId }, Integer.class);
                return new ResponseRestful(true, 1, "文章更新",
count);
        }
}
```

RowMapper 记录映射

```
package cn.netkiller.model;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class CustomerRowMapper implements RowMapper
{
        public Object mapRow(ResultSet rs, int rowNum) throws
SQLException {
                Customer customer = new Customer();
                customer.setId(rs.getInt("ID"));
                customer.setName(rs.getString("NAME"));
                customer.setAge(rs.getInt("AGE"));
                return customer;
        }

}
```

```
public Customer findByCustomerId(int id){

        String sql = "SELECT * FROM CUSTOMER WHERE ID = ?";

        Customer customer =
(Customer)getJdbcTemplate().queryForObject(
                        sql, new Object[] { id }, new
CustomerRowMapper());

        return customer;
}
```

```
Member member = this.jdbcTemplate.queryForObject("select
first_name, last_name from member where id = ?",new Object[]
{112L},
new RowMapper<Member>() {
    public Actor mapRow(ResultSet rs, int rowNum) throws
SQLException {
        Member member = new Member();
        member.setFirstName(rs.getString("first_name"));
        member.setLastName(rs.getString("last_name"));
        return member;
    }
});
```

5.5. queryForList

```
List rows = jdbcTemplate.queryForList("SELECT * FROM USER");
Iterator it = rows.iterator();
while(it.hasNext()) {
    Map userMap = (Map) it.next();
    System.out.print(userMap.get("id") + "\t");
    System.out.print(userMap.get("name") + "\t");
    System.out.print(userMap.get("sex") + "\t");
    System.out.println(userMap.get("age") + "\t");
}
```

Iterator 用法

```
List<Map<String, Object>> rows =
jdbcTemplate.queryForList("select * from user_token where
address=? and contract_address not in (select
contract_address from token)", new Object[] { address });
Iterator<Map<String, Object>> it = rows.iterator();
while (it.hasNext()) {
```

```
        Map<String, Object> userMap = (Map<String, Object>)
it.next();

        String contractAddress = (String)
userMap.get("contract_address");
        String symbol = (String) userMap.get("symbol");
        int decimals = (int) userMap.get("decimals");
}
```

for 循环

```
        @RequestMapping("/article/tag/{siteId}")
        public ResponseRestful tag(@PathVariable int siteId)
{
                List<Tag> tags = new ArrayList<Tag>();
                List<Map<String, Object>> rows = new
ArrayList<Map<String, Object>>();

                String sql = "SELECT id,name FROM cms.tag
WHERE site_id = ?";
                rows = jdbcTemplate.queryForList(sql, new
Object[] { siteId });

                for (Map<String, Object> row : rows) {
                        Tag tag = new Tag();
                        tag.setId((Integer) row.get("id"));
                        tag.setName((String)
row.get("name"));
                        tags.add(tag);
                }
                logger.info("tag {} SQL: {}", siteId, sql);
                return new ResponseRestful(true, tags.size(),
"标签", tags);
        }
```

forEach 用法

```
                jdbcTemplate.queryForList("select id from
public.contract").forEach(item -> {
                        logger.debug(item.toString());
                });
```

5.6. queryForMap

```
Map<String, Object> map =
this.jdbcTemplate.queryForMap("SELECT * FROM USERS WHERE
USERNAME=?", "username");

System.out.println(map.get("USERNAME"));
```

5.7. query

ResultSet

```
HashMap<String,String> member = jdbcTemplate.query("select
name,age from member where id=1", (ResultSet rs) -> {
    HashMap<String,String> results = new HashMap<>();
    while (rs.next()) {
        results.put(rs.getString("name"),
rs.getString("age"));
    }
    return results;
});
```

ResultSetExtractor

ResultSetExtractor

```
HashMap<String,String> member = jdbcTemplate.query("select
name,age from member where id=1", new ResultSetExtractor<Map>
(){
    @Override
    public Map extractData(ResultSet rs) throws
SQLException,DataAccessException {
        HashMap<String,String> mapResult= new
HashMap<String,String>();
        while(rs.next()){

mapResult.put(rs.getString("name"),rs.getString("age"));
        }
        return mapResult;
    }
});
```

RowMapper

```
List<Actor> actors = this.jdbcTemplate.query("select
first_name, last_name from actor",new RowMapper<Actor>() {
    public Actor mapRow(ResultSet rs, int rowNum) throws
SQLException {
        Actor actor = new Actor();
        actor.setFirstName(rs.getString("first_name"));
        actor.setLastName(rs.getString("last_name"));
        return actor;
    }
```

```
});
```

```
public List<Actor> findAllActors() {
    return this.jdbcTemplate.query( "select first_name,
last_name from actor", new ActorMapper());
}

private static final class ActorMapper implements
RowMapper<Actor> {

    public Actor mapRow(ResultSet rs, int rowNum) throws
SQLException {
        Actor actor = new Actor();
        actor.setFirstName(rs.getString("first_name"));
        actor.setLastName(rs.getString("last_name"));
        return actor;
    }
}
```

返回第一条数据，事实上只有一条。

```
        public Token getTokenBySymbol(String symbol) {

                List<Token> response =
jdbcTemplate.query("select * from token where symbol ='" +
symbol + "'", new RowMapper<Token>() {
                        public Token mapRow(ResultSet result,
int rowNum) throws SQLException {
                                Token Token = new Token();

Token.setContractAddress(result.getString(""));

Token.setName(result.getString("name"));
```

```
Token.setSymbol(result.getString("symbol"));

Token.setDecimals(result.getInt("decimals"));
                                return Token;
                        }
                });

                if (response.size() == 1) {
                        return response.get(0);
                }
                return null;
        }
```

5.8. queryForRowSet

```
SqlRowSet rs = jdbcTemplate.queryForRowSet("select * from
test");
```

5.9. update

```
@RequestMapping(value="/comment/add/{siteId}/{articleId}",
method = RequestMethod.POST)
        public ResponseRestful
commentAdd(@PathVariable("siteId") int siteId,
@PathVariable("articleId") int articleId, @RequestBody
Comment comment) {
                String sql = "insert into cms.comment("
                                + "article_id, "
                                + "ctime,"
                                + "content,"
                                + "member_id,"
                                + "nickname,"
```

```
                                    + "picture"
                                    + ")
values(?,?,now(),?,?,?,?,?)";

                int count = jdbcTemplate.update(sql,
                                    comment.getArticleId(),
                                    comment.getContent(),
                                    comment.getMemberId(),
                                    comment.getNickname(),
                                    comment.getPicture()
                );

                return new ResponseRestful(true, count, "评论
添加成功", comment);
        }
```

5.10.

```
new MapSqlParameterSource("symbol", symbol)
```

5.11. 实例参考

参数传递技巧

```
        public List<PendingTransaction>
getPendingTransaction(String address, String contractAddress)
{
                List<PendingTransaction> pendingTransactions
= new ArrayList<PendingTransaction>();
                String sql;
                Object[] param;
                if (contractAddress == null ||
```

```
contractAddress.equals("")) {
                        sql = "select * from
pending_transaction where from_address = ? and
contract_address IS NULL";
                        param = new Object[] { address };
                } else {
                        sql = "select * from
pending_transaction where from_address = ? and
contract_address = ?";
                        param = new Object[] { address,
contractAddress };
                }
                List<Map<String, Object>> rows =
jdbcTemplate.queryForList(sql, param);

                for (Map<String, Object> row : rows) {
                        PendingTransaction pendingTransaction
= new PendingTransaction();
                        pendingTransaction.setHash((String)
row.get("hash"));
                        pendingTransaction.setFrom((String)
row.get("from_address"));
                        pendingTransaction.setTo((String)
row.get("to_address"));
                        pendingTransaction.setValue((String)
row.get("value"));
                        pendingTransaction.setGas((String)
row.get("gas"));
                        pendingTransaction.setSymbol((String)
row.get("symbol"));

pendingTransaction.setContractAddress((String)
row.get("contractAddress"));

pendingTransactions.add(pendingTransaction);
                }
                logger.info("PendingTransaction:" +
pendingTransactions.toString());
                return pendingTransactions;
        }
```

6. Spring Data with Elasticsearch

6.1. 内嵌 Elasticsearch

内嵌 Elasticsearch 应用，你不需要一个 Elasticsearch 服务器，启动 Spring boot 即可使用 Elasticsearch 服务。

Maven

需要下面两个依赖

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
elasticsearch</artifactId>
                </dependency>
                <!-- com.sun.jna for elasticsearch -->
                <dependency>
                        <groupId>com.sun.jna</groupId>
                        <artifactId>jna</artifactId>
                        <version>3.0.9</version>
                </dependency>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.example</groupId>
        <artifactId>api</artifactId>
        <version>0.0.1-SNAPSHOT</version>
```

```
        <packaging>jar</packaging>

        <name>api</name>
        <description>Demo project for Spring
Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.6.RELEASE</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
```

```
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
elasticsearch</artifactId>
                </dependency>
                <!--
https://mvnrepository.com/artifact/javax.persistence/persiste
nce-api -->
                <dependency>
                        <groupId>javax.persistence</groupId>
                        <artifactId>persistence-
api</artifactId>
                        <version>1.0.2</version>
                </dependency>
                <!--
https://mvnrepository.com/artifact/org.json/json -->
```

```
                <dependency>
                        <groupId>org.json</groupId>
                        <artifactId>json</artifactId>
                </dependency>
                <!-- com.sun.jna for elasticsearch -->
                <dependency>
                        <groupId>com.sun.jna</groupId>
                        <artifactId>jna</artifactId>
                        <version>3.0.9</version>
                </dependency>

        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>

                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

src/main/resources/application.properties

```
spring.data.elasticsearch.repositories.enabled=true
```

```
#spring.data.elasticsearch.cluster-name=elasticsearch
#spring.data.elasticsearch.cluster-nodes=119.29.241.95:9200
spring.data.elasticsearch.local=false
spring.data.elasticsearch.properties.transport.tcp.connect_ti
meout=60s
spring.data.elasticsearch.properties.host=127.0.0.1
spring.data.elasticsearch.properties.port=9200
spring.data.elasticsearch.properties.path.home=/tmp
```

Domain Class

```
package com.example.api.domain.elasticsearch;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Id;

import org.springframework.data.annotation.CreatedDate;
import
org.springframework.data.elasticsearch.annotations.DateFormat
;
import
org.springframework.data.elasticsearch.annotations.Document;
import
org.springframework.data.elasticsearch.annotations.Field;
import
org.springframework.data.elasticsearch.annotations.FieldIndex
;
import
org.springframework.data.elasticsearch.annotations.FieldType;

import com.fasterxml.jackson.annotation.JsonFormat;

@Document(indexName = "information", type = "article")
public class Article implements Serializable {

        /**
         *
```

```
         */
        private static final long serialVersionUID =
8789505663320446079L;
        @Id
        private int id;
        private String title;
        private String description;
        private String author;
        private String source;
        private String content;
        @JsonFormat(shape = JsonFormat.Shape.STRING, pattern
= "yyyyMMdd'T'HHmmss.SSS'Z'")
    @Field(type = FieldType.Date, format =
DateFormat.basic_date_time, index = FieldIndex.not_analyzed)
    @CreatedDate
        private Date ctime;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getTitle() {
                return title;
        }

        public void setTitle(String title) {
                this.title = title;
        }

        public String getDescription() {
                return description;
        }

        public void setDescription(String description) {
                this.description = description;
        }

        public Date getCtime() {
                return ctime;
        }
```

```
        public void setCtime(Date ctime) {
                this.ctime = ctime;
        }

        public String getAuthor() {
                return author;
        }

        public void setAuthor(String author) {
                this.author = author;
        }

        public String getSource() {
                return source;
        }

        public void setSource(String source) {
                this.source = source;
        }

        public String getContent() {
                return content;
        }

        public void setContent(String content) {
                this.content = content;
        }

        @Override
        public String toString() {
                return "Article [id=" + id + ", title=" +
title + ", description=" + description + ", author=" + author
+ ", source=" + source + ", content=" + content + ", ctime="
+ ctime + "]";
        }

}
```

ElasticsearchRepository

```
package com.example.api.repository.elasticsearch;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import
org.springframework.data.elasticsearch.repository.Elasticsear
chRepository;
import org.springframework.stereotype.Repository;

import com.example.api.domain.elasticsearch.Article;

@Repository
public interface ArticleElasticsearchRepository extends
ElasticsearchRepository<Article, Integer> {
        Page<Article> findByTitleLike(String title, Pageable
page);

        Page<Article> findByDescription(String description,
Pageable pageable);

        Page<Article> findByDescriptionNot(String
description, Pageable pageable);

        Page<Article> findByDescriptionLike(String
description, Pageable pageable);
}
```

SearchRestController

```
package com.example.api.restful;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
```

```
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

import com.example.api.domain.elasticsearch.Article;
import
com.example.api.repository.elasticsearch.ArticleElasticsearch
Repository;

@RestController
@RequestMapping("/restful/search")
public class SearchRestController {
        @Autowired
        private ArticleElasticsearchRepository
articleElasticsearchRepository;

        @RequestMapping(value = "/article/create")
        public  Article create() {
                Article article = new Article();
                article.setId(1);
                article.setTitle("sssss");
                article.setContent("test");
                return
articleElasticsearchRepository.save(article);
        }
        @RequestMapping(value = "/article/{articleId}")
        public  Article get(@PathVariable int articleId) {
                return
articleElasticsearchRepository.findOne(articleId);
        }
}
```

测试

```
                                MacBook-Pro:~ neo$ curl
```

```
http://test:test@localhost:8443/restful/search/article/create.j
son

{"id":1,"title":"sssss","description":null,"author":null,"sourc
e":null,"content":"test","ctime":null}

                                MacBook-Pro:~ neo$ curl
http://test:test@localhost:8443/restful/search/article/1.json

{"id":1,"title":"sssss","description":null,"author":null,"sourc
e":null,"content":"test","ctime":null}
```

6.2. 集群模式

查看 cluster.name 配置项

```
                        root@netkiller ~ % grep ^cluster.name
/etc/elasticsearch/elasticsearch.yml
                        cluster.name: elasticsearch
```

src/main/resources/application.properties

```
                        spring.data.elasticsearch.cluster-
name=elasticsearch
                        spring.data.elasticsearch.cluster-
nodes=172.16.0.100:9200
                        spring.data.elasticsearch.local=false

spring.data.elasticsearch.repositories.enabled=true
```

6.3. Document

```


@Document(indexName = "customer", type = "external", shards =
1, replicas = 0, refreshInterval = "-1")


```

6.4. Elasticsearch 删除操作

```
package com.example.api.schedule;

import org.elasticsearch.action.delete.DeleteResponse;
import org.elasticsearch.client.transport.TransportClient;
import org.elasticsearch.rest.RestStatus;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.stereotype.Component;

import
com.example.api.domain.elasticsearch.ElasticsearchTrash;
import
com.example.api.repository.elasticsearch.ElasticsearchTrashRe
pository;

@Component
public class ScheduledTasks {
        private static final Logger logger =
LoggerFactory.getLogger(ScheduledTasks.class);

        @Autowired
        private TransportClient client;

        @Autowired
        private ElasticsearchTrashRepository
alasticsearchTrashRepository;

        public ScheduledTasks() {
        }

        @Scheduled(fixedRate = 1000 * 60) // 60秒运行一次调度任
务
        public void cleanTrash() {
```

```
                for (ElasticsearchTrash elasticsearchTrash :
alasticsearchTrashRepository.findAll()) {
                        DeleteResponse response =
client.prepareDelete("information", "article",
elasticsearchTrash.getId() + "").get();
                        RestStatus status =
response.status();
                        logger.info("delete {} {}",
elasticsearchTrash.getId(), status.toString());
                        if (status == RestStatus.OK || status
== RestStatus.NOT_FOUND) {

alasticsearchTrashRepository.delete(elasticsearchTrash);
                        }
                }
        }
}
```

6.5. FAQ

java.lang.IllegalStateException: Received message from unsupported version: [2.0.0] minimal compatible version is: [5.0.0]

spring-boot-starter-data-elasticsearch 目前还不支持 5.0.0 版本

7. Spring boot with Data restful

spring-boot-starter-data-rest 能够提供将 Repository, CrudRepository 等接口直接提供给用户访问

7.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
rest</artifactId>
                </dependency>
```

8. Apache ShardingSphere

8.1. 微服务集群环境，雪花算法出现重复ID

```
Caused by:
com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolati
onException: Duplicate entry '854658443787632640' for key
'PRIMARY'
```

```
# 指定 工作机器数量 最大是2的10次方 ， 即小于 1024 就可以
spring.shardingsphere.sharding.tables.shard.key-
generator.props.worker.id=1000

max-vibration-offset

# 最大容忍的时钟回拨毫秒数，雪花算法依据时间戳来生成的，一旦时间戳回拨就会
造成 id 重复的可能
spring.shardingsphere.sharding.tables.shard.key-
generator.max.tolerate.time.difference.milliseconds=5
```

方案一、配置实现

随机指定 worker.id，这样在kubernetes集群环境，每次启动pod，worker.id 都会自动变化。

```
spring.shardingsphere.sharding.tables.test.key-
generator.props.worker.id=${random.int[1,1024]}
```

查看当前 worker.id

```
package cn.netkiller.controller.test;//package
cn.netkiller.controller;

import org.springframework.beans.factory.annotation.Value;
import
org.springframework.cloud.context.config.annotation.RefreshSc
ope;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

@RefreshScope
@RestController
public class TestRestController {

    @Value("${spring.shardingsphere.sharding.tables.test.key-
generator.props.worker.id}")
    public String workerId;

    public TestRestController() {

    }

    @GetMapping("/workerId")
    public String snow() {
        return this.workerId;
    }
}
```

方案二、代码实现

```
package cn.netkiller.config;

import org.springframework.context.annotation.Configuration;
```

```
import java.net.Inet4Address;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * 动态指定sharding jdbc 的 work.id 雪花算法中的属性，然后通过
System.setProperty() 设置环境变量
 * workId 可以用主机名、IP地址、Mac地址，最大值 1L << 100，就是
1024，即 0<= workId < 1024
 * {@link SnowflakeShardingKeyGenerator#getWorkerId()}
 */
@Configuration
public class SnowFlakeWordIdConfiguration {
    static {
        try {
            InetAddress ip4 = Inet4Address.getLocalHost();
            String addressIp = ip4.getHostAddress();
            System.setProperty("workerId",
(Math.abs(addressIp.hashCode()) % 1024) + "");
        } catch (UnknownHostException e) {
            throw new RuntimeException(e);
        }
    }
}
```

配置文件添加 key-generator.props.worker.id 设置 ${workerId} 变量

```
        key-generator:
          column: id
          props:
            worker:
              id: ${workerId}
          type: SNOWFLAKE
```

9. Spring Data FAQ

9.1. No identifier specified for entity

9.2. Oracle Date 类型显示日期和时间

```
package cn.netkiller.api.domain.oracle;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;

import org.springframework.format.annotation.DateTimeFormat;
import
org.springframework.format.annotation.DateTimeFormat.ISO;

import com.fasterxml.jackson.annotation.JsonFormat;

@Entity
@Table(name = "test")
public class Test {

        @Id
        @Column(name = "ID")
        @GeneratedValue(strategy = GenerationType.SEQUENCE,
generator = "test_id_Sequence")
        @SequenceGenerator(name = "test_id_Sequence",
sequenceName = "test")
        private Long id;

        @NotNull
        @DateTimeFormat(iso = ISO.DATE_TIME)
```

```
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        public Date createdate;

        public Member() {

        }

        public Long getId() {
                return id;
        }

        public void setId(Long id) {
                this.id = id;
        }

        public Date getCreatedate() {
                return createdate;
        }

        public void setCreatedate(Date createdate) {
                this.createdate = createdate;
        }

}
```

9.3. java.lang.ClassCastException: java.lang.Long cannot be cast to java.lang.Integer

问题描述，Restful 请求返回错误，检查数据库 BigInt 修改为 无符号整形，错误依旧存在

```
ALTER TABLE `cms`.`comment`
CHANGE COLUMN `user_id` `user_id` INT(10) UNSIGNED NULL DEFAULT
NULL ;
```

去掉 UNSIGNED 后，错误消失

```
ALTER TABLE `cms`.`comment`
CHANGE COLUMN `user_id` `user_id` INT NULL DEFAULT NULL ;
```

Java 认为 INT(10) UNSIGNED 是 Long 型。

9.4. Executing an update/delete query; nested exception is javax.persistence.TransactionRequiredException: Executing an update/delete query

Internal Server Error","exception":"org.springframework.dao.InvalidDataAccessApiUsageException","message":"Executing an update/delete query; nested exception is javax.persistence.TransactionRequiredException: Executing an update/delete query"



第 9 章 Spring Security

1. Spring Security with HTTP Auth

1.1. 默认配置

如果在 maven 中引入了 spring security当你启动 springboot 的时候会提示

```
Using generated security password: 1cd27b90-1208-4be2-ae8e-
0f564ee427b8
```

默认用户名是 user 可以这样访问

```
neo@MacBook-Pro ~ % curl -s http://user:1cd27b90-1208-4be2-
ae8e-0f564ee427b8@localhost:8080/member/json
{"status":false,"reason":"","code":0,"data":{}}
```

1.2. 设置用户名和密码

```
spring.security.user.name=test
spring.security.user.password=test
spring.security.user.role=USER
```

注意 Springboot 1.x

```
security.user.name=test
security.user.password=passw0rdf
security.user.role=USER
```

1.3. 禁用 Security

方法一

```
@SpringBootApplication(exclude = {
SecurityAutoConfiguration.class })
public class Application {
        public static void main(String[] args) {
                System.out.println("Web Starting...");
                SpringApplication.run(Application.class, args);
        }
}
```

Springboot 1.x 可以在 appliction.properties 中加入

```
security.basic.enabled=false
```

2. Spring boot with Spring security

2.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>api.netkiller.cn</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
```

```
parent</artifactId>
                <version>2.0.2.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
mongodb</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
amqp</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
```

```
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-
devtools</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
mongodb</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
oracle</artifactId>
                        <version>1.0.0.RELEASE</version>
                </dependency>

                <dependency>
                        <groupId>com.oracle</groupId>
                        <artifactId>ojdbc6</artifactId>
                        <!-- <version>12.1.0.1</version> -->
                        <version>11.2.0.3</version>
                        <scope>system</scope>

<systemPath>${basedir}/lib/ojdbc6.jar</systemPath>
                </dependency>

                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
```

```
java</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
mail</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
velocity</artifactId>
                </dependency>
                <dependency>

<groupId>org.apache.velocity</groupId>
                        <artifactId>velocity</artifactId>
                </dependency>
                <dependency>

<groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                        <scope>compile</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
```

```
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-war-
plugin</artifactId>
                                <version>2.6</version>
                                <configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>
                                </configuration>
                        </plugin>
                </plugins>
        </build>

</project>
```

2.2. Reource

src/main/resources/application.properties

添加默认用户，角色user,用户名neo,密码password

```
security.user.name=neo
security.user.password=password
security.user.role=USER
```

现在启动Application，然后尝试访问url，这时会弹出对话框，提示用户用户输入用户名与密码。使用上面的密码便可登陆。

2.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaRepos
itories;
import
org.springframework.data.mongodb.repository.config.EnableMong
oRepositories;
import
org.springframework.web.servlet.config.annotation.CorsRegistr
y;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfi
gurerAdapter;

@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan({ "api.config", "api.web", "api.rest",
"api.service" })
@EnableMongoRepositories
@EnableJpaRepositories
public class Application {

        public @Bean WebMvcConfigurer corsConfigurer() {
                return new WebMvcConfigurerAdapter() {
                        @Override
                        public void
addCorsMappings(CorsRegistry registry) {
```

```
                                registry.addMapping("/**");
                        }
                };
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }

}
```

2.4. WebSecurityConfigurer

注意WebSecurityConfigurer必须在 ComponentScan 的扫描范围

```
package api.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication
.builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends
WebSecurityConfigurerAdapter {

        @Override
        protected void configure(AuthenticationManagerBuilder
```

```
auth) throws Exception {
                auth.inMemoryAuthentication().

withUser("user1").password("secret1").roles("USER")
                .and().

withUser("user2").password("secret2").roles("USER")
                .and().

withUser("admin").password("secret").roles("ADMIN");
        }

        @Override
        protected void configure(HttpSecurity http) throws
Exception {

http.authorizeRequests().anyRequest().fullyAuthenticated();
                http.httpBasic();
                http.csrf().disable();
        }

}
```

2.5. RestController

```
@RestController
@RequestMapping("/service")
public class UserService {
    @RequestMapping(value = "/echo/{in}", method =
RequestMethod.GET)
    public String echo(@PathVariable(value = "in") final
String in, @AuthenticationPrincipal final UserDetails user) {
        return "Hello " + user.getUsername() + ", you said: "
+ in;
    }
}
```

2.6. 测试

```
curl -u user:password http://172.16.0.20:8080/index.html
curl http://user:password@172.16.0.20:8080/index.html
```

2.7. Spring + Security + MongoDB

MongoDB 为 Security 用户认证提供数据存储。

Account

```
package mis.domain;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.index.Indexed;

public class Administrator {
        @Id
        private String id;

        @Indexed(unique = true)
        private String username;
        private String password;
        private String authority;

        public Administrator() {
                // TODO Auto-generated constructor stub
        }

        public Administrator(String username, String
password) {
            this.username = username;
            this.password = password;
          }
```

```
	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getUsername() {
		return username;
	}

	public void setUsername(String username) {
		this.username = username;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public String getAuthority() {
		return authority;
	}

	public void setAuthority(String authority) {
		this.authority = authority;
	}

	@Override
	public String toString() {
		return "User [id=" + id + ", username=" +
username + ", password=" + password + ", authority=" +
authority + "]";
	}
}
```

AccountRepository

```
package mis.repository;

import
org.springframework.data.mongodb.repository.MongoRepository;

import mis.domain.Administrator;

public interface AdministratorRepository extends
MongoRepository<Administrator, String> {

        public Administrator findByUsername(String username);

}
```

WebSecurityConfiguration

```
package mis.config;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication
.builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.authentication
.configurers.GlobalAuthenticationConfigurerAdapter;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
org.springframework.security.config.annotation.web.configurat
```

```
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;
import
org.springframework.security.core.authority.AuthorityUtils;
import org.springframework.security.core.userdetails.User;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsServ
ice;
import
org.springframework.security.core.userdetails.UsernameNotFoun
dException;

import mis.domain.Administrator;
import mis.repository.AdministratorRepository;

@Configuration
class GlobalAuthenticationConfigurer extends
GlobalAuthenticationConfigurerAdapter {

        @Autowired
        AdministratorRepository administratorRepository;

        @Override
        public void init(AuthenticationManagerBuilder auth)
throws Exception {

auth.userDetailsService(userDetailsService());
        }

        @Bean
        UserDetailsService userDetailsService() {
                return new UserDetailsService() {

                        @Override
                        public UserDetails
loadUserByUsername(String username) throws
UsernameNotFoundException {
                                Administrator administrator =
administratorRepository.findByUsername(username);
                                if (administrator != null) {
                                        return new
```

```
User(administrator.getUsername(),
administrator.getPassword(),
AuthorityUtils.createAuthorityList(administrator.getAuthority
()));
                                } else {
                                        throw new
UsernameNotFoundException("could not find the administrator
'" + username + "'");
                                }
                        }

                };
        }
}

@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends
WebSecurityConfigurerAdapter {

        public WebSecurityConfigurer() {
                // TODO Auto-generated constructor stub
        }

        @Override
        protected void configure(HttpSecurity http) throws
Exception {
                //
http.authorizeRequests().anyRequest().fullyAuthenticated().an
d().httpBasic().and().csrf().disable();

                // http.authorizeRequests().antMatchers("/",
"/index.html", "/css/**",
                //
"/js/**","/static/**","/setup.html").permitAll().anyRequest()
.authenticated().and().formLogin().loginPage("/login.html").p
ermitAll().and().logout().permitAll().and().httpBasic();
                // http.authorizeRequests().antMatchers("/**"
                // ).permitAll().and().httpBasic();

                http.authorizeRequests().antMatchers("/ping",
"/v1/*/ping",
"/v1/public/**").permitAll().anyRequest().authenticated().and
().rememberMe().and().httpBasic().and().csrf().disable();
```

```
            }
}
```

3. Spring Boot with Web Security

3.1. EnableWebSecurity

```
package cn.netkiller.config;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication
.builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

        public WebSecurityConfig() {
                // TODO Auto-generated constructor stub
        }

        @Override
        protected void configure(HttpSecurity http) throws
Exception {

                http
                .authorizeRequests()
                    .antMatchers("/", "/about.html",
"/doc/**").permitAll()
```

```
                    .anyRequest().authenticated()
                    .and()
                .formLogin()
                    .loginPage("/login.html")
                    .permitAll()
                    .and()
                .logout()
                    .permitAll();

        }

        @Autowired
    public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {
        auth
            .inMemoryAuthentication()

.withUser("user").password("password").roles("USER")
                .and()

.withUser("admin").password("admin").roles("ADMIN");

    }
}
```

3.2. Web静态资源

用于Web静态资源的权限控制

```
package com.example.api.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication
.builders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
```

```
org.springframework.security.config.annotation.web.builders.W
ebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends
WebSecurityConfigurerAdapter {

        @Override
        public void configure(WebSecurity web) throws
Exception {
                web.ignoring().antMatchers("/static/**",
"/**/*.jsp");
        }

        protected void
registerAuthentication(AuthenticationManagerBuilder auth)
throws Exception {

auth.inMemoryAuthentication().withUser("user1").password("sec
ret1").roles("USER").and().withUser("user2").password("secret
2").roles("USER").and().withUser("admin").password("secret").
roles("ADMIN");
        }

        @Override
        protected void configure(HttpSecurity http) throws
Exception {

http.authorizeRequests().anyRequest().fullyAuthenticated();
                http.httpBasic();
                http.csrf().disable();
        }

}
```

启动 Springboot 可以看到类似日志

```
2018-10-12 18:01:40.692  INFO 4722 --- [           main]
o.s.s.web.DefaultSecurityFilterChain     : Creating filter
chain: Ant [pattern='/**/json'], []
2018-10-12 18:01:40.692  INFO 4722 --- [           main]
o.s.s.web.DefaultSecurityFilterChain     : Creating filter
chain: Ant [pattern='/about'], []
2018-10-12 18:01:40.692  INFO 4722 --- [           main]
o.s.s.web.DefaultSecurityFilterChain     : Creating filter
chain: Ant [pattern='/test/hello'], []
2018-10-12 18:01:40.693  INFO 4722 --- [           main]
o.s.s.web.DefaultSecurityFilterChain     : Creating filter
chain: Ant [pattern='/web/**'], []
```

3.3. 正则匹配

```
@Override
public void configure(WebSecurity web) throws Exception {
    web.ignoring().regexMatchers(XXXXX);
}
```

3.4. 登陆页面，失败页面，登陆中页面

```
        @Override
        protected void configure(HttpSecurity http) throws
Exception {

http.authorizeRequests().antMatchers("/**").hasRole("USER").a
nd().formLogin().usernameParameter("username") // default is
username
```

```
.passwordParameter("password") // default is password

.loginPage("/authentication/login") // default is /login with
an HTTP get

.failureUrl("/authentication/login?failed") // default is
/login?error

.loginProcessingUrl("/authentication/login/process"); //
default is /login

        }
```

3.5. CORS

```
@EnableWebSecurity
public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws
Exception {
        http.cors().and()
            //other config
    }

    @Bean
    CorsConfigurationSource corsConfigurationSource()
    {
        CorsConfiguration configuration = new
CorsConfiguration();

configuration.setAllowedOrigins(Arrays.asList("https://exampl
e.com"));

configuration.setAllowedMethods(Arrays.asList("GET","POST"));
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
```

```
        source.registerCorsConfiguration("/**",
configuration);
        return source;
    }
}
```

3.6. X-Frame-Options 安全

X-Frame-Options: SAMEORIGIN

```
@EnableWebSecurity
public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

        @Override
        protected void configure(HttpSecurity http) throws
Exception {
                http
                        // ...
                        .headers()
                                .frameOptions().sameOrigin()

.httpStrictTransportSecurity().disable();
        }
}
```

安全配置 X-FRAME-OPTIONS 指定允许iframe访问的域名

```
package cn.netkiller.api.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.H
```

```
ttpSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;
import
org.springframework.security.web.header.writers.StaticHeaders
Writer;

@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends
WebSecurityConfigurerAdapter {

        @Override
        protected void configure(HttpSecurity http) throws
Exception {


http.headers().frameOptions().disable().addHeaderWriter(new
StaticHeadersWriter("X-FRAME-OPTIONS", "ALLOW-FROM
netkiller.cn")).and().
                        csrf().disable()
                        .authorizeRequests()

.antMatchers("/","/ping","/v1/*/ping","/public/**","/your/**"
).permitAll()

.antMatchers("/v1/**").authenticated().
                        anyRequest().permitAll().and().
                        httpBasic();
        }

}
```

4. 访问控制列表（Access Control List，ACL）

4.1. antMatchers

/** 表示放行所有请求URL

```
http.authorizeRequests().antMatchers("/**" ).permitAll();
```

匹配精确的URL地址 "/","/products","/product/show/*","/css/**"

```
        @Override
        protected void configure(HttpSecurity httpSecurity)
throws Exception {
    httpSecurity

.authorizeRequests().antMatchers("/","/products","/product/sh
ow/*","/css/**").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin().loginPage("/login").permitAll()
            .and()
            .logout().permitAll();

   httpSecurity.csrf().disable();
   httpSecurity.headers().frameOptions().disable();
}
```

4.2. HTTP Auth

```
        @Override
        protected void configure(HttpSecurity http) throws
Exception {

http.authorizeRequests().antMatchers("/ping","/v1/*/ping","/v
1/public/**" ).permitAll()
                .anyRequest().authenticated()
                .and().rememberMe().and().httpBasic()
                .and().csrf().disable();
        }
```

4.3. Rest

```
protected void configure(HttpSecurity http) throws Exception
{
    http
      .csrf().disable()
      .authorizeRequests()
        .antMatchers(HttpMethod.POST,
"/api/**").authenticated()
        .antMatchers(HttpMethod.PUT,
"/api/**").authenticated()
        .antMatchers(HttpMethod.DELETE,
"/api/**").authenticated()
        .anyRequest().permitAll()
        .and()
      .httpBasic().and()

.sessionManagement().sessionCreationPolicy(SessionCreationPol
icy.STATELESS);
}
```

4.4. hasRole

```
    @Override
    protected void configure(HttpSecurity http) throws
Exception {

      http.authorizeRequests()
        .antMatchers("/", "/member").access("hasRole('USER')
or hasRole('ADMIN') or hasRole('DBA')")
        .and().formLogin().loginPage("/login")

.usernameParameter("sso").passwordParameter("password")
        .and().exceptionHandling().accessDeniedPage("/403");
    }
```

4.5. hasAnyRole()

```
        @Autowired
    private AccessDeniedHandler accessDeniedHandler;

        @Override
    protected void configure(HttpSecurity http) throws
Exception {

        http.csrf().disable()
                .authorizeRequests()
                                        .antMatchers("/",
"/home", "/about").permitAll()

.antMatchers("/admin/**").hasAnyRole("ADMIN")

.antMatchers("/user/**").hasAnyRole("USER")

.anyRequest().authenticated()
                .and()
                .formLogin()
                                        .loginPage("/login")
```

```
                                            .permitAll()
                                            .and()
                .logout()
                                            .permitAll()
                                            .and()

.exceptionHandling().accessDeniedHandler(accessDeniedHandler)
;
    }
```

4.6. withUser

添加用户

```
        @Autowired
    public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {
        auth
            .inMemoryAuthentication()

.withUser("user").password("password").roles("USER");

    }
```

添加多个用户，并指定角色

添加多个用户

```
        @Autowired
    public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {
```

```
        auth.inMemoryAuthentication()

.withUser("user").password("password").roles("USER")
                .and()

.withUser("admin").password("password").roles("ADMIN");
    }
```

```
        @Autowired
    public void configureGlobal(AuthenticationManagerBuilder
auth) throws Exception {
        auth
            .inMemoryAuthentication()

.withUser("user").password("password").roles("USER")
                .and()

.withUser("admin").password("admin").roles("ADMIN")
                .and()

.withUser("admin").password("super").roles("ADMIN","SYS","DBA
")
                ;
    }
```

获取当前用户

```
Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
String currentPrincipalName = authentication.getName();
```

5. Spring Authorization Server

Spring Authorization Server 是 Spring Security OAuth 替代品。

5.1. Oauth2 协议

```
授权模式
oauth2.0提供了四种授权模式，开发者可以根据自己的业务情况自由选择。

授权码授权模式（Authorization Code Grant）
隐式授权模式（Implicit Grant）
密码授权模式（Resource Owner Password Credentials Grant）
客户端凭证授权模式（Client Credentials Grant）
```

token

access_token：访问令牌，必选项。
token_type：令牌类型，该值大小写不敏感，必选项。
expires_in：过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
refresh_token：更新令牌，用来获取下一次的访问令牌，可选项。
scope：权限范围，如果与客户端申请的范围一致，此项可省略。

grant_type

client_credentials

```
                                    grant_type =
'client_credentials' 模式不需要用户去资源服务器登录并授权, 因为客户端
(client)已经有了访问资源服务器的凭证(credentials).
                                    所以当用户访问时,由client直接向资源
```

服务器获取access_token并访问资源即可.

授权码授权模式（Authorization Code Grant）

（A）用户访问客户端，客户端将用户引导向认证服务器。
（B）用户选择是否给予客户端授权。
（C）如用户给予授权，认证服务器将用户引导向客户端指定的redirection uri，同时加上授权码code。
（D）客户端收到code后，通过后台的服务器向认证服务器发送code和redirection uri。
（E）认证服务器验证code和redirection uri，确认无误后，响应客户端访问令牌（access token）和刷新令牌（refresh token）。
请求示例
（A）步骤：客户端申请认证的URI

```
https://www.example.com/oauth/authorize?
response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_UR
L&scope=read&state=xxx
```

参数说明：
response_type：授权类型，必选项，此处的值固定为"code"
client_id：客户端的ID，必选项
redirect_uri：重定向URI，必选项
scope：申请的权限范围，可选项
state：任意值，认证服务器会原样返回,用于抵制CSRF(跨站请求伪造)攻击。

（C）步骤：服务器回应客户端的URI

```
https://client.example.com/cb?
code=SplxlOBeZQQYbYS6WxSbIA&state=xxx
```

参数说明：
code：授权码，必选项。授权码有效期通常设为10分钟，一次性使用。该码与客户端ID、重定向URI以及用户，是一一对应关系。
state：原样返回客户端传的该参数的值。

（D）步骤：客户端向认证服务器申请令牌

```
https://www.example.com/oauth/token?
client_id=CLIENT_ID&grant_type=authorization_code&code=AUTHORIZ
ATION_CODE&redirect_uri=CALLBACK_URL
```

参数说明：
client_id：表示客户端ID，必选项。
grant_type：表示使用的授权模式，必选项，此处的值固定为"authorization_code"。
code：表示上一步获得的授权码，必选项。
redirect_uri：表示重定向URI，必选项，且必须与A步骤中的该参数值保持一致。

注意：协议里没有提及client_secret参数，建议可以使用此参数进行客户端的二次验证。
（E）步骤：响应（D）步骤的数据

```
{ "access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"example", "expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
"example_parameter":"example_value" }
```

参数说明：
access_token：访问令牌，必选项。
token_type：令牌类型，该值大小写不敏感，必选项。
expires_in：过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
refresh_token：更新令牌，用来获取下一次的访问令牌，可选项。
scope：权限范围，如果与客户端申请的范围一致，此项可省略。

使用场景

授权码模式是最常见的一种授权模式，在oauth2.0内是最安全和最完善的。
适用于所有有Server端的应用，如Web站点、有Server端的手机客户端。
可以得到较长期限授权。

密码模式（Resource Owner Password Credentials Grant）

密码模式（Resource Owner Password Credentials Grant）
流程介绍

（A）用户向客户端提供用户名和密码。
（B）客户端将用户名和密码发给认证服务器，向后者请求令牌。
（C）认证服务器确认无误后，向客户端提供访问令牌。
请求示例
（B）步骤：客户端发出https请求

```
https://www.example.com/token?
grant_type=password&username=USERNAME&password=PASSWORD&client_
id=CLIENT_ID
```

参数说明
grant_type：授权类型，此处的值固定为"password"，必选项。
username：用户名，必选项。
password：用户的密码，必选项。
scope：权限范围，可选项。

（C）步骤：向客户端响应（B）步骤的数据

```
{ "access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"example", "expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA" }
```

参数说明
access_token：访问令牌，必选项。
token_type：令牌类型，该值大小写不敏感，必选项。
expires_in：过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
refresh_token：更新令牌，用来获取下一次的访问令牌，可选项。

使用场景

这种模式适用于用户对应用程序高度信任的情况。比如是用户操作系统的一部分。
认证服务器只有在其他授权模式无法执行的情况下，才能考虑使用这种模式。

客户端凭证模式（Client Credentials Grant）

客户端凭证模式（Client Credentials Grant）
流程介绍

（A）客户端向认证服务器进行身份认证，并要求一个访问令牌。
（B）认证服务器确认无误后，向客户端提供访问令牌。
请求示例
（A）步骤：客户端发送https请求

```
https://www.example.com/token?
grant_type=client_credentials&client_id=CLIENT_ID
```

参数说明
granttype：表示授权类型，此处的值固定为"client_credentials"，必选项。
scope：表示权限范围，可选项。

（B）步骤：向客户端响应（A）步骤的数据

```
{ "access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"example", "expires_in":3600,
"example_parameter":"example_value" }
```

参数说明：
access_token：访问令牌，必选项。
token_type：令牌类型，该值大小写不敏感，必选项。
expires_in：过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
example_parameter：其它参数，可选项。

使用场景

客户端模式应用于应用程序想要以自己的名义与授权服务器以及资源服务器进行互动。
例如使用了第三方的静态文件服务

刷新 TOKEN 方式

刷新TOKEN
从上面的四种授权流程可以看出，最终的目的是要获取用户的授权令牌（access_token）。而且授权令牌（access_token）的权限也非常之大，所以在协议中明确表示要设置授权令牌（access_token）的有效期。那么当授权令牌（access_token）过期要怎么办呢，协议里提出了一个刷新token的流程。
流程介绍

（A）--（D）通过授权流程获取access_token，并调用业务api接口。
（F）当调用业务api接口时响应“Invalid Token Error”时。
（G）调用刷新access_token接口，使用参数refresh_token（如果平台方提供，否则需要用户重新进行授权流程）。
（H）响应最新的access_token及refresh_token。
请求示例
（G）步骤：客户端调用刷新token接口

```
https://www.example.com/v1/oauth/token?
grant_type=refresh_token&client_id=CLIENT_ID&client_secret=CLIE
NT_SECRET&refresh_token=REFRESH_TOKEN
```

参数说明
client_id：客户端的ID，必选项。
client_secret：客户端的密钥，必选项。
grant_type：表示使用的授权模式，此处的值固定为"refreshtoken"，必选项。
refresh_token：表示早前收到的更新令牌，必选项。

（H）步骤：响应客户端数据

```
{ "access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"example", "expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
"example_parameter":"example_value" }
```

参数说明
access_token：访问令牌，必选项。
token_type：令牌类型，该值大小写不敏感，必选项。
expires_in：过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。
refresh_token：更新令牌，用来获取下一次的访问令牌，可选项。
scope：权限范围，如果与客户端申请的范围一致，此项可省略。

说明：建议将access_token和refresh_token的过期时间保存下来，每次调用平台方的业务api前先对access_token和refresh_token进行一下时间判断，如果过期则执行刷新access_token或重新授权操作。refersh_token如果过期就只能让用户重新授权。

好，到此oauth2.0的四种授权流程及令牌的刷新流程已经介绍完了，下面来从oauth2.0的安全性上来介绍一下。

5.2. Maven 依赖

```
 <dependency>

<groupId>org.springframework.security.experimental</groupId>
     <artifactId>spring-security-oauth2-authorization-
server</artifactId>
     <version>0.0.1</version>
 </dependency>
```

5.3. Spring cloud with Oauth2

authorization_code

验证服务器器

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
oauth2</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
thymeleaf</artifactId>
                </dependency>
```

```

```

```

```

```

```

测试

```
http://localhost:8080/oauth/authorize?
```

```
response_type=code&client_id=sso&redirect_uri=http://localhos
t:8082/callback&scope=read
http://localhost:8080/oauth/token?
client_id=sso&grant_type=authorization_code&redirect_uri=http
://localhost:8082/callback&code=ZzLi3w

localhost:8082/admin&code=ZzLi3w

localhost:8080/oauth/authorize?
response_type=code&client_id=client&redirect_uri=http://www.n
etkiller.cn&state=123

localhost:8080/oauth/authorize?
response_type=code&client_id=sso&redirect_uri=http://localhos
t:8082/test&scope=app




http://localhost:8080/oauth/token?
client_id=sso&grant_type=authorization_code&redirect_uri=http
://localhost:8082&code=8z5J9L

http://localhost:8080/oauth/authorize?
response_type=code&client_id=sso&redirect_uri=http://localhos
t:8080&scope=app

http://www.netkiller.cn/?code=eX6IMV&state=123
http://localhost:18084/oauth/token?
client_id=client&grant_type=authorization_code&redirect_uri=h
ttp://api.netkiller.cn&code=bzxoHn
```

Spring boot with Oauth2 - Password

下面例子由三个项目组成，分别是 tools, server, client。

其中 tools 是密码生成工具

https://tools.ietf.org/html/rfc6749

Maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>oauth</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>oauth</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.0.2.RELEASE</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>
        <dependencies>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
```

```
jpa</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <!-- Security -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.security.oauth</groupId>
                        <artifactId>spring-security-
oauth2</artifactId>
                </dependency>

        </dependencies>
        <modules>
```

```
                <module>server</module>
                <module>client</module>
        </modules>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                    <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

Password tools

Maven

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>cn.netkiller</groupId>
    <artifactId>oauth</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <artifactId>tools</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>tools</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  </properties>
  <dependencies>
```

```
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

下面的代码用于生成 Spring security 所用的密码

```
package cn.netkiller.oauth.tools;

import
org.springframework.security.crypto.bcrypt.BCryptPasswordEnco
der;

/**
 * Hello world!
 *
 */
public class App {
        public static void main(String[] args) {
                int i = 0;
                while (i < 10) {
                        String password = "123456";
                        BCryptPasswordEncoder passwordEncoder
= new BCryptPasswordEncoder();
                        String hashedPassword =
passwordEncoder.encode(password);

                        System.out.println(hashedPassword);
                        i++;
                }
        }
}
```

运行后生成类似的密码

```
$2a$10$IrDG5Yr3CGorEg9gKG8smeRLnNUieCVyBCJHA80U6h20HDFCxd43W
$2a$10$wseh5xFv1L3a3uHQId0MAOqAN0rKKcuX.RDaBPQ.pV/hsr80TqwxO
$2a$10$xP3Gc/5/PN03BdkDfhUjAemTRVaiwr0lsaqPqD18UI.ho9nRC/ebW
$2a$10$S.wLZ6e5YvmQA6mkX8yXWOdJbvahtDOesRu0ZwPOzAPCwpo7eDAsi
$2a$10$Jo/yuWyiAZ2Lj8.ywoPl7OeOJYuP7RVq8l.qc/zOwtW8MTFp3NYGO
$2a$10$eEvvjPok0fRK.DU6yF0qI.aucuiWr3y5G93SLq9/76ovcOwIuQAuS
$2a$10$BWEkANxbgwATNQCEI9/uNevNEUNlomGY7cZ2CQVm.qCRcnyukT.Si
$2a$10$69wSpyJQvjzJY7ou5PFWlOlEIecQukHV9WEq0nebsz5V6IZKfOVv2
$2a$10$Cyj3hM39V34r5pMeQ.Y9peuUqYMBSvsJ7GTBgp4.stWaTtWMboYGS
$2a$10$0/o4cRN2.tmnc58sH.N4WOsreVI6sWlPl4CCBrmUfJ332TMfRzA42
```

Server

Maven

```
<?xml version="1.0"?>
<project
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>oauth</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>server</artifactId>
        <name>server</name>
        <description>Example Spring Boot REST
project</description>

        <url>http://maven.apache.org</url>
```

```
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>

        </dependencies>

</project>
```

application.properties

```
server.port=8000
server.contextPath=/api

logging.level.com.gigy=DEBUG

# Data source properties
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/netkiller?
useSSL=false
spring.datasource.username=netkiller
spring.datasource.password=123123
spring.datasource.max-idle=10
spring.datasource.max-wait=10000
spring.datasource.min-idle=5
spring.datasource.initial-size=5
spring.datasource.validation-query=SELECT 1
spring.datasource.test-on-borrow=false
spring.datasource.test-while-idle=true
spring.datasource.time-between-eviction-runs-millis=18800
spring.datasource.jdbc-
interceptors=ConnectionState;SlowQueryReport(threshold=0)
```

```
spring.jpa.database=MYSQL
spring.jpa.show-sql=true
#spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.ddl-auto=create-drop
#spring.jpa.hibernate.ddl-auto=validate
#spring.jpa.show-sql=true
```

EnableAuthorizationServer

```
package cn.netkiller.oauth.server.config;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationMan
ager;
import
org.springframework.security.core.userdetails.UserDetailsServ
ice;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEnco
der;
import
org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.config.annotation.configu
rers.ClientDetailsServiceConfigurer;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.AuthorizationServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.EnableAuthorizationServer;
import
```

```
org.springframework.security.oauth2.config.annotation.web.con
figurers.AuthorizationServerEndpointsConfigurer;

@Configuration
@EnableAuthorizationServer
public class AuthorizationServerConfigurer extends
AuthorizationServerConfigurerAdapter {

        @Autowired
        @Qualifier("userDetailsService")
        private UserDetailsService userDetailsService;

        @Autowired
        private AuthenticationManager authenticationManager;

        @Value("${oauth.tokenTimeout:3600}")
        private int expiration;

        @Bean
        public PasswordEncoder passwordEncoder() {
                return new BCryptPasswordEncoder();
        }

        @Override
        public void
configure(AuthorizationServerEndpointsConfigurer configurer)
throws Exception {

configurer.authenticationManager(authenticationManager);

configurer.userDetailsService(userDetailsService);
        }

        @Override
        public void configure(ClientDetailsServiceConfigurer
clients) throws Exception {

clients.inMemory().withClient("api").secret("secret").accessT
okenValiditySeconds(expiration).scopes("read",
"write").authorizedGrantTypes("password",
"refresh_token").resourceIds("resource");
        }

}
```

Spring boot 2.0

```
        @Override
        public void configure(ClientDetailsServiceConfigurer
clients) throws Exception {

clients.inMemory().withClient("api").secret(passwordEncoder()
.encode("secret")).accessTokenValiditySeconds(expiration).sco
pes("read", "write").authorizedGrantTypes("password",
"refresh_token").resourceIds("resource");
        }
```

EnableResourceServer

```
package cn.netkiller.oauth.server.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationMan
ager;
import
org.springframework.security.config.annotation.method.configu
ration.EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
org.springframework.security.config.annotation.web.builders.W
ebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
```

```
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.EnableResourceServer;

@Configuration
@EnableWebSecurity
@EnableResourceServer
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfiguration extends
WebSecurityConfigurerAdapter {

        /**
         * Constructor disables the default security settings
         */
        public WebSecurityConfiguration() {
                super(true);
        }

        @Override
        public void configure(WebSecurity web) throws
Exception {
                web.ignoring().antMatchers("/login");
        }

        @Override
        public void configure(HttpSecurity http) throws
Exception {

http.antMatcher("/api/**").authorizeRequests().anyRequest().a
uthenticated();
        }

        @Bean
        @Override
        public AuthenticationManager
authenticationManagerBean() throws Exception {
                return super.authenticationManagerBean();
        }
}
```

Entity Table

```
package cn.netkiller.oauth.server.model;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.userdetails.UserDetails;

@Entity
@Table(name = "users")
public class User implements UserDetails {

        static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Column(name = "user_id", nullable = false, updatable
= false)
        private Long id;

        @Column(name = "username", nullable = false, unique =
true)
        private String username;

        @Column(name = "password", nullable = false)
        private String password;

        @Column(name = "enabled", nullable = false)
        private boolean enabled;

        public Collection<? extends GrantedAuthority>
```

```
getAuthorities() {
                List<GrantedAuthority> authorities = new
ArrayList<GrantedAuthority>();

                return authorities;
        }

        public boolean isAccountNonExpired() {
                return true;
        }

        public boolean isAccountNonLocked() {
                // we never lock accounts
                return true;
        }

        public boolean isCredentialsNonExpired() {
                // credentials never expire
                return true;
        }

        public boolean isEnabled() {
                return enabled;
        }

        public String getPassword() {
                return password;
        }

        public String getUsername() {
                return username;
        }

}
```

UserRepository

```
package cn.netkiller.oauth.server.repository;
```

```
import org.springframework.data.jpa.repository.JpaRepository;

import cn.netkiller.oauth.server.model.User;

public interface UserRepository extends JpaRepository<User,
Long> {

        /**
         * Find a user by username
         *
         * @param username
         *            the user's username
         * @return user which contains the user with the
given username or null.
         */
        User findOneByUsername(String username);

}
```

UserService

```
package cn.netkiller.oauth.server.service;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsServ
ice;
import
org.springframework.security.core.userdetails.UsernameNotFoun
dException;
import org.springframework.stereotype.Service;

import cn.netkiller.oauth.server.repository.UserRepository;

@Service("userDetailsService")
public class UserService implements UserDetailsService {
```

```
        @Autowired
        private UserRepository userRepository;

        public UserDetails loadUserByUsername(String
username) throws UsernameNotFoundException {

                return
userRepository.findOneByUsername(username);
        }
}
```

TestRestController

```
package cn.netkiller.oauth.server.controller;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.access.prepost.PreAuthorize;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/test")
public class TestRestController {

        @RequestMapping(value = "hello", method =
RequestMethod.GET)
        public ResponseEntity<String> hello() {
                return new ResponseEntity<String>("Hello
world !", HttpStatus.OK);
        }

        @PreAuthorize("#oauth2.hasScope('write')")
        @RequestMapping(value = "set/{string}", method =
```

```
RequestMethod.GET)
        public ResponseEntity<String> set(String string) {
                return new ResponseEntity<String>(string,
HttpStatus.OK);
        }
}
```

数据库初始化

在 src/main/resources 目录创建 data.sql 文件

```
INSERT INTO users (user_id, username, password, enabled)
VALUES
        ('1', 'netkiller@msn.com',
'$2a$10$Cyj3hM39V34r5pMeQ.Y9peuUqYMBSvsJ7GTBgp4.stWaTtWMboYGS
', true);
```

此处密码 $2a$10$Cyj3hM39V34r5pMeQ.Y9peuUqYMBSvsJ7GTBgp4.stWaTtWMboYGS 请使用上面提供的工具生成。

Test

启动 Spring boot Server 项目

```
mvn spring-boot:run
```

启动后 Spring boot 会导入 data.sql 文件

```
mysql> select * from users where username="netkiller@msn.com";
+---------+---------+-----------------------------------------
--------------------+------------------+
| user_id | enabled | password
| username          |
+---------+---------+-----------------------------------------
--------------------+------------------+
|       4 |         |
$2a$10$Cyj3hM39V34r5pMeQ.Y9peuUqYMBSvsJ7GTBgp4.stWaTtWMboYGS |
netkiller@msn.com |
+---------+---------+-----------------------------------------
--------------------+------------------+
1 row in set (0.00 sec)
```

```
MacBook-Pro:Application neo$ curl -X POST --user 'api:secret'
-d
'grant_type=password&username=netkiller@msn.com&password=1234
56' http://localhost:8000/api/oauth/token
{"access_token":"5bc0ee89-cd6d-47be-b31f-
89c9e028159b","token_type":"bearer","refresh_token":"5107c09b
-de85-4faf-8396-941572cf30d2","expires_in":3599,"scope":"read
write"}MacBook-Pro:Application neo$

MacBook-Pro:Application neo$ curl -H "Accept:
application/json" -H "Content-Type: application/json" -H
"Authorization: Bearer 5bc0ee89-cd6d-47be-b31f-89c9e028159b"
-X GET http://localhost:8000/api/test/hello
Hello world !
```

Spring boot with Oauth2 RestTemplate

Maven

```
                <dependency>

<groupId>org.springframework.security.oauth</groupId>
                        <artifactId>spring-security-
oauth2</artifactId>
                </dependency>
```

OAuth2ClientConfiguration.java

```
package cn.netkiller.config;

import static java.util.Arrays.asList;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.oauth2.client.DefaultOAuth2Clien
tContext;
import
org.springframework.security.oauth2.client.OAuth2RestOperatio
ns;
import
org.springframework.security.oauth2.client.OAuth2RestTemplate
;
import
org.springframework.security.oauth2.client.resource.OAuth2Pro
tectedResourceDetails;
import
org.springframework.security.oauth2.client.token.AccessTokenR
equest;
import
org.springframework.security.oauth2.client.token.DefaultAcces
sTokenRequest;
import
org.springframework.security.oauth2.client.token.grant.passwo
rd.ResourceOwnerPasswordResourceDetails;
```

```
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.EnableOAuth2Client;

@EnableOAuth2Client
@Configuration
public class OAuth2ClientConfiguration {

        @Value("${oauth.resource:http://localhost:8080}")
        private String baseUrl;

@Value("${oauth.authorize:http://localhost:8080/oauth/authori
ze}")
        private String authorizeUrl;

@Value("${oauth.token:http://localhost:8080/oauth/token}")
        private String tokenUrl;

        @Bean
        protected OAuth2ProtectedResourceDetails resource() {

                ResourceOwnerPasswordResourceDetails resource
= new ResourceOwnerPasswordResourceDetails();

                resource.setAccessTokenUri(tokenUrl);
                resource.setClientId("api");
                resource.setClientSecret("secret");
                resource.setGrantType("password");
                resource.setScope(asList("read", "write"));

                resource.setUsername("netkiller@msn.com");
                resource.setPassword("123456");

                return resource;
        }

        @Bean
        public OAuth2RestOperations restTemplate() {
                AccessTokenRequest accessTokenRequest = new
DefaultAccessTokenRequest();
                return new OAuth2RestTemplate(resource(), new
DefaultOAuth2ClientContext(accessTokenRequest));
        }

}
```

Application.java

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

        public static void main(String[] args) {

                System.out.println("Spring boot with
Oauth2RestTemplate!");
                SpringApplication.run(Application.class,
args);
        }
}
```

application.properties

```
security.basic.enabled=false
```

Controller

```
package cn.netkiller.controller;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.oauth2.client.OAuth2RestOperatio
ns;
import
org.springframework.security.oauth2.common.OAuth2AccessToken;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class TestController {
        @Autowired
        private OAuth2RestOperations restTemplate;

        @GetMapping("/")
        @ResponseBody
        public String index() {
                OAuth2AccessToken token =
restTemplate.getAccessToken();
                System.out.println(token.getValue());
                String tmp =
restTemplate.getForObject("http://api.netkiller.cn/test.json"
, String.class);
                System.out.println(tmp);
                return tmp;
        }
}
```

Test

```
neo@MacBook-Pro ~/deployment % curl http://localhost:8080

OK
```

Spring boot with Oauth2 jwt

Maven

```
                <dependency>
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>

                <dependency>
<groupId>org.springframework.security.oauth</groupId>
                        <artifactId>spring-security-
oauth2</artifactId>
                </dependency>

                <dependency>
<groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-
jwt</artifactId>
                </dependency>
```

如果是 Springboot 2.0.4 需要制定版本号

```
                <dependency>
<groupId>org.springframework.security.oauth</groupId>
                        <artifactId>spring-security-
oauth2</artifactId>
                        <version>2.3.3.RELEASE</version>
                </dependency>
```

```
                <dependency>

<groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-
jwt</artifactId>
                        <version>1.0.9.RELEASE</version>
                </dependency>

                <dependency>
                        <groupId>javax.xml.bind</groupId>
                        <artifactId>jaxb-api</artifactId>
                </dependency>
                <dependency>
                        <groupId>com.sun.xml.bind</groupId>
                        <artifactId>jaxb-impl</artifactId>
                        <version>2.3.0</version>
                </dependency>
                <dependency>
                        <groupId>com.sun.xml.bind</groupId>
                        <artifactId>jaxb-core</artifactId>
                        <version>2.3.0</version>
                </dependency>
                <dependency>
                        <groupId>com.sun.activation</groupId>

<artifactId>javax.activation</artifactId>
                        <version>1.2.0</version>
                </dependency>
```

Authorization Server

```
package api.config;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.beans.factory.annotation.Qualifier;
```

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import
org.springframework.security.authentication.AuthenticationMan
ager;
import
org.springframework.security.core.userdetails.UserDetailsServ
ice;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEnco
der;
import
org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.config.annotation.configu
rers.ClientDetailsServiceConfigurer;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.AuthorizationServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.EnableAuthorizationServer;
import
org.springframework.security.oauth2.config.annotation.web.con
figurers.AuthorizationServerEndpointsConfigurer;
import
org.springframework.security.oauth2.provider.token.DefaultTok
enServices;
import
org.springframework.security.oauth2.provider.token.TokenStore
;
import
org.springframework.security.oauth2.provider.token.store.JwtA
ccessTokenConverter;
import
org.springframework.security.oauth2.provider.token.store.JwtT
okenStore;

@Configuration
@EnableAuthorizationServer
public class AuthorizationServerConfigurer extends
AuthorizationServerConfigurerAdapter {
```

```
        @Autowired
        @Qualifier("userDetailsService")
        private UserDetailsService userDetailsService;

        @Autowired
        private AuthenticationManager authenticationManager;

        @Value("${oauth.tokenTimeout:3600}")
        private int expiration;

        @Bean
        public PasswordEncoder passwordEncoder() {
                return new BCryptPasswordEncoder();
        }

        // @Override
        // public void
configure(AuthorizationServerSecurityConfigurer oauthServer)
throws Exception {
        // oauthServer.tokenKeyAccess("isAnonymous() ||
hasAuthority('ROLE_TRUSTED_CLIENT')");
        //
oauthServer.checkTokenAccess("hasAuthority('ROLE_TRUSTED_CLIE
NT')");
        // }

        @Override
        public void configure(ClientDetailsServiceConfigurer
clients) throws Exception {

clients.inMemory().withClient("api").secret(passwordEncoder()
.encode("secret")).accessTokenValiditySeconds(expiration).ref
reshTokenValiditySeconds(expiration).scopes("read",
"write").authorizedGrantTypes("password",
"refresh_token").authorities("USER").resourceIds("blockchain"
);
        }

        @Override
        public void
configure(AuthorizationServerEndpointsConfigurer configurer)
throws Exception {

configurer.tokenStore(tokenStore()).accessTokenConverter(acce
ssTokenConverter());
```

```
configurer.authenticationManager(authenticationManager);

configurer.userDetailsService(userDetailsService);
        }

        @Bean
        public TokenStore tokenStore() {
                return new
JwtTokenStore(accessTokenConverter());
        }

        @Bean
        public JwtAccessTokenConverter accessTokenConverter()
{
                JwtAccessTokenConverter converter = new
JwtAccessTokenConverter();
                converter.setSigningKey("123");
                return converter;
        }

        @Bean
        @Primary
        public DefaultTokenServices tokenServices() {
                DefaultTokenServices defaultTokenServices =
new DefaultTokenServices();

defaultTokenServices.setTokenStore(tokenStore());

defaultTokenServices.setSupportRefreshToken(true);
                return defaultTokenServices;
        }

}
```

Resource Server

```
package api.config;
```

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.EnableResourceServer;
import
org.springframework.security.oauth2.config.annotation.web.con
figuration.ResourceServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.con
figurers.ResourceServerSecurityConfigurer;
import
org.springframework.security.oauth2.provider.token.DefaultTok
enServices;
import
org.springframework.security.oauth2.provider.token.TokenStore
;
import
org.springframework.security.oauth2.provider.token.store.JwtA
ccessTokenConverter;
import
org.springframework.security.oauth2.provider.token.store.JwtT
okenStore;

@Configuration
@EnableResourceServer
public class ResourceServerConfigurer extends
ResourceServerConfigurerAdapter {
        public ResourceServerConfigurer() {
                // TODO Auto-generated constructor stub
        }

        @Override
        public void
configure(ResourceServerSecurityConfigurer resources) {
                resources.resourceId("blockchain");
                resources.tokenServices(tokenServices());

        }

        @Bean
        public TokenStore tokenStore() {
                return new
JwtTokenStore(accessTokenConverter());
```

```
        }

        @Bean
        public JwtAccessTokenConverter accessTokenConverter()
{
                JwtAccessTokenConverter converter = new
JwtAccessTokenConverter();
                converter.setSigningKey("123");
                return converter;
        }

        @Bean
        @Primary
        public DefaultTokenServices tokenServices() {
                DefaultTokenServices defaultTokenServices =
new DefaultTokenServices();

defaultTokenServices.setTokenStore(tokenStore());
                return defaultTokenServices;
        }

}
```

Web Security

```
package api.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.authentication.AuthenticationMan
ager;
import
org.springframework.security.config.annotation.method.configu
ration.EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
```

```
import
org.springframework.security.config.annotation.web.builders.W
ebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfiguration extends
WebSecurityConfigurerAdapter {

        @Override
        public void configure(WebSecurity web) throws
Exception {
                web.ignoring().antMatchers("/login");
        }

        @Override
        public void configure(HttpSecurity http) throws
Exception {
                //
http.antMatcher("/**").authorizeRequests().anyRequest().authe
nticated();

http.authorizeRequests().antMatchers(HttpMethod.OPTIONS).perm
itAll().anyRequest().authenticated().and().httpBasic().and().
csrf().disable();
        }

        @Bean
        @Override
        public AuthenticationManager
authenticationManagerBean() throws Exception {
                return super.authenticationManagerBean();
        }

}
```

插入数据

```
INSERT INTO `user` (username, password, enabled) VALUES
('netkiller@msn.com',
'$2a$10$Cyj3hM39V34r5pMeQ.Y9peuUqYMBSvsJ7GTBgp4.stWaTtWMboYGS',
true);
```

使用 **CURL** 测试 **JWT**

```
neo@MacBook-Pro ~ % curl -X POST --user 'api:secret' -d
'grant_type=password&username=netkiller@msn.com&password=123456
' http://localhost:8080/oauth/token
{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiO
lsicmVzb3VyY2UiXSwiZXhwIjoxNTM1MTE4MzYxLCJ1c2VyX25hbWUiOiJuZXRr
aWxsZXJAbXNuLmNvbSIsImp0aSI6ImM5YzA4ODczLWZlMTctNGM2My05MDA1LTI
zZGJlNTE1NTQ0YiIsImNsaWVudF9pZCI6ImFwaSIsInNjb3BlIjpbInJlYWQiLC
J3cml0ZSJdfQ.ydxJQKtdBNWHELL8_aXVoZETNMygXs8T1HQ5XWDBELA","toke
n_type":"bearer","refresh_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6I
kpXVCJ9.eyJhdWQiOlsicmVzb3VyY2UiXSwidXNlcl9uYW1lIjoibmV0a2lsbGV
yQG1zbi5jb20iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiYXRpIjoiYzljMD
g4NzMtZmUxNy00YzYzLTkwMDUtMjNkYmU1MTU1NDRiIiwiZXhwIjoxNTM1MTE4M
zYxLCJqdGkiOiI3ODJiNmY2NC0xYzdiLTQ0YWYtOThlZC1iZDk3Y2QzYzE1NjEi
LCJjbGllbnRfaWQiOiJhcGkifQ.16QUHOrVPUBF97E_972AcLA83zEK7UzaI424
PeAmX6E","expires_in":3599,"scope":"read
write","jti":"c9c08873-fe17-4c63-9005-23dbe515544b"}
```

复制 access_token 粘贴到 Authorization: Bearer 后面

```
neo@MacBook-Pro ~ % curl -H "Accept: application/json" -H
"Content-Type: application/json" -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsiYmxvY2tjaGFpbi
JdLCJleHAiOjE1MzUxMTkyMTUsInVzZXJfbmFtZSI6Im5ldGtpbGxlckBtc24uY
29tIiwianRpIjoiZTNjYmIxNTItYmFkZS00MGM1LTkxOGItZjJjMzZiYTBiMTE2
```

```
IiwiY2xpZW50X2lkIjoiYXBpIiwic2NvcGUiOlsicmVhZCIsIndyaXRlIl19.op
hWAvIT1ZmWCIQyAgyuzmQ98TJsN49OeR3CBSJ_X54" -X GET
http://localhost:8080/test/mongo
{"id":"5b800709e18a211e83c7f355","name":"Netkiller"}
```

测试 Shell

```
URL=http://localhost:8080
TOKEN=$(curl -s  -X POST --user 'api:secret' -d
'grant_type=password&username=netkiller@msn.com&password=123456
' ${URL}/oauth/token | sed 's/.*"access_token":"\
([^"]*\)".*/\1/g')

curl -s -k -H "Accept: application/json" -H "Content-Type:
application/json" -H "Authorization: Bearer ${TOKEN}" -X GET
${URL}/test/hello.json
```

refresh_token

```
curl -s  -X POST --user 'api:secret' -d
'grant_type=refresh_token&client_id=api&client_secret=secret&re
fresh_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsiYm
xvY2tjaGFpbiJdLCJ1c2VyX25hbWUiOiJibG9ja2NoYWluIiwic2NvcGUiOlsic
mVhZCIsIndyaXRlIl0sImF0aSI6ImU2OGE4ODUzLWJlODUtNGUwNy1hYzE0LTM0
MDI1ZDBiZGY0ZiIsImV4cCI6MTU0MDM4MzA5NCwianRpIjoiY2JjNTk2ZWMtOTk
yZi00NmQ2LWFiZGYtMTcyYWFiZmEwNDFiIiwiY2xpZW50X2lkIjoiYXBpIn0.Pq
sGl5Dm8WBqwbhHf-LqmUP1toCpsFS4gLeOP2bMwR4' ${URL}/oauth/token
```

Spring boot with Oauth2 jwt 非对称证书

创建证书

创建证书 keytool -genkeypair -alias jwt -keyalg RSA -keypass passw0rd -keystore jwt.jks -storepass passw0rd

```
neo@MacBook-Pro /tmp/oauth % keytool -genkeypair -alias jwt -
keyalg RSA -keypass passw0rd -keystore jwt.jks -storepass
passw0rd
What is your first and last name?
  [Unknown]:  Neo Chen
What is the name of your organizational unit?
  [Unknown]:  netkiller.cn
What is the name of your organization?
  [Unknown]:  netkiller.cn
What is the name of your City or Locality?
  [Unknown]:  Shenzhen
What is the name of your State or Province?
  [Unknown]:  Guangdong
What is the two-letter country code for this unit?
  [Unknown]:  CN
Is CN=Neo Chen, OU=netkiller.cn, O=netkiller.cn, L=Shenzhen,
ST=Guangdong, C=CN correct?
  [no]:  yes
```

该命令将生成一个名为jwt.jks的文件，其中包含我们的密钥 - 公钥和私钥。 还要牢记keypass和storepass密码。

导出公钥，接下来，我们需要从刚刚生成的JKS中导出我们的公钥，我们可以使用下面的命令来实现：

```
neo@MacBook-Pro /tmp/oauth % keytool -list -rfc --keystore
jwt.jks | openssl x509 -inform pem -pubkey -out certificate.crt
> public.crt

Enter keystore password:  passw0rd
```

公钥内容

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAj6ePdDwBrHKX3kNFnbv
e
T1rTTbyA9GjaiZNwj2X4Y0In7RCFl8auXXBn2DxztQMGqHY2Ydc3/26Gu9Vri44
1
r8/RInA6UpzzDRl5SeYYTobcgfIVpfQ0hTX0xzuMDVLVoLibGfcvGy7ZkrJjQFX
8
lIaO84K8KP/yzma5622XJ+f5hkXmTX5e0tXGDCPjVO1dSrouPWqhcbM0Kf6y3Rd
E
JkNRTHLky6afx8MNobakz1Ab9K7cjD8De6LwScwMQMFU46traN/3Fw0lZFxKkpa
y
+sEUHvHDUYWTuVovUmfiKMX8fj5QCm4imPdA3pF/jjM+xeeVcTID3qffDGOKrGT
F
HQIDAQAB
-----END PUBLIC KEY-----
```

复制 jwt.jks 和 public.crt 到 src/main/resources 目录下

Authorization Server

```
        @Bean
        public JwtAccessTokenConverter accessTokenConverter()
{
                JwtAccessTokenConverter converter = new
JwtAccessTokenConverter();
                KeyStoreKeyFactory keyStoreKeyFactory = new
KeyStoreKeyFactory(new ClassPathResource("jwt.jks"),
"passw0rd".toCharArray());

converter.setKeyPair(keyStoreKeyFactory.getKeyPair("passw0rd"
));
                return converter;
        }
```

Resource Server

```
        @Bean
        public JwtAccessTokenConverter accessTokenConverter()
{
                JwtAccessTokenConverter converter = new
JwtAccessTokenConverter();

                String publicKey = "-----BEGIN PUBLIC KEY----
-\n" +

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAj6ePdDwBrHKX3kNF
nbve\n" +

"T1rTTbyA9GjaiZNwj2X4Y0In7RCFl8auXXBn2DxztQMGqHY2Ydc3/26Gu9Vr
i441\n" +

"r8/RInA6UpzzDRl5SeYYTobcgfIVpfQ0hTX0xzuMDVLVoLibGfcvGy7ZkrJj
QFX8\n" +

"lIaO84K8KP/yzma5622XJ+f5hkXmTX5e0tXGDCPjVO1dSrouPWqhcbM0Kf6y
3RdE\n" +

"JkNRTHLky6afx8MNobakz1Ab9K7cjD8De6LwScwMQMFU46traN/3Fw0lZFxK
kpay\n" +

"+sEUHvHDUYWTuVovUmfiKMX8fj5QCm4imPdA3pF/jjM+xeeVcTID3qffDGOK
rGTF\n" +
                                    "HQIDAQAB\n" +
                                    "-----END PUBLIC KEY-----";

                converter.setVerifierKey(publicKey);
                return converter;
        }
```

Apple iOS 访问 Oauth2

```
//
//  AppDelegate.m
//
//  Created by Apple on 2018/9/8.
//  Copyright © 2018年 Apple. All rights reserved.
//

#import "AppDelegate.h"
#import "YTKNetworkConfig.h"
#import "AFOAuth2Manager.h"
#import <AFNetworking.h>
#import "NSAppConfig.h"


@interface AppDelegate ()
@property (nonatomic, strong) AFOAuthCredential *credential;
@property (nonatomic, strong) AFOAuth2Manager *OAuth2Manager;
@end

@implementation AppDelegate


- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    //授权
    NSURL *url = [NSURL URLWithString:BASE_URL];

    AFOAuth2Manager *OAuth2Manager = [[AFOAuth2Manager
alloc]initWithBaseURL:url clientID:CLIENTID secret:SECRET];
   self.OAuth2Manager = OAuth2Manager;
    [OAuth2Manager
authenticateUsingOAuthWithURLString:OAUTH_TOKEN

username:OAUTH_USERNAME

password:OAUTH_PASSWORD
                                                    scope:@""
    success:^(AFOAuthCredential *credential) {
```

```
      // NSLog(@"********请求OauthToek成功***********");
        self.credential = credential;
        [self testPost];
        [self testGet];
        }
    failure:^(NSError *error) {
     //  NSLog(@"********请求OauthToekn失败
begin***********\r\n%@\r\n*********请求OauthToekn失败
end************",error);

    }];


//    YTKNetworkConfig *config = [YTKNetworkConfig
sharedConfig]; config.baseUrl = @"http://yuantiku.com";
//
    // Override point for customization after application
launch.
    return YES;
}

- (void)testGet{
    NSURL *baseURL = [NSURL
URLWithString:@"http://192.168.0.185:8080"];
    AFHTTPSessionManager *manager =
    [[AFHTTPSessionManager alloc] initWithBaseURL:baseURL];

    [manager.requestSerializer
setAuthorizationHeaderFieldWithCredential:self.credential];

    [manager GET:@"/test/hello"
      parameters:nil
        progress:nil
         success:^(NSURLSessionDataTask * _Nonnull task, id
_Nullable responseObject) {
             NSLog(@"Success: %@", responseObject);
         }
         failure:^(NSURLSessionDataTask * _Nullable task,
NSError * _Nonnull error) {
             NSLog(@"Failure: %@", error);
         }];

}

- (void)testPost{
```

```
    NSURL *baseURL = [NSURL
URLWithString:@"http://192.168.0.185:8080"];
    AFHTTPSessionManager *manager =
    [[AFHTTPSessionManager alloc] initWithBaseURL:baseURL];
    manager.responseSerializer = [AFJSONResponseSerializer
serializer];
    manager.requestSerializer = [AFJSONRequestSerializer
serializer];
    [manager.requestSerializer
setAuthorizationHeaderFieldWithCredential:self.credential];

    NSDictionary *dic = @{@"mobile":@"13113676543",
                          @"password":@"123456",
                          @"role":@"Organization"
                          };

    [manager POST:@"/member/create"
      parameters:dic
        progress:nil
         success:^(NSURLSessionDataTask * _Nonnull task, id
_Nullable responseObject) {
             NSLog(@"Success: %@", responseObject);
         }
         failure:^(NSURLSessionDataTask * _Nullable task,
NSError * _Nonnull error) {
             NSLog(@"Failure: %@ %@", error,task);
         }];

}

- (void)applicationWillResignActive:(UIApplication
*)application {
    // Sent when the application is about to move from active
to inactive state. This can occur for certain types of
temporary interruptions (such as an incoming phone call or
SMS message) or when the user quits the application and it
begins the transition to the background state.
    // Use this method to pause ongoing tasks, disable
timers, and invalidate graphics rendering callbacks. Games
should use this method to pause the game.
}


- (void)applicationDidEnterBackground:(UIApplication
*)application {
```

```
    // Use this method to release shared resources, save user
data, invalidate timers, and store enough application state
information to restore your application to its current state
in case it is terminated later.
    // If your application supports background execution,
this method is called instead of applicationWillTerminate:
when the user quits.
}


- (void)applicationWillEnterForeground:(UIApplication
*)application {
    // Called as part of the transition from the background
to the active state; here you can undo many of the changes
made on entering the background.
}


- (void)applicationDidBecomeActive:(UIApplication
*)application {
    // Restart any tasks that were paused (or not yet
started) while the application was inactive. If the
application was previously in the background, optionally
refresh the user interface.
}


- (void)applicationWillTerminate:(UIApplication *)application
{
    // Called when the application is about to terminate.
Save data if appropriate. See also
applicationDidEnterBackground:.
}


@end
```

Post 出去的数据是 Raw 格式。

```
AFHTTPRequestOperationManager *manager =
[AFHTTPRequestOperationManager manager];
//申明返回的结果是json类型
manager.responseSerializer = [AFJSONResponseSerializer
serializer];
//申明请求的数据是json类型
manager.requestSerializer=[AFJSONRequestSerializer
serializer];
//如果报接受类型不一致请替换一致text/html或别的
manager.responseSerializer.acceptableContentTypes = [NSSet
setWithObject:@"text/html"];
//传入的参数
NSDictionary *parameters =
@{@"1":@"XXXX",@"2":@"XXXX",@"3":@"XXXXX"};
//你的接口地址
NSString *url=@"http://xxxxx";
//发送请求
[manager POST:url parameters:parameters
success:^(AFHTTPRequestOperation *operation, id
responseObject) {
    NSLog(@"JSON: %@", responseObject);
} failure:^(AFHTTPRequestOperation *operation, NSError
*error) {
    NSLog(@"Error: %@", error);
}];
```

Oauth2 客户端

环境：Java11 + Springboot 2.1.3

```
<?xml version="1.0"?>
<project
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>oauth2-client</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>oauth2-client</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
oauth2-client</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
</project>
```

application.yml

```
```

```
server:
  port: 8080

APP-CLIENT-ID: 180579ba67875ffc18e3
APP-CLIENT-SECRET: 8175af8f5691e2f3b6007b9597d8c1e3499e15a0

spring:
#  redis:
#    host: localhost
  security:
    oauth2:
      client:
        provider:
          netkiller:
            authorization-uri:
http://localhost:8080/oauth/authorize
            token-uri: http://localhost:8080/oauth/token
            user-info-uri: http://localhost:8080/user
            user-name-attribute: name
#            token-endpoint-url:
http://localhost:8080/oauth/check_token
          yahoo-oidc:
            issuer-uri: https://api.login.yahoo.com
        registration:
          netkiller:
            client-id: sso
            client-secret: secret
            client-name: Neo
            provider: netkiller
            scope: read
            authorization-grant-type: authorization_code
            redirect-uri:
http://localhost:${server.port}/login/oauth2/code/netkiller
          github-client-1:
            client-id: ${APP-CLIENT-ID}
            client-secret: ${APP-CLIENT-SECRET}
            client-name: Github user
            provider: github
            scope: user
            redirect-uri:
http://localhost:${server.port}/login/oauth2/code/github
          github-client-2:
            client-id: ${APP-CLIENT-ID}
            client-secret: ${APP-CLIENT-SECRET}
            client-name: Github email
```

```
            provider: github
            scope: user:email
            redirect-uri:
http://localhost:${server.port}/login/oauth2/code/github
          yahoo-oidc:
            client-id: ${YAHOO-CLIENT-ID}
            client-secret: ${YAHOO-CLIENT-SECRET}
          github-repos:
            client-id: ${APP-CLIENT-ID}
            client-secret: ${APP-CLIENT-SECRET}
            scope: public_repo
            redirect-uri: "{baseUrl}/github-repos"
            provider: github
            client-name: GitHub Repositories


logging:
  level:
#    org.springframework.web: DEBUG
    org.springframework.security: DEBUG
```

SpringApplication

```
package cn.netkiller.oauth2.client;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguratio
n;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@EnableAutoConfiguration
public class Application {
        public static void main(String[] args) {
                System.out.println("Oauth2 Client!");
                SpringApplication.run(Application.class,
```

```
args);
        }
}
```

WebSecurityConfigurer

```
package cn.netkiller.oauth2.client;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.H
ttpSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configurat
ion.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends
WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http) throws
Exception {

http.authorizeRequests().anyRequest().authenticated().and().o
auth2Login();
        }
}
```

TestController

```
package cn.netkiller.oauth2.client;

import java.security.Principal;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.oauth2.client.registration.Clien
tRegistration;
import
org.springframework.security.oauth2.client.registration.Clien
tRegistrationRepository;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class TestController {
        @Autowired
        private ClientRegistrationRepository
clientRegistrationRepository;

        @RequestMapping("/")
        public Principal email(Principal principal) {
                System.out.println("Hello " +
principal.getName());

                return principal;
        }

        @RequestMapping("/index")
        public ClientRegistration index() {
                ClientRegistration clientRegistration =
this.clientRegistrationRepository.findByRegistrationId("netki
ller");

                return clientRegistration;
        }
}
```

Android Oauth2 + Jwt example

```
package cn.netkiller.okhttp.pojo;

public class Oauth {
    private String access_token;
    private String token_type;
    private String refresh_token;
    private String expires_in;
    private String scope;
    private String jti;

    public String getAccess_token() {
        return access_token;
    }

    public void setAccess_token(String access_token) {
        this.access_token = access_token;
    }

    public String getToken_type() {
        return token_type;
    }

    public void setToken_type(String token_type) {
        this.token_type = token_type;
    }

    public String getRefresh_token() {
        return refresh_token;
    }

    public void setRefresh_token(String refresh_token) {
        this.refresh_token = refresh_token;
    }

    public String getExpires_in() {
        return expires_in;
```

```
    }

    public void setExpires_in(String expires_in) {
        this.expires_in = expires_in;
    }

    public String getScope() {
        return scope;
    }

    public void setScope(String scope) {
        this.scope = scope;
    }

    public String getJti() {
        return jti;
    }

    public void setJti(String jti) {
        this.jti = jti;
    }

    @Override
    public String toString() {
        return "Oauth{" +
                "access_token='" + access_token + '\'' +
                ", token_type='" + token_type + '\'' +
                ", refresh_token='" + refresh_token + '\'' +
                ", expires_in='" + expires_in + '\'' +
                ", scope='" + scope + '\'' +
                ", jti='" + jti + '\'' +
                '}';
    }
}
```

Activity 文件

```
package cn.netkiller.okhttp;
```

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import com.google.gson.Gson;

import java.io.IOException;

import cn.netkiller.okhttp.pojo.Oauth;
import okhttp3.Authenticator;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.Credentials;
import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import okhttp3.Route;

public class Oauth2jwtActivity extends AppCompatActivity {

    private TextView token;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_oauth2jwt);

        token = (TextView) findViewById(R.id.token);

        try {
            get();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }


    public static Oauth accessToken() throws IOException {


        OkHttpClient client = new
```

```
OkHttpClient.Builder().authenticator(
                new Authenticator() {
                    public Request authenticate(Route route,
Response response) {
                        String credential =
Credentials.basic("api", "secret");
                        return
response.request().newBuilder().header("Authorization",
credential).build();
                    }
                }
        ).build();

        String url = "https://api.netkiller.cn/oauth/token";

        RequestBody formBody = new FormBody.Builder()
                .add("grant_type", "password")
                .add("username", "blockchain")
                .add("password", "123456")
                .build();

        Request request = new Request.Builder()
                .url(url)
                .post(formBody)
                .build();

        Response response =
client.newCall(request).execute();
        if (!response.isSuccessful()) {
            throw new IOException("服务器端错误: " + response);
        }

        Gson gson = new Gson();
        Oauth oauth = gson.fromJson(response.body().string(),
Oauth.class);
        Log.i("oauth", oauth.toString());
        return oauth;
    }

    public void get() throws IOException {


        OkHttpClient client = new
OkHttpClient.Builder().authenticator(
                new Authenticator() {
```

```
                    public Request authenticate(Route route,
Response response) throws IOException {
                        return
response.request().newBuilder().header("Authorization",
"Bearer " + accessToken().getAccess_token()).build();
                    }
                }
        ).build();

        String url =
"https://api.netkiller.cn/misc/compatibility";

        Request request = new Request.Builder()
                .url(url)
                .build();

        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                call.cancel();
            }

            @Override
            public void onResponse(Call call, Response
response) throws IOException {

                final String myResponse =
response.body().string();

                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Log.i("oauth", myResponse);
                        token.setText(myResponse);
                    }
                });

            }
        });
    }

}
```

RestTemplate 使用 HttpClient

Maven

```
<?xml version="1.0"?>
<project
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>oauth</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>client</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>client</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.apache.httpcomponents</groupId>
                        <artifactId>httpclient</artifactId>
                        <version>4.5.3</version>
                </dependency>
        </dependencies>
</project>
```

SpringBootApplication

```
package cn.netkiller.oauth.client;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * Hello world!
 *
 */
@SpringBootApplication
public class Application {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                SpringApplication.run(Application.class,
args);
        }
}
```

ClientRestController

```
package cn.netkiller.oauth.client.controller;

import org.apache.http.impl.client.DefaultHttpClient;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.oauth2.client.OAuth2RestOperatio
ns;
import
```

```
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

import com.fasterxml.jackson.databind.JsonNode;

@RestController
public class ClientRestController {

        @RequestMapping("/token")
        public String token() {
            HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_JSON);
            headers.set("Authorization","Bearer "+"6296057a-
ed06-4899-9adc-1993ba7a4946");
            HttpEntity<String> entity = new
HttpEntity<String>(headers);
            ResponseEntity<String> response =
restTemplate.exchange("http://localhost:8000/api/test/hello.j
son",HttpMethod.GET,entity,String.class);
            System.out.println(response.getBody());
                return response.getStatusCode().toString() +
" " + response.getBody();
        }
}
```

Test

首先获取 Token

```
MacBook-Pro:Application neo$ curl -X POST --user 'api:secret' -
d
'grant_type=password&username=netkiller@msn.com&password=123456
' http://localhost:8000/api/oauth/token
{"access_token":"6296057a-ed06-4899-9adc-
1993ba7a4946","token_type":"bearer","refresh_token":"b22d70db-
3253-4f5f-9b6a-8714da23e14d","expires_in":2642,"scope":"read
```

```
write"}
```

将Token写入到 http 头

```
headers.set("Authorization","Bearer "+"6296057a-ed06-4899-9adc-
1993ba7a4946");
```

启动 client 项目

```
                                        mvn spring-boot:run
```

curl 测试

```
                                        MacBook-Pro:Application
neo$ curl http://localhost:8080/client/token.json
                                        200 Hello world !
```

自签名证书信任问题

unable to find valid certification path to requested target

```
package example.controller;

import java.security.KeyManagementException;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;

import javax.net.ssl.SSLContext;

import org.apache.http.conn.ssl.NoopHostnameVerifier;
```

```
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContexts;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import
org.springframework.http.client.HttpComponentsClientHttpReque
stFactory;
import
org.springframework.security.oauth2.client.OAuth2RestOperatio
ns;
import
org.springframework.security.oauth2.common.OAuth2AccessToken;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.client.RestTemplate;

@Controller
public class TestController {
        @Autowired
        private OAuth2RestOperations restTemplate;

        @GetMapping("/ssl")
        @ResponseBody
        public String ssl() throws KeyManagementException,
NoSuchAlgorithmException, KeyStoreException {
                String url =
"https://api.netkiller.cn/news/list.json";

                SSLContext sslcontext =
SSLContexts.custom().loadTrustMaterial(null, (chain,
authType) -> true).build();
                SSLConnectionSocketFactory sslsf = new
SSLConnectionSocketFactory(sslcontext, new String[] { "TLSv1"
}, null, new NoopHostnameVerifier());
                CloseableHttpClient httpClient =
HttpClients.custom().setSSLSocketFactory(sslsf).build();
                HttpComponentsClientHttpRequestFactory
```

```
httpComponentsClientHttpRequestFactory = new
HttpComponentsClientHttpRequestFactory(httpClient);

httpComponentsClientHttpRequestFactory.setConnectTimeout(6000
0);

httpComponentsClientHttpRequestFactory.setReadTimeout(180000)
;

                final RestTemplate restTemplate = new
RestTemplate(httpComponentsClientHttpRequestFactory);

                HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_JSON);
                headers.set("Authorization", "Bearer " +
this.restTemplate.getAccessToken().getValue());
                HttpEntity<String> entity = new
HttpEntity<String>(headers);

                ResponseEntity<String> response =
restTemplate.exchange(url, HttpMethod.GET, entity,
String.class);
                return response.getBody();
        }
}
```

Principal

```
@RestController
public class ResourceController {
    @GetMapping("/getResource")
    public String getResource(Principal principal) {
        return "SUCCESS, 当前用户: " + principal.getName();
    }
    @RequestMapping("/user")
        public Principal user(Principal principal) {
        return principal;
        }
```

```
}
```

SecurityContextHolder 对象

```
        @RequestMapping(value = "principal", method =
RequestMethod.GET)
    public Object getPrincipal() {
        Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrinc
ipal();
        return principal;
    }

    // 查询用户角色
    @RequestMapping(value = "roles", method =
RequestMethod.GET)
    public Object getRoles() {
        return
SecurityContextHolder.getContext().getAuthentication().getAutho
rities();
    }
```

资源服务器配置

ResourceServerConfigurerAdapter

access()

```
        @Override
    public void configure(HttpSecurity http) throws Exception
{
        http
                .httpBasic()
```

```
            .and()
                .authorizeRequests()
                .antMatchers("/home")
                .permitAll()
            .and()
                .authorizeRequests()
                .antMatchers("/user")
                .access("#oauth2.hasScope('read')")
            .and()
                .authorizeRequests()
                .anyRequest()
                .authenticated()
        ;
    }
```

```
        @Override
    public void configure(HttpSecurity http) throws Exception
{
        super.configure(http);
        http
                .authorizeRequests()
                .antMatchers("/login").permitAll()
                .antMatchers("/info","/news")
                .access("#oauth2.hasScope('read') and
hasRole('USER')");
    }
```

```
                        @Override
            public void
configure(ResourceServerSecurityConfigurer resources) throws
Exception {
                TokenStore tokenStore = new
JdbcTokenStore(dataSource);
                resources.resourceId("user-
```

```
services").tokenStore(tokenStore).stateless(true);
            }

            @Override
            public void configure(HttpSecurity http) throws
Exception {
                http
                        .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.NEVER)
                        .and()

.requestMatchers().antMatchers("/user/**")
                        .and()
                        .authorizeRequests()
                        .antMatchers(HttpMethod.GET,
"/user/**").access("#oauth2.hasScope('read')")
                        .antMatchers(HttpMethod.POST,
"/user/**").access("#oauth2.hasScope('write')")
                        .antMatchers(HttpMethod.PATCH,
"/user/**").access("#oauth2.hasScope('write')")
                        .antMatchers(HttpMethod.PUT,
"/user/**").access("#oauth2.hasScope('write')")
                        .antMatchers(HttpMethod.DELETE,
"/user/**").access("#oauth2.hasScope('write')")
                        .and()
                        .headers().addHeaderWriter(new
HeaderWriter() {
                    @Override
                    public void
writeHeaders(HttpServletRequest request, HttpServletResponse
response) {
                        response.addHeader("Access-Control-
Allow-Origin", "*");
                        if
(request.getMethod().equals("OPTIONS")) {
                            response.setHeader("Access-
Control-Allow-Methods", request.getHeader("Access-Control-
Request-Method"));
                            response.setHeader("Access-
Control-Allow-Headers", request.getHeader("Access-Control-
Request-Headers"));
                        }
                    }
                });
```

```
        }
```

Client

Overriding Spring Boot 2.0 Auto-configuration

```
@Configuration
public class OAuth2LoginConfig {

        @Bean
        public ClientRegistrationRepository
clientRegistrationRepository() {
                return new
InMemoryClientRegistrationRepository(this.googleClientRegistr
ation());
        }

        private ClientRegistration googleClientRegistration()
{
                return
ClientRegistration.withRegistrationId("google")
                        .clientId("google-client-id")
                        .clientSecret("google-client-secret")

.clientAuthenticationMethod(ClientAuthenticationMethod.BASIC)

.authorizationGrantType(AuthorizationGrantType.AUTHORIZATION_
CODE)
                        .redirectUriTemplate("
{baseUrl}/login/oauth2/code/{registrationId}")
                        .scope("openid", "profile", "email",
"address", "phone")

.authorizationUri("https://accounts.google.com/o/oauth2/v2/au
th")

.tokenUri("https://www.googleapis.com/oauth2/v4/token")
```

```
.userInfoUri("https://www.googleapis.com/oauth2/v3/userinfo")

.userNameAttributeName(IdTokenClaimNames.SUB)

.jwkSetUri("https://www.googleapis.com/oauth2/v3/certs")
                        .clientName("Google")
                        .build();
        }
}
```

Oauth2 常见问题

修改 /oauth/token 路径

```
        @Override
        public void
configure(AuthorizationServerEndpointsConfigurer configurer)
throws Exception {

configurer.authenticationManager(authenticationManager);

configurer.userDetailsService(userDetailsService);

configurer.pathMapping("/oauth/token","/oauth/token3"); //可以
修改默认/oauth/token路径为 /oauth/token3
        }
```

password 认证方式静态配置用户列表

```
        @Override
        public void
configure(AuthorizationServerEndpointsConfigurer configurer)
throws Exception {
```

```
configurer.authenticationManager(authenticationManager);

configurer.userDetailsService(userDetailsService());
        }

        @Bean
        public UserDetailsService userDetailsService() {
                Map<String, String[]> users = new
HashMap<String, String[]>() {
                        {
                                put("user", new String[] {
"ROLE_USER" });
                                put("admin", new String[] {
"ROLE_USER", "ROLE_ADMIN" });
                                put("client", new String[] {
"ROLE_CLIENT" });
                                put("trust", new String[] {
"ROLE_TRUSTED_CLIENT" });
                        }
                };
                String password =
passwordEncoder().encode("123456");     // 设置默认密码
                // TODO 这里需要自行定义访问数据库的扩展
                return new UserDetailsService() {
                        @Override
                        public UserDetails
loadUserByUsername(String name) throws
UsernameNotFoundException {
                                String[] authList =
users.containsKey(name) ? users.get(name) : new String[] {
"ROLE_USER" };
                                User user = new User(name,
password, AuthorityUtils.createAuthorityList(authList));
                                return user;
                        }
                };

        }
```

第 10 章 Spring Cloud

1. Spring Cloud 相关的 application.properties 配置

1.1. 启用或禁用 bootstrap

```
spring.cloud.bootstrap.enabled=false
spring.cloud.bootstrap.location=classpath:bootstrap.properties
```

1.2. bootstrap.properties 配置文件

bootstrap.properties 是优先级最高配置文件，一般用于 Spring Cloud 配置中心。

bootstrap.yml是由spring.cloud.bootstrap.name（默认:”bootstrap”）或者spring.cloud.bootstrap.location 设置（默认空）。

如果激活 profile（spring.profiles.active=development）对应配置 bootstrap-development.properties

spring.cloud.config.allowOverride=true（允许本地配置覆盖远程配置）。

spring.cloud.config.overrideNone=true 覆盖任何本地属性

spring.cloud.config.overrideSystemProperties=false 仅仅系统属性和环境变量

2. Spring Cloud Config

2.1. Maven 项目 pom.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>microservice</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>microservice</name>
        <url>http://www.netkiller.cn</url>
        <description>Demo project for Spring Boot</description>

        <organization>
                <name>Netkiller Spring Cloud 手札</name>
                <url>http://www.netkiller.cn</url>
        </organization>

        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller Spring Cloud 手
札</organization>

<organizationUrl>http://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
                </developer>
        </developers>

        <properties>
```

```
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>14</java.version>
                <maven.compiler.source>${java.version}
</maven.compiler.source>
                <maven.compiler.target>${java.version}
</maven.compiler.target>
                <maven.compiler.release>${java.version}
</maven.compiler.release>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.3.3.RELEASE</version>
                <relativePath />
        </parent>

        <repositories>
                <repository>
                        <id>alimaven</id>
                        <name>Maven Aliyun Mirror</name>

<url>http://maven.aliyun.com/nexus/content/repositories/central
/</url>
                        <releases>
                                <enabled>true</enabled>
                        </releases>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
        </repositories>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>Hoxton.SR8</version>
```

```
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
config</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                </dependency>
        </dependencies>

        <modules>
                <module>eureka</module>
                <module>gateway</module>
                <module>config</module>
                <module>webflux</module>
                <module>openfeign</module>
                <module>restful</module>
        </modules>
</project>
```

2.2. Server

Maven config 模块

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>microservice</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>config</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>config</name>
        <url>http://www.netkiller.cn</url>
        <description>Config project for Spring
cloud</description>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>11</java.version>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-config-
server</artifactId>
                </dependency>
                <!-- <dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-config-monitor</artifactId>
</dependency> -->
```

```
                <!-- <dependency>
<groupId>org.springframework.boot</groupId> <artifactId>spring-
boot-starter-security</artifactId> </dependency> -->
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <configuration>

<mainClass>cn.netkiller.config.Application</mainClass>
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Application

Application

```
package cn.netkiller.config;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.config.server.EnableConfigServer;
```

```
@EnableConfigServer
@SpringBootApplication
public class Application {
        public static void main(String[] args) {
                System.out.println("Config Server
Starting...");
                SpringApplication.run(Application.class,
args);
        }
}
```

application.properties

```
server.port=8888
spring.cloud.config.server.git.uri=https://github.com/netkiller
/config.git
```

Git 仓库

克隆仓库

```
git clone https://github.com/netkiller/config.git
```

创建配置文件 server-development.properties

```
vim server-development.properties

test.a=KKOOKK
message=Hello world
```

```
```

提交配置文件

```
git commit -a
git push
```

测试服务器

```
neo@netkiller $ curl http://localhost:8888/server-
development.json
{"message":"Hello world","test":{"a":"KKOOKK"}}
```

2.3. Client

Maven pom.xml

```
<?xml version="1.0"?>
<project
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>microservice</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
```

```
        <artifactId>restful</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>restful</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
netflix-eureka-client</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                                <configuration>

<mainClass>cn.netkiller.Application</mainClass>
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Application

```
package cn.netkiller.cloud.client;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.context.config.annotation.RefreshSc
ope;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class,
args);
        }
}

@RefreshScope
@RestController
class MessageRestController {

        @Value("${message:Hello default}")
        private String message;

        @RequestMapping("/message")
        String getMessage() {
                return this.message;
        }
}
```

注意 @RefreshScope 注解

bootstrap.properties

```
spring.application.name=server-development
spring.cloud.config.uri=http://localhost:8888
management.security.enabled=false
```

测试 client

```
neo@netkiller $ curl http://localhost:8080/message.json
Hello world
```

2.4. Config 高级配置

仓库配置

配置文件格式

```
/{application}/{profile}[/{label}]
/{application}-{profile}.yml
/{label}/{application}-{profile}.yml
/{application}-{profile}.properties
/{label}/{application}-{profile}.properties
```

{application} 映射到客户端的"spring.application.name" 或“spring.cloud.config.name”;

{profile}映射到客户端上的"spring.profiles.active" 或“spring.cloud.config.profile”;

{label} 是可选的 git 标签，默认 master;

```
nickname: netkilleriMac:Java neo$ curl
http://localhost:8769/webflux-dev.json
{"name":"Neo","nickname":"netkiller"}

iMac:Java neo$ curl http://localhost:8769/webflux-
dev.properties
name: Neo
nickname: netkiller

iMac:Java neo$ curl http://localhost:8769/webflux-dev.yml
name: Neo
nickname: netkiller

iMac:Java neo$ curl http://localhost:8769/webflux-dev.yaml
name: Neo
nickname: netkiller
```

分支

label 是指 git 的分支

```
spring.cloud.config.label=mybranch
```

basedir

```
spring.cloud.config.server.git.basedir=api/configs
```

HTTP Auth

```
spring.application.name=config-server
spring.cloud.config.server.git.uri=https://netkiller:xxxxxx@g
ithub.com/xyz/microservices-configs.git
```

```
spring.application.name=config-server
spring.cloud.config.server.git.uri=https://github.com/xyz/mic
roservices-configs.git
spring.cloud.config.server.git.username=netkiller
spring.cloud.config.server.git.password=password
```

本地git仓库

创建本地仓库

```
mkdir ~/config
cd config
git init
git config --global user.email "neo.chen@live.com"
git config --global user.name "Neo Chen"
```

创建测试配置文件

```
# cat app-test.properties
name=neo
age=10
```

提交配置文件

```
git add app-test.properties
git commit -a
```

检查文件是否提交成功

```
[root@netkiller config]# git log
commit aee6c35bacf1740004e02f8ecdcf2fd322422405
Author: Neo Chen <neo.chen@live.com>
Date:   Thu Nov 2 14:18:48 2017 +0800

        new file:   app-test.properties
```

配置 Spring cloud config 服务器，修改 application.properties 文件

```
server.port=8888
#spring.cloud.config.server.git.uri=/opt/config
spring.cloud.config.server.git.uri=
file://${user.home}/config
security.user.name=cfg
security.user.password=s3cr3t
```

```
## Spring cloud GIT Repository file
${user.home}/config/root-server.properties
```

检验配置中心

```
[root@netkiller config]# curl
http://cfg:test@localhost:8888/app-test.properties
age: 10
name: neo
```

native 本地配置

载入本地配置文件 resources/shared/config-client-dev.yml

```
server:
  port: 8762
foo: foo version 1
```

配置中心服务端 resources/application.yml 配置文件

```
server:
  port: 8769
spring:
  application:
    name: config-server
  profiles:
    active: native
  cloud:
    config:
```

```
      server:
        native:
          search-locations: classpath:/shared
```

测试配置文件

```
iMac:Java neo$ curl http://localhost:8769/config-client-
dev.json
{"server":{"port":8762},"foo":"foo version 1"}
```

Config server 用户认证

Server 配置

application.properties

```
server.port=8888
spring.cloud.config.server.git.uri=ssh://localhost/config-repo
spring.cloud.config.server.git.clone-on-start=true
security.user.name=cfg
security.user.password=s3cr3t
```

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
```

```
        <groupId>cn.netkiller</groupId>
        <artifactId>config</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>config</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.7.RELEASE</version>
                <relativePath />
        </parent>
        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
config</artifactId>

<version>1.3.1.RELEASE</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-config-
server</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                    <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

测试是否生效

```
neo@MacBook-Pro ~/deployment % curl
http://cfg:s3cr3t@localhost:8888/neo-development.json
{"message":"Hello world","test":{"name":"neo"}}
```

Client 配置

bootstrap.properties:

```
spring.application.name=project
spring.profiles.active=development
spring.cloud.config.uri=http://localhost:8888
spring.cloud.config.username=cfg
spring.cloud.config.password=s3cr3t
```

加密敏感数据

Config server 创建证书

```
keytool -genkeypair -alias config-server-key \
       -keyalg RSA -keysize 4096 -sigalg SHA512withRSA \
       -dname 'CN=Config Server,OU=Spring Cloud,O=Netkiller'
\
       -keypass s3cr3t -keystore config-server.jks \
       -storepass passw0rd
```

application.properties 中配置证书

```
# spring.cloud.config.server.encrypt.enabled=true
encrypt.key-store.location=classpath:/config-server.jks
encrypt.key-store.alias=config-server-key
encrypt.key-store.secret=s3cr3t
encrypt.key-store.password=passw0rd
```

测试加密

```
curl -X POST --data-urlencode mypassword
http://localhost:8888/encrypt
```

```
$ PASSWORD=$(curl -X POST --data-urlencode passw0rd
http://cfg:s3cr3t@localhost:8888/encrypt)
$ echo "user.password=$PASSWORD" >> api-interface-
development.properties
$ git commit -am 'Added encrypted password'

# 刷新配置
$ curl -X POST http://cfg:s3cr3t@localhost:8888/refresh
```

Spring Cloud Config JDBC Backend

Maven pom.xml

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.11.RELEASE</version>
    <relativePath/>
</parent>

<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-config-server</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
```

```
    </dependency>
    <dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
        <version>5.0.3</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.21</version>
    </dependency>
</dependencies>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-
dependencies</artifactId>
            <version>Edgware.SR3</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

数据库表结构

```
CREATE TABLE `properties` (
  `key` varchar(50) NOT NULL,
  `value` varchar(500) NOT NULL,
  `application` varchar(50) NOT NULL,
  `profile` varchar(50) NOT NULL,
  `label` varchar(50) NOT NULL,
  PRIMARY KEY (`KEY`, `APPLICATION`, `PROFILE`, `LABEL`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Config 服务器

```
@EnableConfigServer
@SpringBootApplication
public class Application {

        //@Autowired
        //JdbcTemplate jdbcTemplate;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
        // 测试用数据，仅用于本文测试使用
        JdbcTemplate jdbcTemplate =
context.getBean(JdbcTemplate.class);
        jdbcTemplate.execute("delete from properties");
        jdbcTemplate.execute("INSERT INTO properties
VALUES('neo.message', 'helloworld', 'api', 'stage',
'master')");
        jdbcTemplate.execute("INSERT INTO properties
VALUES('neo.message', 'helloworld', 'new', 'online',
'master')");
        jdbcTemplate.execute("INSERT INTO properties
VALUES('neo.message', 'helloworld', 'test', 'online',
'develop')");
        jdbcTemplate.execute("INSERT INTO properties
VALUES('neo.message', 'helloworld', 'cms', 'online',
'master')");
    }

}
```

application.properties

spring.profiles.active=jdbc 将配置中心的存储实现切换到jdbc的方式, 必须设置。

```
```

```
server.port=8888
spring.profiles.active=jdbc

spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/config-
server-db
spring.datasource.username=root
spring.datasource.password=xxxx

# or

spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=
jdbc:postgresql://localhost:5432/configdb
spring.datasource.username=xxxxxx
spring.datasource.password=xxxxxx
```

2.5. Old

Server (Camden.SR5)

Maven pom.xml 请使用最新版本 1.3.1, 下面的 maven 是早期 Camden.SR5 版本的配置

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>netkiller.cn</groupId>
        <artifactId>cloud</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Neo</name>
        <description>http://www.netkiller.cn</description>
        <packaging>jar</packaging>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
```

```
parent</artifactId>
                <version>1.5.3.RELEASE</version>
                <relativePath />
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-config-
server</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>Camden.SR5</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <build>
                <plugins>
                        <plugin>
```

```
<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

Client (Camden.SR5)

Maven pom.xml Camden.SR5 为早期版本，尽可以使用新版

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>netkiller.cn</groupId>
        <artifactId>cloud</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.2.RELEASE</version>
                <relativePath />
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>
```

```
<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
config</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                    <artifactId>spring-cloud-
dependencies</artifactId>
                                    <version>Camden.SR5</version>
                                    <type>pom</type>
                                    <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                    <artifactId>spring-boot-
```

```
maven-plugin</artifactId>
                            </plugin>
                    </plugins>
            </build>

</project>
```

3. Spring Cloud Consol

3.1. Spring Cloud Consul 配置

核心参数

```
配置项                                                  默认值
spring.cloud.consul.enabled     true
spring.cloud.consul.host        localhost
spring.cloud.consul.port        8500
```

服务发现参数

```
配置项
默认值
spring.cloud.consul.discovery.acl-token
spring.cloud.consul.discovery.catalog-services-watch-delay
10
spring.cloud.consul.discovery.catalog-services-watch-timeout
2
spring.cloud.consul.discovery.datacenters
spring.cloud.consul.discovery.default-query-tag
spring.cloud.consul.discovery.default-zone-metadata-name
zone
spring.cloud.consul.discovery.deregister
true
spring.cloud.consul.discovery.enabled
true
spring.cloud.consul.discovery.fail-fast
true
spring.cloud.consul.discovery.health-check-critical-timeout
spring.cloud.consul.discovery.health-check-interval
10s
spring.cloud.consul.discovery.health-check-path
/actuator/health
```

```
spring.cloud.consul.discovery.health-check-timeout
spring.cloud.consul.discovery.health-check-tls-skip-verify
spring.cloud.consul.discovery.health-check-url
spring.cloud.consul.discovery.heartbeat.enabled
false
spring.cloud.consul.discovery.heartbeat.interval-ratio
spring.cloud.consul.discovery.heartbeat.ttl-unit
s
spring.cloud.consul.discovery.heartbeat.ttl-value
30
spring.cloud.consul.discovery.hostname
spring.cloud.consul.discovery.instance-group
spring.cloud.consul.discovery.instance-id
默认为服务名+环境+端口号
spring.cloud.consul.discovery.instance-zone
spring.cloud.consul.discovery.ip-address
spring.cloud.consul.discovery.lifecycle.enabled
true
spring.cloud.consul.discovery.management-port
spring.cloud.consul.discovery.management-suffix
management
spring.cloud.consul.discovery.management-tags
spring.cloud.consul.discovery.port
spring.cloud.consul.discovery.prefer-agent-address
false
spring.cloud.consul.discovery.prefer-ip-address
false
spring.cloud.consul.discovery.query-passing
false
spring.cloud.consul.discovery.register  true
spring.cloud.consul.discovery.register-health-check
true
spring.cloud.consul.discovery.scheme
http
spring.cloud.consul.discovery.server-list-query-tags
spring.cloud.consul.discovery.service-name
spring.cloud.consul.discovery.tags
spring.cloud.consul.discovery.serviceName
是指注册到 Consul 的服务名称，后期客户端会根据这个名称来进行服务调用。
```

配置服务参数

```
配置项
默认值
spring.cloud.consul.config.enabled
true
spring.cloud.consul.config.prefix
config
spring.cloud.consul.config.default-context
application
spring.cloud.consul.config.profile-separator    ,
spring.cloud.consul.config.data-key
data
spring.cloud.consul.config.format
KEY_VALUE, PROPERTIES, YAML, FILES
spring.cloud.consul.config.name
${spring.application.name}
spring.cloud.consul.config.acl-token
spring.cloud.consul.config.fail-fast                    false
spring.cloud.consul.config.watch.enabled                true
spring.cloud.consul.config.watch.wait-time              55
spring.cloud.consul.config.watch.delay                  1000
```

3.2. Maven 父项目

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>parent</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>
        <name>demo</name>
        <description>Demo project for Spring Boot</description>
        <url>http://www.netkiller.cn</url>
```

```
        <organization>
                <name>Netkiller Spring Cloud 手札</name>
                <url>http://www.netkiller.cn</url>
        </organization>

        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller Spring Cloud 手
札</organization>

<organizationUrl>http://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
                </developer>
        </developers>

        <!--使用aliyun镜像 -->
        <repositories>
                <repository>
                        <id>alimaven</id>
                        <name>Maven Aliyun Mirror</name>

<url>http://maven.aliyun.com/nexus/content/repositories/central
/</url>
                        <releases>
                                <enabled>true</enabled>
                        </releases>
                        <snapshots>
                                <enabled>false</enabled>
                        </snapshots>
                </repository>
        </repositories>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
                <spring-cloud.version>Greenwich.SR1</spring-
cloud.version>
        </properties>
```

```
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-
parent</artifactId>
		<version>2.1.3.RELEASE</version>
		<relativePath />
	</parent>


	<dependencyManagement>
		<dependencies>
			<dependency>

<groupId>org.springframework.cloud</groupId>
				<artifactId>spring-cloud-
dependencies</artifactId>
				<version>${spring-
cloud.version}</version>
				<type>pom</type>
				<scope>import</scope>
			</dependency>
		</dependencies>
	</dependencyManagement>

	<dependencies>
		<dependency>

<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-
test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<scope>test</scope>
		</dependency>
	</dependencies>

	<modules>
		<module>consol-producer</module>
		<module>consol-consumer</module>
		<module>consol-config</module>
		<module>openfeign</module>
```

```
        </modules>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```

3.3. Consul 服务生产者

Maven

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <!-- <groupId>cn.netkiller</groupId> -->
        <artifactId>consul-producer</artifactId>
        <!-- <version>0.0.1-SNAPSHOT</version> -->
        <name>consul-producer</name>
        <url>http://www.netkiller.cn</url>
        <description>Spring Cloud Consul Sample</description>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
```

```
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>11</java.version>
        </properties>
        <dependencies>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
consul-discovery</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <!-- Only needed at compile time -->
                        <optional>true</optional>
                </dependency>

                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <!-- <version>3.8.1</version> -->
                        <scope>test</scope>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
```

```
                                <!-- <configuration>
<mainClass>cn.netkiller.config.Application</mainClass>
</configuration> -->
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

application.properties

```
server.port=8080
spring.application.name=spring-cloud-consul-producer

spring.cloud.consul.host=192.168.4.217
spring.cloud.consul.port=8500

logging.level.org.springframework.cloud.consul=DEBUG
```

SpringApplication

```
package cn.netkiller.consul;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
```

```
org.springframework.cloud.client.discovery.EnableDiscoveryCli
ent;

@SpringBootApplication
@EnableDiscoveryClient
public class Application {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                SpringApplication.run(Application.class,
args);
        }
}
```

TestController

```
package cn.netkiller.consul.controller;

import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class TestController {

        public TestController() {
                // TODO Auto-generated constructor stub
        }

        @GetMapping("/hello")
        public String provider() {
                return "Helloworld!!!";
        }

        @RequestMapping("/hi")
        public String hi(@RequestParam(name = "name") String
name) {
```

```
                return "hi " + name + "!";
        }

}
```

3.4. Consul 服务消费者

Maven

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>consul-consumer</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>consul-consumer</name>
        <url>http://www.netkiller.cn</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>11</java.version>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
```

```
consul-discovery</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <!-- <configuration>
<mainClass>cn.netkiller.config.Application</mainClass>
</configuration> -->
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

application.properties

```
server.port=8082
spring.application.name=spring-cloud-consul-consumer

spring.cloud.consul.host=192.168.4.217
spring.cloud.consul.port=8500
#设置不需要注册到 consul 中
spring.cloud.consul.discovery.register=false
```

SpringApplication

```
package cn.netkiller.consul.consumer;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
        public static void main(String[] args) {
                System.out.println("Consol Consumer!");
                SpringApplication.run(Application.class,
args);
        }
}
```

TestController

```
package cn.netkiller.consul.consumer;
```

```
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import
org.springframework.cloud.client.discovery.DiscoveryClient;
import
org.springframework.cloud.client.loadbalancer.LoadBalancerCli
ent;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
public class TestController {

        public ConsumerController() {
                // TODO Auto-generated constructor stub
        }

        @Autowired
        private LoadBalancerClient loadBalancerClient;
        @Autowired
        private DiscoveryClient discoveryClient;

        /**
         * 获取所有服务
         */
        @RequestMapping("/services")
        public Object services() {
                return discoveryClient.getInstances("spring-
cloud-consul-producer");
        }

        /**
         * 从所有服务中选择一个服务（轮询）
         */
        @RequestMapping("/discover")
        public Object discover() {
                return loadBalancerClient.choose("spring-
cloud-consul-producer").getUri().toString();
        }

        @RequestMapping("/call")
```

```
        public String call() {
                ServiceInstance serviceInstance =
loadBalancerClient.choose("spring-cloud-consul-producer");
                System.out.println("服务地址: " +
serviceInstance.getUri());
                System.out.println("服务名称: " +
serviceInstance.getServiceId());

                String callServiceResult = new
RestTemplate().getForObject(serviceInstance.getUri().toString
() + "/hello", String.class);
                System.out.println(callServiceResult);
                return callServiceResult;
        }
}
```

3.5. Openfeign

Maven

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>parent</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>openfeign</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>openfeign</name>
        <url>http://www.netkiller.cn</url>
        <properties>
```

```
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>11</java.version>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
consul-discovery</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
openfeign</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <!-- <configuration>
<mainClass>cn.netkiller.config.Application</mainClass>
</configuration> -->
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
```

```
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

application.properties

```
server.port=8083
spring.application.name=spring-cloud-consul-openfeign

spring.cloud.consul.host=192.168.4.217
spring.cloud.consul.port=8500

spring.cloud.consul.discovery.register=false
```

SpringApplication

```
package cn.netkiller.openfeign;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryCli
ent;
import
org.springframework.cloud.openfeign.EnableFeignClients;
```

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients

public class Application {
        public static void main(String[] args) {
                System.out.println("openfeign!");
                SpringApplication.run(Application.class,
args);
        }
}
```

Feign 接口

```
package cn.netkiller.openfeign;

import org.springframework.cloud.openfeign.FeignClient;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@FeignClient(value = "spring-cloud-consul-producer", fallback
= FeignFallback.class)
public interface TestFeign {
        @RequestMapping(value = "/hi", method =
RequestMethod.GET)
        String hi(@RequestParam(value = "name") String name);
}
```

```
package cn.netkiller.openfeign;
```

```
public class FeignFallback implements TestFeign {
        @Override
        public String hi(String name) {
                return "sorry,熔断介入";
        }

}
```

TestController

```
package cn.netkiller.openfeign;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class TestController {
        @Autowired
        TestFeign testFeign;

        @RequestMapping("/feign")
        public String testFeign(@RequestParam(name = "name")
String name) {
                return testFeign.hi(name);
        }
}
```

4. Spring Cloud Netflix

4.1. Eureka Server

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller.spring.cloud</groupId>
        <artifactId>netflix.eureka.server</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>eureka.server</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.7.RELEASE</version>
                <relativePath />
        </parent>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
```

```
netflix</artifactId>

<version>1.3.5.RELEASE</version>
                                    <type>pom</type>
                                    <scope>import</scope>
                            </dependency>
                    </dependencies>
            </dependencyManagement>

            <dependencies>
                    <dependency>

<groupId>org.springframework.boot</groupId>
                            <artifactId>spring-boot-starter-
test</artifactId>
                            <scope>test</scope>
                    </dependency>
                    <dependency>

<groupId>org.springframework.cloud</groupId>
                            <artifactId>spring-cloud-starter-
eureka-server</artifactId>
                    </dependency>
            </dependencies>

            <build>
                    <plugins>
                            <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                    <artifactId>maven-surefire-
plugin</artifactId>
                                    <configuration>
                                            <skip>true</skip>
                                    </configuration>
                            </plugin>
                    </plugins>
            </build>
</project>
```

Application

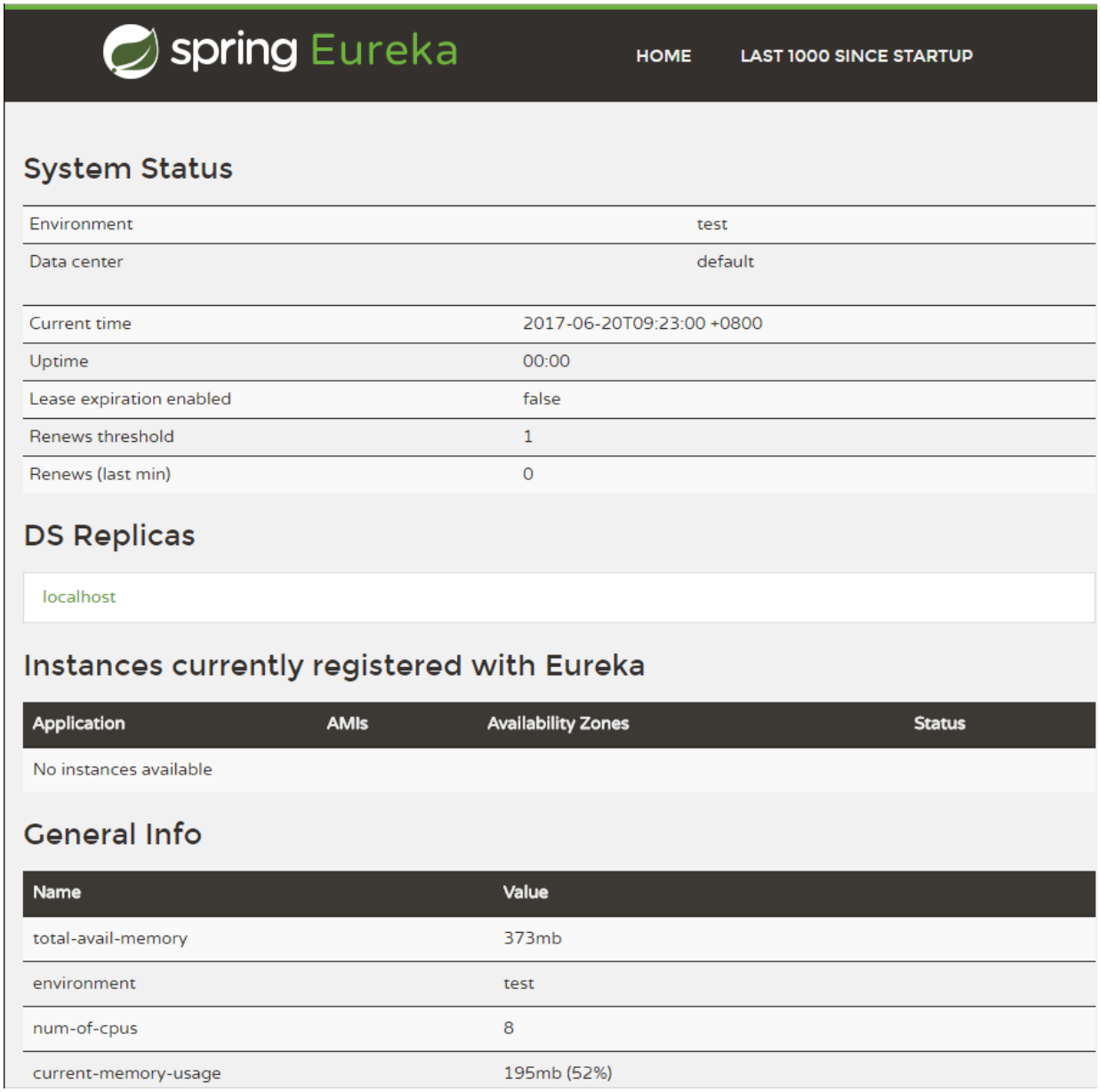
```
package cn.netkiller.spring.cloud.netflix.eureka.server;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.netflix.eureka.server.EnableEurekaSer
ver;

@SpringBootApplication
@EnableEurekaServer
public class Application {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                // new
SpringApplicationBuilder(Application.class).web(true).run(args)
;
                SpringApplication.run(Application.class, args);
        }
}
```

application.properties

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
eureka.client.serviceUrl.defaultZone=http://localhost:${server.
port}/eureka/

logging.level.com.netflix.eureka=OFF
logging.level.com.netflix.discovery=OFF
```

检查注册服务器

http://localhost:8761



4.2. Eureka Client

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller.spring.cloud</groupId>
        <artifactId>eureka.client</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>eureka.client</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.7.RELEASE</version>
                <relativePath />
        </parent>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
netflix</artifactId>

<version>1.3.5.RELEASE</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
```

```
                        <artifactId>spring-cloud-starter-
config</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
eureka</artifactId>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Application

```
package cn.netkiller.spring.cloud.eureka.client;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@Configuration
```

```
@EnableAutoConfiguration
@EnableEurekaClient
@RestController

public class Application {

        @RequestMapping("/")
        public String home() {
                return "Hello World";
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
        }
}
```

RestController

```
package cn.netkiller.spring.cloud.eureka.client;

import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import
org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class TestRestController {
        private static final Logger logger =
LoggerFactory.getLogger(TestRestController.class);

        @RequestMapping("/")
```

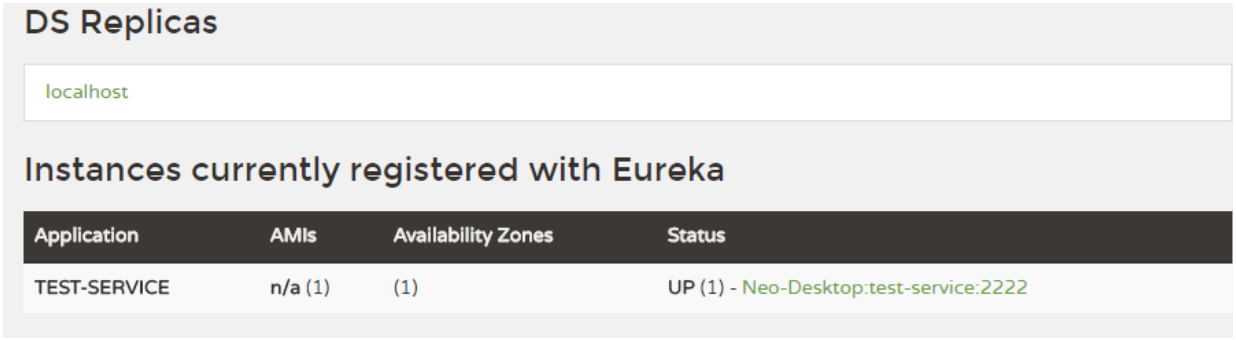
```
        public String version() {
                logger.info("Hello!!!");
                return "Version: v1.0.0";
        }

        @RequestMapping(value = "/add", method =
RequestMethod.GET)
        public Integer add(@RequestParam Integer a,
@RequestParam Integer b) {
                Integer r = a + b;
                return r;
        }

        @RequestMapping("/greeting")
        public String greeting() {
                return "GREETING";
        }
}
```

application.properties

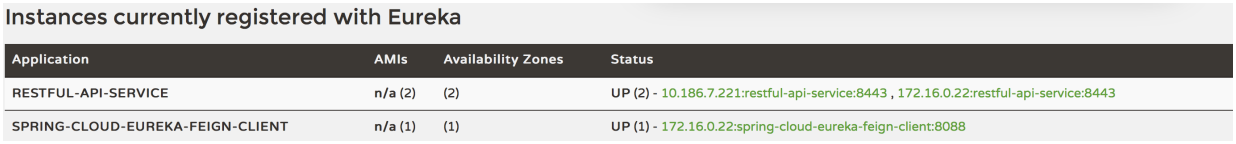
```
spring.application.name=test-service
server.port=8080
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eure
ka/
```

测试

首先确认客户端已经注册到 http://localhost:8761/



你可以启动很多 Eureka 客户端，相同的 spring.application.name 会归为一组，为用户提供负载均衡。




```
neo@MacBook-Pro ~ % curl http://localhost:8080/
Hello World
```

add 接口测试

```
curl http://localhost:8080/add.json?a=5&b=3

8
```

4.3. Feign client

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller.spring.cloud.netflix</groupId>
        <artifactId>feign.client</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>feign.client</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.3.RELEASE</version>
                <relativePath />
        </parent>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
eureka</artifactId>
                        <version>1.3.1.RELEASE</version>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
feign</artifactId>
                        <version>1.3.1.RELEASE</version>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Application

```
package cn.netkiller.spring.cloud.netflix.feign.client;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import
org.springframework.cloud.netflix.feign.EnableFeignClients;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients
@RestController
public class Application {
        @Autowired
        private GreetingClient greetingClient;

        @RequestMapping("/get-greeting")
        public String greeting() {
                return greetingClient.greeting();
        }

        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
                System.out.println("Hello World!");
        }
}
```

interface

```
package cn.netkiller.spring.cloud.netflix.feign.client;

import org.springframework.cloud.netflix.feign.FeignClient;
import org.springframework.web.bind.annotation.RequestMapping;

@FeignClient("test-service")
public interface GreetingClient {
        @RequestMapping("/greeting")
        String greeting();
}
```

@FeignClient("test-service") 是 Eureka Client application.properties 中的 spring.application.name 配置项

@RequestMapping("/greeting") 是 Eureka Client RestController 中的 @RequestMapping

application.properties

```
spring.application.name=spring-cloud-eureka-feign-client
server.port=8088
#eureka.client.register-with-eureka=false
#eureka.client.fetch-registry=false
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eure
ka/
feign.compression.response.enabled=true
feign.compression.request.enabled=true
feign.compression.request.mime-
types=text/xml,application/xml,application/json
feign.compression.request.min-request-size=2048
```

测试

```
$ curl -s http://localhost:8088/get-greeting.json

GREETING
```

fallback

```
@FeignClient(value = "restful-api-service", fallback =
```

```
UserServiceFeignClientFallback.class)
public interface UserServiceFeignClient {
@RequestMapping(value = "/api/user/{id}", method =
RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE,
consumes = MediaType.APPLICATION_JSON_VALUE)
    User getUser(@PathVariable("id") int id);

    @RequestMapping(value = "/api/user/search/findByName?name=
{name}", method = RequestMethod.GET, produces =
MediaType.APPLICATION_JSON_VALUE, consumes =
MediaType.APPLICATION_JSON_VALUE)
    User findUserByName(@PathVariable("name") String name);

    @RequestMapping(value = "/api/user/search/findByAddress?
address={address}", method = RequestMethod.GET)
    String findUserByAddress(@PathVariable("address") String
address);
}
```

```
@Component
public class UserServiceFeignClientFallback implements
UserServiceFeignClient {

    @Override
    public User getUser(int id) {
        return new User("getUser.Fallback", "feignClient
return");
    }

    @Override
    public User findUserByName(String name) {
        return new User("findUserByName.Fallback", "feignClient
return");
    }

    @Override
    public String findUserByAddress(String address) {
        return "fallback";
    }
}
```

4.4. 为 Eureka Server 增加用户认证

Maven

```
                <dependency>
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
```

application.properties

```
                                security.user.name=eureka
                                security.user.password=s3cr3t
```

Eureka Client

```
                                spring.application.name=restful-
api-service

eureka.client.serviceUrl.defaultZone=http://eureka:s3cr3t@localh
ost:8761/eureka/
```

Feign Client

```
eureka.client.serviceUrl.defaultZone=http://eureka:s3cr3t@localh
```

```
ost:8761/eureka/
```

4.5. Eureka 配置项

/eureka/apps

```
neo@MacBook-Pro-Neo ~ % curl http://localhost:8761/eureka/apps
<applications>
  <versions__delta>1</versions__delta>
  <apps__hashcode></apps__hashcode>
</applications>
```

Spring Cloud Eureka 配置参数说明

```
Eureka Client 配置项 (eureka.client.*)

org.springframework.cloud.netflix.eureka.EurekaClientConfigBean

参数名称  说明      默认值
eureka.client.enabled

用于指示Eureka客户端已启用的标志

true

eureka.client.registry-fetch-interval-seconds

指示从eureka服务器获取注册表信息的频率（s）

30

eureka.client.instance-info-replication-interval-seconds

更新实例信息的变化到Eureka服务端的间隔时间，（s）

30
```

```
eureka.client.initial-instance-info-replication-interval-seconds

初始化实例信息到Eureka服务端的间隔时间，（s）

40
eureka.client.eureka-service-url-poll-interval-seconds

询问Eureka Server信息变化的时间间隔（s），默认为300秒 300
eureka.client.eureka-server-read-timeout-seconds

读取Eureka Server 超时时间（s），默认8秒
8

eureka.client.eureka-server-connect-timeout-seconds

连接Eureka Server 超时时间（s），默认5秒
5

eureka.client.eureka-server-total-connections

获取从eureka客户端到所有eureka服务器的连接总数,默认200个

200

eureka.client.eureka-server-total-connections-per-host

获取从eureka客户端到eureka服务器主机允许的连接总数，默认50个

50

eureka.client.eureka-connection-idle-timeout-seconds

连接到 Eureka Server 空闲连接的超时时间（s），默认30
30

eureka.client.registry-refresh-single-vip-address



指示客户端是否仅对单个VIP的注册表信息感兴趣，默认为null

null
eureka.client.heartbeat-executor-thread-pool-size
```

心跳保持线程池初始化线程数，默认2个　2
eureka.client.heartbeat-executor-exponential-back-off-bound

心跳超时重试延迟时间的最大乘数值，默认10

10
eureka.client.serviceUrl.defaultZone

可用区域映射到与eureka服务器通信的完全限定URL列表。每个值可以是单个URL或逗号分隔的备用位置列表。
(http://${eureka.instance.hostname}:${server.port}/eureka/)

eureka.client.use-dns-for-fetching-service-urls

指示eureka客户端是否应使用DNS机制来获取要与之通信的eureka服务器列表。当DNS名称更新为具有其他服务器时，eureka客户端轮询
eurekaServiceUrlPollIntervalSeconds中指定的信息后立即使用该信息。

false
eureka.client.register-with-eureka

指示此实例是否应将其信息注册到eureka服务器以供其他服务发现，默认为false

True
eureka.client.prefer-same-zone-eureka

实例是否使用同一zone里的eureka服务器，默认为true，理想状态下，eureka客户端与服务端是在同一zone下

true
eureka.client.log-delta-diff

是否记录eureka服务器和客户端之间在注册表的信息方面的差异，默认为false

false
eureka.client.disable-delta

指示eureka客户端是否禁用增量提取

false
eureka.client.fetch-remote-regions-registry

逗号分隔的区域列表，提取eureka注册表信息

eureka.client.on-demand-update-status-change

客户端的状态更新到远程服务器上，默认为true

true

eureka.client.allow-redirects

指示服务器是否可以将客户端请求重定向到备份服务器/集群。如果设置为false，则服务器将直接处理请求。如果设置为true，则可以将HTTP重定向发送到具有新服务器位置的客户端。

false

eureka.client.availability-zones.*

获取此实例所在区域的可用区域列表（在AWS数据中心中使用）。更改在运行时在registryFetchIntervalSeconds指定的下一个注册表获取周期生效。

eureka.client.backup-registry-impl

获取实现BackupRegistry的实现的名称，该实现仅在eureka客户端启动时第一次作为后备选项获取注册表信息。 对于需要额外的注册表信息弹性的应用程序，可能需要这样做，否则它将无法运行。

eureka.client.cache-refresh-executor-exponential-back-off-bound

在发生一系列超时的情况下，它是重试延迟的最大乘数值。

10

eureka.client.cache-refresh-executor-thread-pool-size

缓存刷新线程池初始化线程数量
2

eureka.client.client-data-accept

客户端数据接收的名称 full
eureka.client.decoder-name

解码器名称
eureka.client.dollar-replacement

eureka服务器序列化/反序列化的信息中获取“$”符号的替换字符串。默认为“_-”

eureka.client.encoder-name

编码器名称
eureka.client.escape-char-replacement

eureka服务器序列化/反序列化的信息中获取“_”符号的的替换字符串。默认为“__“

eureka.client.eureka-server-d-n-s-name

获取要查询的DNS名称来获得eureka服务器，此配置只有在eureka服务器ip地址列表是在DNS中才会用到。默认为null

null

eureka.client.eureka-server-port

获取eureka服务器的端口，此配置只有在eureka服务器ip地址列表是在DNS中才会用到。默认为null

null

eureka.client.eureka-server-u-r-l-context

表示eureka注册中心的路径，如果配置为eureka，则为
http://ip:port/eureka/,

在eureka的配置文件中加入此配置表示eureka作为客户端向注册中心注册，从而构成eureka集群。此配置只有在eureka服务器ip地址列表是在DNS中才会用到，默认为
null

null

eureka.client.fetch-registry

客户端是否获取eureka服务器注册表上的注册信息，默认为true

true

eureka.client.filter-only-up-instances

是否过滤掉非up实例，默认为true
true

eureka.client.g-zip-content

当服务端支持压缩的情况下，是否支持从服务端获取的信息进行压缩。默认为true

eureka.client.property-resolver

属性解析器
eureka.client.proxy-host

获取eureka server 的代理主机名
null

eureka.client.proxy-password

获取eureka server 的代理主机密码

null

eureka.client.proxy-port

获取eureka server 的代理主机端口

null

eureka.client.proxy-user-name

获取eureka server 的代理用户名

null

eureka.client.region

获取此实例所在的区域（在AWS数据中心中使用）。

us-east-1

eureka.client.should-enforce-registration-at-init

client 在初始化阶段是否强行注册到注册中心

false

eureka.client.should-unregister-on-shutdown

client在shutdown情况下，是否显示从注册中心注销

true

服务实例配置项（eureka.instance.*）

org.springframework.cloud.netflix.eureka.EurekaInstanceConfigBean

参数名称 说明 默认值
eureka.instance.appname

注册到注册中心的应用名称
unknown

eureka.instance.a-s-g-name

注册到注册中心的应用所属分组名称（AWS服务器） null
eureka.instance.app-group-name

注册到注册中心的应用所属分组名称

null
eureka.instance.data-center-info

指定服务实例所属数据中心

eureka.instance.instance-enabled-onit

指示是否应在eureka注册后立即启用实例以获取流量

false
eureka.instance.non-secure-port

http通信端口
80

eureka.instance.non-secure-port-enabled

是否启用HTTP通信端口　　ture
eureka.instance.secure-port

HTTPS通信端口
443

eureka.instance.secure-port-enabled

是否启用HTTPS通信端口　　false
eureka.instance.secure-virtual-host-name

服务实例安全主机名称（HTTPS）　　unknown
eureka.instance.virtual-host-name

该服务实例非安全注解名称（HTTP）　　unknown
eureka.instance.secure-health-check-url

该服务实例安全健康检查地址（URL），绝对地址

eureka.instance.lease-renewal-interval-in-seconds

该服务实例向注册中心发送心跳间隔（s）
30

eureka.instance.lease-expiration-duration-in-seconds

指示eureka服务器在删除此实例之前收到最后一次心跳之后等待的时间（s）

90

eureka.instance.metadata-map.*

eureka.instance.ip-address

该服务实例的IP地址 null
eureka.instance.prefer-ip-address

是否优先使用服务实例的IP地址，相较于hostname false
eureka.instance.status-page-url

```
该服务实例的状态检查地址（url），绝对地址      null
eureka.instance.status-page-url-path

该服务实例的状态检查地址，相对地址
/actuator/info

eureka.instance.home-page-url

该服务实例的主页地址（url），绝对地址


eureka.instance.home-page-url-path

该服务实例的主页地址，相对地址
/

eureka.instance.health-check-url

该服务实例的健康检查地址（url），绝对地址
null

eureka.instance.health-check-url-path

该服务实例的健康检查地址，相对地址
/actuator/health

eureka.instance.instance-id

该服务实例在注册中心的唯一实例ID


eureka.instance.hostname

该服务实例所在主机名


eureka.instance.namespace

获取用于查找属性的命名空间。 在Spring Cloud中被忽略。

eureka

eureka.instance.environment

该服务实例环境配置
```

```
eureka.instance.default-address-resolution-order

默认地址解析顺序


eureka.instance.initial-status

该服务实例注册到Eureka Server 的初始状态     up
eureka.instance.registry.default-open-for-traffic-count

【Eureka Server 端属性】默认开启通信的数量
1

eureka.instance.registry.expected-number-of-renews-per-min

【Eureka Server 端属性】每分钟续约次数
1

Eureka Server 配置项（eureka.server.*）

org.springframework.cloud.netflix.eureka.server.EurekaServerConf
igBean

参数名称  说明      默认值
eureka.server.enable-self-preservation

启用自我保护机制，默认为true         true
eureka.server.eviction-interval-timer-in-ms

清除无效服务实例的时间间隔（ms），默认1分钟
60000

eureka.server.delta-retention-timer-interval-in-ms

清理无效增量信息的时间间隔（ms），默认30秒
30000

eureka.server.disable-delta

禁用增量获取服务实例信息     false
eureka.server.log-identity-headers

是否记录登录日志   true
```

eureka.server.rate-limiter-burst-size

限流大小
10

eureka.server.rate-limiter-enabled

是否启用限流　　false
eureka.server.rate-limiter-full-fetch-average-rate

平均请求速率
100

eureka.server.rate-limiter-throttle-standard-clients

是否对标准客户端进行限流　　false
eureka.server.rate-limiter-registry-fetch-average-rate

服务注册与拉取的平均速率
500

eureka.server.rate-limiter-privileged-clients

信任的客户端列表

eureka.server.renewal-percent-threshold

15分钟内续约服务的比例小于0.85，则开启自我保护机制，再此期间不会清除已注册的任何服务（即便是无效服务）

0.85

eureka.server.renewal-threshold-update-interval-ms

更新续约阈值的间隔（分钟），默认15分钟
15

eureka.server.response-cache-auto-expiration-in-seconds

注册信息缓存有效时长（s），默认180秒
180

eureka.server.response-cache-update-interval-ms

注册信息缓存更新间隔（s），默认30秒
30

eureka.server.retention-time-in-m-s-in-delta-queue

保留增量信息时长（分钟），默认3分钟
3

eureka.server.sync-when-timestamp-differs

当时间戳不一致时，是否进行同步　　true
eureka.server.use-read-only-response-cache

是否使用只读缓存策略　　true

自定义工具设置

eureka.server.json-codec-name

Json编解码器名称

eureka.server.property-resolver

属性解析器名称

eureka.server.xml-codec-name

Xml编解码器名称

Eureka Server 集群配置

eureka.server.enable-replicated-request-compression

复制数据请求时，数据是否压缩 false
eureka.server.batch-replication

节点之间数据复制是否采用批处理　　false

```
eureka.server.max-elements-in-peer-replication-pool

备份池最大备份事件数量，默认1000
1000

eureka.server.max-elements-in-status-replication-pool

状态备份池最大备份事件数量，默认1000
1000

eureka.server.max-idle-thread-age-in-minutes-for-peer-
replication

节点之间信息同步线程最大空闲时间（分钟）
15

eureka.server.max-idle-thread-in-minutes-age-for-status-
replication

节点之间状态同步线程最大空闲时间（分钟）
10

eureka.server.max-threads-for-peer-replication

节点之间信息同步最大线程数量
20

eureka.server.max-threads-for-status-replication

节点之间状态同步最大线程数量
1

eureka.server.max-time-for-replication

节点之间信息复制最大通信时长（ms）
30000

eureka.server.min-available-instances-for-peer-replication

集群中服务实例最小数量，-1 表示单节点
-1

eureka.server.min-threads-for-peer-replication

节点之间信息复制最小线程数量
```

```
5

eureka.server.min-threads-for-status-replication

节点之间信息状态同步最小线程数量
1

eureka.server.number-of-replication-retries

节点之间数据复制时，可重试次数
5

eureka.server.peer-eureka-nodes-update-interval-ms

节点更新数据间隔时长（分钟）
10

eureka.server.peer-eureka-status-refresh-time-interval-ms

节点之间状态刷新间隔时长（ms）
30000

eureka.server.peer-node-connect-timeout-ms

节点之间连接超时时长（ms）
200

eureka.server.peer-node-connection-idle-timeout-seconds

节点之间连接后，空闲时长（s）
30

eureka.server.peer-node-read-timeout-ms

几点之间数据读取超时时间（ms）
200

eureka.server.peer-node-total-connections

集群中节点连接总数
1000

eureka.server.peer-node-total-connections-per-host

节点之间连接，单机最大连接数量
```

```
500

eureka.server.registry-sync-retries

节点启动时，尝试获取注册信息的次数
500

eureka.server.registry-sync-retry-wait-ms

节点启动时，尝试获取注册信息的间隔时长（ms）
30000

eureka.server.wait-time-in-ms-when-sync-empty

在Eureka服务器获取不到集群里对等服务器上的实例时，需要等待的时间（分钟）

5
```

Eureka instance 配置项

```
#服务注册中心实例的主机名
eureka.instance.hostname=localhost
#注册在Eureka服务中的应用组名
eureka.instance.app-group-name=
#注册在的Eureka服务中的应用名称
eureka.instance.appname=
#该实例注册到服务中心的唯一ID
eureka.instance.instance-id=
#该实例的IP地址
eureka.instance.ip-address=
#该实例，相较于hostname是否优先使用IP
eureka.instance.prefer-ip-address=false

#用于AWS平台自动扩展的与此实例关联的组名，
eureka.instance.a-s-g-name=
#部署此实例的数据中心
eureka.instance.data-center-info=
#默认的地址解析顺序
```

```
eureka.instance.default-address-resolution-order=
#该实例的环境配置
eureka.instance.environment=
#初始化该实例，注册到服务中心的初始状态
eureka.instance.initial-status=up
#表明是否只要此实例注册到服务中心，立马就进行通信
eureka.instance.instance-enabled-onit=false
#该服务实例的命名空间,用于查找属性
eureka.instance.namespace=eureka
#该服务实例的子定义元数据，可以被服务中心接受到
eureka.instance.metadata-map.test = test

#服务中心删除此服务实例的等待时间(秒为单位),时间间隔为最后一次服务中心接受到的心跳时间
eureka.instance.lease-expiration-duration-in-seconds=90
#该实例给服务中心发送心跳的间隔时间，用于表明该服务实例可用
eureka.instance.lease-renewal-interval-in-seconds=30
#该实例，注册服务中心，默认打开的通信数量
eureka.instance.registry.default-open-for-traffic-count=1
#每分钟续约次数
eureka.instance.registry.expected-number-of-renews-per-min=1

#该实例健康检查url,绝对路径
eureka.instance.health-check-url=
#该实例健康检查url,相对路径
eureka.instance.health-check-url-path=/health
#该实例的主页url,绝对路径
eureka.instance.home-page-url=
#该实例的主页url,相对路径
eureka.instance.home-page-url-path=/
#该实例的安全健康检查url,绝对路径
eureka.instance.secure-health-check-url=
#https通信端口
eureka.instance.secure-port=443
#https通信端口是否启用
eureka.instance.secure-port-enabled=false
#http通信端口
eureka.instance.non-secure-port=80
#http通信端口是否启用
eureka.instance.non-secure-port-enabled=true
#该实例的安全虚拟主机名称(https)
eureka.instance.secure-virtual-host-name=unknown
#该实例的虚拟主机名称(http)
eureka.instance.virtual-host-name=unknown
#该实例的状态呈现url,绝对路径
```

```
eureka.instance.status-page-url=
#该实例的状态呈现url,相对路径
eureka.instance.status-page-url-path=/status
```

Eureka client 配置项

```
#该客户端是否可用
eureka.client.enabled=true
#实例是否在eureka服务器上注册自己的信息以供其他服务发现，默认为true
eureka.client.register-with-eureka=false
#此客户端是否获取eureka服务器注册表上的注册信息，默认为true
eureka.client.fetch-registry=false
#是否过滤掉，非UP的实例。默认为true
eureka.client.filter-only-up-instances=true
#与Eureka注册服务中心的通信zone和url地址
eureka.client.serviceUrl.defaultZone=http://${eureka.instance.ho
stname}:${server.port}/eureka/

#client连接Eureka服务端后的空闲等待时间，默认为30 秒
eureka.client.eureka-connection-idle-timeout-seconds=30
#client连接eureka服务端的连接超时时间，默认为5秒
eureka.client.eureka-server-connect-timeout-seconds=5
#client对服务端的读超时时长
eureka.client.eureka-server-read-timeout-seconds=8
#client连接all eureka服务端的总连接数，默认200
eureka.client.eureka-server-total-connections=200
#client连接eureka服务端的单机连接数量，默认50
eureka.client.eureka-server-total-connections-per-host=50
#执行程序指数回退刷新的相关属性，是重试延迟的最大倍数值，默认为10
eureka.client.cache-refresh-executor-exponential-back-off-
bound=10
#执行程序缓存刷新线程池的大小，默认为5
eureka.client.cache-refresh-executor-thread-pool-size=2
#心跳执行程序回退相关的属性，是重试延迟的最大倍数值，默认为10
eureka.client.heartbeat-executor-exponential-back-off-bound=10
#心跳执行程序线程池的大小,默认为5
eureka.client.heartbeat-executor-thread-pool-size=5
# 询问Eureka服务url信息变化的频率（s），默认为300秒
eureka.client.eureka-service-url-poll-interval-seconds=300
#最初复制实例信息到eureka服务器所需的时间（s），默认为40秒
```

```
eureka.client.initial-instance-info-replication-interval-
seconds=40
#间隔多长时间再次复制实例信息到eureka服务器，默认为30秒
eureka.client.instance-info-replication-interval-seconds=30
#从eureka服务器注册表中获取注册信息的时间间隔（s），默认为30秒
eureka.client.registry-fetch-interval-seconds=30

# 获取实例所在的地区。默认为us-east-1
eureka.client.region=us-east-1
#实例是否使用同一zone里的eureka服务器，默认为true，理想状态下，eureka客户
端与服务端是在同一zone下
eureka.client.prefer-same-zone-eureka=true
# 获取实例所在的地区下可用性的区域列表，用逗号隔开。（AWS）
eureka.client.availability-
zones.china=defaultZone,defaultZone1,defaultZone2
#eureka服务注册表信息里的以逗号隔开的地区名单，如果不这样返回这些地区名单，
则客户端启动将会出错。默认为null
eureka.client.fetch-remote-regions-registry=
#服务器是否能够重定向客户端请求到备份服务器。 如果设置为false，服务器将直接
处理请求，如果设置为true，它可能发送HTTP重定向到客户端。默认为false
eureka.client.allow-redirects=false
#客户端数据接收
eureka.client.client-data-accept=
#增量信息是否可以提供给客户端看，默认为false
eureka.client.disable-delta=false
#eureka服务器序列化/反序列化的信息中获取“_”符号的的替换字符串。默认为“__“
eureka.client.escape-char-replacement=__
#eureka服务器序列化/反序列化的信息中获取“$”符号的替换字符串。默认为“_-”
eureka.client.dollar-replacement="_-"
#当服务端支持压缩的情况下，是否支持从服务端获取的信息进行压缩。默认为true
eureka.client.g-zip-content=true
#是否记录eureka服务器和客户端之间在注册表的信息方面的差异，默认为false
eureka.client.log-delta-diff=false
# 如果设置为true,客户端的状态更新将会点播更新到远程服务器上，默认为true
eureka.client.on-demand-update-status-change=true
#此客户端只对一个单一的VIP注册表的信息感兴趣。默认为null
eureka.client.registry-refresh-single-vip-address=
#client是否在初始化阶段强行注册到服务中心，默认为false
eureka.client.should-enforce-registration-at-init=false
#client在shutdown的时候是否显示的注销服务从服务中心，默认为true
eureka.client.should-unregister-on-shutdown=true

# 获取eureka服务的代理主机，默认为null
eureka.client.proxy-host=
#获取eureka服务的代理密码，默认为null
```

```
eureka.client.proxy-password=
# 获取eureka服务的代理端口，默认为null
eureka.client.proxy-port=
# 获取eureka服务的代理用户名，默认为null
eureka.client.proxy-user-name=

#属性解释器
eureka.client.property-resolver=
#获取实现了eureka客户端在第一次启动时读取注册表的信息作为回退选项的实现名称
eureka.client.backup-registry-impl=
#这是一个短暂的×××的配置，如果最新的×××是稳定的，则可以去除，默认为null
eureka.client.decoder-name=
#这是一个短暂的编码器的配置，如果最新的编码器是稳定的，则可以去除，默认为
null
eureka.client.encoder-name=

#是否使用DNS机制去获取服务列表，然后进行通信。默认为false
eureka.client.use-dns-for-fetching-service-urls=false
#获取要查询的DNS名称来获得eureka服务器，此配置只有在eureka服务器ip地址列表
是在DNS中才会用到。默认为null
eureka.client.eureka-server-d-n-s-name=
#获取eureka服务器的端口，此配置只有在eureka服务器ip地址列表是在DNS中才会用
到。默认为null
eureka.client.eureka-server-port=
#表示eureka注册中心的路径，如果配置为eureka，则为
http://x.x.x.x:x/eureka/，在eureka的配置文件中加入此配置表示eureka作为
客户端向注册中心注册，从而构成eureka集群。此配置只有在eureka服务器ip地址列
表是在DNS中才会用到，默认为null
eureka.client.eureka-server-u-r-l-context=
```

Eureka Server配置项

```
#服务端开启自我保护模式。无论什么情况，服务端都会保持一定数量的服务。避免
client与server的网络问题，而出现大量的服务被清除。
eureka.server.enable-self-preservation=true
#开启清除无效服务的定时任务，时间间隔。默认1分钟
eureka.server.eviction-interval-timer-in-ms= 60000
#间隔多长时间，清除过期的delta数据
eureka.server.delta-retention-timer-interval-in-ms=0
#过期数据，是否也提供给client
```

```
eureka.server.disable-delta=false
#eureka服务端是否记录client的身份header
eureka.server.log-identity-headers=true
#请求频率限制器
eureka.server.rate-limiter-burst-size=10
#是否开启请求频率限制器
eureka.server.rate-limiter-enabled=false
#请求频率的平均值
eureka.server.rate-limiter-full-fetch-average-rate=100
#是否对标准的client进行频率请求限制。如果是false，则只对非标准client进行限制
eureka.server.rate-limiter-throttle-standard-clients=false
#注册服务、拉去服务列表数据的请求频率的平均值
eureka.server.rate-limiter-registry-fetch-average-rate=500
#设置信任的client list
eureka.server.rate-limiter-privileged-clients=
#在设置的时间范围类，期望与client续约的百分比。
eureka.server.renewal-percent-threshold=0.85
#多长时间更新续约的阈值
eureka.server.renewal-threshold-update-interval-ms=0
#对于缓存的注册数据，多长时间过期
eureka.server.response-cache-auto-expiration-in-seconds=180
#多长时间更新一次缓存中的服务注册数据
eureka.server.response-cache-update-interval-ms=0
#缓存增量数据的时间，以便在检索的时候不丢失信息
eureka.server.retention-time-in-m-s-in-delta-queue=0
#当时间戳不一致的时候，是否进行同步
eureka.server.sync-when-timestamp-differs=true
#是否采用只读缓存策略，只读策略对于缓存的数据不会过期。
eureka.server.use-read-only-response-cache=true

################server 自定义实现的配置####################33
#json的转换的实现类名
eureka.server.json-codec-name=
#PropertyResolver
eureka.server.property-resolver=
#eureka server xml的编解码实现名称
eureka.server.xml-codec-name=

################server node 与 node 之间关联的配置
####################33
#发送复制数据是否在request中，总是压缩
eureka.server.enable-replicated-request-compression=false
#指示群集节点之间的复制是否应批处理以提高网络效率。
eureka.server.batch-replication=false
```

```
#允许备份到备份池的最大复制事件数量。而这个备份池负责除状态更新的其他事件。可以根据内存大小，超时和复制流量，来设置此值得大小
eureka.server.max-elements-in-peer-replication-pool=10000
#允许备份到状态备份池的最大复制事件数量
eureka.server.max-elements-in-status-replication-pool=10000
#多个服务中心相互同步信息线程的最大空闲时间
eureka.server.max-idle-thread-age-in-minutes-for-peer-replication=15
#状态同步线程的最大空闲时间
eureka.server.max-idle-thread-in-minutes-age-for-status-replication=15
#服务注册中心各个instance相互复制数据的最大线程数量
eureka.server.max-threads-for-peer-replication=20
#服务注册中心各个instance相互复制状态数据的最大线程数量
eureka.server.max-threads-for-status-replication=1
#instance之间复制数据的通信时长
eureka.server.max-time-for-replication=30000
#正常的对等服务instance最小数量。-1表示服务中心为单节点。
eureka.server.min-available-instances-for-peer-replication=-1
#instance之间相互复制开启的最小线程数量
eureka.server.min-threads-for-peer-replication=5
#instance之间用于状态复制，开启的最小线程数量
eureka.server.min-threads-for-status-replication=1
#instance之间复制数据时可以重试的次数
eureka.server.number-of-replication-retries=5
#eureka节点间间隔多长时间更新一次数据。默认10分钟。
eureka.server.peer-eureka-nodes-update-interval-ms=600000
#eureka服务状态的相互更新的时间间隔。
eureka.server.peer-eureka-status-refresh-time-interval-ms=0
#eureka对等节点间连接超时时间
eureka.server.peer-node-connect-timeout-ms=200
#eureka对等节点连接后的空闲时间
eureka.server.peer-node-connection-idle-timeout-seconds=30
#节点间的读数据连接超时时间
eureka.server.peer-node-read-timeout-ms=200
#eureka server 节点间连接的总共最大数量
eureka.server.peer-node-total-connections=1000
#eureka server 节点间连接的单机最大数量
eureka.server.peer-node-total-connections-per-host=10
#在服务节点启动时，eureka尝试获取注册信息的次数
eureka.server.registry-sync-retries=
#在服务节点启动时，eureka多次尝试获取注册信息的间隔时间
eureka.server.registry-sync-retry-wait-ms=
#当eureka server启动的时候，不能从对等节点获取instance注册信息的情况，应等待多长时间。
```

```
eureka.server.wait-time-in-ms-when-sync-empty=0

###############server 与 remote 关联的配置####################33
#过期数据，是否也提供给远程region
eureka.server.disable-delta-for-remote-regions=false
#回退到远程区域中的应用程序的旧行为 (如果已配置) 如果本地区域中没有该应用程序的实例，则将被禁用。
eureka.server.disable-transparent-fallback-to-other-region=false
#指示在服务器支持的情况下，是否必须为远程区域压缩从尤里卡服务器获取的内容。
eureka.server.g-zip-content-from-remote-region=true
#连接eureka remote note的连接超时时间
eureka.server.remote-region-connect-timeout-ms=1000
#remote region 应用白名单
eureka.server.remote-region-app-whitelist.
#连接eureka remote note的连接空闲时间
eureka.server.remote-region-connection-idle-timeout-seconds=30
#执行remote region 获取注册信息的请求线程池大小
eureka.server.remote-region-fetch-thread-pool-size=20
#remote region 从对等eureka加点读取数据的超时时间
eureka.server.remote-region-read-timeout-ms=1000
#从remote region 获取注册信息的时间间隔
eureka.server.remote-region-registry-fetch-interval=30
#remote region 连接eureka节点的总连接数量
eureka.server.remote-region-total-connections=1000
#remote region 连接eureka节点的单机连接数量
eureka.server.remote-region-total-connections-per-host=50
#remote region抓取注册信息的存储文件，而这个可靠的存储文件需要全限定名来指定
eureka.server.remote-region-trust-store=
#remote region 储存的文件的密码
eureka.server.remote-region-trust-store-password=
#remote region url.多个逗号隔开
eureka.server.remote-region-urls=
#remote region url.多个逗号隔开
eureka.server.remote-region-urls-with-name.

################server 与 ASG/AWS/EIP/route52 之间关联的配置
####################33
#缓存ASG信息的过期时间。
eureka.server.a-s-g-cache-expiry-timeout-ms=0
#查询ASG信息的超时时间
eureka.server.a-s-g-query-timeout-ms=300
#服务更新ASG信息的频率
eureka.server.a-s-g-update-interval-ms=0
#AWS访问ID
```

```
eureka.server.a-w-s-access-id=
#AWS安全密钥
eureka.server.a-w-s-secret-key=
#AWS绑定策略
eureka.server.binding-strategy=eip
#用于从第三方AWS 帐户描述自动扩展分组的角色的名称。
eureka.server.list-auto-scaling-groups-role-name=
#是否应该建立连接引导
eureka.server.prime-aws-replica-connections=true
#服务端尝试绑定候选EIP的次数
eureka.server.e-i-p-bind-rebind-retries=3
#服务端绑定EIP的时间间隔.如果绑定就检查;如果绑定失效就重新绑定。当且仅当已经绑定的情况
eureka.server.e-i-p-binding-retry-interval-ms=10
#服务端绑定EIP的时间间隔.当且仅当服务为绑定的情况
eureka.server.e-i-p-binding-retry-interval-ms-when-unbound=
#服务端尝试绑定route53的次数
eureka.server.route53-bind-rebind-retries=3
#服务端间隔多长时间尝试绑定route53
eureka.server.route53-binding-retry-interval-ms=30
#
eureka.server.route53-domain-t-t-l=10
```

4.6. ribbon

```
@Configuration
public class RibbonConfigure {

    @LoadBalanced
    @Bean
    public RestTemplate restTemplate(){
        return new RestTemplate();
    }

    //指定Ribbon使用随机策略
    @Bean
    public IRule loadBalanceRule(){
        //return new RandomRule();
        List<Integer> ports = new ArrayList<>();
        ports.add(8081);
```

```
        return new CustomRule(ports);
    }
}
```

LoadBalancerClient 实例

application.properties

```
web.ribbon.listOfServers=localhost:7900,localhost:7901,localhost
:7902
```

LoadBalancerClient 获取服务器列表

```
package cn.netkiller.openfeign.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import
org.springframework.cloud.client.loadbalancer.LoadBalancerClient
;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class TestController {

        @Autowired
        private LoadBalancerClient loadBalancerClient;

        @GetMapping("/lb")
        public String LoadBalancer() {
                ServiceInstance serviceInstance =
this.loadBalancerClient.choose("web");
                System.out.println("Server: " +
serviceInstance.getServiceId() + ":" + serviceInstance.getHost()
```

```
+ ":"
                                    + serviceInstance.getPort());

                return serviceInstance.toString();
        }

}
```

Ribbon 相关配置

```
spring.cloud.loadbalancer.ribbon.enabled=false
```

内置负载均衡策略

```
provider.ribbon.NFLoadBalancerRuleClassName=com.netflix.loadbala
ncer.RandomRule
```

```
RoundRobinRule
轮询策略。Ribbon 默认采用的策略。若经过一轮轮询没有找到可用的 provider，其
最多 轮询 10 轮。若最终还没有找到，则返回 null。

RandomRule
随机策略，从所有可用的 provider 中随机选择一个。

RetryRule
重试策略。先按照 RoundRobinRule 策略获取 provider，若获取失败，则在指定
的时限内重试。默认的时限为 500 毫秒。

BestAvailableRule
最可用策略。选择并发量最小的 provider，即连接的消费者数量最少的 provider。
```

AvailabilityFilteringRule
可用过滤算法。该算法规则是:过滤掉处于熔断状态的 provider 与已经超过连接极限的 provider，对剩余 provider 采用轮询策略。

ZoneAvoidanceRule
zone 回避策略。根据 provider 所在 zone 及 provider 的可用性，对 provider 进行选择。

WeightedResponseTimeRule
“权重响应时间”策略。根据每个 provider 的平均响应时间计算其权重，响应时间越快权重越大，被选中的机率就越高。在刚启动时采用轮询策略。后面就会根据权重进行选择了。

4.7. 获取 EurekaClient 信息

```
package cn.netkiller.sample;

import com.netflix.discovery.EurekaClient;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.context.annotation.Lazy;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

@SpringBootApplication
@EnableEurekaClient
@RestController
public class WebFluxApplication {

    @Autowired
    @Lazy
    private EurekaClient eurekaClient;

    @Value("${spring.application.name}")
```

```
    private String appName;

    public static void main(String[] args) {
        SpringApplication.run(WebFluxApplication.class, args);
    }

    @GetMapping("/client")
    public Mono<String> greeting() {
        String idInEureka =
eurekaClient.getApplication(appName).getInstances().get(0).getId
();
        return Mono.just(String.format("Hello from '%s'!",
idInEureka));
    }

    @GetMapping("/client2")
    public Mono<String> greetingWithParam(@RequestParam(value =
"id") Long id) {
        String idInEureka =
eurekaClient.getApplication(appName).getInstances().get(0).getId
();
        return Mono.just(String.format("Hello with param from
'%s'!", idInEureka));
    }
}
```

4.8. Zuul

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>zuul</artifactId>
        <version>0.0.1-SNAPSHOT</version>
```

```
        <packaging>jar</packaging>

        <name>zuul</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.6.RELEASE</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>
        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>Dalston.SR2</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
zuul</artifactId>
                </dependency>
        </dependencies>
</project>
```

EnableZuulProxy

```
package cn.netkiller.zuul;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
public class Application {
        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
        }
}
```

application.yml

```
server:
  port: 8765

logging:
  level:
    ROOT: INFO
    org.springframework.web: DEBUG

zuul:
  routes:
    restful:
      path: /restful/**
      url: http://api:password@api.netkiller.com:8080/restful
```

负载均衡配置

```
zuul:
  routes:
    httpbin:
      path: /**
      serviceId: httpslb

httpslb:
  ribbon:
    listOfServers: api1.netkiller.org, api2.netkiller.cn
```

5. Openfeign

5.1. Openfeign 扫描包配置

```
@EnableFeignClients(basePackages = {"cn.netkiller.openfeign"})
```

5.2. 用户认证

```
@Configuration
@EnableFeignClients
public class FeignConfiguration {
    @Bean
    public Contract feignContract() {
        return new feign.Contract.Default();
    }

    @Bean
    public BasicAuthRequestInterceptor
basicAuthRequestInterceptor() {
        return new BasicAuthRequestInterceptor("user",
"password");
    }
}
```

5.3. 应用实例

```
@FeignClient(name="myServiceName", url="localhost:8888")
public interface OpenfeignService {
        @RequestMapping("/")
        public String getName();
```

```
}
```

5.4. 配置连接方式

系统默认是 httpclient

httpclient

```
            <dependency>
        <groupId>io.github.openfeign</groupId>
        <artifactId>feign-httpclient</artifactId>
    </dependency>
```

httpclient

```
feign.httpclient.enabled=true
feign.httpclient.max-connections=1000
feign.httpclient.max-connections-per-route=50
```

okhttp

Maven 依赖

```
<dependency>
    <groupId>io.github.openfeign</groupId>
    <artifactId>feign-okhttp</artifactId>
</dependency>
```

```

```

启用 okhttp

```
feign.httpclient.enabled=false
feign.okhttp.enabled=true
feign.httpclient.max-connections=1000
feign.httpclient.max-connections-per-route=50
```

5.5. 配置手册

https://docs.spring.io/spring-cloud-openfeign/docs/current/reference/html/appendix.html

6. Spring Cloud Gateway

SpringCloud Gateway是基于WebFlux框架实现的网关服务器

gateway网关路由配置有两种方式

1. 通过@Bean自定义RouteLocator，在启动主类Application中配置
2. 在配置文件yml中配置

这两种方式都可以实现网关路由，还可以同时使用，写在配置配置文件中对于运维更友好。

6.1. Gateway 例子

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>gateway</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>gateway</name>
        <url>http://www.netkiller.cn</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>11</java.version>
                <spring-cloud.version>Greenwich.SR1</spring-
```

```
cloud.version>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.1.3.RELEASE</version>
                <relativePath />
        </parent>


        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>${spring-
cloud.version}</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
gateway</artifactId>
                </dependency>
        </dependencies>
</project>
```

SpringApplication

```
package cn.netkiller.gateway;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
        public static void main(String[] args) {

                SpringApplication.run(Application.class,
args);

        }
}
```

application.yml

resources/application.yml

```
server:
  port: 8080
spring:
  application:
    name: spring-cloud-gateway
  cloud:
    gateway:
      routes:
      - id: linux
        uri: http://www.netkiller.cn
```

```
        predicates:
        - Path=/linux

logging:
  level:
    org.springframework.cloud.gateway: TRACE
    org.springframework.http.server.reactive: DEBUG
    org.springframework.web.reactive: DEBUG
    reactor.ipc.netty: DEBUG
```

RouteLocator 方式

```
package com.springcloud.gateway;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.gateway.route.RouteLocator;
import
org.springframework.cloud.gateway.route.builder.RouteLocatorB
uilder;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class GatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class,
args);
    }

    @Bean
    public RouteLocator
customRouteLocator(RouteLocatorBuilder builder) {
        return builder.routes()
                .route("path_route", r -> r.path("/linux")
                        .uri("http://www.netkiller.cn"))
                .build();
    }
```

```
}
```

6.2. 路由配置

转发操作

```
        @Bean
        public RouteLocator
customRouteLocator(RouteLocatorBuilder builder) {
                return builder.routes().route("path_route", r
-> r.path("/ch/history/").uri("https://new.qq.com")).build();
        }
```

URL 参数

```
        @Bean
        public RouteLocator
customRouteLocator(RouteLocatorBuilder builder) {
                return builder.routes().route("query_route",
r -> r.query("id",
"1000").uri("https://news.netkiller.cn/news")).build();
        }
```

```
curl http://localhost:8080/?id=1000
```

传递参数

```
        @Bean
        public RouteLocator
customRouteLocator(RouteLocatorBuilder builder) {
                return builder.routes().route("query_route",
r -> r.query("q").uri("https://cn.bing.com/search")).build();
        }
```

```
http://localhost:8080/?q=netkiller
```

```
        @Bean
        public RouteLocator
customRouteLocator(RouteLocatorBuilder builder) {
                return builder.routes().route("query_route",
r ->
r.query("q").and().path("/search").uri("https://cn.bing.com")
)
                                .build();
        }
```

```
http://localhost:8080/search?q=netkiller
```

7. Spring Cloud Stream

8. Spring Cloud Bus

9. Spring Cloud Sleuth

分布式链路追踪工具

```
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
sleuth</artifactId>
                </dependency>
```

字段定义

```
字段                                    描述
trace                                   从客户发起请求(request)抵达被追踪系
统的边界开始，到被追踪系统向客户返回响应(response)为止的整个过程
span                                    每个trace中会调用若干个服务，为了记
录调用了哪些服务，以及每次调用的消耗时间等信息，在每次调用服务时，埋入一个调
用记录
X-B3-ParentSpanId       标识当前工作单元所属的上一个工作单元
X-B3-Sampled            是否采样，1表示需要被输出，0表示不需要被输出
X-B3-TraceId            一条请求链路(trace)的唯一标识，必须值
X-Span-Name                     工作单元的名称
X-B3-SpanId                     一个工作单元(span)的唯一标识，必须值
```

9.1. logback 安装

```
logging.level.org.springframework.web=DEBUG
spring.sleuth.traceId128=true
spring.sleuth.sampler.probability=1.0
logging.pattern.level=[%X{traceId}/%X{spanId}] %-5p [%t] %C{2}
```

```
- %m%n
```

10. Spring Cloud with Kubernetes

10.1. Config

Spring Cloud 使用 Kubernetes 提供的 Config Maps 作为配置中心。这样的好处是我们无需再启动一个 config 服务。

Maven 依赖

父项目

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>kubernetes</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>kubernetes</name>

        <url>http://www.netkiller.cn</url>
        <description>Spring Cloud with Kubernetes</description>

        <organization>
                <name>Netkiller Spring Cloud 手札</name>
                <url>http://www.netkiller.cn</url>
        </organization>

        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller Spring Cloud 手
```

```
札</organization>

<organizationUrl>http://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
                </developer>
        </developers>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <spring-cloud.version>Hoxton.SR8</spring-
cloud.version>


<docker.registry>registry.netkiller.cn:5000</docker.registry>

<docker.registry.name>netkiller</docker.registry.name>
                <docker.image>mcr.microsoft.com/java/jre:15-
zulu-alpine</docker.image>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.3.4.RELEASE</version>
                <relativePath />
        </parent>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>${spring-
cloud.version}</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
```

```
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <modules>
                <module>service</module>
                <module>ConfigMaps</module>
        </modules>

</project>
```

项目模块

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>kubernetes</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>configmaps</artifactId>
        <version>0.0.1-SNAPSHOT</version>
```

```
        <name>configmaps</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
kubernetes-config</artifactId>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <executions>
                                        <execution>
                                                <goals>

<goal>repackage</goal>
                                                </goals>
                                        </execution>
                                </executions>
                        </plugin>
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.2</version>
                                <configuration>

<imageName>${docker.registry}/${docker.registry.name}/${project
```

```
.artifactId}</imageName>

<baseImage>${docker.image}</baseImage>

<maintainer>netkiller@msn.com</maintainer>
                                        <volumes>/tmp</volumes>
                                        <workdir>/srv</workdir>
                                        <exposes>8080</exposes>
                                        <env>
                                                <JAVA_OPTS>-
server -Xms128m -Xmx256m</JAVA_OPTS>
                                        </env>
                                        <entryPoint>["sh", "-
c", "java ${JAVA_OPTS} -jar /srv/${project.build.finalName}.jar
${SPRING_OPTS}"]</entryPoint>
                                        <resources>
                                                <resource>

<targetPath>/srv</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>
                                        <!--
<image>${docker.image.prefix}/${project.artifactId}</image> -->
                                        <!--
<newName>${docker.image.prefix}/${project.artifactId}:${project
.version}</newName> -->
                                        <!-- <serverId>docker-
hub</serverId> -->

<registryUrl>http://${docker.registry}/v2/</registryUrl>
                                        <imageTags>
                                                <!--
<imageTag>${project.version}</imageTag> -->

<imageTag>latest</imageTag>
                                        </imageTags>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
```

```
</project>
```

Spring Cloud 配置文件

src/main/resources/bootstrap.yml

```
spring:
  application:
    name: spring-cloud-kubernetes-configmaps
  profiles:
    active: dev
  cloud:
    kubernetes:
      reload:
        enabled: true
        mode: polling
        period: 5000
      config:
        sources:
          - name: ${spring.application.name}
            group: cn.netkiller
            namespace: default

management:
  security:
    enabled: false
  #context-path: /
  endpoints:
    web:
      exposure:
        include: refresh
```

程序文件

SpringBootApplication 启动文件

```
package cn.netkiller.config;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class App {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                SpringApplication.run(App.class, args);
        }
}
```

配置类

```
package cn.netkiller.config;

import
org.springframework.boot.context.properties.ConfigurationProp
erties;
import org.springframework.context.annotation.Configuration;

@Configuration
@ConfigurationProperties(prefix = "greeting")
public class KeyValueConfig {
        private String message = "This is a default message";

        public String getMessage() {
                return this.message;
        }

        public void setMessage(String message) {
                this.message = message;
        }
}
```

控制器

```
package cn.netkiller.config;

import java.text.SimpleDateFormat;
import java.util.Date;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.context.properties.EnableConfigurati
onProperties;
import
org.springframework.cloud.context.config.annotation.RefreshSc
ope;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

@RestController
@EnableConfigurationProperties(KeyValueConfig.class)
@RefreshScope
public class TestController {
        @Autowired
        private KeyValueConfig keyValueConfig;

        @GetMapping("/")
        public String index() {
                return "Hello world\r\n";
        }

        @GetMapping("/hello")
        public String hello() {
                return keyValueConfig.getMessage() + " [" +
new SimpleDateFormat().format(new Date()) + "]";
        }
}
```

Kubernetes 编排脚本

config.yaml

```
apiVersion: v1
kind: Service
metadata:
  annotations:
    fabric8.io/git-commit:
729badc5e8578b67c1f9387ac0d1949b0646a991
    fabric8.io/git-branch: master
    fabric8.io/git-url:
https://netkiller@github.com/netkiller/spring-cloud-
kubernetes.git
    fabric8.io/scm-url: https://github.com/spring-
projects/spring-boot/kubernetes/ConfigMaps
    fabric8.io/scm-tag: HEAD
    prometheus.io/port: "9779"
    prometheus.io/scrape: "true"
  labels:
    expose: "true"
    app: ConfigMaps
    provider: fabric8
    version: 0.0.1-SNAPSHOT
    group: cn.netkiller
  name: config
spec:
  ports:
  - name: http
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: ConfigMaps
    provider: fabric8
    group: cn.netkiller
  type: NodePort
---
```

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: spring-cloud-kubernetes-configmaps
data:
  application.yml: |-
    greeting:
      message: Say Hello to the World
    farewell:
      message: Say Goodbye
    ---
    spring:
      profiles: development
    greeting:
      message: Say Hello to the Developers
    farewell:
      message: Say Goodbye to the Developers
    ---
    spring:
      profiles: production
    greeting:
      message: Say Hello to the Ops
---
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    fabric8.io/git-commit:
729badc5e8578b67c1f9387ac0d1949b0646a991
    fabric8.io/git-branch: master
    fabric8.io/git-url:
https://netkiller@github.com/netkiller/spring-cloud-
kubernetes.git
    fabric8.io/scm-url: https://github.com/spring-
projects/spring-boot/kubernetes/ConfigMaps
    fabric8.io/scm-tag: HEAD
  labels:
    app: ConfigMaps
    provider: fabric8
    version: 0.0.1-SNAPSHOT
    group: cn.netkiller
  name: config
spec:
  replicas: 1
  revisionHistoryLimit: 2
```

```
  selector:
    matchLabels:
      app: ConfigMaps
      provider: fabric8
      group: cn.netkiller
  template:
    metadata:
      annotations:
        fabric8.io/git-commit:
729badc5e8578b67c1f9387ac0d1949b0646a991
        fabric8.io/git-branch: master
        fabric8.io/scm-tag: HEAD
        fabric8.io/git-url:
https://netkiller@github.com/netkiller/spring-cloud-
kubernetes.git
        fabric8.io/scm-url: https://github.com/spring-
projects/spring-boot/kubernetes/ConfigMaps
      labels:
        app: ConfigMaps
        provider: fabric8
        version: 0.0.1-SNAPSHOT
        group: cn.netkiller
    spec:
      containers:
      - env:
        - name: KUBERNETES_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        image:
registry.netkiller.cn:5000/netkiller/configmaps:latest
        #imagePullPolicy: IfNotPresent
        name: spring-boot
        ports:
        - containerPort: 8080
          name: http
          protocol: TCP
        - containerPort: 9779
          name: prometheus
          protocol: TCP
        - containerPort: 8778
          name: jolokia
          protocol: TCP
        securityContext:
          privileged: false
```

测试

编译并构建 Docker 镜像

```
iMac:ConfigMaps neo$ mvn package docker:build
```

查看镜像是否正确产生

```
iMac:ConfigMaps neo$ docker images
REPOSITORY                                          TAG
IMAGE ID            CREATED             SIZE
registry.netkiller.cn:5000/netkiller/configmaps   latest
93280aec434f        4 minutes ago       219MB
```

上传镜像到私有库

```
iMac:ConfigMaps neo$ docker push
registry.netkiller.cn:5000/netkiller/configmaps
The push refers to repository
[registry.netkiller.cn:5000/netkiller/configmaps]
f56e553c8b82: Pushed
7c1edc21f93f: Layer already exists
50644c29ef5a: Layer already exists
latest: digest:
sha256:3ef48e858254ee4d578fe1737fd948b2679c33d28d0dc573cf1e8076
d0a054a1 size: 952
```

开启 Spring cloud 访问 Kubernetes Config Maps 的权限。

```
kubectl create clusterrolebinding permissive-binding \
  --clusterrole=cluster-admin \
  --user=admin \
  --user=kubelet \
  --group=system:serviceaccounts
```

部署镜像

```
iMac:ConfigMaps neo$ kubectl create -f config.yaml
service/config created
configmap/spring-cloud-kubernetes-configmaps created
deployment.apps/config created
```

查看服务地址

```
iMac:ConfigMaps neo$ minikube service list
|----------------------|---------------------------|-----------
---|---------------------------|
|      NAMESPACE       |           NAME            | TARGET
PORT  |            URL            |
|----------------------|---------------------------|-----------
---|---------------------------|
| default              | config                    | http/8080
| http://192.168.64.12:30662 |
| default              | kubernetes                | No node
port |
| kube-system          | kube-dns                  | No node
port |
| kubernetes-dashboard | dashboard-metrics-scraper | No node
port |
| kubernetes-dashboard | kubernetes-dashboard      | No node
```

```
port |
|----------------------|--------------------------|-----------
---|--------------------------|
```

访问地址，从 Config Maps 中获取配置项。

```
iMac:ConfigMaps neo$ curl http://192.168.64.12:30662/hello
Say Hello to the World [10/7/20, 11:25 AM]
```

修改配置增加了=*=，然后使用 kubectl apply -f config.yaml 刷新

```
iMac:ConfigMaps neo$ curl http://192.168.64.12:30662/hello
Say Hello to the World=*= [10/7/20, 11:26 AM]
```

配置刷新成功

10.2. 注册发现

有了 Kubernetes 注册发现，我们就可以抛弃 Eureka Server。

Maven 父项目

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
```

```
        <groupId>cn.netkiller</groupId>
        <artifactId>kubernetes</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>pom</packaging>

        <name>kubernetes</name>

        <url>http://www.netkiller.cn</url>
        <description>Spring Cloud with Kubernetes</description>

        <organization>
                <name>Netkiller Spring Cloud 手札</name>
                <url>http://www.netkiller.cn</url>
        </organization>

        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller Spring Cloud 手
札</organization>

<organizationUrl>http://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
                </developer>
        </developers>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <spring-cloud.version>Hoxton.SR8</spring-
cloud.version>


<docker.registry>registry.netkiller.cn:5000</docker.registry>

<docker.registry.name>netkiller</docker.registry.name>
                <docker.image>mcr.microsoft.com/java/jre:15-
zulu-alpine</docker.image>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
```

```
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.3.4.RELEASE</version>
                <relativePath />
        </parent>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>${spring-
cloud.version}</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                </dependency>
        </dependencies>

        <modules>
                <module>service</module>
                <module>ConfigMaps</module>
                <module>provider</module>
```

```
                <module>consumer</module>
        </modules>

</project>
```

provider

Maven 依赖

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>kubernetes</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>provider</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>provider</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
kubernetes</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
```

```
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
webflux</artifactId>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>

                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.2</version>
                                <configuration>

<imageName>${docker.registry}/${docker.registry.name}/${project
.artifactId}</imageName>

<baseImage>${docker.image}</baseImage>

<maintainer>netkiller@msn.com</maintainer>
                                        <volumes>/tmp</volumes>
                                        <workdir>/srv</workdir>
                                        <exposes>8080</exposes>
                                        <env>
                                                <JAVA_OPTS>-
server -Xms128m -Xmx256m</JAVA_OPTS>
```

```
                                        </env>
                                        <entryPoint>["sh", "-
c", "java ${JAVA_OPTS} -jar /srv/${project.build.finalName}.jar
${SPRING_OPTS}"]</entryPoint>
                                        <resources>
                                                <resource>

<targetPath>/srv</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>
                                        <!--
<image>${docker.image.prefix}/${project.artifactId}</image> -->
                                        <!--
<newName>${docker.image.prefix}/${project.artifactId}:${project
.version}</newName> -->
                                        <!-- <serverId>docker-
hub</serverId> -->

<registryUrl>http://${docker.registry}/v2/</registryUrl>
                                        <imageTags>
                                                <!--
<imageTag>${project.version}</imageTag> -->

<imageTag>latest</imageTag>
                                        </imageTags>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Springboot 启动类

注意：这里必须使用 @EnableDiscoveryClient 注解，不能使用 @EnableEurekaClient。

他们的区别是 @EnableEurekaClient 只能注册到 Eureka Server，而 @EnableDiscoveryClient 不仅可以注册进 Eureka Server 还能注册到 ZooKeeper，Consul，Kubernetes 等等.....

```
package cn.netkiller.provider;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryCli
ent;

@SpringBootApplication
@EnableDiscoveryClient
public class ProviderApplication {
        public static void main(String[] args) {
                System.out.println("Hello World!");

SpringApplication.run(ProviderApplication.class, args);
        }
}
```

控制器

```
package cn.netkiller.provider.controller;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.GetMapping;
```

```
import
org.springframework.web.bind.annotation.RestController;

import lombok.extern.slf4j.Slf4j;

@RestController
@Slf4j
public class ProviderController {

        @Autowired
        private DiscoveryClient discoveryClient;

        @GetMapping("/")
        public String index() {
                return "Hello world!!!";
        }

        @GetMapping("/ping")
        public String ping() {
                try {
                        return
InetAddress.getLocalHost().getHostName();
                } catch (UnknownHostException e) {
                        return "Pong";
                }
        }

        @GetMapping("/services")
        public List<String> services() {
                return this.discoveryClient.getServices();
        }

}
```

@GetMapping("/services") 可以返回注册中心已经注册的服务。

application.properties 配置文件

src/main/resource/application.properties 配置文件

```
spring.application.name=provider
server.port=8080
```

Pod 启动后会以 spring.application.name 设置的名字注册到注册中心，Openfeign 将改名字访问微服务。

Kubernetes provider 编排脚本

```
apiVersion: v1
kind: Service
metadata:
  name: provider
  labels:
    app.kubernetes.io/name: provider
spec:
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    app.kubernetes.io/name: provider

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: provider
  labels:
    app.kubernetes.io/name: provider
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: provider
  template:
```

```
    metadata:
      labels:
        app.kubernetes.io/name: provider
        app.kubernetes.io/instance: sad-markhor
    spec:
      containers:
        - name: provider
          image:
registry.netkiller.cn:5000/netkiller/provider
          #imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 8080
              protocol: TCP
          env:
            - name: "KUBERNETES_NAMESPACE"
              valueFrom:
                fieldRef:
                  fieldPath: "metadata.namespace"
```

consumer

Maven 依赖

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>kubernetes</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>consumer</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>consumer</name>
```

```
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
kubernetes</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
openfeign</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
kubernetes-ribbon</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
webflux</artifactId>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
```

```
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.2</version>
                                <configuration>

<imageName>${docker.registry}/${docker.registry.name}/${project
.artifactId}</imageName>

<baseImage>${docker.image}</baseImage>

<maintainer>netkiller@msn.com</maintainer>
                                        <volumes>/tmp</volumes>
                                        <workdir>/srv</workdir>
                                        <exposes>8080</exposes>
                                        <env>
                                                <JAVA_OPTS>-
server -Xms128m -Xmx256m</JAVA_OPTS>
                                        </env>
                                        <entryPoint>["sh", "-
c", "java ${JAVA_OPTS} -jar /srv/${project.build.finalName}.jar
${SPRING_OPTS}"]</entryPoint>
                                        <resources>
                                                <resource>

<targetPath>/srv</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>
                                        <!--
<image>${docker.image.prefix}/${project.artifactId}</image> -->
                                        <!--
<newName>${docker.image.prefix}/${project.artifactId}:${project
.version}</newName> -->
```

```
                                        <!-- <serverId>docker-
hub</serverId> -->

<registryUrl>http://${docker.registry}/v2/</registryUrl>
                                        <imageTags>
                                                <!--
<imageTag>${project.version}</imageTag> -->

<imageTag>latest</imageTag>
                                        </imageTags>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Springboot 启动类

```
package cn.netkiller.consumer;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryCli
ent;
import
org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class ConsumerApplication {
        public static void main(String[] args) {
                System.out.println("Hello World!");

SpringApplication.run(ConsumerApplication.class, args);
        }
```

```
}
```

控制器

```
package cn.netkiller.consumer.controller;

import java.util.List;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import
org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RestController;

import cn.netkiller.consumer.feign.ProviderClient;
import lombok.extern.slf4j.Slf4j;

@RestController
@Slf4j
public class ConsumerController {
        @Autowired
        private DiscoveryClient discoveryClient;

        @Autowired
        private ProviderClient providerClient;

        @GetMapping("/")
        public String index() {
                return "Consumer OK\r\n";
        }

        @GetMapping("/service")
        public Object getClient() {
                return discoveryClient.getServices();
        }
```

```
        @GetMapping("/instance")
        public List<ServiceInstance> getInstance(String
instanceId) {
                return
discoveryClient.getInstances(instanceId);
        }

        @GetMapping("/ping")
        public String ping() {
                return
OperationResponse.builder().status(true).data(providerClient.
ping()).build();
        }
}
```

@GetMapping("/ping") 将经过注册中心，获取到可用的服务，运行后从微服务返回结果。

```
package cn.netkiller.consumer.controller;

public class OperationResponse {

        public boolean status = false;
        public String data = "";

        public static OperationResponse builder() {
                // TODO Auto-generated method stub
                return new OperationResponse();
        }

        public OperationResponse status(boolean status) {
                this.status = status;
                return this;
        }

        public OperationResponse data(String data) {
                this.data = data;
                return this;
        }
```

```
        public String build() {
                return String.format("Status: %s, Data: %s",
this.status, this.data);
        }
}
```

FeignClient 接口

```
package cn.netkiller.consumer.feign;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.stereotype.Component;
import org.springframework.web.bind.annotation.GetMapping;

@FeignClient(name = "provider", fallback =
ProviderClientFallback.class)
public interface ProviderClient {
        @GetMapping("/ping")
        String ping();
}

@Component
class ProviderClientFallback implements ProviderClient {

        @Override
        public String ping() {
                return "Error";
        }
}
```

application.properties 配置文件

src/main/resource/application.properties 配置文件

```
spring.application.name=consumer
server.port=8080

provider.ribbon.KubernetesNamespace=default
provider.ribbon.NFLoadBalancerRuleClassName=com.netflix.loadbalancer.RandomRule
```

Kubernetes consumer 编排脚本

```
apiVersion: v1
kind: Service
metadata:
  name: consumer
  labels:
    app.kubernetes.io/name: consumer
spec:
  type: NodePort
  ports:
    - port: 8080
      nodePort: 30080
      protocol: TCP
      name: http
  selector:
    app.kubernetes.io/name: consumer

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: consumer
  labels:
    app.kubernetes.io/name: consumer
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
      app.kubernetes.io/name: consumer
  template:
    metadata:
      labels:
        app.kubernetes.io/name: consumer
    spec:
      containers:
        - name: consumer
          image:
registry.netkiller.cn:5000/netkiller/consumer
          #imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 8080
              protocol: TCP
```

测试

编译，打包，构建Docker镜像

```
iMac:provider neo$ mvn package docker:build
iMac:consumer neo$ mvn package docker:build
```

推送镜像

```
iMac:spring-cloud-kubernetes neo$ docker images | grep
registry.netkiller.cn
registry.netkiller.cn:5000/netkiller/consumer     latest
63921dc9b81d        27 seconds ago      238MB
registry.netkiller.cn:5000/netkiller/provider     latest
fbcc6b3a91ef        4 minutes ago       221MB
registry.netkiller.cn:5000/netkiller/configmaps   latest
93280aec434f        23 hours ago        219MB
```

```
iMac:spring-cloud-kubernetes neo$ docker push
registry.netkiller.cn:5000/netkiller/consumer
The push refers to repository
[registry.netkiller.cn:5000/netkiller/consumer]
51c839989a66: Pushed
7c1edc21f93f: Pushed
50644c29ef5a: Pushed
latest: digest:
sha256:2591686cfc63888f3b97ca1e950205b58a47b3d3c618242465baa079
6cf7e056 size: 952

iMac:spring-cloud-kubernetes neo$ docker push
registry.netkiller.cn:5000/netkiller/provider
The push refers to repository
[registry.netkiller.cn:5000/netkiller/provider]
47eb1c86b415: Pushed
7c1edc21f93f: Mounted from netkiller/consumer
50644c29ef5a: Mounted from netkiller/consumer
latest: digest:
sha256:42cdeb63cd67a5edf9e769538878a0c8f18048bcde1ef9ba22d58766
afa2d52d size: 952

iMac:spring-cloud-kubernetes neo$ curl -s
http://registry.netkiller.cn:5000/v2/_catalog | jq
{
  "repositories": [
    "netkiller/configmaps",
    "netkiller/consumer",
    "netkiller/provider"
  ]
}
```

将 provider 和 consumer 应用部署到 Kubernetes

```
iMac:spring-cloud-kubernetes neo$ kubectl create -f
provider/src/main/kubernetes/provider.yaml
service/provider created
deployment.apps/provider created
```

```
iMac:spring-cloud-kubernetes neo$ kubectl create -f
consumer/src/main/kubernetes/consumer.yaml
service/consumer created
deployment.apps/consumer created
```

查看 consumer 端口

```
iMac:spring-cloud-kubernetes neo$ minikube service list
|---------------------|---------------------------|-----------
---|---------------------------|
|      NAMESPACE      |           NAME            | TARGET
PORT  |            URL            |
|---------------------|---------------------------|-----------
---|---------------------------|
| default             | config                    | http/8080
| http://192.168.64.12:30662 |
| default             | consumer                  | http/8080
| http://192.168.64.12:30080 |
| default             | kubernetes                | No node
port |
| default             | provider                  | No node
port |
| kube-system         | kube-dns                  | No node
port |
| kubernetes-dashboard | dashboard-metrics-scraper | No node
port |
| kubernetes-dashboard | kubernetes-dashboard      | No node
port |
|---------------------|---------------------------|-----------
---|---------------------------|
```

测试 provider 是否工作正常

```
$ curl -s http://10.10.0.121:8080/ping
provider-54875bf44-4gf49
```

```
$ curl -s http://10.10.0.121:8080/services |jq
[
  "config",
  "kubernetes",
  "provider"
]
```

查看 consumer 是否已经注册成功

```
iMac:spring-cloud-kubernetes neo$ curl -s
http://192.168.64.12:30080/service |jq
[
  "config",
  "consumer",
  "kubernetes",
  "provider"
]

iMac:spring-cloud-kubernetes neo$ curl
http://192.168.64.12:30080/ping
Status: true, Data: provider-54875bf44-4gf49
```

增加 provider 节点，然后反复请求可以看到返回不同节点的主机名

```
iMac:spring-cloud-kubernetes neo$ kubectl scale deployment
provider --replicas=3
deployment.apps/provider scaled

iMac:spring-cloud-kubernetes neo$ curl -s
http://192.168.64.12:30080/ping
Status: true, Data: provider-54875bf44-8vs72

iMac:spring-cloud-kubernetes neo$ curl -s
http://192.168.64.12:30080/ping
```

```
Status: true, Data: provider-54875bf44-4gf49i
```

测试完毕销毁服务

```
iMac:spring-cloud-kubernetes neo$ kubectl delete -f
consumer/src/main/kubernetes/consumer.yaml
service "consumer" deleted
deployment.apps "consumer" deleted

iMac:spring-cloud-kubernetes neo$ kubectl delete -f
provider/src/main/kubernetes/provider.yaml
service "provider" deleted
deployment.apps "provider" deleted
```

```
spring.cloud.kubernetes.discovery.enabled=false
```

11. Spring Cloud Alibaba

11.1. 安装 Nacos

Docker 安装 Nacos

安装 netkiller-devops 库

```
pip install netkiller-devops
```

创建 docker.py 编排文件

```
#!/usr/bin/env python3
from netkiller.docker import *

volume = Volumes()
volume.create('mysql')

mysql = Services('mysql')
mysql.image('mysql:5.7').container_name('mysql').restart('alway
s').hostname('db.netkiller.cn').env_file(os.getcwd()+'/nacos/en
v/mysql.env')
mysql.ports(['3306:3306']).volumes([
        'mysql:/var/lib/mysql'
]).command([
        '--socket=/var/lib/mysql/mysql.sock',
        '--default-authentication-
plugin=mysql_native_password',
    '--character-set-server=utf8mb4',
    '--collation-server=utf8mb4_general_ci',
    '--explicit_defaults_for_timestamp=true',
    '--lower_case_table_names=1',
    '--max_execution_time=0'
])
```

```
nacos = Services('nacos')
nacos.container_name('nacos').env_file(os.getcwd()+'/nacos/env/
nacos-mysql.env')
# .environment([
#       'PREFER_HOST_MODE=hostname',
#       'MODE=standalone'
# ])
nacos.image('nacos/nacos-server').volumes([
        '../nacos/logs/:/home/nacos/logs',

'../nacos/init.d/custom.properties:/home/nacos/init.d/custom.pr
operties'
]).ports([
        "8848:8848",
    "9848:9848",
        '9555:9555'
]).depends_on('mysql').restart('on-failure')

experiment = Composes('experiment')
experiment.version('3.9')
experiment.volumes(volume)
experiment.services(mysql)
experiment.services(nacos)

if __name__ == '__main__':
        try:
                docker = Docker()
                docker.sysctl([{'vm.max_map_count':'262144'}])
                docker.environment(experiment)
                docker.main()
        except KeyboardInterrupt:
                print ("Crtl+C Pressed. Shutting down.")
```

查看帮助信息

```
[root@localhost ~]# python3 docker.py
Python controls the docker manager.
Usage: docker.py [options]
up|rm|start|stop|restart|logs|top|images|exec <service>
```

```
Options:
  -h, --help            show this help message and exit
  --debug               debug mode
  -e development|testing|production, --
environment=development|testing|production
                        environment
  -d, --daemon          run as daemon
  --logfile=LOGFILE     logs file.
  -l, --list            print service of environment
  -f, --follow          following logging
  -c, --compose         show docker compose
  --export              export docker compose

Homepage: http://www.netkiller.cn       Author: Neo
<netkiller@msn.com>
```

启动 nacos

```
[root@localhost ~]# python3 docker.py -e experiment up nacos
mysql is up-to-date
Starting nacos ... done

[root@localhost ~]# python3 docker.py -e experiment ps
    Name                   Command               State
Ports
------------------------------------------------------------------
------------------------------------------------------------------
-----------------------------------------
mysql           docker-entrypoint.sh --soc ...   Up
0.0.0.0:3306->3306/tcp,:::3306->3306/tcp, 33060/tcp
nacos           bin/docker-startup.sh            Up
0.0.0.0:8848->8848/tcp,:::8848->8848/tcp, 0.0.0.0:9555-
>9555/tcp,:::9555->9555/tcp,

0.0.0.0:9848->9848/tcp,:::9848->9848/tcp
```

查看启动端口

```
[root@localhost ~]# ss -lnt | grep -E "(8848|9848)"
LISTEN 0      1024         0.0.0.0:8848       0.0.0.0:*
LISTEN 0      1024         0.0.0.0:9848       0.0.0.0:*
LISTEN 0      1024            [::]:8848          [::]:*
LISTEN 0      1024            [::]:9848          [::]:*
```

测试配置中心

```
[root@localhost ~]# curl -X POST
"http://127.0.0.1:8848/nacos/v1/cs/configs?
dataId=nacos.cfg.dataId&group=test&content=helloWorld"
true

[root@localhost ~]# curl -X GET
"http://127.0.0.1:8848/nacos/v1/cs/configs?
dataId=nacos.cfg.dataId&group=test"
helloWorld
```

登陆 Web 界面 http://192.168.30.12:8848/nacos/ 默认的账号密码是：nacos/nacos

Kubernetes 安装 Nacos

创建 nacos 数据库用户

```
CREATE USER 'nacos'@'%' IDENTIFIED BY 'nacos';

GRANT ALL PRIVILEGES ON nacos.* TO 'nacos'@'%';

SHOW GRANTS FOR 'nacos'@'%';
```

前往 https://github.com/alibaba/nacos/blob/master/distribution/conf/nacos-mysql.sql 下来SQL文件，恢复到 nacos 数据中。

```
import sys
sys.path.insert(0, '/Users/neo/workspace/devops')
from netkiller.kubernetes import *
namespace = 'default'

config = ConfigMap('nacos')
config.apiVersion('v1')
config.metadata().name('nacos').namespace(namespace)
config.data({
    'mysql.host': "rm-
bp1g441na9an26wsb.mysql.rds.aliyuncs.com",
    'mysql.port': "3306",
    'mysql.dbname': "nacos",
    'mysql.user': "nacos",
    'mysql.password': "nacos"
})
# config.debug()

statefulSet = StatefulSet()
statefulSet = StatefulSet()
statefulSet.apiVersion('apps/v1')
statefulSet.metadata().name('nacos').labels(
    {'app': 'nacos'}).namespace(namespace)
statefulSet.spec().replicas(3)
statefulSet.spec().serviceName('nacos')
statefulSet.spec().selector({'matchLabels': {'app':
'nacos'}})
statefulSet.spec().template().metadata().labels({'app':
'nacos'})
statefulSet.spec().template().metadata().annotations(
    {'pod.alpha.kubernetes.io/initialized': "true"})
#
statefulSet.spec().template().spec().affinity().nodeAffinity(
{
```

```
#     'requiredDuringSchedulingIgnoredDuringExecution': [
#         {'labelSelector': {
#             'matchExpressions': [
#                 {'key': 'app',
#                  'operator': 'In',
#                  'values': ['nacos']
#                  }]
#             },
#         'topologyKey': "kubernetes.io/hostname"
#         }
#     ]
# })
statefulSet.spec().template().spec().containers().name('nacos
').imagePullPolicy(Define.containers.imagePullPolicy.IfNotPre
sent).image(
    'nacos/nacos-server:latest').resources(
        # {'requests': {
        # # 'cpu':'200m',
        # 'memory': "2Gi"}}
        ).ports([
        {'name':'client','containerPort': 8848},
        {'name':'client-rpc','containerPort': 9848},
        {'name':'raft-rpc','containerPort': 9849}
    ]).env([
        {'name': 'TZ', 'value': 'Asia/Shanghai'},
        {'name': 'LANG', 'value': 'en_US.UTF-8'},
        {'name': 'NACOS_REPLICAS', 'value': '1'},

        # {'name': 'SPRING_DATASOURCE_PLATFORM', 'value':
'mysql'},
        # {'name': 'MYSQL_SERVICE_HOST', 'value': 'mysql-
0.mysql.default.svc.cluster.local'},
        {'name': 'MYSQL_SERVICE_HOST', 'valueFrom':
{'configMapKeyRef':{'name': 'nacos','key': 'mysql.host'}}},
        {'name': 'MYSQL_SERVICE_PORT', 'valueFrom':
{'configMapKeyRef':{'name': 'nacos','key': 'mysql.port'}}},
        {'name': 'MYSQL_SERVICE_DB_NAME', 'valueFrom':
{'configMapKeyRef':{'name': 'nacos','key': 'mysql.dbname'}}},
        {'name': 'MYSQL_SERVICE_USER', 'valueFrom':
{'configMapKeyRef':{'name': 'nacos','key': 'mysql.user'}}},
        {'name': 'MYSQL_SERVICE_PASSWORD', 'valueFrom':
{'configMapKeyRef':{'name': 'nacos','key':
'mysql.password'}}},
        # {'name': 'MYSQL_SERVICE_DB_PARAM', 'value':
'characterEncoding=utf8&connectTimeout=1000&socketTimeout=300
```

```
0&autoReconnect=true&useSSL=false&serverTimezone=GMT%2B8'},

        {'name': 'NACOS_SERVER_PORT', 'value': '8848'},
        {'name': 'NACOS_APPLICATION_PORT', 'value': '8848'},
        {'name': 'PREFER_HOST_MODE', 'value': 'hostname'},
        {'name': 'NACOS_SERVERS', 'value': 'nacos-
0.nacos.default.svc.cluster.local:8848 nacos-
1.nacos.default.svc.cluster.local:8848 nacos-
2.nacos.default.svc.cluster.local:8848'},

        # {'name': 'JVM_XMX', 'value': '4g'},
        # {'name': 'NACOS_DEBUG', 'value': 'true'},
        # {'name': 'TOMCAT_ACCESSLOG_ENABLED', 'value':
'true'},
    ])

# statefulSet.debug()

service = Service()
service.metadata().name('nacos')
service.metadata().namespace(namespace)
service.metadata().labels({'app':'nacos'})
service.spec().selector({'app': 'nacos'})
service.spec().type('ClusterIP')
service.spec().ports([
    {'name': 'server', 'protocol': 'TCP', 'port': 8848,
'targetPort': 8848},
    {'name': 'client-rpc', 'protocol': 'TCP', 'port': 9848,
'targetPort': 9848},
    {'name': 'raft-rpc', 'protocol': 'TCP', 'port': 9555,
'targetPort': 9555}
])

# service.debug()

ingress = Ingress()
ingress.apiVersion('networking.k8s.io/v1')
ingress.metadata().name('nacos')
ingress.metadata().namespace(namespace)
#
ingress.metadata().annotations({'kubernetes.io/ingress.class'
: 'nginx'})
ingress.spec().rules([{
    'host': 'nacos.netkiller.com',
    'http':{
```

```
        'paths': [{
            'pathType': Define.Ingress.pathType.Prefix,
            'path': '/nacos',
            'backend':{
                'service':{
                    'name':'nacos',
                    'port':{'number': 8848}
                }
            }}]
}}])
# ingress.debug()

kubernetes = Kubernetes('/Volumes/Data/kubeconfig')
compose = Compose('nacos')
compose.add(config)
compose.add(statefulSet)
compose.add(service)
compose.add(ingress)
kubernetes.compose(compose)
kubernetes.main()
```

```
```

IP限制，白名单

```
        location /nacos {
        allow 192.168.0.0/24;
        allow 172.18.0.0/16;
        allow 202.104.66.10;
        deny all;

        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
        proxy_pass http://192.168.0.10:8848;
    }
```

防火墙配置

配置防火墙，限制 8848 端口的访问策略，防止本地或其他服务注册到 Nacos 中。

```
$ iptables -A INPUT -s 172.18.5.0/24 -p tcp --dport 8848 -j
REJECT
```

172.18.5.0/24 是办公网络，添加上面IP规则，可以防止开发人的电脑注册到测试环境。

删除规则

删除方法一

```
$ iptables -L -n --line-number | grep 8848
8    REJECT     tcp  --  172.18.5.0/24        0.0.0.0/0
tcp dpt:8848 reject-with icmp-port-unreachable
134  ACCEPT     tcp  --  0.0.0.0/0            172.17.0.119
tcp dpt:8848

$ iptables -D INPUT 8
```

删除方法二

```
$ iptables -D INPUT -s 172.18.5.0/24 -p tcp --dport 8848 -j
```

```
REJECT
```

11.2. Kubernetes 部署微服务

pom.xml 中加入 docker 插件

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.example</groupId>
        <artifactId>demo</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>demo</name>
        <description>Demo project for Spring
Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.6.3</version>
                <relativePath /> <!-- lookup parent from
repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
```

```
<sonar.projectKey>netkiller.cn_java_AX0HsoVkT19KeT2iVgUT</son
ar.projectKey>

<sonar.qualitygate.wait>true</sonar.qualitygate.wait>


<docker.registry>registry.netkiller.cn/netkiller.cn</docker.r
egistry>

        </properties>

        <repositories>
                <repository>
                        <id>gitlab-maven</id>

<url>${env.CI_API_V4_URL}/projects/${env.CI_PROJECT_ID}/packa
ges/maven</url>
                </repository>
        </repositories>
        <distributionManagement>
                <repository>
                        <id>gitlab-maven</id>

<url>${CI_API_V4_URL}/projects/${env.CI_PROJECT_ID}/packages/
maven</url>
                </repository>
                <snapshotRepository>
                        <id>gitlab-maven</id>

<url>${CI_API_V4_URL}/projects/${env.CI_PROJECT_ID}/packages/
maven</url>
                </snapshotRepository>
        </distributionManagement>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>

                <dependency>
```

```
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <!-- <version>4.13.2</version> -->
                        <scope>test</scope>
                </dependency>

        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-
maven-plugin</artifactId>
                        </plugin>
                        <plugin>

<groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.2</version>
                                <configuration>

<imageName>${docker.registry}/${project.artifactId}
</imageName>
                                        <baseImage>openjdk:8-
alpine</baseImage>

<maintainer>netkiller@msn.com</maintainer>

<volumes>/srv</volumes>

<workdir>/srv</workdir>
                                        <env>
                                                <JAVA_OPTS>-
server -Xms512m -Xmx4096m -
Djava.security.egd=file:/dev/./urandom</JAVA_OPTS>
```

```
                                        </env>

<exposes>8080</exposes>
                                        <entryPoint>["sh", "-
c", "/srv/docker-entrypoint.sh"]</entryPoint>
                                        <resources>
                                                <resource>

<targetPath>/srv</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                                <resource>

<targetPath>/srv</targetPath>

<directory>.</directory>

<include>docker-entrypoint.sh</include>
                                                </resource>
                                        </resources>

<registryUrl>http://${docker.registry}/v2/</registryUrl>
                                        <imageTags>

<imageTag>${project.version}</imageTag>

<imageTag>latest</imageTag>
                                        </imageTags>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

容器启动脚本

在项目目录创建 docker-entrypoint.sh 文件

```
#!/bin/sh

if [ ! -z $1 ]; then
    MODULE=$1
    shift
fi

if [ -z $JAVA_OPTS ]; then
    JAVA_OPTS='-Xms1024m -Xmx4096m -XX:MetaspaceSize=128m -
XX:MaxMetaspaceSize=512m -
Djava.security.egd=file:/dev/./urandom -Duser.timezone=GMT+8
-Dfile.encoding=utf-8'
fi

if [ -z $MODULE ]; then
    echo "MODULE environment is not set"
    exit 127
else
    PACKAGE=/srv/$MODULE.jar
fi

DEBUG='-Xdebug -
Xrunjdwp:transport=dt_socket,suspend=n,server=y,address=5555'
SKYWALKING="-javaagent:/srv/skywalking/agent/skywalking-
agent.jar -
Dskywalking.collector.backend_service=oap.netkiller.cn:11800
-Dskywalking.agent.service_name=${MODULE}"

exec java ${JAVA_OPTS}  -jar ${PACKAGE} $@
```

暂时 DEBUG，SKYWALKING 没有使用，放在一遍不碍事。脚本的用法

```
./docker-entrypoint.sh your_module --server.port=8080
```

构建 **docker** 镜像

运行 mvn 命令构建 docker 镜像

```
neo@Netkiller-iMac ~/w/java.netkiller.cn (master)> mvn package
docker:build docker:push
```

不出预料，你会看到下面输出

```
[INFO] Building image registry.netkiller.cn/netkiller.cn/demo
Step 1/9 : FROM openjdk:8-alpine

 ---> a3562aa0b991
Step 2/9 : MAINTAINER netkiller@msn.com

 ---> Using cache
 ---> b4a79be602ae
Step 3/9 : ENV JAVA_OPTS -server -Xms512m -Xmx4096m -
Djava.security.egd=file:/dev/./urandom

 ---> Using cache
 ---> 9d685ea4a0d3
Step 4/9 : WORKDIR /srv

 ---> Using cache
 ---> e2feea451bb1
Step 5/9 : ADD /srv/demo-0.0.1-SNAPSHOT.jar /srv/

 ---> 7ad53fb991b8
Step 6/9 : ADD /srv/docker-entrypoint.sh /srv/

 ---> 39def6507064
Step 7/9 : EXPOSE 8080

 ---> Running in 338a99e6ec36
Removing intermediate container 338a99e6ec36
 ---> f192b73ab3b9
```

```
Step 8/9 : ENTRYPOINT ["sh", "-c", "/srv/docker-entrypoint.sh"]

 ---> Running in 5bda82acd305
Removing intermediate container 5bda82acd305
 ---> 85c1b2615a97
Step 9/9 : VOLUME /srv

 ---> Running in 27d71c55bf7e
Removing intermediate container 27d71c55bf7e
 ---> 64e0d8992fdd
ProgressMessage{id=null, status=null, stream=null, error=null,
progress=null, progressDetail=null}
Successfully built 64e0d8992fdd
Successfully tagged
registry.netkiller.cn/netkiller.cn/demo:latest
[INFO] Built registry.netkiller.cn/netkiller.cn/demo
[INFO] Tagging registry.netkiller.cn/netkiller.cn/demo with
0.0.1-SNAPSHOT
[INFO] Tagging registry.netkiller.cn/netkiller.cn/demo with
latest
```

查看镜像

```
neo@Netkiller-iMac ~/w/java.netkiller.cn (master)> docker image
ls | grep netkiller
registry.netkiller.cn/netkiller.cn/demo
0.0.1-SNAPSHOT   64e0d8992fdd   3 minutes ago    122MB
registry.netkiller.cn/netkiller.cn/demo
latest           64e0d8992fdd   3 minutes ago    122MB
```

编排 kubernetes 容器

```
from netkiller.kubernetes import *
namespace = 'default'
```

```
compose = Compose('development')

module = 'demo'
# version = '0.0.1-SNAPSHOT'
version = 'latest'

deployment = Deployment()
deployment.apiVersion('apps/v1')

deployment.metadata().name(module).labels({'app':
module}).namespace(namespace)
deployment.spec().replicas(1)
deployment.spec().selector({'matchLabels': {'app': module}})
deployment.spec().template().metadata().labels({'app':
module})
deployment.spec().template().spec().containers().name(module)
.image(

'registry.netkiller.cn/netkiller.cn/cloud.netkiller.cn:%s' %
version).ports([{
        'containerPort': 8080
    }]).env([
        {'name': 'TZ', 'value': 'Asia/Shanghai'},
        {'name': 'LANG', 'value': 'en_US.UTF-8'},
    ]).args([module,'--server.port=8080'])

# deployment.debug()
# deployment.json()

service = Service()
service.metadata().name(module)
service.metadata().namespace(namespace)
service.spec().selector({'app': module})
service.spec().type('NodePort')
service.spec().ports([{
    'name': 'http',
    'protocol': 'TCP',
    'port': 8080,
    'targetPort': 8080
}])

compose.add(deployment)
compose.add(service)

print("=" * 40, "Compose", "=" * 40)
```

```
compose.debug()
compose.delete()
compose.create()⏎
```

查看容器运行状态

```
neo@Netkiller-iMac ~/w/java.netkiller.cn (master)> kubectl get
pods
NAME                        READY   STATUS             RESTARTS
AGE
nginx-88c84c4d8-8pmzp       1/1     Running            1
3d20h
demo-76b7598b76-5hstp       1/1     Running            0
5h43m
busybox                     0/1     CrashLoopBackOff   52
4h44m
```

启动指定 nacos

容器中不方便修改配置文件，我们可以使用环境变量覆盖配置

```
JAVA_OPTS=-Dspring.cloud.nacos.username=nacos \
-Dspring.cloud.nacos.password=nacos \
-Dspring.cloud.nacos.config.server-addr=mse-032dbef0-nacos-
ans.mse.aliyuncs.com:8848 \
-Dspring.cloud.nacos.discovery.server-addr=mse-032dbef0-nacos-
ans.mse.aliyuncs.com:8848 \
```

相当于

```
```

```
java -Dspring.cloud.nacos.username=nacos \
-Dspring.cloud.nacos.password=nacos \
-Dspring.cloud.nacos.config.server-addr=mse-032dbef0-nacos-
ans.mse.aliyuncs.com:8848 \
-Dspring.cloud.nacos.discovery.server-addr=mse-032dbef0-nacos-
ans.mse.aliyuncs.com:8848 \
-jar netkiller.jar
```

11.3. Nacos 配置中心/注册中心代码实例

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>cn.netkiller</groupId>
        <artifactId>bottleneck</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>bottleneck</name>
        <description>bottleneck</description>
        <organization>
                <name>Netkiller Spring Cloud 手札</name>
                <url>https://www.netkiller.cn</url>
        </organization>
        <developers>
                <developer>
                        <name>Neo</name>
                        <email>netkiller@msn.com</email>
                        <organization>Netkiller Spring Cloud 手
札</organization>

<organizationUrl>https://www.netkiller.cn</organizationUrl>
                        <roles>
                                <role>Author</role>
                        </roles>
                </developer>
        </developers>
```

```
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>2.7.3</version>
                <relativePath />
        </parent>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                        <!-- Exclude the Tomcat dependency -->
                        <exclusions>
                                <exclusion>

<groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-
boot-starter-tomcat</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <!-- Use Undertow instead Tomcat -->
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
undertow</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
webflux</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
jdbc</artifactId>
```

```
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis</artifactId>
                        <exclusions>
                                <exclusion>

<groupId>io.lettuce</groupId>
                                        <artifactId>lettuce-
core</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
                <dependency>
                        <groupId>redis.clients</groupId>
                        <artifactId>jedis</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-
commons</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.projectlombok</groupId>
```

```
                        <artifactId>lombok</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
elasticsearch</artifactId>
                </dependency>
                <dependency>
                        <groupId>com.alibaba.cloud</groupId>
                        <artifactId>spring-cloud-starter-
alibaba-nacos-config</artifactId>
                        <version>2.2.9.RELEASE</version>
                </dependency>
                <dependency>
                        <groupId>com.alibaba.cloud</groupId>
                        <artifactId>spring-cloud-starter-
alibaba-nacos-discovery</artifactId>
                        <version>2.2.9.RELEASE</version>
                </dependency>

                <!--
https://mvnrepository.com/artifact/org.fluentd/fluent-logger --
>
                <dependency>
                        <groupId>org.fluentd</groupId>
                        <artifactId>fluent-logger</artifactId>
                        <version>0.3.4</version>
                </dependency>
                <dependency>
                        <groupId>com.sndyuk</groupId>
                        <artifactId>logback-more-
appenders</artifactId>
                        <version>1.8.7</version>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-
context</artifactId>
                        <version>3.1.4</version>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
```

```
                        <artifactId>spring-cloud-
commons</artifactId>
                        <version>3.1.4</version>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
loadbalancer</artifactId>
                        <version>3.1.4</version>
                </dependency>

                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
bootstrap</artifactId>
                        <version>3.1.4</version>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                                <configuration>

<mainClass>cn.netkiller.Application</mainClass>
                                </configuration>
                        </plugin>
                        <plugin>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>

                        <plugin>
                                <groupId>com.spotify</groupId>
                                <artifactId>docker-maven-
plugin</artifactId>
                                <version>1.2.2</version>
```

```
                                <configuration>

<imageName>netkiller/${project.artifactId}</imageName>
                                        <baseImage>openjdk:19-
alpine</baseImage>

<maintainer>netkiller@msn.com</maintainer>
                                        <volumes>/tmp</volumes>
                                        <workdir>/srv</workdir>
                                        <exposes>8080</exposes>
                                        <env>
                                                <JAVA_OPTS>-
server -Xms128m -Xmx2048m</JAVA_OPTS>
                                        </env>
                                        <entryPoint>["sh", "-
c", "java ${JAVA_OPTS} -jar /srv/${project.build.finalName}.jar
${SPRING_OPTS}"]</entryPoint>
                                        <resources>
                                                <resource>

<targetPath>/srv</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>
                                                </resource>
                                        </resources>

<image>netkiller/${project.artifactId}</image>

<newName>netkiller/${project.artifactId}:${project.version}
</newName>
                                        <!-- <serverId>docker-
hub</serverId> -->

<registryUrl>http://${docker.registry}/v2/</registryUrl>
                                        <imageTags>

<imageTag>undertow</imageTag>
                                                <!--
<imageTag>tomcat</imageTag> <imageTag>${project.version}
</imageTag> <imageTag>latest</imageTag> -->
                                        </imageTags>
                                </configuration>
                        </plugin>
```

```
                </plugins>
        </build>
</project>
```

SpringBootApplication

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryClien
t;

@EnableDiscoveryClient
@SpringBootApplication
public class Application {

        public static void main(String[] args) {
                System.out.println("Netkiller bottleneck
tool!");
                SpringApplication.run(Application.class, args);
        }
}
```

ConfigController

```
package cn.netkiller.controller;

import org.springframework.beans.factory.annotation.Value;
import
org.springframework.cloud.context.config.annotation.RefreshScop
e;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RefreshScope
@RestController
public class ConfigController {

	public ConfigController() {
		// TODO Auto-generated constructor stub
	}

	@Value("${key}")
	public String name;

	@GetMapping("/config")
	public String config() {

		String name = this.name;
		return name;
	}
}
```

配置文件

```
#debug=true
server.port=8080

#server.tomcat.max-connections=10000
#server.tomcat.max-threads=4096
#server.tomcat.accept-count=1000
#server.tomcat.min-spare-threads=100

server.undertow.max-http-post-size=0
server.undertow.io-threads=16
server.undertow.worker-threads=4096
server.undertow.buffer-size=1024
server.undertow.buffers-per-region=1024
server.undertow.direct-buffers=true
```

```
spring.application.name=bottleneck
spring.profiles.active=dev
#spring.profiles.active=test
##spring.profiles.active=prod
#
#logging.file.path=/tmp
##logging.file.name=spring.log
#
endpoints.metrics.enabled=true
management.endpoints.jmx.exposure.include=*
management.endpoints.web.exposure.include=*
management.endpoints.health.show-details=always
#
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://172.18.200.5:3306/test?
useUnicode=true&characterEncoding=UTF-8
spring.datasource.username=root
spring.datasource.password=passw0rd

spring.datasource.type=com.zaxxer.hikari.HikariDataSource
spring.datasource.hikari.minimum-idle=5
spring.datasource.hikari.maximum-pool-size=200
spring.datasource.hikari.auto-commit=true
spring.datasource.hikari.idle-timeout=30000
spring.datasource.hikari.pool-name=Hikari
spring.datasource.hikari.max-lifetime=55000
spring.datasource.hikari.connection-timeout=30000
spring.datasource.hikari.connection-test-query=SELECT 1

spring.redis.host=172.18.200.5
spring.redis.port=6379
spring.redis.password=passw0rd
spring.redis.database=0

spring.redis.jedis.pool.max-active=1000
spring.redis.jedis.pool.max-idle=80
spring.redis.jedis.pool.min-idle=20
spring.redis.jedis.pool.max-wait=-1

spring.cloud.nacos.server-addr=nacos.netkiller.cn:8848
spring.cloud.nacos.username=neo
spring.cloud.nacos.password=netkiller

#spring.cloud.nacos.config.enable-remote-sync-config=true
spring.cloud.nacos.config.namespace=${spring.profiles.active}
```

```
spring.cloud.nacos.config.group=DEFAULT_GROUP
spring.cloud.nacos.config.prefix=${spring.application.name}
spring.cloud.nacos.config.file-extension=yaml
spring.cloud.nacos.discovery.service=${spring.application.name:
DEFAULT-SERVICE-NAME}
spring.cloud.nacos.discovery.namespace=${spring.profiles.active
}
```

11.4. FAQ

禁用 Nacos

当 Maven 引入了 nacos 依赖，启动就会要求配置 Nacos，可以通过下面方法禁用 Nacos

```
spring.cloud.nacos.config.enabled=false
spring.cloud.nacos.discovery.enabled=false
spring.cloud.nacos.config.refresh-enabled=false
spring.cloud.nacos.discovery.instance-enabled=false
```

禁止注册

有时我们希望只获取配置，并不希望服务注册

register-enabled: false

```
spring:
  profiles:
    active: dev
  application:
    name: netkiller
  main:
    allow-bean-definition-overriding: true
```

```
  cloud:
    nacos:
      username: dev
      password: passw0rd
      config:
        server-addr:
http://${spring.profiles.active}.netkiller.cn:8848
        file-extension: yaml
        enabled: true
        namespace: ${spring.profiles.active}
        enable-remote-sync-config: true
      discovery:
        server-addr:
http://${spring.profiles.active}.netkiller.cn:8848
        namespace: ${spring.profiles.active}
        register-enabled: true
```

Failed to bind properties under 'server.tomcat.basedir' to java.io.File:

环境 Docker 安装 nacos 提示 server.tomcat.basedir 错误

```
nacos   | ***************************
nacos   | APPLICATION FAILED TO START
nacos   | ***************************
nacos   |
nacos   | Description:
nacos   |
nacos   | Failed to bind properties under
'server.tomcat.basedir' to java.io.File:
nacos   |
nacos   |     Property: server.tomcat.basedir
nacos   |     Value:
nacos   |     Origin: InputStream resource [resource loaded
through InputStream] - 34:0
nacos   |     Reason: failed to convert java.lang.String to
java.io.File (caused by java.lang.IllegalStateException: Could
not retrieve file for class path resource []: class path
resource [] cannot be resolved to absolute file path because it
does not reside in the file system:
```

```
jar:file:/home/nacos/target/nacos-server.jar!/BOOT-
INF/classes!/)
nacos   |
nacos   | Action:
nacos   |
nacos   | Update your application's configuration
```

解决方案

进入容器复制 application.properties 文件到本地，修改 server.tomcat.basedir= 配置，改为 server.tomcat.basedir=. 然后在挂载到容器卷中。

```
[root@cloud ops.sfzito.com]# docker run -it --entrypoint sh
nacos/nacos-server
sh-4.2# cat /home/nacos/conf/application.properties
# spring
server.servlet.contextPath=${SERVER_SERVLET_CONTEXTPATH:/nacos}
server.contextPath=/nacos
server.port=${NACOS_APPLICATION_PORT:8848}
spring.datasource.platform=${SPRING_DATASOURCE_PLATFORM:""}
nacos.cmdb.dumpTaskInterval=3600
nacos.cmdb.eventTaskInterval=10
nacos.cmdb.labelTaskInterval=300
nacos.cmdb.loadDataAtStart=false
db.num=${MYSQL_DATABASE_NUM:1}
db.url.0=jdbc:mysql://${MYSQL_SERVICE_HOST}:${MYSQL_SERVICE_POR
T:3306}/${MYSQL_SERVICE_DB_NAME}?
${MYSQL_SERVICE_DB_PARAM:characterEncoding=utf8&connectTimeout=
1000&socketTimeout=3000&autoReconnect=true&useSSL=false}
db.url.1=jdbc:mysql://${MYSQL_SERVICE_HOST}:${MYSQL_SERVICE_POR
T:3306}/${MYSQL_SERVICE_DB_NAME}?
${MYSQL_SERVICE_DB_PARAM:characterEncoding=utf8&connectTimeout=
1000&socketTimeout=3000&autoReconnect=true&useSSL=false}
db.user=${MYSQL_SERVICE_USER}
db.password=${MYSQL_SERVICE_PASSWORD}

### The auth system to use, currently only 'nacos' and 'ldap'
is supported:
```

```
nacos.core.auth.system.type=${NACOS_AUTH_SYSTEM_TYPE:nacos}

### worked when nacos.core.auth.system.type=nacos
### The token expiration in seconds:
nacos.core.auth.plugin.nacos.token.expire.seconds=${NACOS_AUTH_
TOKEN_EXPIRE_SECONDS:18000}
### The default token:
nacos.core.auth.plugin.nacos.token.secret.key=${NACOS_AUTH_TOKE
N:SecretKey012345678901234567890123456789012345678901
23456789}

### Turn on/off caching of auth information. By turning on this
switch, the update of auth information would have a 15 seconds
delay.
nacos.core.auth.caching.enabled=${NACOS_AUTH_CACHE_ENABLE:false
}
nacos.core.auth.enable.userAgentAuthWhite=${NACOS_AUTH_USER_AGE
NT_AUTH_WHITE_ENABLE:false}
nacos.core.auth.server.identity.key=${NACOS_AUTH_IDENTITY_KEY:s
erverIdentity}
nacos.core.auth.server.identity.value=${NACOS_AUTH_IDENTITY_VAL
UE:security}
server.tomcat.accesslog.enabled=${TOMCAT_ACCESSLOG_ENABLED:fals
e}
server.tomcat.accesslog.pattern=%h %l %u %t "%r" %s %b %D
# default current work dir
server.tomcat.basedir=
## spring security config
### turn off security
nacos.security.ignore.urls=${NACOS_SECURITY_IGNORE_URLS:/,/erro
r,/**/*.css,/**/*.js,/**/*.html,/**/*.map,/**/*.svg,/**/*.png,/
**/*.ico,/console-
fe/public/**,/v1/auth/**,/v1/console/health/**,/actuator/**,/v1
/console/server/**}
# metrics for elastic search
management.metrics.export.elastic.enabled=false
management.metrics.export.influx.enabled=false

nacos.naming.distro.taskDispatchThreadCount=10
nacos.naming.distro.taskDispatchPeriod=200
nacos.naming.distro.batchSyncKeyCount=1000
nacos.naming.distro.initDataRatio=0.9
nacos.naming.distro.syncRetryDelay=5000
nacos.naming.data.warmup=true
```

```

```

不读取 bootstrap.yaml 文件

Nacos 是一个独立项目，并不依赖 Spring Cloud，如果只是想使用配置中心，在 Springboot 中引入 Nacos 依赖，你会发现 Springboot 不读取 bootstrap.yaml。

解决方法

```
                <dependency>
<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
bootstrap</artifactId>
                        <version>3.1.4</version>
                </dependency>
```

WARN [com.alibaba.nacos.client.naming:177] [,] - out of date data received, old-t: 1665711914993, new-t: 1665711902390

出错信息如下，故障是因为每个节点的时间不同步造成的。

```
[2022-10-14 09:44:51.289
[com.alibaba.nacos.naming.push.receiver] WARN
[com.alibaba.nacos.client.naming:177] [,] - out of date data
received, old-t: 1665711914993, new-t: 1665711902390
```

解决方法，安装时间同步软件。例如 NTP

User limit of inotify instances reached or too many open files

```
nacos  | 09:22:23.431 [main] ERROR
org.springframework.boot.SpringApplication - Application run
failed
nacos  |
com.alibaba.nacos.api.exception.runtime.NacosRuntimeException:
ErrCode:500, ErrMsg:User limit of inotify instances reached or
too many open files
nacos  |        at
com.alibaba.nacos.core.listener.StartingApplicationListener.loa
dPreProperties(StartingApplicationListener.java:161)
nacos  |        at
com.alibaba.nacos.core.listener.StartingApplicationListener.env
ironmentPrepared(StartingApplicationListener.java:100)
nacos  |        at
com.alibaba.nacos.core.code.SpringApplicationRunListener.enviro
nmentPrepared(SpringApplicationRunListener.java:60)
nacos  |        at
org.springframework.boot.SpringApplicationRunListeners.lambda$e
nvironmentPrepared$2(SpringApplicationRunListeners.java:66)
nacos  |        at
java.util.ArrayList.forEach(ArrayList.java:1259)
nacos  |        at
org.springframework.boot.SpringApplicationRunListeners.doWithLi
steners(SpringApplicationRunListeners.java:120)
nacos  |        at
org.springframework.boot.SpringApplicationRunListeners.doWithLi
steners(SpringApplicationRunListeners.java:114)
nacos  |        at
org.springframework.boot.SpringApplicationRunListeners.environm
entPrepared(SpringApplicationRunListeners.java:65)
nacos  |        at
org.springframework.boot.SpringApplication.prepareEnvironment(S
pringApplication.java:343)
nacos  |        at
org.springframework.boot.SpringApplication.run(SpringApplicatio
n.java:301)
nacos  |        at
org.springframework.boot.SpringApplication.run(SpringApplicatio
n.java:1317)
nacos  |        at
org.springframework.boot.SpringApplication.run(SpringApplicatio
n.java:1306)
```

```
nacos  |        at com.alibaba.nacos.Nacos.main(Nacos.java:35)
nacos  |        at
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
nacos  |        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccesso
rImpl.java:62)
nacos  |        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMetho
dAccessorImpl.java:43)
nacos  |        at
java.lang.reflect.Method.invoke(Method.java:498)
nacos  |        at
org.springframework.boot.loader.MainMethodRunner.run(MainMethod
Runner.java:49)
nacos  |        at
org.springframework.boot.loader.Launcher.launch(Launcher.java:1
08)
nacos  |        at
org.springframework.boot.loader.Launcher.launch(Launcher.java:5
8)
nacos  |        at
org.springframework.boot.loader.PropertiesLauncher.main(Propert
iesLauncher.java:467)
nacos  | Caused by:
com.alibaba.nacos.api.exception.NacosException:
java.io.IOException: User limit of inotify instances reached or
too many open files
nacos  |        at
com.alibaba.nacos.sys.file.WatchFileCenter$WatchDirJob.<init>
(WatchFileCenter.java:184)
nacos  |        at
com.alibaba.nacos.sys.file.WatchFileCenter.registerWatcher(Watc
hFileCenter.java:92)
nacos  |        at
com.alibaba.nacos.core.listener.StartingApplicationListener.reg
isterWatcher(StartingApplicationListener.java:167)
nacos  |        at
com.alibaba.nacos.core.listener.StartingApplicationListener.loa
dPreProperties(StartingApplicationListener.java:159)
nacos  |        ... 20 common frames omitted
nacos  | Caused by: java.io.IOException: User limit of inotify
instances reached or too many open files
nacos  |        at sun.nio.fs.LinuxWatchService.<init>
(LinuxWatchService.java:64)
nacos  |        at
```

```
sun.nio.fs.LinuxFileSystem.newWatchService(LinuxFileSystem.java
:47)
nacos  |        at
com.alibaba.nacos.sys.file.WatchFileCenter$WatchDirJob.<init>
(WatchFileCenter.java:179)
nacos  |        ... 23 common frames omitted
nacos  | 09:22:23.435 [Thread-4] WARN
com.alibaba.nacos.common.executor.ThreadPoolManager -
[ThreadPoolManager] Start destroying ThreadPool
nacos  | 09:22:23.435 [Thread-4] WARN
com.alibaba.nacos.common.executor.ThreadPoolManager -
[ThreadPoolManager] Destruction of the end
```

加大 fs.inotify.max_user_instances 配置即可，默认是 128

```
sysctl fs.inotify.max_user_watches
sudo sysctl -w fs.inotify.max_user_watches=50889300

sysctl fs.inotify.max_user_instances
sudo sysctl -w fs.inotify.max_user_instances=65535
```

开启权限

```
nacos.core.auth.enabled= true
```

在容器中设置开启验证：NACOS_AUTH_ENABLE=true

ERROR Whitelabel

当Nacos开启了认证配置nacos.core.auth.enabled=true时，当前账号没有该命名空间的权限，就会出现下面的错误提示。

```
2022-11-07 18:42:04,370 [,,,] INFO
com.alibaba.nacos.client.config.impl.LocalConfigInfoProcessor:2
12 [main] - LOCAL_SNAPSHOT_PATH:/root/nacos/config
Mon, Nov 7 2022 6:42:04 pm      2022-11-07 18:42:04,393 [,,,]
ERROR com.alibaba.nacos.client.config.impl.ClientWorker:304
[main] - [fixed-nacos.default.svc.cluster.local_8848-test]
[sub-server-error] no right, dataId=netkiller,
group=DEFAULT_GROUP, tenant=test
Mon, Nov 7 2022 6:42:04 pm      2022-11-07 18:42:04,393 [,,,]
ERROR
com.alibaba.cloud.nacos.client.NacosPropertySourceBuilder:104
[main] - get data from Nacos error,dataId:netkiller
Mon, Nov 7 2022 6:42:04 pm
com.alibaba.nacos.api.exception.NacosException: <html><body>
<h1>Whitelabel Error Page</h1><p>This application has no
explicit mapping for /error, so you are seeing this as a
fallback.</p><div id='created'>Mon Nov 07 18:42:04 CST
2022</div><div>There was an unexpected error (type=Forbidden,
status=403).</div></body></html>
```

解决方案，在权限控制->权限管理中「添加权限」可以解决。

```
spring:
  cloud:
    nacos:
      discovery:
        server-addr: 127.0.0.1:8848
        metadata:
          preserved.heart.beat.interval: 1000   # 客户端上报心跳的
间隔时间。(单位:毫秒)
          preserved.heart.beat.timeout: 3000    # 不发送心跳后，从
健康到不健康的时间。(单位:毫秒)
          preserved.ip.delete.timeout: 3000             # 不发送
心跳后，被nacos下掉该实例的时间。(单位:毫秒)
```

Spring cloud 网关 Gateway 的 ribbion 配置

```
#ribbon config,Interval to refresh the server list from the
source
ribbon:
  ServerListRefreshInterval: 3000
```

最终 Openfeign 获得注册服务的时间是 Nacos + Gateway = 6ms

12. FAQ

12.1. Cannot execute request on any known server

com.netflix.discovery.shared.transport.TransportException: Cannot execute request on any known server

解决方法，禁用 CSRF

```
package cn.netkiller.eureka.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.b
uilders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.Htt
pSecurity;
import
org.springframework.security.config.annotation.web.configuratio
n.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuratio
n.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class SecurityConfigurerAdapter extends
WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http) throws
Exception {
                http.csrf().disable();
                super.configure(http);
        }

        @Override
        protected void configure(AuthenticationManagerBuilder
auth) throws Exception {
```

```
                super.configure(auth);
        }

}
```

12.2. @EnableDiscoveryClient与@EnableEurekaClient 区别

相同点：@EnableDiscoveryClient、@EnableEurekaClient 这二个注解作用，都可以让该服务注册到注册中心上去。
不同点：@EnableEurekaClient 只支持Eureka注册中心，
@EnableDiscoveryClient 支持Eureka、Zookeeper、Consul 这三个注册中心。

12.3. Feign请求超时

```
方法一，修改配置是让Hystrix的超时时间改为5秒
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds: 5000

方法二，修改配置，禁用Hystrix的超时时间
hystrix.command.default.execution.timeout.enabled: false

方法三，修改配置，用于索性禁用feign的hystrix。
feign.hystrix.enabled: false
```

12.4. 已停止的微服务节点注销慢或不注销

由于 Eureka Server 清理无效节点周期长默认为90秒，可能会遇到微服务注销慢甚至不注销的问题。

Eureka Server 配置，注意仅适合开发环境。

```
# 设为false, 关闭自我保护， 从而保证会注销微服务
eureka.server.enable-self-preservation=false

# 清理间隔（单位毫秒，默认是60 * 1000）
eureka.server.eviction-interval-timer-in-ms=30000
```

Eureka Client

配置开启健康检查，续约更新时间和到期时间。

```
# 设为true，开启健康检查（需要spring-boot-starter-actuator 依赖）
eureka.client.healthcheck.enabled=ture

# 续约更新时间间隔（默认是30秒）
eureka.instance.lease-renewal-interval-in-seconds=20000

# 续约到期时间（默认90秒）
eureka.instance.lease-expiration-duration-in-seconds=30000
```

12.5. Feign 启动出错 PathVariable annotation was empty on param 0.

问题分析，@PathVariable 找不到对应的参数

```
package api.feign;

import java.util.List;
import java.util.Map;

import org.springframework.cloud.netflix.feign.FeignClient;
```

```
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

@FeignClient("restful-api-service")
public interface Search {

        @RequestMapping("/search/article/list")
        public List<Map<String, Object>> list();

        @RequestMapping("/search/article/{articleId}")
        public Object read(@PathVariable String articleId);
}
```

解决方案

```
        @RequestMapping("/search/article/{articleId}")
        public Object read(@PathVariable("articleId") String
articleId);
```

12.6. Feign 提示 Consider defining a bean of type 'common.feign.Cms' in your configuration.

背景：我们需要共用 Feign 接口，故将 Feign 放到共用的 common-version.jar 包中，供其他项目使用。

启动提示：Consider defining a bean of type 'common.feign.Cms' in your configuration.

注解加入包位置后解决

```
@EnableFeignClients("common.feign")
```

例 10.1. Share feign interface.

```
package cn.netkiller.feign;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import
org.springframework.cloud.netflix.feign.EnableFeignClients;

@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients("common.feign")
public class Application {

        public static void main(String[] args) {
                System.out.println("Feign Starting...");
                SpringApplication.run(Application.class, args);
        }
}
```

12.7. Load balancer does not have available server for client

```
com.netflix.client.ClientException: Load balancer does not have
available server for client: restful
```

12.8. Eureka Client (Dalston.SR1)

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller.spring.cloud</groupId>
        <artifactId>eureka.client</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>eureka.client</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.3.RELEASE</version>
                <relativePath />
        </parent>

        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
dependencies</artifactId>
                                <version>Dalston.SR1</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
```

```
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
config</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
eureka</artifactId>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-
plugin</artifactId>
                                <configuration>
                                        <skip>true</skip>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

Application

```
```

```
package cn.netkiller.spring.cloud.eureka.client;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryClien
t;

@SpringBootApplication
@EnableDiscoveryClient
public class Application {
        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
        }
}
```

RestController

```
package cn.netkiller.spring.cloud.eureka.client;

import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.cloud.client.ServiceInstance;
import
org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class TestRestController {
        private static final Logger logger =
LoggerFactory.getLogger(TestRestController.class);
```

```
        @RequestMapping("/")
        public String home() {
                logger.info("Hello!!!");
                return "Hello World";
        }

        @Autowired
        private DiscoveryClient discoveryClient;

        @RequestMapping("/service-instances/{applicationName}")
        public List<ServiceInstance>
serviceInstancesByApplicationName(@PathVariable String
applicationName) {
                return
this.discoveryClient.getInstances(applicationName);
        }

        @RequestMapping(value = "/add", method =
RequestMethod.GET)
        public Integer add(@RequestParam Integer a,
@RequestParam Integer b) {
                @SuppressWarnings("deprecation")
                ServiceInstance instance =
discoveryClient.getLocalServiceInstance();
                Integer r = a + b;
                logger.info("/add, host:" + instance.getHost()
+ ", service_id:" + instance.getServiceId() + ", result:" + r);
                return r;
        }

        @RequestMapping("/greeting")
        public String greeting() {
                return "GREETING";
        }
}
```

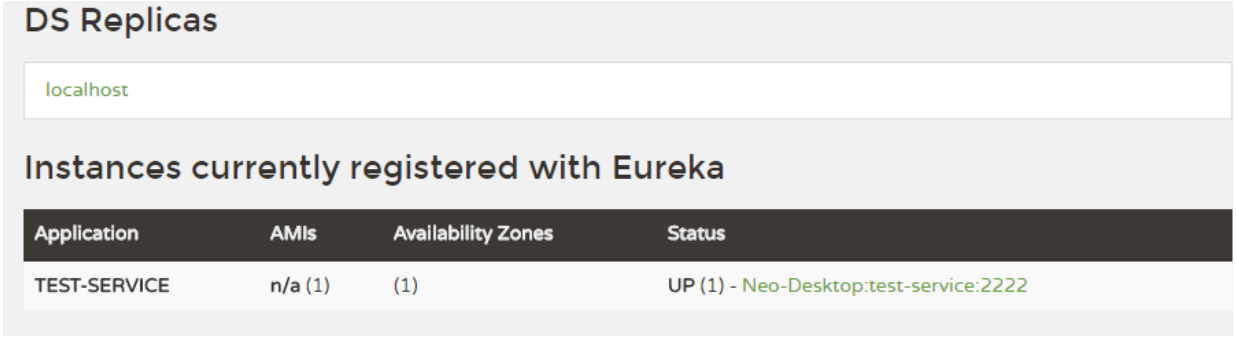
application.properties

```
spring.application.name=test-service
```

```
server.port=8080
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eure
ka/
```

测试

首先确认客户端已经注册到 http://localhost:8761/




```
$ curl http://localhost:8080/service-instances/test-service

[
        {
                "host": "Neo-Desktop",
                "port": 8080,
                "secure": false,
                "uri": "http://Neo-Desktop:8080",
                "serviceId": "TEST-SERVICE",
                "metadata": {},
                "instanceInfo": {
                        "instanceId": "Neo-Desktop:test-
service:8080",
                        "app": "TEST-SERVICE",
                        "appGroupName": null,
                        "ipAddr": "172.25.10.150",
                        "sid": "na",
                        "homePageUrl": "http://Neo-
Desktop:8080/",
                        "statusPageUrl": "http://Neo-
Desktop:8080/info",
```

```
                        "healthCheckUrl": "http://Neo-
Desktop:8080/health",
                        "secureHealthCheckUrl": null,
                        "vipAddress": "test-service",
                        "secureVipAddress": "test-service",
                        "countryId": 1,
                        "dataCenterInfo": {
                        "@class":
"com.netflix.appinfo.InstanceInfo$DefaultDataCenterInfo",
                        "name": "MyOwn"
                },
                "hostName": "Neo-Desktop",
                "status": "UP",
                "leaseInfo": {
                        "renewalIntervalInSecs": 30,
                        "durationInSecs": 90,
                        "registrationTimestamp": 1497922681680,
                        "lastRenewalTimestamp": 1497922681680,
                        "evictionTimestamp": 0,
                        "serviceUpTimestamp": 1497922003783
                },
                "isCoordinatingDiscoveryServer": false,
                "metadata": {},
                "lastUpdatedTimestamp": 1497922681680,
                "lastDirtyTimestamp": 1497922681025,
                "actionType": "ADDED",
                "asgName": null,
                "overriddenStatus": "UNKNOWN"
                }
        }
]
```

add 接口测试

```
curl http://localhost:8080/add.json?a=5&b=3

8
```

12.9. Config Server(1.3.1.RELEASE)

Server

Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>config</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>config</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.6.RELEASE</version>
                <relativePath />
        </parent>
        <dependencyManagement>
                <dependencies>
                        <dependency>

<groupId>org.springframework.cloud</groupId>
                                <artifactId>spring-cloud-
config</artifactId>
```

```
                                <version>1.3.1.RELEASE</version>
                                <type>pom</type>
                                <scope>import</scope>
                        </dependency>
                </dependencies>
        </dependencyManagement>
        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-config-
server</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>


        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

Application

Application

```

```

```
package cn.netkiller.cloud;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryClien
t;
import
org.springframework.cloud.config.server.EnableConfigServer;
import org.springframework.context.annotation.Configuration;

@Configuration
@EnableAutoConfiguration
@EnableDiscoveryClient
@EnableConfigServer
@SpringBootApplication
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
        }

}
```

application.properties

```
server.port=8888
spring.cloud.config.server.git.uri=https://github.com/netkiller/
config.git
```

Git 仓库

克隆仓库

```
git clone https://github.com/netkiller/config.git
```

创建配置文件 server-development.properties

```
vim server-development.properties

test.a=KKOOKK
message=Hello world
```

提交配置文件

```
git commit -a
git push
```

测试服务器

```
neo@netkiller $ curl http://localhost:8888/server-
development.json
{"message":"Hello world","test":{"a":"KKOOKK"}}
```

Client

Maven pom.xml

```
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>netkiller.cn</groupId>
        <artifactId>cloud</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.5.2.RELEASE</version>
                <relativePath />
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencyManagement>
            <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                    <artifactId>spring-cloud-
config</artifactId>
                    <version>1.3.1.RELEASE</version>
                    <type>pom</type>
                    <scope>import</scope>
                </dependency>
            </dependencies>
        </dependencyManagement>

        <dependencies>
                <dependency>

<groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-starter-
config</artifactId>
                </dependency>
                <dependency>
```

```
<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
actuator</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                    <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```

Application

```
package cn.netkiller.cloud.client;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import
```

```
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.context.config.annotation.RefreshScop
e;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
public class Application {

        public static void main(String[] args) {
                SpringApplication.run(Application.class, args);
        }
}

@RefreshScope
@RestController
class MessageRestController {

        @Value("${message:Hello default}")
        private String message;

        @RequestMapping("/message")
        String getMessage() {
                return this.message;
        }
}
```

注意 @RefreshScope 注解

bootstrap.properties

```
spring.application.name=server-development
spring.cloud.config.uri=http://localhost:8888
management.security.enabled=false
```

测试 client

```
neo@netkiller $ curl http://localhost:8080/message.json
Hello world
```

12.10. feign.RetryableException: Read timed out executing

```
ribbon:
  #请求连接的超时时长
  ConnectTimeout: 60000
  #请求处理的超时时长
  ReadTimeout: 60000
```

第 11 章 Tomcat Spring 运行环境

1. Maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.example</groupId>
        <artifactId>demo</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>demo</name>
        <description>Demo project for Spring Boot</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.3.0.RELEASE</version>
                <relativePath/> <!-- lookup parent from
repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
```

```
web</artifactId>
                </dependency>

                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>

<groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-
plugin</artifactId>
                        </plugin>
                </plugins>
        </build>


</project>
```

2. Spring Boot Quick start

2.1. 创建项目

```
curl https://start.spring.io/starter.tgz  \
  -d artifactId=creds-example-server \
  -d dependencies=security,web \
  -d language=java \
  -d type=maven-project \
  -d baseDir=example-server \
| tar -xzvf -
```

2.2. pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>api.netkiller.cn</groupId>
        <artifactId>api.netkiller.cn</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Skyline</name>
        <description>skylinechencf@gmail.com</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
parent</artifactId>
                <version>1.4.0.RELEASE</version>
        </parent>
        <dependencies>
                <dependency>

<groupId>org.springframework.boot</groupId>
```

```
                        <artifactId>spring-boot-starter-
web</artifactId>
                </dependency>
        </dependencies>

        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source />
                                        <target />
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

2.3. Controller

```
package hello;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
@EnableAutoConfiguration
public class SampleController {

    @RequestMapping("/")
    @ResponseBody
    String home() {
        return "Hello World!";
    }
```

```
    public static void main(String[] args) throws Exception {
        SpringApplication.run(SampleController.class, args);
    }
}
```

测试

```
curl http://127.0.0.1:8080/
```

3. Spring MVC configuration

```
  <!--
*************************************************************
*** -->
  <!--  RESOURCE FOLDERS CONFIGURATION
-->
  <!--  Dispatcher configuration for serving static resources
-->
  <!--
*************************************************************
*** -->
  <mvc:resources location="/images/" mapping="/images/**" />
  <mvc:resources location="/css/" mapping="/css/**" />
```

4. Tomcat

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        id="WebApp_ID" version="3.1">
        <display-name>m.cf88.com</display-name>
        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
                <welcome-file>index.htm</welcome-file>
                <welcome-file>index.jsp</welcome-file>
                <welcome-file>default.html</welcome-file>
                <welcome-file>default.htm</welcome-file>
                <welcome-file>default.jsp</welcome-file>
        </welcome-file-list>

        <servlet>
        <servlet-name>netkiller</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>netkiller</servlet-name>
        <url-pattern>/welcome.jsp</url-pattern>
        <url-pattern>/welcome.html</url-pattern>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>

</web-app>
```

netkiller-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-
beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-
mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context.xsd">

        <context:component-scan base-
package="cn.netkiller.controller" />

        <bean id="viewResolver"

class="org.springframework.web.servlet.view.UrlBasedViewResolve
r">
                <property name="viewClass"

value="org.springframework.web.servlet.view.JstlView" />
                <property name="prefix" value="/WEB-INF/jsp/"
/>
                <property name="suffix" value=".jsp" />
        </bean>

</beans>
```

5. 集成 Mybatis

5.1. pom.xml

```
                <dependency>
                        <groupId>org.mybatis</groupId>
                        <artifactId>mybatis</artifactId>
                        <version>3.3.0</version>
                </dependency>
                <dependency>
                        <groupId>org.mybatis</groupId>
                        <artifactId>mybatis-
spring</artifactId>
                        <version>1.2.3</version>
                </dependency>
```

5.2. properties

```
        <bean id="configuracion"

class="org.springframework.beans.factory.config.PropertyPlace
holderConfigurer">
                <property name="location"
value="classpath:resources/development.properties" />
        </bean>
```

```
jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
jdbc.url=jdbc:oracle:thin:@192.168.4.9:1521:orcl
#jdbc.url=jdbc:mysql://127.0.0.1:3306/mybatis
```

```
jdbc.username=test
jdbc.password=123456
```

```
        <bean id="dataSource"

class="org.springframework.jdbc.datasource.DriverManagerDataS
ource">
                <property name="driverClassName"
value="${jdbc.driverClassName}" />
                <property name="url" value="${jdbc.url}" />
                <property name="username"
value="${jdbc.username}" />
                <property name="password"
value="${jdbc.password}" />
        </bean>
```

5.3. dataSource

```
       <bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSou
rce">
               <property name="driverClassName"
value="${driver}" />
               <property name="url" value="${url}" />
               <property name="username" value="${username}"
/>
               <property name="password" value="${password}"
/>
       </bean>
```

5.4. SqlSessionFactory

创建SqlSessionFactory，需指定数据源，property名称必须为dataSource

```
        <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
                <property name="dataSource" ref="dataSource"
/>
        </bean>
```

5.5. Mapper 扫描

```
        <bean
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
                <property name="basePackage"
value="cn.netkiller.mappers" />
                <property name="annotationClass"
value="cn.netkiller.mappers.annotation.MybatisMapper"/>
                <property name="sqlSessionFactoryBeanName"
value="sqlSessionFactory"/>
        </bean>
```

5.6. Mapper 单一class映射

创建数据映射器Mapper，属性mapperInterface的value必须为接口类

```
        <bean id="userMapper"
class="org.mybatis.spring.mapper.MapperFactoryBean">
                <property name="mapperInterface"
value="com.mybatis.demo.UserMapper" />
```

```
                <property name="sqlSessionFactory"
ref="sqlSessionFactory" />
        </bean>
```

5.7. Service

```
        <bean id="userService"
class="cn.netkiller.service.UserService">
        </bean>
```

5.8. 测试实例

例 11.1. MyBatis

建立映射

```
package cn.netkiller.mapper;

import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Param;

import cn.netkiller.model.User;

public interface UserMapper {
        @Select("SELECT * FROM `user` WHERE id = #{id}")
        public User findById(@Param("id") int id);
}
```

建立模型

```
package cn.netkiller.model;

public class User {
	private String id;
	private String name;
	private int age;

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	@Override
	public String toString() {
		return "User [id=" + id + ", name=" + name +
", age=" + age + "]";
	}
}
```

建立 service

```
package cn.netkiller.service;

import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import cn.netkiller.mapper.UserMapper;
import cn.netkiller.model.User;

@Service
public class UserService {
        @Autowired
        private UserMapper userMapper;

        public UserMapper getUserMapper() {
                return userMapper;
        }

        public void setUserMapper(UserMapper userMapper) {
                this.userMapper = userMapper;
        }

        public User findById(int id) {
        return userMapper.findById(id);
    }
}
```

建立控制器

```
package cn.netkiller.controller;

import
org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import
org.springframework.web.context.support.WebApplicationContext
Utils;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;

import cn.netkiller.mapper.UserMapper;
import cn.netkiller.model.User;
import cn.netkiller.service.UserService;

@Controller
public class Index {

        @Autowired
        private UserMapper userMapper;

        @Autowired
    private UserService userService;


        @RequestMapping("/index")
        // @ResponseBody
        public ModelAndView index() {

                String message = "Hello";
                return new ModelAndView("index/index",
"variable", message);
        }

        @RequestMapping("/user")
        public ModelAndView user() {

                User user = userService.findById(2);
                String message = user.toString();
                return new ModelAndView("index/index",
"variable", message);
        }

        @RequestMapping("/member")
        public ModelAndView member() {
```

```
                User user = userMapper.findById(2);
                String message = user.toString();
                return new ModelAndView("index/index",
"variable", message);
        }
}
```

第 12 章 杂项 Miscellaneous

1. URL 拼装/解析

```
        UriComponents https =
UriComponentsBuilder.newInstance()
                .scheme("https")
                .host("www.netkiller.cn")
                .port("8080")
                .path("/article")
                .queryParam("id", "9527")
                .encode(StandardCharsets.UTF_8)
                .build();

        log.info(https.toUriString());
```

URL 解析

```
String httpUrl = "https://www.netkiller.cn:8080/article?
id=9527";
UriComponents uriComponents =
UriComponentsBuilder.fromHttpUrl(httpUrl).build();

提取协议头
String scheme = uriComponents.getScheme();
// scheme = https
System.out.println("scheme = " + scheme);

获取host操作。
String host = uriComponents.getHost();
// host = felord.cn
System.out.println("host = " + host);

提取 Port 端口。
```

```
int port = uriComponents.getPort();
// port = -1
System.out.println("port = " + port);
```
但是很奇怪的是上面的是 -1,很多人误以为会是80。其实 Http 协议确实是80，但是java.net.URL#getPort()规定，若 URL 的实例未申明（省略）端口号，则返回值为-1。所以当返回了-1就等同于80，但是 URL 中不直接体现它们。

```
提取 Path 路径
String path = uriComponents.getPath();
// path = /spring-security/{article}
System.out.println("path = " + path);

提取 Query 参数

String query = uriComponents.getQuery();
// query = version=1&timestamp=123123325
System.out.println("query = " + query);
```
更加合理的提取方式：

```
MultiValueMap<String, String> queryParams =
uriComponents.getQueryParams();
// queryParams = {version=[1], timestamp=[123123325]}
System.out.println("queryParams = " + queryParams);
```

替换变量

```
        UriComponents uriComponents =
UriComponentsBuilder.newInstance()
                .scheme("https")
                .host("www.netkiller.cn")
                .port("8080")
                .path("/article/{category}")
                .queryParam("id", "9527")
                .encode(StandardCharsets.UTF_8)
                .build();

        UriComponents expand = uriComponents.expand("story");
```

```
        log.info(expand.toUriString());
        # https://www.netkiller.cn:8080/article/story?id=9527

        UriComponents uriComponents =
UriComponentsBuilder.newInstance()
                .scheme("https")
                .host("www.netkiller.cn")
                .port("8080")
                .path("/book/{chapter}/{section}")
                .queryParam("id", "9527")
                .encode(StandardCharsets.UTF_8)
                .build();
        UriComponents expand =
uriComponents.expand(Map.of("chapter", "chapter1", "section",
"section2"));

        log.info(expand.toUriString());
        #
https://www.netkiller.cn:8080/book/chapter1/section2?id=9527
```

2. ServletUriComponentsBuilder

```
String locationUri = ServletUriComponentsBuilder
                    .fromCurrentRequest()
                    .path("/{id}")
                    .buildAndExpand(employeeId)
                    .toUriString();
```

3. URL 路径相关

过滤路径

```
        PathPattern pattern = new
PathPatternParser().parse("/test/**");
        PathContainer pathContainer =
exchange.getRequest().getPath().pathWithinApplication();
        if (pattern.matches(pathContainer)) {
            log.info("custom webFilter");
            return chain.filter(exchange);
        }
```

```
        PathPatternParser pathPatternParser = new
PathPatternParser();

        List<String> paths = List.of("/token", "/verifier",
"/mock/*");
        List<PathPattern> parsedPatterns = new ArrayList<>();

        for (String path : paths) {
            PathPattern pathPattern =
pathPatternParser.parse(path);
            parsedPatterns.add(pathPattern);
        }

        PathContainer pathContainer =
exchange.getRequest().getPath().pathWithinApplication();
        for (PathPattern pattern : parsedPatterns) {
            if (pattern.matches(pathContainer)) {
                System.out.println("Path " + pathContainer +
" matches pattern " + pattern.getPatternString());
            }
        }
```

```
        return chain.filter(exchange);
```

第 13 章 FAQ

1. org.hibernate.dialect.Oracle10gDialect does not support identity key generation

```
@GeneratedValue(strategy=GenerationType.IDENTITY)
换成
@GeneratedValue(strategy=GenerationType.AUTO)

or

@Id
@Column(name = "ID")
@GeneratedValue(strategy=GenerationType.SEQUENCE, generator =
"id_Sequence")
@SequenceGenerator(name = "id_Sequence", sequenceName =
"ID_SEQ")
private int id;
```

2. No identifier specified for entity

在实体中使用

```
import javax.persistence.Id;
替换
import org.springframework.data.annotation.Id;
```

3. Could not read document: Invalid UTF-8 middle byte 0xd0

Spring 默认不支持 UTF-8

```
2016-08-17 16:04:53.148 WARN 7700 --- [nio-8080-exec-1]
.w.s.m.s.DefaultHandlerExceptionResolver : Failed to read HTTP
message:
org.springframework.http.converter.HttpMessageNotReadableExcept
ion:Could not read document: Invalid UTF-8 middle byte 0xd0 at
[Source: java.io.PushbackInputStream@33aa54cc; line: 1, column:
38](through reference chain:
api.domain.oracle.Withdraw["bankname"]); nested exception is
com.fasterxml.jackson.databind.JsonMappingException: Invalid
UTF-8 middle byte 0xd0 at [Source:
java.io.PushbackInputStream@33aa54cc; line: 1, column: 38]
                        (through reference chain:
api.domain.oracle.Withdraw["bankname"])
```

解决方案 application.properties 配置文件中加入如下配置:

```
                        spring.messages.encoding=UTF-8
                        server.tomcat.uri-encoding=UTF-8
                        spring.http.encoding.charset=UTF-8
                        spring.http.encoding.enabled=true
                        spring.http.encoding.force=true
```

4. java.sql.SQLRecoverableException: IO Error: The Network Adapter could not establish the connection

分析，Oracle 数据库无法连接，确认用户密码正确，日志提示 The Network Adapter could not establish the connection 看上去更像网络故障，同事还有下面两条日志。

```
                        Caused by: oracle.net.ns.NetException:
The Network Adapter could not
                        establish the connection
                        Caused by:
java.net.SocketTimeoutException: connect timed out
```

通过 ss 命令可以看到有tcp操作，可以排除不是网络故障。

```
                        [root@iZ62m7362hwZ ~]# ss -ant | grep
1521
                        TIME-WAIT 0 0 47.90.18.24:45780
15.84.21.59:1521
```

检查你的用户名与密码是否含有特殊字符，特殊字符需要使用转义字符"\".

```
spring.datasource.url=jdbc:oracle:thin:neo/\
[y7\$ghM\~3b@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
(HOST=215.184.211.50)(PORT=1521))(LOAD_BALANCE=YES)
(FAILOVER=ON)(CONNECT_DATA=(SERVER=DEDICATED)
(SERVICE_NAME=orcl)(FAILOVER_MODE=(TYPE=SESSION)
(METHOD=BASIC))))
                        #spring.datasource.username=neo
                        #spring.datasource.password=[y7$ghM~3b
```

将用户名写入spring.datasource.url中，格式jdbc:oracle:thin:用户名/密码@(.....)，禁用spring.datasource.username和spring.datasource.password两个配置项。

5. Field javaMailSender in cn.netkiller.rest.EmailRestController required a bean of type 'org.springframework.mail.javamail.JavaMailSender' that could not be found.

启动提示 'org.springframework.mail.javamail.JavaMailSender' that could not be found 这句话很误导人。实际上是 (spring.mail.host) did not find property 'host'

```
                        ***************************
                        APPLICATION FAILED TO START
                        ***************************

                        Description:

                        Field javaMailSender in
cn.netkiller.rest.EmailRestController required a
                        bean of type
'org.springframework.mail.javamail.JavaMailSender' that
                        could not be found.
                        - Bean method 'mailSender' not loaded
because AnyNestedCondition 0
                        matched 2 did not; NestedCondition on

MailSenderAutoConfiguration.MailSenderCondition.JndiNamePropert
y
                        @ConditionalOnProperty
(spring.mail.jndi-name) did not find property
                        'jndi-name';
                        NestedCondition on

MailSenderAutoConfiguration.MailSenderCondition.HostProperty
                        @ConditionalOnProperty
(spring.mail.host) did not find property
                        'host'
```

```
                        Action:

                        Consider revisiting the conditions
above or defining a bean of type

'org.springframework.mail.javamail.JavaMailSender' in your
                        configuration.
```

解决方案，application.properties 增加 spring.mail.host=localhost

6. org.postgresql.util.PSQLException: FATAL: no pg_hba.conf entry for host "172.16.0.3", user "test", database "test ", SSL off

确认 pg_hba.conf 配置正确，并且 psql 可以正常链接，spring仍然报错

```
spring.datasource.url=jdbc:postgresql://47.90.18.244:5432/test
                        spring.datasource.username=test
                        spring.datasource.password=test
                        spring.jpa.show-sql=true
                        spring.jpa.hibernate.ddl-auto=create-
drop
                        spring.jpa.generate-ddl=true
```

请检查 jdbc:postgresql://47.90.18.244:5432/test 后面test是否多了一个空格或者有特殊字符。删除test后面的空格可以解决

7. Spring boot 怎样显示执行的SQL语句

```
spring.jpa.show-sql=true
```

8. Cannot determine embedded database driver class for database type NONE

错误如下

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Cannot determine embedded database driver class for database
type NONE

Action:

If you want an embedded database please put a supported one on
the classpath. If you have database settings to be loaded from
a particular profile you may need to active it (no profiles are
currently active).
```

背景：Maven 项目中并不包含任何与数据库有关的依赖。问题出在另一个公共包中如： common-version.jar

解决方案：排除不需要的包

```
                <dependency>
                        <groupId>cn.netkiller</groupId>
                        <artifactId>common</artifactId>
                        <version>0.0.1-SNAPSHOT</version>
                        <exclusions>
                                <exclusion>
```

```
<groupId>mysql</groupId>
                                        <artifactId>mysql-
connector-java</artifactId>
                                </exclusion>
                                <exclusion>

<groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-
boot-starter-data-jpa</artifactId>
                                </exclusion>
                                <exclusion>

<groupId>org.springframework.boot</groupId>
                                        <artifactId>spring-
boot-starter-jdbc</artifactId>
                                </exclusion>
                        </exclusions>
                </dependency>
```

9. Spring boot / Spring cloud 时区差8个小时

经过检查：操作系统时区 CST，数据库是 SYSTEM，Spring boot 获取时间相差8个小时。

分析：认为是 @JsonFormat 格式化造成的。

解决方案：在 @JsonFormat 中增加时区设置。

```
        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "Asia/Shanghai")
        public Date ctime;
```

期间尝试多种方式无效:

下面例子无效
spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.mvc.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=GMT+8

# 下面方法无效
spring.datasource.url=jdbc:mysql://119.29.241.95:3306/5kwords?
useSSL=false&serverTimezone=UTC

# 下面配置仍然无效
spring.jpa.properties.jadira.usertype.autoRegisterUserTypes = true
spring.jpa.properties.jadira.usertype.javaZone=Asia/Shanghai
spring.jpa.properties.jadira.usertype.databaseZone=Asia/Shanghai

根源在 Json 转化。

完成的例子

```
package common.domain;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

import org.springframework.format.annotation.DateTimeFormat;

import com.fasterxml.jackson.annotation.JsonFormat;

@Entity
@Table(name = "article", catalog = "cms")
public class Article implements Serializable {
	private static final long serialVersionUID =
76037726829502713 21L;

	@Id
	public int id;
	public String title;
	@Column(name = "short")
	public String shortTitle;
	public String description;
	public String author;
	public String star;
	public String tags;
	public boolean status;
	public String content;
	public int typeId;
	public int siteId;

	@DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
	@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "Asia/Shanghai")
```

```
        public Date ctime;

        @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")
        @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone
= "America/Phoenix")
        public Date mtime;

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getTitle() {
                return title;
        }

        public void setTitle(String title) {
                this.title = title;
        }

        public String getDescription() {
                return description;
        }

        public void setDescription(String description) {
                this.description = description;
        }

        public Date getCtime() {
                return ctime;
        }

        public void setCtime(Date ctime) {
                this.ctime = ctime;
        }

        public String getShortTitle() {
                return shortTitle;
        }

        public void setShortTitle(String shortTitle) {
                this.shortTitle = shortTitle;
```

```
	}

	public String getAuthor() {
		return author;
	}

	public void setAuthor(String author) {
		this.author = author;
	}

	public String getStar() {
		return star;
	}

	public void setStar(String star) {
		this.star = star;
	}

	public String getTags() {
		return tags;
	}

	public void setTags(String tags) {
		this.tags = tags;
	}

	public boolean isStatus() {
		return status;
	}

	public void setStatus(boolean status) {
		this.status = status;
	}

	public String getContent() {
		return content;
	}

	public void setContent(String content) {
		this.content = content;
	}

	public int getTypeId() {
		return typeId;
	}
```

```
	public void setTypeId(int typeId) {
		this.typeId = typeId;
	}

	public int getSiteId() {
		return siteId;
	}

	public void setSiteId(int siteId) {
		this.siteId = siteId;
	}

	public Date getMtime() {
		return mtime;
	}

	public void setMtime(Date mtime) {
		this.mtime = mtime;
	}

	@Override
	public String toString() {
		return "Article [id=" + id + ", title=" +
title + ", shortTitle=" + shortTitle + ", description=" +
description + ", author=" + author + ", star=" + star + ",
tags=" + tags + ", status=" + status + ", content=" + content
+ ", typeId=" + typeId + ", siteId=" + siteId + ", ctime=" +
ctime + ", mtime=" + mtime + "]";
	}

}
```

10. @Value 取不到值

在构造方法中引用@value为null，由于spring实例化顺序为先执行构造方法，再注入成员变量，所以取值为null。

调用spring组件时使用new对象，而不是@Autowired。

11. Spring boot 2.1.0

application.properties 需要加入下面参数，否则 @Bean 不允许。

```
spring.main.allow-bean-definition-overriding=true
```

另外 MySQL驱动必须使用最新的 com.mysql.cj.jdbc.Driver

12. Field authenticationManager in cn.netkiller.oauth2.config.AuthorizationServerConfigurer required a bean of type 'org.springframework.security.authentication.AuthenticationManager' that could not be found.

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Field authenticationManager in
cn.netkiller.oauth2.config.AuthorizationServerConfigurer
required a bean of type
'org.springframework.security.authentication.AuthenticationMana
ger' that could not be found.

The injection point has the following annotations:
        -
@org.springframework.beans.factory.annotation.Autowired(require
d=true)


Action:

Consider defining a bean of type
'org.springframework.security.authentication.AuthenticationMana
ger' in your configuration.
```

注入 @Bean @Override public AuthenticationManager authenticationManagerBean() 即可解决

```
package cn.netkiller.oauth2.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationManag
er;
import
org.springframework.security.config.annotation.method.configura
tion.EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.Htt
pSecurity;
import
org.springframework.security.config.annotation.web.configuratio
n.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuratio
n.WebSecurityConfigurerAdapter;

@EnableGlobalMethodSecurity(prePostEnabled = true)
@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends
WebSecurityConfigurerAdapter {

        public WebSecurityConfigurer() {
                // TODO Auto-generated constructor stub
        }

        // 此处必须声明这个bean类，否则无法注入AuthenticationManager
        @Bean
        @Override
        public AuthenticationManager
authenticationManagerBean() throws Exception {
                return super.authenticationManagerBean();
        }

        protected void configure(HttpSecurity http) throws
Exception {

http.formLogin().and().authorizeRequests().anyRequest().authent
```

```
icated().and().logout().permitAll().and().csrf().disable();
        }
}
```

13. 打印 Bean 信息

```
import java.util.Arrays;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Bean
    public CommandLineRunner
commandLineRunner(ApplicationContext ctx) {
        return args -> {

            System.out.println("Let's inspect the beans
provided by Spring Boot:");

            String[] beanNames =
ctx.getBeanDefinitionNames();
            Arrays.sort(beanNames);
            for (String beanName : beanNames) {
                System.out.println(beanName);
            }

        };
    }

}
```

14. The dependencies of some of the beans in the application context form a cycle

@Service 中的 @Autowired 出现了相互注入，引起循环。

```
api    | ***************************
api    | APPLICATION FAILED TO START
api    | ***************************
api    |
api    | Description:
api    |
api    | The dependencies of some of the beans in the
application context form a cycle:
api    |
api    | ┌─────┐
api    | |  incarOrderFlowServiceImpl defined in URL
[jar:file:/app/api.netkiller.cn.jar!/BOOT-
INF/lib/api.netkiller.cn-
1.0.0.jar!/com/zito/incar/service/impl/IncarOrderFlowServiceImp
l.class]
api    | ↑     ↓
api    | |  incarOrderServiceImpl (field private
cn.netkiller.service.IIncarAttachService
cn.netkiller.service.impl.IncarOrderServiceImpl.iIncarAttachSer
vice)
api    | ↑     ↓
api    | |  incarAttachServiceImpl (field private
cn.netkiller.service.IIncarOrderFlowService
cn.netkiller.service.impl.IncarAttachServiceImpl.iIncarOrderFlo
wService)
api    | └─────┘
```

解决方案

增加 @Lazy 注解

```
@Slf4j
@Service
@Lazy
public class IncarOrderFlowServiceImpl implements
IIncarOrderFlowService {

        @Autowired
        private ISysRoleService iSysRoleService;

        @Autowired
        private  IIncarOrderService iIncarOrderService;
        ...
        ...
        ...
}
```

15. no main manifest attribute, in /srv/job-admin.jar

```
[root@localhost cloud.netkiller.cn]# java -jar job-admin.jar
no main manifest attribute, in job-admin.jar
```

解压 jar 包，查看 META-INF/MANIFEST.MF 文件

```
[root@localhost ~]# unzip job-admin.jar
[root@localhost ~]# cat META-INF/MANIFEST.MF
Manifest-Version: 1.0
Archiver-Version: Plexus Archiver
Built-By: gitlab-runner
Created-By: Apache Maven 3.8.4
Build-Jdk: 1.8.0_312
```

解决方法，pom.xml 中增加 repackage 配置项

```
                <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-
plugin</artifactId>
                <version>${spring-boot.version}</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
```

```
            </plugin>
        </plugins>
```

重新打包，再解压开查看 MANIFEST.MF 文件

```
[root@localhost ~]# cat META-INF/MANIFEST.MF
Manifest-Version: 1.0
Spring-Boot-Classpath-Index: BOOT-INF/classpath.idx
Archiver-Version: Plexus Archiver
Built-By: gitlab-runner
Spring-Boot-Layers-Index: BOOT-INF/layers.idx
Start-Class: cn.netkiller.admin.Application
Spring-Boot-Classes: BOOT-INF/classes/
Spring-Boot-Lib: BOOT-INF/lib/
Spring-Boot-Version: 2.6.2
Created-By: Apache Maven 3.8.4
Build-Jdk: 1.8.0_312
Main-Class: org.springframework.boot.loader.JarLauncher
```

第 14 章 MyBatis

http://blog.mybatis.org/

1. Mybatis 入门

创建数据库与表并插入测试数据

```
CREATE DATABASE `mybatis` /*!40100 COLLATE 'utf8_general_ci'
*/;

CREATE USER 'mybatis'@'192.168.%' IDENTIFIED BY 'mybatis';
GRANT USAGE ON *.* TO 'mybatis'@'192.168.%';
GRANT SELECT, EXECUTE, SHOW VIEW, ALTER, ALTER ROUTINE,
CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW,
DELETE, DROP, EVENT, INDEX, INSERT, REFERENCES, TRIGGER,
UPDATE, LOCK TABLES  ON `mybatis`.* TO 'mybatis'@'192.168.%'
WITH GRANT OPTION;
FLUSH PRIVILEGES;
SHOW GRANTS FOR 'mybatis'@'192.168.%';


CREATE TABLE IF NOT EXISTS `user` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(50) DEFAULT NULL,
  `age` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;


INSERT INTO `user` (`id`, `name`, `age`) VALUES
        (1, 'Neo', 35),
        (2, 'Jerry', 36);
```

Maven pom.xml 中加入依赖包

```
        <dependencies>
                <dependency>
                        <groupId>org.mybatis</groupId>
                        <artifactId>mybatis</artifactId>
                        <version>3.3.0</version>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                        <version>5.1.37</version>
                </dependency>
        </dependencies>
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>MyBatis</groupId>
        <artifactId>MyBatis</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>org.mybatis</groupId>
                        <artifactId>mybatis</artifactId>
                        <version>3.3.0</version>
                </dependency>
                <dependency>
```

```
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-
java</artifactId>
                        <version>5.1.37</version>
                </dependency>
        </dependencies>
        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.3</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

src/mybatis.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config
3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
        <environments default="development">
                <environment id="development">
                        <transactionManager type="JDBC" />
                        <!-- 配置数据库连接信息 -->
                        <dataSource type="POOLED">
                                <property name="driver"
value="com.mysql.jdbc.Driver" />
                                <property name="url"
value="jdbc:mysql://192.168.6.1:3306/mybatis" />
                                <property name="username"
value="mybatis" />
```

```
                                    <property name="password"
value="mybatis" />
                            </dataSource>
                </environment>
        </environments>
        <mappers>

                <mapper
resource="cn/netkiller/mapping/userMapping.xml" />
        </mappers>
</configuration>
```

src/cn/netkiller/mapping/userMapping.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="cn.netkiller.mapping.UserMapping">
        <select id="getUser" parameterType="String"
resultType="cn.netkiller.model.User">
                select * from user where id=#{id}
        </select>
</mapper>
```

resultType 文件

```
package cn.netkiller.model;

public class User {
        private String id;
        private String name;
        private int age;
```

```
        public String getId() {
                return id;
        }

        public void setId(String id) {
                this.id = id;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this.name = name;
        }

        public int getAge() {
                return age;
        }

        public void setAge(int age) {
                this.age = age;
        }

        @Override
        public String toString() {
                return "User [id=" + id + ", name=" + name +
", age=" + age + "]";
        }
}
```

测试代码

```
package cn.netkiller.test;

import java.io.InputStream;

import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
```

```
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import cn.netkiller.model.*;

public class Tests {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                String resource = "mybatis.xml";

                InputStream is =
Tests.class.getClassLoader().getResourceAsStream(resource);
                SqlSessionFactory sessionFactory = new
SqlSessionFactoryBuilder().build(is);
                SqlSession session =
sessionFactory.openSession();

                String statement =
"cn.netkiller.mapping.UserMapping.getUser";// 映射sql的标识字符
串

                User user = session.selectOne(statement,
"2");
                System.out.println(user.toString());
        }

}
```

2. 接口注解

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config
3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
        <environments default="development">
                <environment id="development">
                        <transactionManager type="JDBC" />
                        <dataSource type="POOLED">
                                <property name="driver"
value="com.mysql.jdbc.Driver" />
                                <property name="url"
value="jdbc:mysql://192.168.6.1:3306/mybatis" />
                                <property name="username"
value="mybatis" />
                                <property name="password"
value="mybatis" />
                        </dataSource>
                </environment>
        </environments>
        <mappers>
                <mapper
class="cn.netkiller.mapper.UserMapper" />
        </mappers>
</configuration>
```

```
package cn.netkiller.mapper;

import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Param;

import cn.netkiller.model.User;

public interface UserMapper {
```

```
        @Select("SELECT * FROM `user` WHERE id = #{id}")
        public User findById(@Param("id") int id);
}
```

```
package cn.netkiller.test;

import java.io.InputStream;

import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import cn.netkiller.mapper.UserMapper;
import cn.netkiller.model.User;

public class AnnotationsTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                String resource = "mybatis.xml";

                InputStream is =
XmlTest.class.getClassLoader().getResourceAsStream(resource);
                SqlSessionFactory sessionFactory = new
SqlSessionFactoryBuilder().build(is);
                SqlSession session =
sessionFactory.openSession();

                UserMapper mapper =
session.getMapper(UserMapper.class);
                User user = mapper.findById(2);

                System.out.println(user.toString());

        }
}
```

第 15 章 Apache Struts

http://struts.apache.org/

You can checkout all the example applications from the Struts 2 GitHub repository at https://github.com/apache/struts-examples.

1. struts.xml

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        id="WebApp_ID" version="3.0">
        <display-name>helloworld</display-name>
        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
                <welcome-file>index.htm</welcome-file>
                <welcome-file>index.jsp</welcome-file>
                <welcome-file>default.html</welcome-file>
                <welcome-file>default.htm</welcome-file>
                <welcome-file>default.jsp</welcome-file>
        </welcome-file-list>

        <filter>
                <filter-name>struts2</filter-name>
                <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAn
dExecuteFilter</filter-class>
        </filter>

        <filter-mapping>
```

```
                <filter-name>struts2</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>

</web-app>
```

1.1. include

```
        <include file="/cn/netkiller/struts/ajax.xml" />
        <include file="/cn/netkiller/struts/admin.xml" />
        <include file="/cn/netkiller/struts/logs.xml" />
```

2. Struts Tags

使用Struts Tags 需要在jsp页面中加入下面一行。

```
<%@ taglib prefix="s" uri="/struts-tags" %>
```

2.1. property

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
<head>
    <title>Hello</title>
</head>
<body>

Hello, <s:property value="name"/>

</body>
</html>
```

```
<s:property value="messageStore.message" />
<s:property value="#session.user.username" />

<s:bean name="cn.netkiller.Person" var="personBean" />
<s:property value="#personBean.name" />
```

2.2. set

```
<s:set var="personName" value="person.name"/>
Hello, <s:property value="#personName"/>

<s:set var="janesName">Jane Doe</s:set>
<s:property value="#janesName"/>
```

禁止HTML转义，如果你的字符串中含有&, <, > 等字符输出就会出现 &, <, > escapeHtml="false" 可以禁止这样的转义，原样输出。

```
<s:property value="url" escapeHtml="false"/>
```

https://struts.apache.org/docs/property.html

```
Name    Required        Default Evaluated       Type
Description
default false           false   String  The default value to
be used if value attribute is null
escapeCsv       false   false   false   Boolean Whether to
escape CSV (useful to escape a value for a column)
escapeHtml      false   true    false   Boolean Whether to
escape HTML
escapeJavaScript        false   false   false   Boolean
Whether to escape Javascript
escapeXml       false   false   false   Boolean Whether to
escape XML
```

2.3. url

```
<p><a href="<s:url action='hello'/>">Hello World</a></p>

<s:url action="hello" var="helloLink">
  <s:param name="userName">Bruce Phillips</s:param>
</s:url>

<p><a href="${helloLink}">Hello Bruce Phillips</a></p>
```

2.4. s:include

```
<s:include value="/pages/example.jsp"></s:include>
```

2.5. s:action

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<s:action name="index" namespace="/news" executeResult="true"
/>
```

```
<s:action name="index" namespace="/member"
executeResult="true">
        <s:param name="name">Neo</s:param>
</s:action>
```

```

```

2.6. HTML Form

form

```
<p>Get your own personal hello by filling out and submitting
this form.</p>

<s:form action="hello">

  <s:textfield name="userName" label="Your name" />

   <s:submit value="Submit" />

</s:form>
```

textfield

```
<s:textfield name="variable"/>
```

s:hidden

隐藏表单

```
<s:hidden id="unique" name="form.unique" value=""/>
```

select

```
<s:select name="city" list="
{'Beijing','Shanghai','Guangdong','Shenzhen'}" theme="simple"
headerKey="Shenzhen" headerValue="Shenzhen"></s:select>

<select name="city" id="searchCriteriaForm_city">
    <option value="Shenzhen">Shenzhen</option>
    <option value="Beijing">Beijing</option>
    <option value="Shanghai">Shanghai</option>
    <option value="Guangdong">Guangdong</option>
    <option value="Shenzhen">Shenzhen</option>
</select>
```

```
<s:select name="city" id="city" list="#
{1:'Beijing',2:'Shanghai',3:'Guangdong',4:'Shenzhen'}"
label="city" listKey="key" listValue="value"  headerKey="4"
headerValue="Shenzhen" />

<select name="city" id="city">
    <option value="4">Shenzhen</option>
    <option value="1">Beijing</option>
    <option value="2">Shanghai</option>
    <option value="3">Guangdong</option>
    <option value="4">Shenzhen</option>
</select>
```

2.7. iterator

```
<s:iterator value="people">
        <s:property value="lastName"/>, <s:property
value="firstName"/>
</s:iterator>
```

2.8. if elseif else

```
<s:if test="%{false}">
    <div>Will Not Be Executed</div>
</s:if>
<s:elseif test="%{true}">
    <div>Will Be Executed</div>
</s:elseif>
<s:else>
    <div>Will Not Be Executed</div>
</s:else>
```

3. Action

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

        <constant name="struts.devMode" value="true" />

        <package name="basicstruts2" extends="struts-
default">
                <action name="index">
                        <result>/index.jsp</result>
                </action>

                <action name="hello"

class="org.apache.struts.helloworld.action.HelloWorldAction"
method="execute">
                        <result
name="success">/HelloWorld.jsp</result>
                </action>

                <action name="register"
class="org.apache.struts.register.action.Register"
                        method="execute">
                        <result
name="success">/thankyou.jsp</result>
                        <result
name="input">/register.jsp</result>
                </action>

        </package>
</struts>
```

3.1. redirect

```
<action name="hello"
   class="com.tutorialspoint.struts2.HelloWorldAction"
   method="execute">
   <result name="success" type="redirect">
       <param name="location">
         /NewWorld.jsp
      </param >
   </result>
</action>
```

3.2. redirectAction

```
<package name="public" extends="struts-default">
    <action name="login" class="...">
        <!-- Redirect to another namespace -->
        <result type="redirectAction">
            <param name="actionName">dashboard</param>
            <param name="namespace">/secure</param>
        </result>
    </action>
</package>

<package name="secure" extends="struts-default"
namespace="/secure">
    <-- Redirect to an action in the same namespace -->
    <action name="dashboard" class="...">
        <result>dashboard.jsp</result>
        <result name="error"
type="redirectAction">error</result>
    </action>
```

```
    <action name="error" class="...">
        <result>error.jsp</result>
    </action>
</package>

<package name="passingRequestParameters" extends="struts-
default" namespace="/passingRequestParameters">
   <!-- Pass parameters (reportType, width and height) -->
   <!--
   The redirectAction url generated will be :
   /genReport/generateReport.action?
reportType=pie&amp;width=100&amp;height=100#summary
   -->
   <action name="gatherReportInfo" class="...">
      <result name="showReportResult" type="redirectAction">
         <param name="actionName">generateReport</param>
         <param name="namespace">/genReport</param>
         <param name="reportType">pie</param>
         <param name="width">100</param>
         <param name="height">100</param>
         <param name="empty"></param>
         <param name="suppressEmptyParameters">true</param>
         <param name="anchor">summary</param>
      </result>
   </action>

   <action name="Auth" class="cn.netkiller.api.action.Auth">
                <result name="credit" type="redirectAction">
                        <param
name="actionName">history</param>
                        <param
name="namespace">/api/report</param>
                        <param name="LoginName">${LoginName}
</param>
                        <param name="Direction">${Direction}
</param>
                        <param name="StartDate">${StartDate}
</param>
                        <param name="EndDate">${EndDate}
</param>
                </result>
        </action>

</package>
```

```
```

3.3. JSON

JSON 拦截器

```
        <action name="withdraw"
class="cn.netkiller.api.action.Report" method="getHistory">
                <interceptor-ref name="defaultStack" />
                <interceptor-ref name="json">
                        <param name="enableSMD">true</param>
                </interceptor-ref>
                <result name="success" type="json">
                        <param name="enableGZIP">true</param>
                        <param
name="excludeProperties">.*direction</param>
                </result>
        </action>
```

enableGZIP 压缩传输

```
                <result name="success" type="json">
                        <param name="enableGZIP">true</param>
                </result>
```

excludeProperties 排除 Properties

```
                <result name="success" type="json">
```

```
                        <param
name="excludeProperties">.*direction</param>
                </result>
```

```
```

3.4. 传递 Timestamp 变量

Struts 从URL传递 Timestamp 类型的变量 StartDate

```
http://www.netkiller.cn/api/report/history.do?
LoginName=888666&StartDate=2016-01-01&EndDate=2016-02-29
```

解决方法 参考 setStartDate / setEndDate 两个方法，以字符串方式传入，然后转换为Timestamp类型

```
package cn.netkiller.api.action;

import java.sql.Timestamp;
import java.util.Calendar;
import org.apache.struts2.json.annotations.JSON;
import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

public class Report extends ActionSupport {
        private static final long serialVersionUID =
6484202866632836225L;

        private String loginName;
```

```
        private Timestamp startDate = null;
        private Timestamp endDate = null;

        public Report() {

        }

        public String execute() {
                return Action.SUCCESS;
        }

        public String getLoginName() {
                return loginName;
        }

        public void setLoginName(String loginName) {
                this.loginName = loginName;
        }

        public Timestamp getStartDate() {
                return startDate;
        }

        public void setStartDate(String startDate) {
                this.startDate = Timestamp.valueOf(startDate
+ " 00:00:00");
        }

        public Timestamp getEndDate() {
                return endDate;
        }

        public void setEndDate(String endDate) {
                this.endDate = Timestamp.valueOf(endDate + "
00:00:00");
        }

}
```

4. Ajax + JSON

struts.xml 中加入

```
                <action name="Captcha"
class="com.example.action.ajax.Captcha">
                        <result name="success" type="json">
</result>
                </action>
```

Java 文件

```
package com.example.action.ajax;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

public class Captcha extends ActionSupport {
        private static final long serialVersionUID =
7284583098398030297L;

        private String string1 = "A";
        private String[] stringarray1 = {"A1","B1"};
        private int number1 = 123456789;
        private int[] numberarray1 = {1,2,3,4,5,6,7,8,9};
        private List<String> lists = new ArrayList<String>();
        private Map<String, String> maps = new
HashMap<String, String>();
```

```
        //no getter method, will not include in the JSON

        public Captcha(){
                lists.add("list1");
                lists.add("list2");
                lists.add("list3");
                lists.add("list4");
                lists.add("list5");

                maps.put("key1", "value1");
                maps.put("key2", "value2");
                maps.put("key3", "value3");
                maps.put("key4", "value4");
                maps.put("key5", "value5");
        }

        public String execute() {
               return Action.SUCCESS;
        }

        public String getString1() {
                return string1;
        }

        public void setString1(String string1) {
                this.string1 = string1;
        }

        public String[] getStringarray1() {
                return stringarray1;
        }

        public void setStringarray1(String[] stringarray1) {
                this.stringarray1 = stringarray1;
        }

        public int getNumber1() {
                return number1;
        }

        public void setNumber1(int number1) {
                this.number1 = number1;
        }

        public int[] getNumberarray1() {
```

```
                return numberarray1;
        }

        public void setNumberarray1(int[] numberarray1) {
                this.numberarray1 = numberarray1;
        }

        public List<String> getLists() {
                return lists;
        }

        public void setLists(List<String> lists) {
                this.lists = lists;
        }

        public Map<String, String> getMaps() {
                return maps;
        }

        public void setMaps(Map<String, String> maps) {
                this.maps = maps;
        }

}
```

测试URL http://localhost:8080/ajax/Captcha.action

4.1. GET/POST JSON

struts.xml 文件加入 enableSMD

```
                <action name="Captcha"
class="com.example.action.ajax.Captcha">
                        <interceptor-ref name="defaultStack"
/>
                        <interceptor-ref name="json">
                                <param
name="enableSMD">true</param>
```

```
                        </interceptor-ref>
                        <result name="success" type="json">
</result>
                </action>
```

Java 代码

```
package com.example.action.ajax;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

public class Captcha extends ActionSupport {
        private static final long serialVersionUID =
7284583098398030297L;

        private String phone = "13113668890";
        private String email = "netkiller@msn.com";
        private Boolean status = false;

        //no getter method, will not include in the JSON

        public Captcha() {

        }

        public String getEmail() {
                return email;
        }

        public void setEmail(String email) {
                this.email = email;
        }

        public String execute() {
                this.status = true;
                System.out.printf("%s, %s, %s, %b\n",
this.getClass().getName(), this.getPhone(), this.getEmail(),
this.getStatus());
```

```
                return Action.SUCCESS;
        }

        public String getPhone() {
                return this.phone;
        }

        public void setPhone(String phone) {
                this.phone = phone;
        }

        public void setStatus(Boolean status) {
                this.status = status;
        }
        public boolean getStatus() {
                return this.status;
        }
}
```

使用 curl 模拟 post 测试

```
# curl http://192.168.4.34:8080/ajax/Captcha.do
{"email":"netkiller@msn.com","phone":"13113668890","status":t
rue}

# curl -X POST -H "Content-Type:application/json" -d
'{"phone":"13066884444", "email":"neo.chan@live.com"}'
http://192.168.4.34:8080/ajax/Captcha.do
{"email":"neo.chan@live.com","phone":"13066884444","status":t
rue}
```

GET 例子

```
# curl "http://192.168.4.34:8080/ajax/Captcha.do?
```

```
phone=13322993040&email=netkiller@mac.com"
{"email":"netkiller@mac.com","phone":"13322993040","status":t
rue}
```

5. Json 内容展示

Struts 配置文件

```
        <package name="information" extends="main"
namespace="/inf">
                <action name="Information"
class="com.example.action.Infomation">
                        <result
type="tiles">information</result>
                </action>
        </package>
```

Action 文件

```
package cn.netkiller.action;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.StringReader;
import java.net.URL;
import java.net.URLConnection;
import java.nio.charset.Charset;

import javax.json.Json;
import javax.json.JsonObject;
import javax.json.JsonReader;

import com.opensymphony.xwork2.Action;
import com.opensymphony.xwork2.ActionSupport;

public class Infomation extends ActionSupport{
        /**
```

```
         *
         */
        private static final long serialVersionUID = 1L;


        private JsonObject jsonObject = null;
        private String jsonString = "";

        @Override
        public String execute() throws IOException {

                String URL =
"http://inf.example.com/list/json/93/20/0.html";
                System.out.printf("%s Requeted URL is %s",
this.getClass().getName(), URL);

                StringBuilder sb = new StringBuilder();
                URLConnection urlConn = null;
                InputStreamReader in = null;
                try {
                        URL url = new URL(URL);
                        urlConn = url.openConnection();
                        if (urlConn != null)
                                urlConn.setReadTimeout(60 *
1000);
                        if (urlConn != null &&
urlConn.getInputStream() != null) {
                                in = new
InputStreamReader(urlConn.getInputStream(),
Charset.defaultCharset());
                                BufferedReader bufferedReader
= new BufferedReader(in);
                                if (bufferedReader != null) {
                                        int cp;
                                        while ((cp =
bufferedReader.read()) != -1) {

sb.append((char) cp);
                                        }

bufferedReader.close();
                                }
                        }
                        in.close();
```

```
                        jsonString = sb.toString();

                        System.out.println(jsonString);

                        JsonReader reader =
Json.createReader(new StringReader(jsonString));

                        JsonObject jsonObject =
reader.readObject();
                        this.setJsonObject(jsonObject);

                        reader.close();

                        //
System.out.println(jsonObject.size());

                        /*for (int i = 0; i <
jsonObject.size() - 2; i++) {
                                JsonObject rowObject =
jsonObject.getJsonObject(Integer.toString(i));
                                //
System.out.println(rowObject.toString());

System.out.printf("%s\t%s\t%s\n",
rowObject.getJsonString("id"),
rowObject.getJsonString("title"),

rowObject.getJsonString("ctime"));
                        }*/



                } catch (Exception e) {
                        throw new RuntimeException("Exception
while calling URL:" + URL, e);
                }

                return Action.SUCCESS;
        }

        public JsonObject getJsonObject() {
                return jsonObject;
        }

        public void setJsonObject(JsonObject jsonObject) {
```

```
                this.jsonObject = jsonObject;
        }

        public String getJsonString() {
                return jsonString;
        }

        public void setJsonString(String jsonString) {
                this.jsonString = jsonString;
        }
}
```

JSP 文件

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
        pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>

<table>
        <c:forEach items="${jsonObject.entrySet()}"
var="json"
                varStatus="status">
                <c:if test="${not status.last}">
                        <tr>
                                <td>${json.key+1}</td>

<td>${json.value.getJsonString("id")}</td>

<td>${json.value.getJsonString("title")}</td>

<td>${json.value.getJsonString("ctime")}</td>
                        </tr>
                </c:if>
        </c:forEach>
</table>
==================
```

```
<c:forEach items="${jsonObject.entrySet()}" var="json">
      ${json.value.toString()}
</c:forEach>

==========

<s:property value="jsonString" />
```

解决双引号问题

```
<%@ page language="java" contentType="text/html; charset=UTF-
8"
        pageEncoding="UTF-8"%>

<%@ taglib prefix="s" uri="/struts-tags"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions"
prefix="fn"%>

<table>
        <c:forEach items="${jsonObject.entrySet()}"
var="json"
                varStatus="status">
                <c:if test="${not status.last}">

                        <c:set var="id"

value="${fn:replace(json.value.getJsonString('id'),'\"',
'')}" />
                        <c:set var="title"
value="${fn:replace(json.value.getJsonString('title'),'\"',
'')}" />
                        <c:set var="ctime"

value="${fn:replace(json.value.getJsonString('ctime'),'\"',
'')}" />
                        <tr>
```

```
                                    <td><c:out
value="${status.count}" /></td>
                                    <td><a href="/inf/Detail.do?
id=<c:out value="${id}"/>"><c:out

value="${title}" /></a></td>
                                    <td><c:out value="${ctime}"
/></td>
                            </tr>
                    </c:if>
            </c:forEach>
</table>

<c:set var="pages"
value="${jsonObject.getJsonObject('pages')}" />


<a href="${pages.first}">首页</a>
<a href="${pages.before}">上一页</a>
<a href="${pages.next}">下一页</a>
<a href="${pages.last}">尾页</a>

<span>Count: ${pages.count}, Total: ${pages.total}</span>
```

5.1. 禁止方法

```
        @JSON(serialize = false)
        public String getDatas() {
                return datas;
        }
```

使用 excludeProperties 在 Action 中排除

```
        <action name="withdraw"
class="cn.netkiller.api.action.Report" method="getHistory">
```

```
                <interceptor-ref name="defaultStack" />
                <interceptor-ref name="json">
                        <param name="enableSMD">true</param>
                </interceptor-ref>
                <result name="success" type="json">
                        <param name="enableGZIP">true</param>
                        <param
name="excludeProperties">.*direction</param>
                </result>
        </action>
```

5.2. 格式化日期

```
@JSON(format="yyyy-MM-dd")
public Date getDate(){
        return date;
}
```

5.3. 重命名变量名

```
@JSON(name="mypassword")
public String getPassword() {
        return password;
}
```

5.4. org.apache.struts2.json

```
JSONUtil.serialize(sb.toString());
JSONUtil.deserialize(sb.toString());
```

6. Interceptor

6.1. Session

在 web.xml 文件中定义 Session 超时时间

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

创建拦截器程序

```
package cn.netkiller.interceptor;

import java.util.Map;
import java.lang.Override;

import com.opensymphony.xwork2.ActionInvocation;
import
com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class SessionInterceptor extends AbstractInterceptor {

        private static final long serialVersionUID =
8347994918002285514L;

        @Override
        public String intercept(ActionInvocation invocation)
throws Exception {
                Map<String, Object> session =
invocation.getInvocationContext().getSession();
                if (session.isEmpty())
                        return "nosession"; // session is
empty/expired
```

```
                return invocation.invoke();
        }
}
```

配置拦截器

```
        <package name="mobile" extends="main"
namespace="/mobile">
                <global-results>
                        <result name="nosession"
type="redirectAction">
                                <param
name="actionName">Index</param>
                                <param
name="namespace">/mobile</param>
                        </result>
                </global-results>
                <interceptor name="session"
class="cn.netkiller.SessionInterceptor" />
                <interceptor-stack
name="sessionExpirayStack">
                <interceptor-ref name="defaultStack"/>
                <interceptor-ref name="session"/>
                </interceptor-stack>
                <default-interceptor-ref
name="sessionExpirayStack" />

                <action name="testAction" class="TestClass">
                <interceptor-ref name="sessionExpirayStack"
/>
                <result name="success">success.jsp</result>
                <result name="error">error.jsp</result>
                </action>
        </package>
```

7. Action 中使用线程

背景，在Action中发送邮件，阻塞程序继续执行并返回500，使用Thread 实现异步发送，因为我们并不关心邮件是否到达,只需正常发送即可。

```
        public String execute(){

                ...
                ...

                        try {
                                // Send email

                                Thread sendmail = new
Thread(new Runnable() {
                                        @Override
                                        public void run() {
                                                try {

log.info("sendEmail Begin");

sender.sendEmail(form.getEmail(), form.getText());

log.info("sendEmail End");
                                                } catch
(Exception e) {

e.printStackTrace();
                                                }
                                        }
                                });
                                sendmail.setName("sendmail" +
sendmail.getId() + "logingName:" + form.getLoginname());
                                sendmail.start();

                        } catch (Exception e) {
                                e.printStackTrace();
                                log.info("sendEmail Error");
```

```
                        }

                        ...
                        ...

                        log.info("CreateTrialAccount:" +
form.toString());
                        return Action.SUCCESS;
        }
```

8. 日志

WEB-INF/classes/log4j.properties

```
## # Struts2
log4j.logger.freemarker=INFO
log4j.logger.org.apache.struts2=DEBUG
log4j.logger.org.apache.struts2.components=INFO
log4j.logger.org.apache.struts2.dispatcher=INFO
log4j.logger.org.apache.struts2.convention=INFO
log4j.logger.org.apache.struts2.util=DEBUG
log4j.logger.org.apache.tiles.access.TilesAccess=DEBUG
log4j.logger.org.apache.struts2.interceptor=DEBUG

## # OpenSymphony Stuff
log4j.logger.com.opensymphony=INFO
log4j.logger.com.opensymphony.xwork2.ognl=INFO
log4j.logger.com.opensymphony.xwork2.util=INFO
```

9. FAQ

9.1. Struts 怎样判断用户来自电脑还是移动设备

```
package cn.netkiller.action;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import javax.servlet.http.HttpServletRequest;

public class Index extends ActionSupport {

        private static final long serialVersionUID =
7782483098148890753L;

        public String execute() {
                HttpServletRequest request =
(HttpServletRequest)
ActionContext.getContext().get(org.apache.struts2.StrutsStati
cs.HTTP_REQUEST);
                String userAgent = request.getHeader("User-
Agent").toLowerCase();
                System.out.println(this.getClass().getName()
+ ": " + userAgent);
                String[] mobileAgents = { "iphone",
"android", "phone", "mobile", "wap", "netfront", "java",
"opera mobi", "opera mini", "ucweb", "windows ce", "symbian",
"series", "webos", "sony", "blackberry", "dopod", "nokia",
"samsung", "palmsource", "xda", "pieplus", "meizu", "midp",
"cldc", "motorola", "foma", "docomo", "up.browser",
"up.link", "blazer", "helio", "hosin", "huawei", "novarra",
"coolpad", "webos", "techfaith", "palmsource", "alcatel",
"amoi", "ktouch", "nexian", "ericsson", "philips", "sagem",
"wellcom", "bunjalloo", "maui", "smartphone", "iemobile",
"spice", "bird", "zte-", "longcos", "pantech", "gionee",
"portalmmm", "jig browser", "hiptop", "benq", "haier",
"^lct", "320x320", "240x320", "176x220", "w3c ", "acs-",
"alav", "alca", "amoi", "audi", "avan", "benq", "bird",
"blac", "blaz", "brew", "cell", "cldc", "cmd-", "dang",
```

```
"doco", "eric", "hipt", "inno", "ipaq", "java", "jigs",
"kddi", "keji", "leno", "lg-c", "lg-d", "lg-g", "lge-",
"maui", "maxo", "midp", "mits", "mmef", "mobi", "mot-",
                                "moto", "mwbp", "nec-",
"newt", "noki", "oper", "palm", "pana", "pant", "phil",
"play", "port", "prox", "qwap", "sage", "sams", "sany", "sch-
", "sec-", "send", "seri", "sgh-", "shar", "sie-", "siem",
"smal", "smar", "sony", "sph-", "symb", "t-mo", "teli", "tim-
", "tsm-", "upg1", "upsi", "vk-v", "voda", "wap-", "wapa",
"wapi", "wapp", "wapr", "webc", "winw", "winw", "xda", "xda-
", "Googlebot-Mobile" };
                if (userAgent != null) {
                        for (String mobileAgent :
mobileAgents) {
                                if
(userAgent.indexOf(mobileAgent) >= 0) {
                                        return "mobile";
                                }
                        }
                }

                return "computer";
        }
}
```

```
        <action name="Index"
class="cn.netkiller.action.Index">
                <result name="computer"
type="redirectAction">
                        <param
name="actionName">index</param>
                        <param
name="namespace">/computer</param>
                </result>
                <result name="mobile" type="redirect">
                        <param
name="location">http://m.netkiller.cn/index.html</param>
                </result>
        </action>
```

第 16 章 Apache Tiles

1. 配置 Tiles

1.1. Maven

```
    <dependency>
      <groupId>org.apache.struts</groupId>
          <artifactId>struts-tiles</artifactId>
      <version>1.3.10</version>
    </dependency>
```

1.2. web.xml

Tiles 有两种配置方法，选择一种最适合你的方式添加到 web.xml 文件即可

第一种方式 Tiles servlet

```
<servlet>
    <servlet-name>tiles</servlet-name>
    <servlet-
class>org.apache.tiles.web.startup.TilesServlet</servlet-class>
    ...
    <load-on-startup>2</load-on-startup>
</servlet>
```

第二种方式 Tiles listener

```
<listener>
    <listener-
class>org.apache.tiles.web.startup.TilesListener</listener-
class>
</listener>
```

2. Template 配置模板

WEB-INF/tiles.xml

```
<tiles-definitions>
        <definition name="index" path="/WEB-INF/jsp/index.jsp">
            <put name="title"  value="Tiles Example" />
            <put name="header" value="/WEB-INF/jsp/header.jsp"
/>
            <put name="menu"   value="/WEB-INF/jsp/menu.jsp" />
            <put name="body"   value="/WEB-INF/jsp/body.jsp" />
            <put name="footer" value="/WEB-INF/jsp/footer.jsp"
/>
        </definition>
        <definition name="template_mobile" template="/WEB-
INF/jsp/mobile/template.jsp">
                <put-attribute name="header" value="/WEB-
INF/jsp/mobile/header.jsp" />
                <put-attribute name="content" value="" />
                <put-attribute name="footer" value="/WEB-
INF/jsp/mobile/footer.jsp" />
        </definition>
</tiles-definitions>
```

/WEB-INF/jsp/mobile/template.jsp

```
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%@taglib prefix="tiles" uri="http://tiles.apache.org/tags-
tiles" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
        <title>www.netkiller.cn</title>
</head>
<body>
```

```
<tiles:insertAttribute name="header" ignore="true" />

<tiles:insertAttribute name="content" ignore="true" />

<tiles:insertAttribute name="footer" ignore="true" />

</body>
</html>
```

3. Struts tiles

/WEB-INF/tiles.xml

```
        <definition name="Index" extends="template_mobile">
                <put-attribute name="content" value="/WEB-
INF/jsp/mobile/Index.jsp" />
        </definition>
```

src/struts.xml

```
<struts>
        <package name="mobile" extends="main"
namespace="/mobile">
                <action name="Index"
class="cn.netkiller.mobile.action.Index" method="execute">
                        <result name="success"
type="tiles">Index</result>
                </action>
        </package>
</struts>
```

第 17 章 Play

https://www.playframework.com/

安装Play

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/lang/java/f
ramework/play.sh | bash
```

或者

```
cd /usr/local/src/

wget http://downloads.typesafe.com/typesafe-
activator/1.3.2/typesafe-activator-1.3.2-minimal.zip

unzip typesafe-activator-1.3.2-minimal.zip

mv activator-1.3.2-minimal /srv/
ln -s activator-1.3.2-minimal /srv/activator

cat >> /etc/profile.d/activator.sh <<'EOF'
export PATH=$PATH:/srv/activator
EOF
```

首次运行会下载所需的包

```
# activator
Getting com.typesafe.activator activator-launcher 1.3.2 ...
downloading https://repo.typesafe.com/typesafe/ivy-
releases/com.typesafe.activator/activator-
launcher/1.3.2/jars/activator-launcher.jar ...
```

```
        [SUCCESSFUL ] com.typesafe.activator#activator-
launcher;1.3.2!activator-launcher.jar (3004ms)
downloading https://repo1.maven.org/maven2/org/scala-
lang/scala-library/2.11.5/scala-library-2.11.5.jar ...
        [SUCCESSFUL ] org.scala-lang#scala-
library;2.11.5!scala-library.jar (49602ms)
downloading https://repo.typesafe.com/typesafe/ivy-
releases/com.typesafe.activator/activator-
props/1.3.2/jars/activator-props.jar ...
        [SUCCESSFUL ] com.typesafe.activator#activator-
props;1.3.2!activator-props.jar (2091ms)
downloading https://repo.typesafe.com/typesafe/ivy-
releases/com.typesafe.activator/activator-ui-
common/1.3.2/jars/activator-ui-common.jar ...
        [SUCCESSFUL ] com.typesafe.activator#activator-ui-
common;1.3.2!activator-ui-common.jar (2480ms)
downloading https://repo.typesafe.com/typesafe/ivy-
releases/org.scala-sbt/launcher-interface/0.13.8-
M5/jars/launcher-interface.jar ...
        [SUCCESSFUL ] org.scala-sbt#launcher-interface;0.13.8-
M5!launcher-interface.jar (2489ms)
downloading https://repo.typesafe.com/typesafe/ivy-
releases/org.scala-sbt/completion_2.11/0.13.8-
M5/jars/completion_2.11.jar ...
        [SUCCESSFUL ] org.scala-sbt#completion_2.11;0.13.8-
M5!completion_2.11.jar (6234ms)
downloading https://repo.typesafe.com/typesafe/ivy-
releases/com.typesafe.activator/activator-templates-cache/1.0-
a0afb008ea619bf9d87dc010156cddffa8a6f880/jars/activator-
templates-cache.jar ...
        [SUCCESSFUL ] com.typesafe.activator#activator-
templates-cache;1.0-
a0afb008ea619bf9d87dc010156cddffa8a6f880!activator-templates-
cache.jar (8632ms)
downloading https://repo.typesafe.com/typesafe/ivy-
releases/com.typesafe.activator/activator-common/1.0-
a0afb008ea619bf9d87dc010156cddffa8a6f880/jars/activator-
common.jar ...
        [SUCCESSFUL ] com.typesafe.activator#activator-
common;1.0-a0afb008ea619bf9d87dc010156cddffa8a6f880!activator-
common.jar (23184ms)
downloading https://repo1.maven.org/maven2/org/scala-
lang/modules/scala-xml_2.11/1.0.1/scala-xml_2.11-1.0.1.jar ...
        [SUCCESSFUL ] org.scala-lang.modules#scala-
xml_2.11;1.0.1!scala-xml_2.11.jar(bundle) (8692ms)
```

```
downloading https://repo1.maven.org/maven2/org/scala-
lang/modules/scala-parser-combinators_2.11/1.0.1/scala-parser-
combinators_2.11-1.0.1.jar ...
        [SUCCESSFUL ] org.scala-lang.modules#scala-parser-
combinators_2.11;1.0.1!scala-parser-
combinators_2.11.jar(bundle) (5927ms)
downloading
https://repo1.maven.org/maven2/org/apache/lucene/lucene-
core/4.3.0/lucene-core-4.3.0.jar ...
        [SUCCESSFUL ] org.apache.lucene#lucene-
core;4.3.0!lucene-core.jar (22313ms)
downloading
https://repo1.maven.org/maven2/org/apache/lucene/lucene-
analyzers-common/4.3.0/lucene-analyzers-common-4.3.0.jar ...
        [SUCCESSFUL ] org.apache.lucene#lucene-analyzers-
common;4.3.0!lucene-analyzers-common.jar (12576ms)
downloading
https://repo1.maven.org/maven2/org/apache/lucene/lucene-
queryparser/4.3.0/lucene-queryparser-4.3.0.jar ...
        [SUCCESSFUL ] org.apache.lucene#lucene-
queryparser;4.3.0!lucene-queryparser.jar (6739ms)
downloading
https://repo1.maven.org/maven2/com/typesafe/akka/akka-
actor_2.11/2.3.9/akka-actor_2.11-2.3.9.jar ...
        [SUCCESSFUL ] com.typesafe.akka#akka-
actor_2.11;2.3.9!akka-actor_2.11.jar (14051ms)
downloading https://repo1.maven.org/maven2/com/amazonaws/aws-
java-sdk/1.3.29/aws-java-sdk-1.3.29.jar ...
```

第 18 章 Log

1. Logback

http://logback.qos.ch/index.html

Logback 是 log4j 作者开发，目前的趋势Log4j逐步被Logback取代。

1.1. Maven 包

```
                <!--
https://mvnrepository.com/artifact/org.slf4j/slf4j-api -->
                <dependency>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-api</artifactId>
                        <version>1.7.25</version>
                </dependency>
                <!--
https://mvnrepository.com/artifact/ch.qos.logback/logback-
core -->
                <dependency>
                        <groupId>ch.qos.logback</groupId>
                        <artifactId>logback-core</artifactId>
                        <version>1.2.3</version>
                </dependency>
                <dependency>
                        <groupId>ch.qos.logback</groupId>
                        <artifactId>logback-
access</artifactId>
                        <version>1.2.3</version>
                </dependency>
                <!--
https://mvnrepository.com/artifact/ch.qos.logback/logback-
classic -->
                <dependency>
```

```
                        <groupId>ch.qos.logback</groupId>
                        <artifactId>logback-
classic</artifactId>
                        <version>1.2.3</version>
                </dependency>
```

1.2. Example

```
package com.logs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class MyApp {
        final static Logger logger =
LoggerFactory.getLogger("MyApp.class");
        public static void main(String[] args) {

                logger.trace("trace");
                logger.debug("debug str");
                logger.info("info str");
                logger.warn("warn");
                logger.error("error");
        }
}
```

2. slf4j

http://www.slf4j.org/

log4j 已经被 Logback

3. log4j

3.1. 安装 Log4j

手工安装

log4j

http://logging.apache.org/

```
wget http://government-
grants.org/mirrors/apache.org/logging/log4j/1.2.14/logging-
log4j-1.2.14.tar.gz
tar zxvf logging-log4j-1.2.14.tar.gz
cd logging-log4j-1.2.14
cp dist/lib/log4j-1.2.14.jar /srv/java/lib
```

Maven

```
        <dependencies>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-api</artifactId>
                        <version>2.5</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-core</artifactId>
                        <version>2.5</version>
                </dependency>
        </dependencies>
```

3.2. log4j 环境变量

${catalina.home}

```
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=${catalina.home}/logs/logs_tomcat.log
log4j.appender.R.MaxFileSize=10KB
```

3.3. Log4j Example

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
        <Appenders>
                <Console name="Console" target="SYSTEM_OUT">
                        <PatternLayout pattern="%d{yyyy-MM-dd
HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n" />
                </Console>
        </Appenders>
        <Loggers>
                <Logger name="cn.netkiller.Logging"
level="trace">
                        <AppenderRef ref="Console" />
                </Logger>
                <Logger name="cn.netkiller" level="debug">
                        <AppenderRef ref="Console" />
                </Logger>
                <Root level="error">
                        <AppenderRef ref="Console" />
                </Root>
        </Loggers>
</Configuration>
```

```
package cn.netkiller;
```

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class Logging {
        private static final Logger logger =
LogManager.getLogger("appender");

        public void application() {

                String parameter = "sssssssssssss";
                if (logger.isDebugEnabled()) {
                        logger.debug("This is debug : " +
parameter);
                }

                if (logger.isInfoEnabled()) {
                        logger.info("This is info : " +
parameter);
                }

                logger.trace("trace");
                logger.debug("debug");
                logger.info("info");
                logger.warn("warn");
                logger.error("error");
                logger.fatal("fatal");

        }

        public static void main(String[] args) {
                Logging log = new Logging();
                log.application();

        }
}
```

3.4. log4j.properties

stdout 标准输出

```
# Root logger option
log4j.rootLogger=INFO, stdout

# Direct log messages to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd
HH:mm:ss} %-5p %c{1}:%L - %m%n
```

第 19 章 JSON (JavaScript Object Notation)

1. javax.json.*

1.1. Json 编码

```
package netkiller.json;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.json.*;

public final class Writer {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                JsonObjectBuilder jsonBuilder =
Json.createObjectBuilder();
                JsonObjectBuilder addressBuilder =
Json.createObjectBuilder();
                JsonArrayBuilder phoneNumBuilder =
Json.createArrayBuilder();

                phoneNumBuilder.add("12355566688").add("0755-
2222-3333");

                addressBuilder.add("street",
"Longhua").add("city",  "Shenzhen").add("zipcode", 518000);

                jsonBuilder.add("nickname",
"netkiller").add("name", "Neo").add("department",
```

```
"IT").add("role",       "Admin");

                jsonBuilder.add("phone", phoneNumBuilder);
                jsonBuilder.add("address", addressBuilder);

                JsonObject jsonObject = jsonBuilder.build();

                System.out.println(jsonObject);

                try {
                        // write to file
                        File file = new File("json.txt");
                        if (!file.exists()) {
                                file.createNewFile();
                        }
                        OutputStream os = null;
                        os = new FileOutputStream(file);
                        JsonWriter jsonWriter =
Json.createWriter(os);
                        jsonWriter.writeObject(jsonObject);
                        jsonWriter.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

}
```

运行后输出

```
{"nickname":"netkiller","name":"Neo","department":"IT","role":"
Admin","phone":["12355566688","0755-2222-3333"],"address":
{"street":"Longhua","city":"Shenzhen","zipcode":"518000"}}
```

1.2. Json 解码

```
package netkiller.json;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import javax.json.Json;
import javax.json.JsonArray;
import javax.json.JsonObject;
import javax.json.JsonReader;
import javax.json.JsonValue;

public final class Reader {

        public static final String JSON_FILE="json.txt";

        public static void main(String[] args) throws
IOException {
                InputStream fis = new
FileInputStream(JSON_FILE);
                //create JsonReader object
                JsonReader jsonReader =
Json.createReader(fis);

                //get JsonObject from JsonReader
                JsonObject jsonObject =
jsonReader.readObject();

                //we can close IO resource and JsonReader now
                jsonReader.close();
                fis.close();

                System.out.printf("nickname: %s \n",
jsonObject.getString("nickname"));
                System.out.printf("name: %s \n",
jsonObject.getString("name"));
                System.out.printf("department: %s \n",
                jsonObject.getString("department"));
                System.out.printf("role: %s \n",
jsonObject.getString("role"));
                JsonArray jsonArray =
jsonObject.getJsonArray("phone");
```

```
                //long[] numbers = new
long[jsonArray.size()];
                int index = 0;
                for(JsonValue value : jsonArray){
                        //numbers[index++] =
Long.parseLong(value.toString());
                        System.out.printf("phone[%d]: %s \n",
index++, value.toString());
                }

                //reading inner object from json object
                JsonObject innerJsonObject =
jsonObject.getJsonObject("address");

                System.out.printf("address: %s, %s, %d \n",
                innerJsonObject.getString("street"),
                innerJsonObject.getString("city"),
                innerJsonObject.getInt("zipcode"));

        }

}
```

运行结果

```
nickname: netkiller
name: Neo
department: IT
role: Admin
phone[0]: +8612355566688
phone[1]: 0755-2222-3333
address: Longhua, Shenzhen, 518000
```

1.3. URL获取Json

```
package netkiller.json;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.StringReader;
import java.net.URL;
import java.net.URLConnection;
import java.nio.charset.Charset;

import javax.json.*;

public class HttpUrl {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                String URL =
"http://www.example.com/json/2/20/0.html";
                // system.out.println("Requeted URL:" + URL);
                StringBuilder sb = new StringBuilder();
                URLConnection urlConn = null;
                InputStreamReader in = null;
                try {
                        URL url = new URL(URL);
                        urlConn = url.openConnection();
                        if (urlConn != null)
                                urlConn.setReadTimeout(60 *
1000);
                        if (urlConn != null &&
urlConn.getInputStream() != null) {
                                in = new
InputStreamReader(urlConn.getInputStream(),
Charset.defaultCharset());
                                BufferedReader bufferedReader
= new BufferedReader(in);
                                if (bufferedReader != null) {
                                        int cp;
                                        while ((cp =
bufferedReader.read()) != -1) {

sb.append((char) cp);
                                        }
```

```
bufferedReader.close();
                                }
                        }
                        in.close();

                        String jsonString = sb.toString();

                        //System.out.println(jsonString);

                        JsonReader reader =
Json.createReader(new StringReader(jsonString));

                        JsonObject jsonObject =
reader.readObject();

                        reader.close();

                        //
System.out.println(jsonObject.size());

                        for (int i = 0; i < jsonObject.size()
- 2; i++) {
                                JsonObject rowObject =
jsonObject.getJsonObject(Integer.toString(i));
                                //
System.out.println(rowObject.toString());

System.out.printf("%s\t%s\t%s\n",
rowObject.getJsonString("id"),
rowObject.getJsonString("title"),

rowObject.getJsonString("ctime"));
                        }

                } catch (Exception e) {
                        throw new RuntimeException("Exception
while calling URL:" + URL, e);
                }

        }

}
```

2. Jackson

2.1. ObjectToJSON

```
package cn.netkiller;
import java.util.Date;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;

public class ObjectToJSON {
  public static void main(String[] args) throws
JsonProcessingException {
     Test test = new Test("Neo", 123, new Date());
     ObjectMapper mapper = new ObjectMapper();
     String jsonData =
mapper.writerWithDefaultPrettyPrinter()
                  .writeValueAsString(test);
     System.out.println(jsonData);
  }
}
```

2.2. JSONToObject

```
package cn.netkiller;
import java.io.IOException;
import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;

public class JSONToObject {
        public static void main(String[] args) throws
JsonParseException, JsonMappingException, IOException {
                  String jsonData =
                     "{"
```

```
                            +"\"name\" : \"Neo\","
                            +"\"age\" : 12,"
                            +"\"birthDate\" : \"2019-Jun-11
07:00:46 CST\""
                      +"}";
                    ObjectMapper mapper = new ObjectMapper();
                    Test test = mapper.readValue(jsonData,
Test.class);
                    System.out.println(test.getName()+" |
"+test.getBirthDate()+" | "+ test.getAge());
            }
}
```

2.3. JsonNode

```
String[] picture;
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.convertValue(picture,
JsonNode.class);
```

3. org.json

3.1. JSONArray forEach

```
            JSONArray playlists =
object.getJSONArray("result");
            Log.i(TAG, "播放数量：" + playlists.size() + " 列表"
+ playlists);

            List<String> lists = new ArrayList<String>();

            playlists.forEach(jsonObject -> {
                JSONObject obj = (JSONObject) jsonObject;
                lists.add(obj.getString("url"));
            });
```

4. Google Json

4.1. json 转 map

```
                HashMap<String, HashMap<String, String>> data
= new HashMap<String, HashMap<String, String>>();
        Gson gson = new Gson();
        data = gson.fromJson(jsonString, data.getClass());
```

4.2. LinkedHashMap 转 Json

```
Gson gson = new Gson();
Map<String, String> data = new LinkedHashMap<String, String>
() {{
    put("text", text);
    put("scene", "talk");
}};

String json = gson.toJson(data, LinkedHashMap.class);
```

注意：String json = gson.toJson(data); 这样转换不成功，返回 null，必须指定 class 才能成功 String json = gson.toJson(data, LinkedHashMap.class);

第 20 章 AMQP(Advanced Message Queuing Protocol)

1. Send and Recv

Send

```
package cn.netkiller;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

public class Send {

        private final static String QUEUE_NAME = "hello";

        public static void main(String[] argv) throws Exception
{
                ConnectionFactory factory = new
ConnectionFactory();
                factory.setHost("192.168.6.1");
                Connection connection =
factory.newConnection();
                Channel channel = connection.createChannel();

                channel.queueDeclare(QUEUE_NAME, false, false,
false, null);
                String message = "Hello World!";
                channel.basicPublish("", QUEUE_NAME, null,
message.getBytes("UTF-8"));
                System.out.println(" [x] Sent '" + message +
"'");

                channel.close();
                connection.close();
        }
```

```
}
```

Recv.java

```
package cn.netkiller;

import com.rabbitmq.client.*;
import java.io.IOException;

public class Recv {

        private final static String QUEUE_NAME = "hello";

        public static void main(String[] argv) throws Exception
{
                ConnectionFactory factory = new
ConnectionFactory();
                factory.setHost("192.168.6.1");
                Connection connection =
factory.newConnection();
                Channel channel = connection.createChannel();

                channel.queueDeclare(QUEUE_NAME, false, false,
false, null);
                System.out.println(" [*] Waiting for messages.
To exit press CTRL+C");

                Consumer consumer = new
DefaultConsumer(channel) {
                        @Override
                        public void handleDelivery(String
consumerTag, Envelope envelope, AMQP.BasicProperties
properties,
                                                byte[] body) throws
IOException {
                                String message = new
String(body, "UTF-8");
                                System.out.println(" [x]
Received '" + message + "'");
                        }
                };
                channel.basicConsume(QUEUE_NAME, true,
```

```
consumer);
        }
}
```

2. direct

Send.java

```
package cn.netkiller;

import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

public class DirectPush {

	private static final String EXCHANGE_NAME =
"cn.netkiller";

	public static void main(String[] args) throws Exception
{
		try {
			ConnectionFactory factory = new
ConnectionFactory();
			factory.setHost("192.168.6.1");
			Connection connection =
factory.newConnection();
			Channel channel =
connection.createChannel();
			channel.exchangeDeclare(EXCHANGE_NAME,
"direct");

			String routingKey = "demo";
			String message = "Hello";

			channel.basicPublish(EXCHANGE_NAME,
routingKey, null, message.getBytes());
			System.out.println(" [x] Sent '" +
routingKey + "':'" + message + "'");

			channel.close();
			connection.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
```

```
        }

}
```

Recv.java

```
package cn.netkiller;

import com.rabbitmq.client.*;

import java.io.IOException;

public class DirectReceive {

        private static final String EXCHANGE_NAME =
"cn.netkiller";
        //private final static String QUEUE_NAME = "customer";

        public static void main(String[] args) {
                try {
                        ConnectionFactory factory = new
ConnectionFactory();
                        factory.setHost("192.168.6.1");
                        Connection connection =
factory.newConnection();
                        Channel channel =
connection.createChannel();
                        channel.exchangeDeclare(EXCHANGE_NAME,
"direct");

                        String queueName =
channel.queueDeclare().getQueue();
                        System.out.println(queueName);
                        channel.queueBind(queueName,
EXCHANGE_NAME, "demo");
                        channel.queueBind(queueName,
EXCHANGE_NAME, "real");

                        System.out.println(" [*] Waiting for
messages. To exit press CTRL+C");
```

```
                        Consumer consumer = new
DefaultConsumer(channel) {
                                @Override
                                public void
handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties,
                                                        byte[] body)
throws IOException {
                                        String message = new
String(body, "UTF-8");
                                        System.out.println("
[x] Received '" + envelope.getRoutingKey() + "':'" + message +
"'");
                                }
                        };
                        channel.basicConsume(queueName, true,
consumer);

                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

}
```

第 21 章 NoSQL

1. MongoDB

1.1. pom.xml

```
<dependency>
        <groupId>org.mongodb</groupId>
        <artifactId>mongodb-driver</artifactId>
        <version>3.2.2</version>
</dependency>
```

1.2. 插入操作

```
package cn.netkiller.controller;

import java.net.UnknownHostException;

import com.mongodb.BasicDBObject;
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.MongoClient;

public final class Tracker {

        public Tracker() {

        }

        public static void main(String[] args) {

                MongoClient mongo = null;
                try {
```

```
                        mongo = new MongoClient("192.168.4.1",
27017);

                        DB db = mongo.getDB("finance");
                        DBCollection table =
db.getCollection("tracker");
                        BasicDBObject document = new
BasicDBObject();
                        document.put("test", "helloworld");
                        table.insert(document);
                } catch (UnknownHostException e) {
                e.printStackTrace();
                }
        }
}
```

1.3. 读取操作

```
```

第 22 章 Elasticsearch API

```
<?xml version="1.0"?>
<project
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>cn.netkiller</groupId>
                <artifactId>example</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </parent>
        <groupId>cn.netkiller</groupId>
        <artifactId>elastic</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>elastic</name>
        <url>http://maven.apache.org</url>
        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>

                <dependency>
                        <groupId>org.elasticsearch</groupId>

<artifactId>elasticsearch</artifactId>
                        <version>5.6.3</version>
                </dependency>
                <dependency>
```

```
<groupId>org.elasticsearch.client</groupId>
                        <artifactId>transport</artifactId>
                        <version>5.6.3</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-api</artifactId>
                        <version>2.8.2</version>
                </dependency>
                <dependency>

<groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-core</artifactId>
                        <version>2.8.2</version>
                </dependency>

        </dependencies>
</project>
```

1. Client

```
Settings settings = Settings.builder()
                                            .put("cluster.name",
"elasticsearch") //集群名称

.put("client.transport.sniff", true) //自动嗅探

.put("discovery.type", "zen")

.put("discovery.zen.minimum_master_nodes", 1)

.put("discovery.zen.ping_timeout", "500ms")

.put("discovery.initial_state_timeout", "500ms")
                                            .build();
Client client = new PreBuiltTransportClient(settings)
```

```
.addTransportAddress(new
InetSocketTransportAddress(InetAddress.getByName(ip), 9300));
```

2. insert

```
import org.elasticsearch.action.index.IndexResponse;
import org.elasticsearch.client.Client;

IndexResponse response = client.prepareIndex("information",
"news", "1")
.setSource("{ \"title\": \"ElasticSearch 5.5.2\"}")
.get();
```

使用 HashMap 插入

```
public void createData() {
        Map<String, Object> map = new HashMap<String, Object>
();
        map.put("name", "Neo Chen");
        map.put("age", 30);
        map.put("book", new String[]{"Netkiller Linux 手
札","Netkiller Java 手札"});
        map.put("website", "http://www.netkiller.cn");
        map.put("email", "netkiller@msn.com");

        IndexResponse response = client.prepareIndex("book",
"author", UUID.randomUUID().toString())
                        .setSource(map).get();
        System.out.println("Status: " +
response.status().getStatus() + "! id=" + response.getId());
}
```

3. Get

```
        @Autowired
        private TransportClient client;

        @RequestMapping(value = "/article/{articleId}")
        public GetResponse read(@PathVariable String
articleId) {
                GetResponse response =
client.prepareGet("information", "article", articleId).get();
                return response;
        }
```

4. delete

```
import org.elasticsearch.action.delete.DeleteResponse;
import org.elasticsearch.client.Client;

DeleteResponse response = client.prepareDelete("book",
"computer", "1")
.get();
```

5. Search

```
        @RequestMapping(value = "/article/list")
        public List<Map<String, Object>> list() {
                List<Map<String, Object>> list = new
ArrayList<Map<String, Object>>();
                SearchResponse response =
client.prepareSearch("information").setTypes("article")

.setSearchType(SearchType.DFS_QUERY_THEN_FETCH).addSort("ctim
e", SortOrder.DESC)

.setFrom(0).setSize(60).setExplain(true).get();
                for (final SearchHit hit :
response.getHits().getHits()) {

System.out.println(hit.getSourceAsString());

hit.getSourceAsMap().remove("content");
                        list.add(hit.getSourceAsMap());
                }
                return list;
        }
```

6. Query 查询

6.1. match all 匹配所有数据

```
SearchResponse response = client.prepareSearch()
        .setIndices(index)
        .setTypes(type)
        .setSearchType(SearchType.QUERY_AND_FETCH)
        .setFetchSource(new String[]{"title"}, null)
        .setQuery(QueryBuilders.matchAllQuery())
        .setSize(10).execute().actionGet();
```

6.2. match 匹配查询

```
public void match() {
        SearchRequestBuilder requestBuilder =
client.prepareSearch("company").setTypes("employee")

.setQuery(QueryBuilders.matchQuery("name", "neo"));
        System.out.println(requestBuilder.toString());

        SearchResponse response = requestBuilder.get();

        System.out.println(response.status());
        if (response.status().getStatus() == 200) {
                for (SearchHit hits :
response.getHits().getHits()) {

System.out.println(hits.getSourceAsString());
                }
        }
}
```

6.3. match phrase 短语精准匹配

```
public void matchPhrase() {
        SearchRequestBuilder requestBuilder =
client.prepareSearch("company").setTypes("employee")

.setQuery(QueryBuilders.matchPhraseQuery("name", "neo"));

        SearchResponse response = requestBuilder.get();

        if (response.status().getStatus() == 200) {
                for (SearchHit hits :
response.getHits().getHits()) {

System.out.println(hits.getSourceAsString());
                }
        }
}
```

7. Filter 过滤

```
FilterBuilder filterBuilder = FilterBuilders.andFilter(

FilterBuilders.existsFilter("title").filterName("exist"),
        FilterBuilders.termFilter("title", "elastic")
);

SearchResponse response = client.prepareSearch("netkiller")
        .setPostFilter(filterBuilder)
        .get();
```

7.1. term

```
.setPostFilter(QueryBuilders.termQuery("site_id", siteId))
```

7.2. range

```
QueryBuilders.rangeQuery("age").from(12).to(18)
```

8. Sorting

```
SearchResponse response = client.prepareSearch("netkiller")
        .setQuery(QueryBuilders.matchAllQuery())
        .addSort(SortBuilders.fieldSort("title"))
        .addSort("_score", SortOrder.DESC)
        .get()
```

9. 返回 Source 字段

下面例子查询返回 "id", "title", "description", "image", "ctime"

```
        /*
         * 范例：/restful/search/article/list/23/0/20.json?
tags=美国
         */
        @RequestMapping(value =
"/article/list/{siteId}/{from}/{size}")
        public List<Map<String, Object>>
listBySiteIdAndTags(@PathVariable String siteId,
@PathVariable int from, @PathVariable int size,
@RequestParam(value = "tags", required = false) String tags)
{
                List<Map<String, Object>> list = new
ArrayList<Map<String, Object>>();

                SearchRequestBuilder searchRequestBuilder =
client.prepareSearch("information").setTypes("article").setSe
archType(SearchType.DFS_QUERY_THEN_FETCH).addSort("ctime",
SortOrder.DESC);

                searchRequestBuilder.setFetchSource(new
String[] { "id", "title", "description", "image", "ctime" },
null);
                if (tags != null && !tags.equals("")) {
                        // logger.info(tags);

searchRequestBuilder.setQuery(QueryBuilders.matchQuery("tags"
, tags));
                }

searchRequestBuilder.setPostFilter(QueryBuilders.termQuery("s
ite_id",
siteId)).setFrom(from).setSize(size).setExplain(true);

                logger.info(searchRequestBuilder.toString());
                SearchResponse response =
searchRequestBuilder.get();
```

```
                for (final SearchHit hit :
response.getHits().getHits()) {
                        //
logger.info(hit.getSourceAsString());

hit.getSourceAsMap().remove("content");
                        list.add(hit.getSourceAsMap());
                }
                logger.info(tags);
                return list;
        }
```

10. Count

```
QueryBuilders qb =
QueryBuilders.boolQuery().must(QueryBuilders.termQuery("statu
s", false));

CountResponse response = client.prepareCount("book") // 索引
                .setQuery(qb) //类型
                .get();
```

11. Example 范例

11.1. Spring boot 案例

```
        /*
         * 范例：/restful/search/article/list/23/0/20.json?
tags=美国
         */
        @RequestMapping(value =
"/article/list/{siteId}/{from}/{size}")
        public List<Map<String, Object>>
listBySiteIdAndTags(@PathVariable String siteId,
@PathVariable int from, @PathVariable int size,
@RequestParam(value = "tags", required = false) String tags)
{
                List<Map<String, Object>> list = new
ArrayList<Map<String, Object>>();

                SearchRequestBuilder searchRequestBuilder =
client.prepareSearch("information").setTypes("article").setSe
archType(SearchType.DFS_QUERY_THEN_FETCH).addSort("ctime",
SortOrder.DESC)
                // .addFields("_source","title",
"description", "ctime")
                ;

                if (tags != null && !tags.equals("")) {
                        // logger.info(tags);

searchRequestBuilder.setQuery(QueryBuilders.matchQuery("tags"
, tags));
                }

searchRequestBuilder.setPostFilter(QueryBuilders.termQuery("s
ite_id",
siteId)).setFrom(from).setSize(size).setExplain(true);

                //
logger.info(searchRequestBuilder.toString());
```

```
                SearchResponse response =
searchRequestBuilder.get();

                for (final SearchHit hit :
response.getHits().getHits()) {
                        //
logger.info(hit.getSourceAsString());

hit.getSourceAsMap().remove("content");
                        list.add(hit.getSourceAsMap());
                }
                logger.info(tags);
                return list;
        }
}
```

12. FAQ

12.1. 显示查询 JSON 字符串

```
logger.info(searchRequestBuilder.toString());
```

```
2017-09-05 14:34:25 14858 [http-nio-8443-exec-1] INFO
c.e.a.restful.SearchRestController : {
  "from" : 0,
  "size" : 2,
  "query" : {
    "match" : {
      "tags" : {
        "query" : "ioi",
        "operator" : "OR",
        "prefix_length" : 0,
        "max_expansions" : 50,
        "fuzzy_transpositions" : true,
        "lenient" : false,
        "zero_terms_query" : "NONE",
        "boost" : 1.0
      }
    }
  },
  "post_filter" : {
    "term" : {
      "site_id" : {
        "value" : "22",
        "boost" : 1.0
      }
    }
  },
  "explain" : true,
  "sort" : [
    {
      "ctime" : {
        "order" : "desc"
      }
```

```
      }
   ]
}
```

第 23 章 Jersey - RESTful Web Services in Java.

https://jersey.java.net/

1. Client 2.x

1.1. Maven 版本

1.x

```
<!--
https://mvnrepository.com/artifact/com.sun.jersey/jersey-
client -->
<dependency>
    <groupId>com.sun.jersey</groupId>
    <artifactId>jersey-client</artifactId>
    <version>1.19.2</version>
</dependency>
```

2.x 版本

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>example</artifactId>
```

```
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>example</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <dependencies>

                <dependency>

<groupId>org.glassfish.jersey.core</groupId>
                        <artifactId>jersey-
client</artifactId>
                        <version>2.25.1</version>
                </dependency>

        </dependencies>
</project>
```

两个版本差异非常大，本文的例子使用2.23.2

1.2. GET 操作

```
package cn.netkiller.jersey;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import org.glassfish.jersey.client.ClientConfig;
```

```
public class JerseyClientGet {

        public static void main(String[] args) {

                ClientConfig clientConfig = new
ClientConfig();

                Client client =
ClientBuilder.newClient(clientConfig);
                WebTarget webTarget =
client.target("http://inf.netkiller.cn").path("/list/json/2.h
tml");

                Invocation.Builder invocationBuilder =
webTarget.request(MediaType.APPLICATION_JSON);
                Response response = invocationBuilder.get();

                System.out.println(response.getStatus());
                System.out.println(response.getStatusInfo());

                if (response.getStatus() == 200) {

                        String output =
response.readEntity(String.class);
                        System.out.println(output);

                }

        }
}
```

1.3. GET + Auth 用户认证

```
package cn.netkiller.jersey;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Invocation;
import javax.ws.rs.client.WebTarget;
```

```
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import org.glassfish.jersey.client.ClientConfig;
import
org.glassfish.jersey.client.authentication.HttpAuthentication
Feature;

public class JerseyClientGetAuth {

        public static void main(String[] args) {

                ClientConfig clientConfig = new
ClientConfig();

                HttpAuthenticationFeature feature =
HttpAuthenticationFeature.basic("neo", "chen");
                clientConfig.register(feature);

                Client client =
ClientBuilder.newClient(clientConfig);
                WebTarget webTarget =
client.target("http://api.netkiller.cn/v1/withdraw/ping.json"
).path("");

                Invocation.Builder invocationBuilder =
webTarget.request(MediaType.APPLICATION_JSON);
                Response response = invocationBuilder.get();

                System.out.println(response.getStatus());
                System.out.println(response.getStatusInfo());

                if (response.getStatus() == 200) {

                        String output =
response.readEntity(String.class);
                        System.out.println(output);

                }

        }
}
```

2. Client 1.x

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>example</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>example</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <dependencies>

                <!--
https://mvnrepository.com/artifact/com.sun.jersey/jersey-
client -->
                <dependency>
                        <groupId>com.sun.jersey</groupId>
                        <artifactId>jersey-
client</artifactId>
                        <version>1.19.3</version>
                </dependency>

        </dependencies>
</project>
```

```
package cn.netkiller.jersey;

import javax.ws.rs.core.MediaType;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import com.sun.jersey.api.client.filter.HTTPBasicAuthFilter;

public class HttpAuth1 {

	public HttpAuth1() {
		// TODO Auto-generated constructor stub
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		try {

			ClientConfig clientConfig = new
DefaultClientConfig();

			Client client =
Client.create(clientConfig);
			client.addFilter(new
HTTPBasicAuthFilter("user", "password"));

			WebResource webResource =
client.resource("http://api.netkiller.cn/v1/config/read.json?
name=cache");

			ClientResponse response =
webResource.accept(MediaType.APPLICATION_JSON).type(MediaType
.APPLICATION_JSON).get(ClientResponse.class);

			if (response.getStatus() != 200) {
				throw new
RuntimeException("Failed : HTTP error code : " +
response.getStatus());
			}
```

```
                        String output =
response.getEntity(String.class);

                        System.out.println("Server response
.... \n");
                        System.out.println(output);

                } catch (Exception e) {

                        e.printStackTrace();

                }

        }

}
```

2.1. Jersey + Auth + HTTP2 + SSL

我的应用场景 Jersey client -> CDN HTTP2 SSL -> api.netkiller.cn (HTTP2 SSL Auth) 下面代码100% 可运行。

```
package cn.netkiller.jersey;

import javax.ws.rs.core.MediaType;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
import com.sun.jersey.api.client.filter.HTTPBasicAuthFilter;

public class HttpAuth1 {

        public HttpAuth1() {
```

```
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                try {

                        ClientConfig clientConfig = new
DefaultClientConfig();
                        Client client =
Client.create(clientConfig);
                        client.addFilter(new
HTTPBasicAuthFilter("user", "password"));

                        WebResource webResource =
client.resource("https://api.netkiller.cn/v1/config/read.json
?name=cache");

                        ClientResponse response =
webResource.accept(MediaType.APPLICATION_JSON).type(MediaType
.APPLICATION_JSON).get(ClientResponse.class);

                        if (response.getStatus() != 200) {
                                throw new
RuntimeException("Failed : HTTP error code : " +
response.getStatus());
                        }

                        String output =
response.getEntity(String.class);

                        System.out.println("Server response
.... \n");
                        System.out.println(output);

                } catch (Exception e) {

                        e.printStackTrace();

                }

        }

}
```

如果SSL证书配置正确将会输出返回内容，如果SSL证书不正确会返回下面错误，请检查你的SSL证书

```
com.sun.jersey.api.client.ClientHandlerException:
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building
failed:
sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
        at
com.sun.jersey.client.urlconnection.URLConnectionClientHandler.
handle(URLConnectionClientHandler.java:155)
        at
com.sun.jersey.api.client.filter.HTTPBasicAuthFilter.handle(HTT
PBasicAuthFilter.java:105)
        at
com.sun.jersey.api.client.Client.handle(Client.java:652)
        at
com.sun.jersey.api.client.WebResource.handle(WebResource.java:6
82)
        at
com.sun.jersey.api.client.WebResource.access$200(WebResource.ja
va:74)
        at
com.sun.jersey.api.client.WebResource$Builder.get(WebResource.j
ava:509)
        at
cn.netkiller.jersey.HttpAuth1.main(HttpAuth1.java:31)
Caused by: javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building
failed:
sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
        at
sun.security.ssl.Alerts.getSSLException(Alerts.java:192)
        at
sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1949)
        at
```

```
sun.security.ssl.Handshaker.fatalSE(Handshaker.java:302)
        at
sun.security.ssl.Handshaker.fatalSE(Handshaker.java:296)
        at
sun.security.ssl.ClientHandshaker.serverCertificate(ClientHands
haker.java:1509)
        at
sun.security.ssl.ClientHandshaker.processMessage(ClientHandshak
er.java:216)
        at
sun.security.ssl.Handshaker.processLoop(Handshaker.java:979)
        at
sun.security.ssl.Handshaker.process_record(Handshaker.java:914)
        at
sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:10
62)
        at
sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocke
tImpl.java:1375)
        at
sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.jav
a:1403)
        at
sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.jav
a:1387)
        at
sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient
.java:559)
        at
sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.c
onnect(AbstractDelegateHttpsURLConnection.java:185)
        at
sun.net.www.protocol.http.HttpURLConnection.getInputStream0(Htt
pURLConnection.java:1513)
        at
sun.net.www.protocol.http.HttpURLConnection.getInputStream(Http
URLConnection.java:1441)
        at
java.net.HttpURLConnection.getResponseCode(HttpURLConnection.ja
va:480)
        at
sun.net.www.protocol.https.HttpsURLConnectionImpl.getResponseCo
de(HttpsURLConnectionImpl.java:338)
        at
com.sun.jersey.client.urlconnection.URLConnectionClientHandler.
```

```
_invoke(URLConnectionClientHandler.java:253)
        at
com.sun.jersey.client.urlconnection.URLConnectionClientHandler.
handle(URLConnectionClientHandler.java:153)
        ... 6 more
Caused by: sun.security.validator.ValidatorException: PKIX path
building failed:
sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
        at
sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java
:387)
        at
sun.security.validator.PKIXValidator.engineValidate(PKIXValidat
or.java:292)
        at
sun.security.validator.Validator.validate(Validator.java:260)
        at
sun.security.ssl.X509TrustManagerImpl.validate(X509TrustManager
Impl.java:324)
        at
sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509TrustMan
agerImpl.java:229)
        at
sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509Tr
ustManagerImpl.java:124)
        at
sun.security.ssl.ClientHandshaker.serverCertificate(ClientHands
haker.java:1491)
        ... 21 more
Caused by:
sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
        at
sun.security.provider.certpath.SunCertPathBuilder.build(SunCert
PathBuilder.java:141)
        at
sun.security.provider.certpath.SunCertPathBuilder.engineBuild(S
unCertPathBuilder.java:126)
        at
java.security.cert.CertPathBuilder.build(CertPathBuilder.java:2
80)
        at
sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java
:382)
```

```
        ... 27 more
```

第 24 章 Apache HttpComponents

1. org.apache.commons.lang3

1.1. HTML 标签处理

```
package cn.netkiller.apache.lang;

import org.apache.commons.lang3.StringEscapeUtils;

@SuppressWarnings("deprecation")
public class LangTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                String html = "<span>Neo's book</span>";
                String encode =
StringEscapeUtils.escapeHtml4(html);
                String decode =
StringEscapeUtils.unescapeHtml4(encode);
                System.out.println(encode);
                System.out.println(decode);

        }

}
```

1.2. StringUtils.join 使用特定字符链接字符串

下面例子使用逗号链接字符串

```
```

```
org.apache.commons.lang.StringUtils.join(arraylist, ',')
```

1.3. RandomStringUtils

```
        String project =
RandomStringUtils.randomAlphanumeric(10);
        System.out.print(project);
```

随机输出 ASCII

```
       System.out.println(RandomStringUtils.randomAscii(10));
```

随机输出数字

```
System.out.println(RandomStringUtils.randomNumeric(10));
```

指定字符串随机输出

```
       String project = RandomStringUtils.random(10,
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ").toStrin
g();
       System.out.println(project);
```

2. commons-text

```
<dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-text</artifactId>
        <version>1.6</version>
</dependency>
```

2.1. 禁止转译 json

```
package cn.netkiller.test;

import org.apache.commons.text.StringEscapeUtils;

public class TestJson {

        public static void main(String[] args) {

System.out.println(StringEscapeUtils.escapeJson("
{\"name\":\"Neo\"}"));
        }

}
```

输出

```
{\"name\":\"Neo\"}
```

3. Apache HttpClient

3.1. Maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>cn.netkiller</groupId>
        <artifactId>example</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
                <!--
https://mvnrepository.com/artifact/org.apache.httpcomponents/
httpclient -->
                <dependency>

<groupId>org.apache.httpcomponents</groupId>
                        <artifactId>httpclient</artifactId>
                        <version>4.5.6</version>
                </dependency>

        </dependencies>
        <build>
                <sourceDirectory>src</sourceDirectory>
                <plugins>
                        <plugin>
                                <artifactId>maven-compiler-
plugin</artifactId>
                                <version>3.5.1</version>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

3.2. HTTP POST 操作

Post Data

```
package cn.netkiller.httpclient;

import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.xml.bind.DatatypeConverter;

import org.apache.http.Consts;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;

public class HttpClientPost {

        public static void main(String[] args) throws
ClientProtocolException, IOException,
NoSuchAlgorithmException {
                // TODO Auto-generated method stub

                String url =
```

```
"http://api.netkiller.cn:8080/api/comment/list.jspx";
                String appId = "3175755150424665";
                String appKey =
"yEjnjoSEOQpOP49od1IexLkyVB4HTi9c";
                String contentId = "1103";

                DateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

                String date = dateFormat.format(new Date());

                String token =
DatatypeConverter.printHexBinary(MessageDigest.getInstance("M
D5").digest(String.format("%s&%s", appKey,
date).getBytes("UTF-8"))).toLowerCase();

                CloseableHttpClient httpclient =
HttpClients.createDefault();

                //
appId=3175755150424665&contentId=1103&pageSize=100&timeStamp=
2017-08-03
                //
10:20:00&token=e1180c0306aff7792c3e25699900dd0d
                List<NameValuePair> params = new
ArrayList<NameValuePair>();
                params.add(new BasicNameValuePair("appId",
appId));
                params.add(new
BasicNameValuePair("contentId", contentId));
                params.add(new BasicNameValuePair("pageSize",
"10"));
                params.add(new
BasicNameValuePair("timeStamp", date));
                params.add(new BasicNameValuePair("token",
token));

                System.out.println(params.toString());
                UrlEncodedFormEntity urlEncodedFormEntity =
new UrlEncodedFormEntity(params, Consts.UTF_8);

System.out.println(urlEncodedFormEntity.toString());

                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(urlEncodedFormEntity);
```

```
                CloseableHttpResponse response =
httpclient.execute(httpPost);

                System.out.println(response.getStatusLine());
                HttpEntity entity = response.getEntity();
                String responseBody =
EntityUtils.toString(entity, "UTF-8");
                System.out.println(responseBody.toString());
                response.close();
        }

}
```

POST RAW 数据

```
import java.io.IOException;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;

@SuppressWarnings("deprecation")
public class HTTPREST {

        public static void main(String[] args) throws
ClientProtocolException, IOException {
                HttpClient httpClient = new
DefaultHttpClient();

                try {
                        HttpPost request = new
HttpPost("http://test:123456@api.netkiller.cn/v1/test/create.
json");
```

```
                        StringEntity params = new
StringEntity("{\"name\":\"neo\",
\"nickname\":\"netkiller\"}", "UTF-8");
                        request.addHeader("content-type",
"application/json");
                        request.addHeader("Accept",
"application/json");
                        request.setEntity(params);
                        HttpResponse response =
httpClient.execute(request);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println(statusCode);

                        if (response != null) {

                                String responseBody =
EntityUtils.toString(response.getEntity(), "UTF-8");

System.out.println(responseBody.toString());
                        }

                } catch (Exception ex) {
                        ex.printStackTrace();
                } finally {

httpClient.getConnectionManager().shutdown();
                }

        }

}
```

POST GBK 编码得数据

有些国内短信运营商发送短信的接口只能接收 GBK 编码，下面就是一个POST GBK编码数据的范例。

```
```

```
package api.test.sms;

import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import org.apache.http.Consts;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;

public class SMS {

	public SMS() {
		// TODO Auto-generated constructor stub
	}

	public void sendSMS(String mobile, String message)
throws ClientProtocolException, IOException {

		String url =
"http://api.netkiller.cn/sms/v2/send";

		SimpleDateFormat sdf = new
SimpleDateFormat("MMddHHmmss");
		String timestamp =
sdf.format(Calendar.getInstance().getTime());

		CloseableHttpClient httpclient =
HttpClients.createDefault();

		List<NameValuePair> params = new
ArrayList<NameValuePair>();
		params.add(new BasicNameValuePair("apikey",
```

```
"2863300994052170feb"));
                params.add(new BasicNameValuePair("mobile",
mobile));
                params.add(new BasicNameValuePair("content",
message));

                UrlEncodedFormEntity urlEncodedFormEntity =
new UrlEncodedFormEntity(params, "GBK");

System.out.println(urlEncodedFormEntity.toString());

                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(urlEncodedFormEntity);

                CloseableHttpResponse response =
httpclient.execute(httpPost);

                HttpEntity entity = response.getEntity();
                String responseBody =
EntityUtils.toString(entity, "UTF-8");

                System.out.println(params.toString());
                System.out.println(response.getStatusLine());
                System.out.println(responseBody.toString());
                response.close();

        }

        public static void main(String[] args) throws
ClientProtocolException, IOException {
                // TODO Auto-generated method stub
                SMS sms = new SMS();
                String message = String.format("您的验证码是%s,
在%s分钟内输入有效。如非本人操作请忽略此短信。", 8888, 10);
                sms.sendSMS("13113668800", message);
        }

}
```

3.3. HTTPS

Get https 接口

环境 Nginx SSL(openssl自颁发)，nginx 通过proxy_pass连接 Tomcat

下面是 nginx 配置

```
server {
        listen 443 ssl spdy;
        server_name api.netkiller.cn;

        ssl_certificate /etc/nginx/ssl/api.netkiller.cn.crt;
        ssl_certificate_key
/etc/nginx/ssl/api.netkiller.cn.key;
        ssl_session_cache shared:SSL:20m;
        ssl_session_timeout 60m;
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

        charset utf-8;
        access_log /var/log/nginx/api.netkiller.cn.access.log;
        error_log /var/log/nginx/api.netkiller.cn.error.log;

        location / {
                proxy_pass http://127.0.0.1:7000;
                proxy_http_version 1.1;
                proxy_set_header X-Forwarded-For
                $proxy_add_x_forwarded_for;
        }

}
```

下面是 Java 程序

```
package cn.netkiller.example;

import java.io.IOException;
```

```
import java.security.KeyManagementException;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;

import org.apache.http.HttpEntity;
import org.apache.http.ParseException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContextBuilder;
import org.apache.http.util.EntityUtils;

public class NginxAndOpenSSLAndTomcatAndHttpclient {
        public static void main(String[] args) throws
ParseException, IOException, NoSuchAlgorithmException,
KeyStoreException, KeyManagementException {
                SSLContextBuilder builder = new
SSLContextBuilder();
                builder.loadTrustMaterial(null, new
TrustSelfSignedStrategy());
                SSLConnectionSocketFactory sslFactory = new
SSLConnectionSocketFactory(builder.build());
                CloseableHttpClient httpclient =
HttpClients.custom().setSSLSocketFactory(sslFactory).build();

                HttpGet httpGet = new
HttpGet("https://neo:netkiller@api.netkiller.cn/v1/news/today
.json");
                CloseableHttpResponse response =
httpclient.execute(httpGet);
                try {

System.out.println(response.getStatusLine());
                        HttpEntity entity =
response.getEntity();
                        String responseBody =
EntityUtils.toString(entity, "UTF-8");

System.out.println(responseBody.toString());
                } finally {
                        response.close();
                }
```

```
        }
}
```

如果遇到配置问题，可以看一下 《Netkiller Linux Web 手札》

POST json 数据

```
package cn.netkiller.example;

import java.io.IOException;
import java.security.KeyManagementException;
import java.security.KeyStoreException;
import java.security.NoSuchAlgorithmException;

import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContextBuilder;
import org.apache.http.util.EntityUtils;

public class HttpClientSSLPost {

        public HttpClientSSLPost() {
                // TODO Auto-generated constructor stub
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SSLContextBuilder builder = new
SSLContextBuilder();
                try {
                        builder.loadTrustMaterial(null, new
```

```
TrustSelfSignedStrategy());

                        SSLConnectionSocketFactory sslFactory
= new SSLConnectionSocketFactory(builder.build());
                        CloseableHttpClient httpclient =
HttpClients.custom().setSSLSocketFactory(sslFactory).build();

                        HttpPost httpPost = new
HttpPost("https://neo:YruuUCNXKe@api.netkiller.cn/v1/member/c
reate.json");
                        httpPost.addHeader("content-type",
"application/json");
                        httpPost.addHeader("Accept",
"application/json");

                        HttpEntity httpEntity = new
StringEntity("{\"name\":\"neo\",
\"nickname\":\"netkiler\",\"age\":\"18\"}", "UTF-8");

                        httpPost.setEntity(httpEntity);

                        CloseableHttpResponse response =
httpclient.execute(httpPost);


System.out.println(response.getStatusLine());
                        HttpEntity entity =
response.getEntity();
                        String responseBody =
EntityUtils.toString(entity, "UTF-8");

System.out.println(responseBody.toString());
                        response.close();
                } catch (NoSuchAlgorithmException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (KeyStoreException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (KeyManagementException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (ClientProtocolException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
```

```
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                finally {

                }
        }

}
```

3.4. HTTP/2

HTTP/2实例

```
    @Test
    public void testHttp2() throws URISyntaxException,
IOException, InterruptedException {
        HttpClient.newBuilder()
                .followRedirects(HttpClient.Redirect.SECURE)
                .version(HttpClient.Version.HTTP_2)
                .build()
                .sendAsync(HttpRequest.newBuilder()
                                .uri(new
URI("https://http2.akamai.com/demo"))
                                .GET()
                                .build(),
                        HttpResponse.BodyHandler.asString())
                .whenComplete((resp,t) -> {
                    if(t != null){
                        t.printStackTrace();
                    }else{
                        System.out.println(resp.body());

System.out.println(resp.statusCode());
                    }
                }).join();
    }
```

3.5. Java11

sync get

```
        /**
     * --add-modules jdk.incubator.httpclient
     * @throws IOException
     * @throws InterruptedException
     * @throws URISyntaxException
     */
    @Test
    public void testGet() throws IOException,
InterruptedException, URISyntaxException {
        HttpClient httpClient = HttpClient.newHttpClient();
        HttpRequest httpRequest = HttpRequest.newBuilder()
                .uri(new URI("https://www.baidu.com"))
                .header("User-Agent", "jdk 9 http client")
                .GET()
                .build();
        HttpResponse<String> httpResponse =
httpClient.send(httpRequest,
HttpResponse.BodyHandler.asString());
        System.out.println(httpResponse.statusCode());
        System.out.println(httpResponse.body());
    }
```

async get

```
    @Test
    public void testAsyncGet() throws URISyntaxException,
InterruptedException, ExecutionException {
        HttpClient client = HttpClient.newHttpClient();
```

```
        HttpRequest request = HttpRequest.newBuilder()
                .uri(new URI("https://www.baidu.com"))
                .GET()
                .build();

        CompletableFuture<HttpResponse<String>> response =
client.sendAsync(request, HttpResponse.BodyHandler.asString());

        response.whenComplete((resp,t) -> {
            if(t != null){
                t.printStackTrace();
            }else{
                System.out.println(resp.body());
                System.out.println(resp.statusCode());
            }
        }).join();
    }
```

post form

post form

```
        @Test
    public void testPostForm() throws URISyntaxException {
        HttpClient client = HttpClient.newHttpClient();

        HttpRequest request = HttpRequest.newBuilder()
                .uri(new
URI("http://www.w3school.com.cn/demo/demo_form.asp"))
                .header("Content-Type","application/x-www-form-
urlencoded")

.POST(HttpRequest.BodyProcessor.fromString("name1=value1&name2=
value2"))
                .build();
        client.sendAsync(request,
HttpResponse.BodyHandler.asString())
                .whenComplete((resp,t) -> {
```

```
                    if(t != null){
                        t.printStackTrace();
                    }else{
                        System.out.println(resp.body());
                        System.out.println(resp.statusCode());
                    }
        }).join();
    }
```

3.6. Host name 'api.netkiller.cn' does not match the certificate subject provided

这个问题是CN不匹配造成的，日志如下

```
Exception in thread "main"
javax.net.ssl.SSLPeerUnverifiedException: Host name
'api.netkiller.cn' does not match the certificate subject
provided by the peer (EMAILADDRESS=netkiller@msn.com,
CN=netkiller.cn, OU=CF, O=CF, L=Shenzhen, ST=Guangdong, C=CN)
```

解决方案一：重新做证书

```
Country Name (2 letter code) [XX]:CN
State or Province Name (full name) []:Guangdong
Locality Name (eg, city) [Default City]:Shenzhen
Organization Name (eg, company) [Default Company Ltd]:CF
Organizational Unit Name (eg, section) []:CF
Common Name (eg, your name or your server's hostname)
[]:api.netkiller.cn
Email Address []:netkiller@msn.com
```

解决方案二：是用CN上的域名链接

3.7. HttpStatus

```
package cn.netkiller.consul.controller;

import org.apache.http.HttpStatus;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import
org.springframework.web.bind.annotation.RestController;

@RestController
public class TestController {

        public TestController() {
                // TODO Auto-generated constructor stub
        }

        @RequestMapping("/health")
        public int health() {
                return HttpStatus.SC_OK;
        }
}
```

3.8.

```
        String url =
"https://www.netkiller.cn:8080/book/chapter1/section2?
id=9527";
    URIBuilder uriBuilder = new URIBuilder(url);
    System.out.println(uriBuilder.getScheme());
    System.out.println(uriBuilder.getUserInfo());
    System.out.println(uriBuilder.getHost());
    System.out.println(uriBuilder.getPort());
    System.out.println(uriBuilder.getPath());
    System.out.println(uriBuilder.getQueryParams());
```

```
    System.out.println(uriBuilder.getFragment());
    System.out.println(uriBuilder.getCharset());
```

添加参数

```
        String url = "https://www.netkiller.cn/test?id=1";
    URIBuilder uriBuilder = new URIBuilder(url);
    uriBuilder.setParameter("id", "2"); // set 会覆盖原来的参数
    uriBuilder.addParameter("cat" ,"5"); // add 添加参数
    System.out.println(uriBuilder.build());
```

去除参数

```
        String url =
"https://www.netkiller.cn:8080/book/chapter1/section2?
id=9527";
    URIBuilder uriBuilder = new URIBuilder(url);
    uriBuilder.clearParameters();
    System.out.println(uriBuilder.build());
    # https://www.netkiller.cn:8080/book/chapter1/section2
```

第 25 章 Cache

1. java memcached client

```
wget http://img.whalin.com/memcached/jdk6/log4j/java_memcached-
release_1.5.1.tar.gz
tar zxvf java_memcached-release_1.5.1.tar.gz
cd java_memcached-release_1.5.1
cp java_memcached-release_1.5.1.jar
/usr/local/memcached/api/java
```

export CLASSPATH="./:/usr/local/java/lib:/usr/local/java/jre/lib:/usr/local/memcached/api/java/java_memcached-release_1.5.1.jar:/usr/local/memcached/api/java/log4j-1.2.14.jar"

例 25.1. memcached.java

```
import com.danga.MemCached.*;
import org.apache.log4j.*;
public class memcached {

        public static void main(String[] args) {
                try{
                        BasicConfigurator.configure();

                        String[] serverlist = {
"127.0.0.1:11211"  };

                        // initialize the pool for memcache
servers
                        SockIOPool pool =
SockIOPool.getInstance( "test" );
                        pool.setServers( serverlist );
                        pool.setInitConn( 10 );
```

```
                        pool.setMinConn( 5 );
                        pool.setMaxConn( 250 );
                        pool.setMaintSleep( 30 );
                        pool.setNagle( false );
                        pool.setSocketTO( 3000 );
                        pool.initialize();

                MemCachedClient mc = new MemCachedClient();

                // compression is enabled by default
                mc.setCompressEnable(true);

                // set compression threshhold to 4 KB (default:
15 KB)
                mc.setCompressThreshold(4096);

                // turn on storing primitive types as a string
representation
                // Should not do this in most cases.
                mc.setPrimitiveAsString(true);

                mc.setPoolName( "test" );

                        for ( int i = 0; i < 10; i++ ) {
                                boolean success = mc.set( "" +
i, "Hello!" );
                                String result = (String)mc.get(
"" + i );
                                System.out.println(
String.format( "set( %d ): %s", i, success ) );
                                System.out.println(
String.format( "get( %d ): %s", i, result ) );
                        }

                        System.out.println( "\n\t -- sleeping -
-\n" );
                        try { Thread.sleep( 10000 ); } catch (
Exception ex ) { }

                        for ( int i = 0; i < 10; i++ ) {
                                boolean success = mc.set( "" +
i, "Hello!" );
                                String result = (String)mc.get(
"" + i );
                                System.out.println(
```

```
String.format( "set( %d ): %s", i, success ) );
                                System.out.println(
String.format( "get( %d ): %s", i, result ) );
                            }

                }
                catch (Exception e)
                {
                        System.out.println("[Exception] - " +
e.toString());
                }
                finally {}

        }

}
```

test memcache

```
javac memcached.java
java memcached
```

2. Jedis

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.7.2</version>
    <type>jar</type>
    <scope>compile</scope>
</dependency>
```

```
package cn.netkiller.redis;

import redis.clients.jedis.Jedis;

public class JedisTest {

        private static Jedis jedis;

        public static void main(String[] args) {
                jedis = new Jedis("192.168.2.1");
                System.out.println("Server is running:
"+jedis.ping());
                jedis.set("foo", "bar");
                String value = jedis.get("foo");
                System.out.println(value);
        }

}
```

2.1. 认证

```
jedis = new Jedis("localhost", 6379, 15000);
jedis.auth("foobared");
```

2.2. jedis.keys

```
        public Map<String, String> getCaptcha() {

                Map<String, String> captchas = new
HashMap<String, String>();
                this.jedis = new
Jedis(this.config.getProperty("captcha.redis"));

                System.out.printf("%s: Redis server is
running %s\n", this.getClass().getName(), this.jedis.ping());

                Set<String> keys =
jedis.keys("captcha::phone::*");
                if (!keys.isEmpty()) {
                        for(String key:keys){
                                if (this.jedis.exists(key)) {

captchas.put(key.replace("captcha::phone::", ""),
this.jedis.get(key)+" ["+ String.valueOf(this.jedis.ttl(key))
+ "s]");
                                }
                        }
                }

                this.jedis.close();

                return captchas;
        }
```

```
import redis.clients.jedis.Jedis;
public class RedisKeys {
   public static void main(String[] args) {

      Jedis jedis = new Jedis("localhost");
      System.out.println("Connection to server sucessfully");

      List<String> list = jedis.keys("*");
      for(int i=0; i<list.size(); i++) {
         System.out.println("List of stored keys::
"+list.get(i));
      }
   }
}
```

3. Ehcache

第 26 章 Kafka

1. 安装 Kafka 环境

请参考 《Netkiller Linux 手札》电子书 的 Kafka 章节，电子书中又详细的安装步骤。

2. Maven

```
                <!--
https://mvnrepository.com/artifact/org.apache.kafka/kafka-
clients -->
                <dependency>
                        <groupId>org.apache.kafka</groupId>
                        <artifactId>kafka-
clients</artifactId>
                        <version>1.0.0</version>
                </dependency>
```

3. 启动 kafka

有两种方式启动 kafka, 一种是命令行，另一种是通过 Java 程序，命令行方式请参考《Netkiller Linux 手札》，这里只介绍如何使用 Java 程序启动 Kafka。

首先启动 Zookeeper

```
QuorumPeerConfig config = new QuorumPeerConfig();
InputStream inputStream =
KafkaTest.class.getResourceAsStream("/srv/kafka/config/zookee
per.properties");
Properties properties = new Properties();
properties.load(inputStream);
inputStream.close();
config.parseProperties(properties);
ServerConfig   serverConfig = new ServerConfig();
serverConfig.readFrom(config);
new ZooKeeperServerMain().runFromConfig(serverConfig);
```

然后启动 Kafka

```
InputStream inputStream =
KafkaTest.class.getResourceAsStream("/srv/kafka/config/server
.properties");
Properties properties = new Properties();
properties.load(is);
inputStream.close();
KafkaServerStartable kafkaServerStartable =
KafkaServerStartable.fromProps(properties);
kafkaServerStartable.startup();
kafkaServerStartable.awaitShutdown();
```

4. 入门例子

4.1. 订阅例子

```
package cn.netkiller.kafka.test;

import java.util.Arrays;
import java.util.Properties;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;

public class KafkaConsumerExample {

        private static KafkaConsumer<String, String>
consumer;

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Properties props = new Properties();
                props.put("bootstrap.servers",
"kafka.netkiller.cn:9092");
                props.put("group.id", "test");
                props.put("enable.auto.commit", "true");
                props.put("auto.commit.interval.ms", "1000");
                props.put("session.timeout.ms", "30000");
                props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
                props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
                consumer = new KafkaConsumer<String, String>
(props);
                consumer.subscribe(Arrays.asList("test"));
                while (true) {
                        ConsumerRecords<String, String>
records = consumer.poll(100);
                        for (ConsumerRecord<String, String>
record : records)
```

```
                                    System.out.printf("offset =
%d, key = %s, value = %s\n", record.offset(), record.key(),
record.value());
                }
        }

}
```

测试方法

```
root@netkiller ~ % /srv/kafka/bin/kafka-console-producer.sh -
-broker-list localhost:9092 --topic test
>Helloworld
```

下面详细讲解上面的程序。首先我们通过Properties文件来配置消费属性。

```
                Properties props = new Properties();
                props.put("bootstrap.servers",
"kafka.netkiller.cn:9092");
                props.put("group.id", "test");
                props.put("enable.auto.commit", "true");
                props.put("auto.commit.interval.ms", "1000");
                props.put("session.timeout.ms", "30000");
                props.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
                props.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
```

然后订阅TOPIC，从那个TOPIC 读取消息，Kafka 可以同时订阅多个 TOPIC。下面的例子中，同时订阅了foo和bar两个topic：

```
consumer.subscribe(Arrays.asList("foo", "bar"));
```

取出消息（消费消息），通过循环调用poll方法，从队列中取出消息。

```
                while (true) {
                        ConsumerRecords<String, String>
records = consumer.poll(100);
                        for (ConsumerRecord<String, String>
record : records)
                                System.out.printf("offset =
%d, key = %s, value = %s\n", record.offset(), record.key(),
record.value());
                }
```

poll方法里面的参数是等待消息的时间(Long类型)，如果队列里面有消息，会立马返回，如果没有，会等待指定的时间然后返回。

```
```

4.2. 发布例子

```
package cn.netkiller.kafka.test;

import org.apache.kafka.clients.producer.KafkaProducer;
```

```
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.util.Properties;

public class KafkaProducerExample {
    public static void main(String[] args) {
        Properties props = new Properties();
        props.put("bootstrap.servers",
"kafka.netkiller.cn:9092");
        props.put("acks", "all");
        props.put("retries", 0);
        props.put("batch.size", 16384);
        props.put("linger.ms", 1);
        props.put("buffer.memory", 33554432);
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

        Producer<String, String> producer = new
KafkaProducer<>(props);
        for(int i = 0; i < 100; i++)
            producer.send(new ProducerRecord<>("test",
Integer.toString(i), Integer.toString(i)));

        producer.close();
    }
}
```

5. 线程例子

```
package cn.netkiller.ipo.test;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import org.apache.kafka.common.errors.WakeupException;
import
org.apache.kafka.common.serialization.StringDeserializer;

public class KafkaConsumerThread implements Runnable {
        private final KafkaConsumer<String, String> consumer;
        private final List<String> topics;

        public KafkaConsumerThread(String groupId,
List<String> topics) {
                this.topics = topics;
                Properties props = new Properties();
                props.put("bootstrap.servers",
"kafka.netkiller.cn:9092");
                props.put("group.id", groupId);
                props.put("key.deserializer",
StringDeserializer.class.getName());
                props.put("value.deserializer",
StringDeserializer.class.getName());
                this.consumer = new KafkaConsumer<String,
String>(props);
        }
```

```
        public void run() {
                try {
                        consumer.subscribe(this.topics);

                        while (true) {
                                ConsumerRecords<String,
String> records = consumer.poll(Long.MAX_VALUE);
                                for (ConsumerRecord<String,
String> record : records) {
                                        Map<String, Object>
data = new HashMap<>();
                                        data.put("partition",
record.partition());
                                        data.put("offset",
record.offset());
                                        data.put("value",
record.value());

System.out.println(data);
                                }
                        }
                } catch (WakeupException e) {
                        // ignore for shutdown
                } finally {
                        consumer.close();
                }
        }

        public void shutdown() {
                consumer.wakeup();
        }

        public static void main(String[] args) {
                int numConsumers = 3;
                String groupId = "consumer-tutorial-group";
                List<String> topics = Arrays.asList("test");
                ExecutorService executor =
Executors.newFixedThreadPool(numConsumers);

                final List<KafkaConsumerThread> consumers =
new ArrayList<>();
                for (int i = 0; i < numConsumers; i++) {
                        KafkaConsumerThread consumer = new
KafkaConsumerThread(groupId, topics);
```

```
                        consumers.add(consumer);
                        executor.submit(consumer);
                }

                Runtime.getRuntime().addShutdownHook(new
Thread() {
                        @Override
                        public void run() {
                                for (KafkaConsumerThread
consumer : consumers) {
                                        consumer.shutdown();
                                }
                                executor.shutdown();
                                try {

executor.awaitTermination(5000, TimeUnit.MILLISECONDS);
                                } catch (InterruptedException
e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }
}
```

第 27 章 Software Development Kit

1. JAVE(Java Audio Video Encoder)

http://www.sauronsoftware.it/projects/jave/manual.php#10

2. Google

2.1. com.google.gson

https://github.com/google/gson

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>sender</groupId>
        <artifactId>sender</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>Sender</name>
        <description>EDM</description>

        <dependencies>
                <dependency>
                        <groupId>com.rabbitmq</groupId>
                        <artifactId>amqp-client</artifactId>
                        <version>3.6.0</version>
                </dependency>
                <dependency>

<groupId>com.google.code.gson</groupId>
                        <artifactId>gson</artifactId>
                        <version>2.6.2</version>
                        <scope>compile</scope>
                </dependency>
        </dependencies>

</project>
```

map 处理

Example

```
package io.github.netkiller;

import java.util.HashMap;
import java.util.Map;

import com.google.gson.*;

public class GsonTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                Gson gson = new Gson();
                String json = "
{\"k1\":\"v1\",\"k2\":\"v2\"}";
                Map<String, String> map = new HashMap<String,
String>();
                map = (Map<String, String>)
gson.fromJson(json, map.getClass());
                System.out.println(map.get("k1"));
        }

}
```

多层 Map 剥离

```
                Gson gson = new Gson();
                String inf= "{\"0\":
{\"id\":\"2\",\"category_id\":\"1\",\"title\":\"Test2\",\"aut
hor\":\"\",\"ctime\":\"2016-03-05 11:59:56\"},\"1\":
{\"id\":\"1\",\"category_id\":\"1\",\"title\":\"Test1\",\"aut
hor\":\"\u6d4b\u8bd5\",\"ctime\":\"2016-03-05
11:57:30\"},\"pages\":
{\"count\":2,\"first\":0,\"last\":0,\"before\":0,\"current\":
```

```
0,\"next\":0,\"total\":0}}";
                Map<String, Map> map = new HashMap<String,
Map>();

                map = (Map<String, Map>) gson.fromJson(inf,
map.getClass());

System.out.println(map.get("1").get("title"));

System.out.println(map.get("pages").get("count"));
```

POJO

```
package cn.netkiller.gson;

import java.util.ArrayList;
import java.util.List;

public class Personal {
        private int age = 30;
        private String name = "neo";
        private List<String> telphone = new ArrayList<String>
() {
                {
                        add("13113668890");
                        add("13322993040");
                        add("29812080");
                }
        };

        // getter and setter methods

        @Override
        public String toString() {
                return "Personal [age=" + age + ", name=" +
name + ", telphone=" + telphone + "]";
        }
}
```

toJson

```
package cn.netkiller.gson;

import com.google.gson.Gson;

public class GsonExampleToJson {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Personal obj = new Personal();
                Gson gson = new Gson();

                // convert java object to JSON format, and
returned as JSON formatted string
                String json = gson.toJson(obj);
                System.out.println(json);
        }

}
```

fromJson

```
package cn.netkiller.gson;

import com.google.gson.Gson;

public class GsonExampleFromJson {
        public static void main(String[] args) {

                Personal obj = new Personal();
                Gson gson = new Gson();
```

```
                // convert the json string back to object
                obj = gson.fromJson("
{\"age\":30,\"name\":\"neo\",\"telphone\":
[\"13113668890\",\"13322993040\",\"29812080\"]}",
Personal.class);

                System.out.println(obj);
        }
}
```

JsonParser

```
package cn.netkiller.gson;

import java.util.Map.Entry;

import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonArray;
import com.google.gson.JsonParser;

public class GsonJsonParser {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                String jsonString = "
{\"age\":30,\"name\":\"neo\",\"telphone\":
[\"13113668890\",\"13322993040\",\"29812080\"],\"address\":
{\"province\":\"Guangdong\",\"city\":\"Shenzhen\"}}";

                JsonElement root = new
JsonParser().parse(jsonString);
                System.out.println(root.toString());

System.out.println(root.getAsJsonObject().get("age").getAsInt
());

System.out.println(root.getAsJsonObject().get("name").getAsSt
```

```
ring());

                // Get the content of the first map
                JsonArray jsonArray =
root.getAsJsonObject().get("telphone").getAsJsonArray();

                for (JsonElement tel : jsonArray) {
                        System.out.println(tel);
                }

                JsonObject object =
root.getAsJsonObject().get("address").getAsJsonObject();
                for (Entry<String, JsonElement> entry :
object.entrySet()) {
                        System.out.println(entry.getKey() +
":" + entry.getValue());
                }
        }

}
```

Exmaple 范例

Map to Json

```
Gson gson = new
GsonBuilder().enableComplexMapKeySerialization().create();
String jsonString = gson.toJson(map);
```

Exmaple 范例

Map to Json

```
Gson gson = new
GsonBuilder().enableComplexMapKeySerialization().create();
Map<String, String> source = gson.fromJson(output,
HashMap.class);
```

处理复杂的类型

通过 TypeToken 定义复杂数据库类型。

```
package cn.netkiller.ipo.process.json;

import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.reflect.TypeToken;

import cn.netkiller.ipo.process.ProcessInterface;

public class JsonValueLength implements ProcessInterface {
        private final static Logger logger =
LoggerFactory.getLogger(JsonValueLength.class);
        private Gson gson = new
GsonBuilder().enableComplexMapKeySerialization().create();
        private int maxLength = 0;

        public JsonValueLength(int maxLength) {
                this.maxLength = maxLength;
        }

        @Override
        public String run(String line) {
```

```
                Map<String, String> map = gson.fromJson(line,
new TypeToken<Map<String, String>>() {
                }.getType());
                for (String key : map.keySet()) {
                        if (map.get(key).length() >
this.maxLength) {
                                map.put(key,
map.get(key).substring(0, this.maxLength));
                        }
                }
                logger.debug("{}", map);
                return gson.toJson(map);
        }

}
```

2.2. Guava

https://github.com/google/guava/wiki

maven

```
                <dependency>
                        <groupId>com.google.guava</groupId>
                        <artifactId>guava</artifactId>
                        <version>23.0</version>
                </dependency>
```

27 版本需要指定 27.1-jre，如果是安卓系统 27.1-android

```
                <dependency>
                        <groupId>com.google.guava</groupId>
                        <artifactId>guava</artifactId>
```

```
                        <version>27.1-jre</version>
                </dependency>
```

删除不可显示的字符

```
package cn.netkiller.test;

import com.google.common.base.CharMatcher;

public class Test {

        public static void main(String[] args) {

                String string = "佛山市南海区123华泰ABC精密abc机械有限公司消防维保□□□□□□,";

                // 版本 23.0
                // String printable =
CharMatcher.INVISIBLE.removeFrom(string);
                // String clean =
CharMatcher.ASCII.retainFrom(printable);

                String printable =
CharMatcher.invisible().removeFrom(string);
                String clean =
CharMatcher.ascii().retainFrom(printable);
                System.out.println(printable);
                System.out.println(clean);


        }

}
```

3. Mahout

3.1. 推荐系统

Maven pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>cn.netkiller</groupId>
        <artifactId>mahout</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>mahout</name>
        <url>http://maven.apache.org</url>

        <properties>
                <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>org.apache.mahout</groupId>
                        <artifactId>mahout-core</artifactId>
                        <version>0.9</version>
                </dependency>
                <dependency>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-nop</artifactId>
                        <version>1.7.30</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
```

```
                        <artifactId>junit</artifactId>
                        <version>3.8.1</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
</project>
```

推荐程序

```
package cn.netkiller.mahout;

import java.io.File;
import java.util.List;

import
org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import
org.apache.mahout.cf.taste.impl.neighborhood.NearestNUserNeig
hborhood;
import
org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedR
ecommender;
import
org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelation
Similarity;
import org.apache.mahout.cf.taste.model.DataModel;
import
org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;
import
org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.Recommender;
import org.apache.mahout.cf.taste.similarity.UserSimilarity;

public class App {

        public App() {
                // TODO Auto-generated constructor stub
        }
```

```
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                try {
                        // 从文件加载数据
                        DataModel model = new
FileDataModel(new File("target/classes/test.csv"));
                        // 指定用户相似度计算方法，这里采用皮尔森相
关度
                        UserSimilarity similarity = new
PearsonCorrelationSimilarity(model);
                        // 指定用户邻居数量，这里为2
                        UserNeighborhood neighborhood = new
NearestNUserNeighborhood(2, similarity, model);
                        // 构建基于用户的推荐系统
                        Recommender recommender = new
GenericUserBasedRecommender(model, neighborhood, similarity);
                        // 得到指定用户的推荐结果，这里是得到用户1的
两个推荐
                        List<RecommendedItem> recommendations
= recommender.recommend(1, 2);
                        // 打印推荐结果
                        for (RecommendedItem recommendation :
recommendations) {

System.out.println(recommendation);
                        }
                } catch (Exception e) {
                        System.out.println(e);
                }
        }

}
```

数据文件

```
1,101,5.0
1,102,3.0
```

```
1,103,2.5
2,101,2.0
2,102,2.5
2,103,5.0
2,104,2.0
3,101,2.5
3,104,4.0
3,105,4.5
3,107,5.0
4,101,5.0
4,103,3.0
4,104,4.5
4,106,4.0
5,101,4.0
5,102,3.0
5,103,2.0
5,104,4.0
5,105,3.5
5,106,4.0
```

4. Hessian

基于Binary-RPC协议实现

5. quartz-scheduler

http://quartz-scheduler.org/

6. Redisson

http://redisson.org/

部分 II. Android 9 Pie

第 28 章 Android Studio

1. 卸载 Android Studio

卸载 Mac Android Studio

```
rm -Rf /Applications/Android\ Studio.app
rm -Rf ~/Library/Preferences/AndroidStudio*
rm ~/Library/Preferences/com.google.android.studio.plist
rm -Rf ~/Library/Application\ Support/AndroidStudio*
rm -Rf ~/Library/Logs/AndroidStudio*
rm -Rf ~/Library/Caches/AndroidStudio*
rm -Rf ~/Library/Android*
```

删除 Projects

```
rm -Rf ~/AndroidStudioProjects
```

删除 gradle

```
rm -rf ~/.gradle
```

卸载Android Virtual Devices(AVDs) and *.keystore.

```
rm -Rf ~/.android
```

2. 代码格式化

```
Option + Command + L
```

3. 设置兼容最低SDK版本

Android 9 Pie 的设置方法

打开 build.gradle 修改 minSdkVersion 项，这里的 26 表示 Android 8.0

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "cn.netkiller.video"
        minSdkVersion 26
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation
'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
```

```
}
```

4. SDK Tools

```
wget https://dl.google.com/android/repository/sdk-tools-linux-
4333796.zip
unzip sdk-tools-linux-4333796.zip
cd tools/
```

查看帮助信息

```
neo@ubuntu:~/tmp/tools$ bin/sdkmanager --help
Usage:
  sdkmanager [--uninstall] [<common args>] [--package_file=
<file>] [<packages>...]
  sdkmanager --update [<common args>]
  sdkmanager --list [<common args>]
  sdkmanager --licenses [<common args>]
  sdkmanager --version

With --install (optional), installs or updates packages.
    By default, the listed packages are installed or (if
already installed)
    updated to the latest version.
With --uninstall, uninstall the listed packages.

    <package> is a sdk-style path (e.g. "build-tools;23.0.0" or
              "platforms;android-23").
    <package-file> is a text file where each line is a sdk-
style path
                    of a package to install or uninstall.
    Multiple --package_file arguments may be specified in
combination
    with explicit paths.

With --update, all installed packages are updated to the latest
version.
```

```
With --list, all installed and available packages are printed
out.

With --licenses, show and offer the option to accept licenses
for all
     available packages that have not already been accepted.

With --version, prints the current version of sdkmanager.

Common Arguments:
    --sdk_root=<sdkRootPath>: Use the specified SDK root
instead of the SDK
                              containing this tool

    --channel=<channelId>: Include packages in channels up to
<channelId>.
                           Common channels are:
                           0 (Stable), 1 (Beta), 2 (Dev), and 3
(Canary).

    --include_obsolete: With --list, show obsolete packages in
the
                        package listing. With --update, update
obsolete
                        packages as well as non-obsolete.

    --no_https: Force all connections to use http rather than
https.

    --proxy=<http | socks>: Connect via a proxy of the given
type.

    --proxy_host=<IP or DNS address>: IP or DNS address of the
proxy to use.

    --proxy_port=<port #>: Proxy port to connect to.

    --verbose: Enable verbose output.

* If the env var REPO_OS_OVERRIDE is set to "windows",
  "macosx", or "linux", packages will be downloaded for that
OS.
```

4.1. 接受 License

```
$ yes|tools/bin/sdkmanager --licenses
```

4.2. 查看 SDK 列表

```
neo@ubuntu:~/tmp/tools$ bin/sdkmanager --list | grep
"platforms;"
Warning: File /home/neo/.android/repositories.cfg could not be
loaded.
  platforms;android-26 | 2       | Android SDK Platform 26  |
platforms/android-26/
  platforms;android-10
| 2             | Android SDK Platform 10
  platforms;android-11
| 2             | Android SDK Platform 11
  platforms;android-12
| 3             | Android SDK Platform 12
  platforms;android-13
| 1             | Android SDK Platform 13
  platforms;android-14
| 4             | Android SDK Platform 14
  platforms;android-15
| 5             | Android SDK Platform 15
  platforms;android-16
| 5             | Android SDK Platform 16
  platforms;android-17
| 3             | Android SDK Platform 17
  platforms;android-18
| 3             | Android SDK Platform 18
  platforms;android-19
| 4             | Android SDK Platform 19
  platforms;android-20
| 2             | Android SDK Platform 20
```

```
  platforms;android-21
| 2             | Android SDK Platform 21
  platforms;android-22
| 2             | Android SDK Platform 22
  platforms;android-23
| 3             | Android SDK Platform 23
  platforms;android-24
| 2             | Android SDK Platform 24
  platforms;android-25
| 3             | Android SDK Platform 25
  platforms;android-26
| 2             | Android SDK Platform 26
  platforms;android-27
| 3             | Android SDK Platform 27
  platforms;android-28
| 6             | Android SDK Platform 28
  platforms;android-7
| 3             | Android SDK Platform 7
  platforms;android-8
| 3             | Android SDK Platform 8
  platforms;android-9
| 2             | Android SDK Platform 9
```

4.3. 按照 Android SDK

最新版本 Android 9 SDK

```
bin/sdkmanager "platform-tools" "platforms;android-28"
```

按照旧版本的 Android 8 SDK

```
bin/sdkmanager "platforms;android-26"
```

5. 命令行操作

会同时生成debug和release两个包

```
./gradlew assemble
```

只生成release的包

```
./gradlew assembleRelease
```

6. adb 命令

默认情况执行 adb 会提示找不到命令

```
neo@MacBook-Pro-M2 ~ % adb
zsh: command not found: adb
```

提示

这里我使用的是 zsh shell

```
neo@MacBook-Pro-M2 ~ % open -e .zprofile

export PATH=${PATH}:~/Library/Android/sdk/platform-tools
```

现在可以正常使用了

```
neo@MacBook-Pro-M2 ~ % adb version
Android Debug Bridge version 1.0.41
Version 34.0.4-10411341
Installed as /Users/neo/Library/Android/sdk/platform-tools/adb
Running on Darwin 23.0.0 (arm64)
```

6.1. 获得 root 权限

```
neo@MacBook-Pro-M2 ~ % adb push public.libraries.txt
/system/etc/public.libraries.txt
```

```
public.libraries.txt: 1 file pushed, 0 skipped. 1.0 MB/s (485
bytes in 0.000s)
adb: error: failed to copy 'public.libraries.txt' to
'/system/etc/public.libraries.txt': remote couldn't create
file: Read-only file system
```

```
neo@MacBook-Pro-M2 ~ % adb root
neo@MacBook-Pro-M2 ~ % adb push public.libraries.txt
/system/etc/public.libraries.txt
public.libraries.txt: 1 file pushed, 0 skipped. 1.4 MB/s (485
bytes in 0.000s)
adb: error: failed to copy 'public.libraries.txt' to
'/system/etc/public.libraries.txt': remote couldn't create
file: Read-only file system
```

```
neo@MacBook-Pro-M2 ~ % adb remount
Using overlayfs for /system
Using overlayfs for /vendor
Using overlayfs for /odm
Using overlayfs for /product
Using overlayfs for /system_ext
Now reboot your device for settings to take effect
remount succeeded
```

```
neo@MacBook-Pro-M2 ~ % adb push public.libraries.txt
/system/etc/public.libraries.txt
public.libraries.txt: 1 file pushed, 0 skipped. 1.5 MB/s (485
bytes in 0.000s)
```

6.2. 设备管理

```
neo@MacBook-Pro-M2 ~ % adb devices
List of devices attached
0123456789ABCDEF        device
CFE6R21625003544        device
```

查看详细信息

```
neo@MacBook-Pro-M2 ~ % adb devices -l
List of devices attached
0123456789ABCDEF       device usb:1310720X
product:full_aiv8167sm3_bsp model:aiv8167sm3_bsp
device:aiv8167sm3_bsp transport_id:2
CFE6R21625003544       device usb:1114112X product:MRR-W29
model:MRR_W29 device:HWMRR-Q transport_id:1
```

6.3. Shell

```
neo@MacBook-Pro-M2 ~ % adb -s CFE6R21625003544 shell
HWMRR-Q:/ $
```

网络相关

查看 IP 地址

```
neo@MacBook-Pro-M2 ~> adb shell ifconfig
```

```
wlan0     Link encap:Ethernet  HWaddr c0:84:7d:2b:3c:24
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:0

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope: Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:340 errors:0 dropped:0 overruns:0 frame:0
          TX packets:340 errors:0 dropped:0 overruns:0
carrier:0
          collisions:0 txqueuelen:1
          RX bytes:27313 TX bytes:27313

eth0      Link encap:Ethernet  HWaddr 86:7a:05:cc:ae:72
          inet6 addr: fe80::847a:5ff:fecc:ae72/64 Scope: Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:508
          Interrupt:42
```

无线 IP 地址

只查看无线 IP 地址

```
adb shell ifconfig wlan0
```

Mac 地址

```
rk3568_r:/ $ cat /sys/class/net/wlan0/address
```

```
30:7b:c9:0f:12:b9
```

查看 MAC 地址

```
neo@MacBook-Pro-M2 ~> adb shell cat
/sys/class/net/wlan0/address
c0:84:7d:2b:3c:24
```

内存信息

```
neo@MacBook-Pro-M2 ~> adb shell cat /proc/meminfo
MemTotal:        2043916 kB
MemFree:          844392 kB
MemAvailable:    1334032 kB
Buffers:            6376 kB
Cached:           609984 kB
SwapCached:            0 kB
Active:           562872 kB
Inactive:         346688 kB
Active(anon):     297196 kB
Inactive(anon):   116108 kB
Active(file):     265676 kB
Inactive(file):   230580 kB
Unevictable:         256 kB
Mlocked:             256 kB
HighTotal:       1564672 kB
HighFree:         661800 kB
LowTotal:         479244 kB
LowFree:          182592 kB
SwapTotal:        520908 kB
SwapFree:         520908 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:        293468 kB
Mapped:           348972 kB
```

```
Shmem:            120132 kB
Slab:             194608 kB
SReclaimable:     172360 kB
SUnreclaim:        22248 kB
KernelStack:        5376 kB
PageTables:        12448 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     1542864 kB
Committed_AS:   25076844 kB
VmallocTotal:     499712 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
CmaTotal:          16384 kB
CmaFree:           14540 kB
```

查看硬件与系统属性

```
neo@MacBook-Pro-M2 ~> adb shell cat /system/build.prop

# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=NHG47K
ro.build.display.id=rk3288-userdebug 7.1.2 NHG47K
eng.server22.20230423.034518 test-keys
ro.build.version.incremental=eng.server22.20230423.034518
ro.build.version.sdk=25
ro.build.version.preview_sdk=0
ro.build.version.codename=REL
ro.build.version.all_codenames=REL
ro.build.version.release=7.1.2
ro.build.version.security_patch=2017-04-05
ro.build.version.base_os=
ro.build.date=Sun Apr 23 03:45:18 UTC 2023
ro.build.date.utc=1682221518
ro.build.type=userdebug
ro.build.user=server22
```

```
ro.build.host=server-zysj-03
ro.build.tags=test-keys
ro.build.flavor=rk3288-userdebug
ro.product.model=rk3288
ro.product.brand=Android
ro.product.name=rk3288
ro.product.device=rk3288
ro.product.board=rk30sdk
# ro.product.cpu.abi and ro.product.cpu.abi2 are obsolete,
# use ro.product.cpu.abilist instead.
ro.product.cpu.abi=armeabi-v7a
ro.product.cpu.abi2=armeabi
ro.product.cpu.abilist=armeabi-v7a,armeabi
ro.product.cpu.abilist32=armeabi-v7a,armeabi
ro.product.cpu.abilist64=
ro.product.manufacturer=rockchip
ro.product.locale.language=zh
ro.product.locale.region=CN
persist.sys.timezone=Asia/Shanghai
ro.wifi.channels=
ro.board.platform=rk3288
# ro.build.product is obsolete; use ro.product.device
ro.build.product=rk3288
# Do not try to parse description, fingerprint, or thumbprint
ro.build.description=rk3288-userdebug 7.1.2 NHG47K
eng.server22.20230423.034518 test-keys
ro.build.fingerprint=Android/rk3288/rk3288:7.1.2/NHG47K/server0
4230345:userdebug/test-keys
ro.build.characteristics=tablet
# end build properties
#
# from device/rockchip/rk3288/system.prop
#
#
# system.prop
#

# modify by alvin, support for 4G patch.
rild.libpath=/system/lib/libreference-ril.so
rild.libargs=-d /dev/ttyUSB3
# Default ecclist
ro.ril.ecclist=112,911
ro.opengles.version=196610
wifi.interface=wlan0
# modify by alvin, support for 4G patch.
```

```
#rild.libpath=/system/lib/libril-rk29-dataonly.so
#rild.libargs=-d /dev/ttyACM0
persist.tegra.nvmmlite = 1
ro.audio.monitorOrientation=true

#NFC
debug.nfc.fw_download=false
debug.nfc.se=false

#add Rockchip properties here
ro.rk.screenoff_time=2147483647
ro.rk.screenshot_enable=true
ro.rk.def_brightness=200
ro.rk.homepage_base=http://m.baidu.com/?from=844&vit=fps
ro.rk.install_non_market_apps=false
sys.hwc.compose_policy=0
sys.wallpaper.rgb565=0
sf.power.control=2073600
sys.rkadb.root=0
ro.sf.fakerotation=false
ro.sf.hwrotation=0
ro.rk.MassStorage=false
ro.rk.systembar.voiceicon=true
ro.rk.systembar.tabletUI=false
ro.rk.LowBatteryBrightness=false
ro.tether.denied=false
#repair by alvin, surport change system density value.
sys.resolution.changed=true
ro.default.size=100
#persist.sys.timezone=
ro.product.usbfactory=rockchip_usb
wifi.supplicant_scan_interval=15
ro.factory.tool=0
ro.kernel.android.checkjni=0
#set default lcd density to Rockchip tablet
ro.sf.lcd_density=240
ro.adb.secure=0
ro.rk.displayd.enable=false

#/*add by yfc for show vendor id*/
ro.source.code.version = 220

#add by alvin for hdmi rotation
ro.same.orientation=true
ro.orientation.einit=0
```

```
ro.rotation.external=true

#add by alvin, support for config camera rotation.
ro.camera.param.degree=0
ro.camera.back=0
ro.camera.place=0
# add by alvin for system 4k ui
# default main framebuffer resolution
persist.sys.framebuffer.main=1536x2048
sys.hwc.device.primary=HDMI-A

#
# ADDITIONAL_BUILD_PROPERTIES
#
ro.target.product=tablet
dalvik.vm.heapstartsize=16m
dalvik.vm.heapgrowthlimit=480m
dalvik.vm.heapsize=520m
dalvik.vm.heaptargetutilization=0.75
dalvik.vm.heapminfree=512k
dalvik.vm.heapmaxfree=8m
ro.config.ringtone=Ring_Synth_04.ogg
ro.config.notification_sound=pixiedust.ogg
ro.carrier=unknown
ro.config.alarm_alert=Alarm_Classic.ogg
ro.rksdk.version=RK30_ANDROID7.1.2-SDK-v1.00.00
camera2.portability.force_api=1
persist.sys.strictmode.visual=false
ro.rk.bt_enable=true
ro.rk.flash_enable=true
ro.rk.hdmi_enable=true
ro.factory.hasUMS=false
persist.sys.usb.config=mtp,adb
testing.mediascanner.skiplist=/mnt/shell/emulated/Android/
ro.factory.hasGPS=false
ro.factory.storage_suppntfs=true
ro.factory.without_battery=false
ro.rk.screenoff_time=2147483647
ro.com.widevine.cachesize=16777216
ro.enable.optee=true
ro.product.first_api_level=23
ro.boot.noril=true
keyguard.no_require_sim=true
ro.com.android.dataroaming=true
ril.function.dataonly=1
```

```
ro.config.enable.remotecontrol=false
ro.udisk.visible=true
ro.safemode.disabled=true
ro.wallpaper.fixsize=true
ro.hwui.texture_cache_size=72
ro.hwui.layer_cache_size=48
ro.hwui.r_buffer_cache_size=8
ro.hwui.path_cache_size=32
ro.hwui.gradient_cache_size=1
ro.hwui.drop_shadow_cache_size=6
ro.hwui.texture_cache_flushrate=0.4
ro.hwui.text_small_cache_width=1024
ro.hwui.text_small_cache_height=1024
ro.hwui.text_large_cache_width=2048
ro.hwui.text_large_cache_height=1024
ro.hwui.disable_scissor_opt=true
ro.rk.screenshot_enable=true
sys.status.hidebar_enable=false
persist.sys.ui.hw=true
ro.product.version=1.0.0
ro.product.ota.host=www.rockchip.com:2300
ro.sys.sdcardfs=true
persist.sys.dalvik.vm.lib.2=libart.so
dalvik.vm.isa.arm.variant=cortex-a15
dalvik.vm.isa.arm.features=default
dalvik.vm.lockprof.threshold=500
net.bt.name=Android
dalvik.vm.stack-trace-file=/data/anr/traces.txt
ro.expect.recovery_id=0x182fbd9a6eea8693a3aeac4bfab86ba6271f55d
100000000000000000000000
```

获取指定的属性 adb shell getprop net.bt.name

```
neo@MacBook-Pro-M2 ~> adb shell getprop net.bt.name
Android
```

6.4. 设备 ID

获取变量

```
neo@MacBook-Pro-M2 ~> adb shell settings get secure android_id
95c27630f4559e58
```

设置变量

```
RK3566:/ # settings put global policy_control
immersive.full=*
RK3566:/ # settings put global policy_control
immersive.status=*
RK3566:/ # settings put global policy_control
immersive.navigation=*
```

显示/关闭虚拟键

设置属性值为0表示一直打开虚拟按键，属性值为1 表示隐藏虚拟按键

```
adb root
adb remount
adb shell
$ getprop qemu.hw.mainkeys
$ setprop qemu.hw.mainkeys 0
$ stop
$ start
```

```
setprop persist.qemu.hw.mainkeys 0
```

6.5. 查看安卓版本

```
neo@MacBook-Pro-M2 ~> adb shell getprop
ro.build.version.release
7.1.2
```

产品型号

```
neo@MacBook-Pro-M2 ~> adb shell getprop ro.product.model
rk3288
```

6.6. Logcat

```
neo@MacBook-Pro-M2 ~> adb logcat
neo@MacBook-Pro-M2 ~> adb -s CFE6R21625003544 logcat
```

6.7. 上传文件

```
adb push libtinyalsa.so /system/lib/
```

```
neo@MacBook-Pro-M2 ~> adb push netkiller.wav /sdcard/
```

6.8. 下载文件

```
neo@MacBook-Pro-M2 tmp % adb pull /sdcard/file.wav
/sdcard/file.wav: 1 file pulled, 0 skipped. 0.0 MB/s (44 bytes
in 0.002s)
```

6.9. 安卓 .apk bk

```
        -l ： 锁定应用程序
        -t ： 允许测试包
        -d ： 允许降级覆盖安装
        -p ： 部分应用安装
        -g ： 为应用程序授予所有运行时的权限
```

```
neo@MacBook-Pro-M2 ~> adb install netkiller.apk
```

6.10. 屏幕尺寸

```
```

```
neo@MacBook-Pro-M2 ~> adb shell wm size
Physical size: 1536x2048

neo@MacBook-Pro-M2 ~> adb shell wm density
Physical density: 240
```

```
neo@MacBook-Pro-M2 ~> adb shell dumpsys window displays
WINDOW MANAGER DISPLAY CONTENTS (dumpsys window displays)
  Display: mDisplayId=0
    init=1536x2048 240dpi cur=1536x2048 app=1536x1964
rng=1536x1416-2048x1928
    deferred=false layoutNeeded=false

  Application tokens in top down Z order:
    mStackId=1
    mDeferDetach=false
    mFullscreen=true
    mBounds=[0,0][1536,2048]
      taskId=56
        mFullscreen=true
        mBounds=[0,0][1536,2048]
        mdr=false
        appTokens=[AppWindowToken{1c392c token=Token{ff4ab7e
ActivityRecord{379f939 u0
com.wc.holoos/.player.PlayClockActivity t56}}}]
        mTempInsetBounds=[0,0][0,0]
          Activity #0 AppWindowToken{1c392c token=Token{ff4ab7e
ActivityRecord{379f939 u0
com.wc.holoos/.player.PlayClockActivity t56}}}
          windows=[Window{73ec1eb u0
com.wc.holoos/com.wc.holoos.player.PlayClockActivity}]
          windowType=2 hidden=false hasVisible=true
          app=true voiceInteraction=false
          allAppWindows=[Window{73ec1eb u0
com.wc.holoos/com.wc.holoos.player.PlayClockActivity}]
          task={taskId=56 appTokens=[AppWindowToken{1c392c
token=Token{ff4ab7e ActivityRecord{379f939 u0
com.wc.holoos/.player.PlayClockActivity t56}}}] mdr=false}
           appFullscreen=true requestedOrientation=1
          hiddenRequested=false clientHidden=false
```

```
reportedDrawn=true reportedVisible=true
          numInterestingWindows=1 numDrawnWindows=1
inPendingTransaction=false allDrawn=true (animator=true)
          startingData=null removed=false firstWindowDrawn=true
mIsExiting=false
    mStackId=0
    mDeferDetach=false
    mFullscreen=true
    mBounds=[0,0][1536,2048]
      taskId=55
        mFullscreen=true
        mBounds=[0,0][1536,2048]
        mdr=false
        appTokens=[AppWindowToken{56cdf68 token=Token{362805a
ActivityRecord{b30e805 u0 com.wc.holoos/.MainActivity t55}}}]
        mTempInsetBounds=[0,0][0,0]
          Activity #0 AppWindowToken{56cdf68
token=Token{362805a ActivityRecord{b30e805 u0
com.wc.holoos/.MainActivity t55}}}
          windows=[Window{90133ba u0
com.wc.holoos/com.wc.holoos.MainActivity}]
          windowType=2 hidden=true hasVisible=true
          app=true voiceInteraction=false
          allAppWindows=[Window{90133ba u0
com.wc.holoos/com.wc.holoos.MainActivity}]
          task={taskId=55 appTokens=[AppWindowToken{56cdf68
token=Token{362805a ActivityRecord{b30e805 u0
com.wc.holoos/.MainActivity t55}}}] mdr=false}
           appFullscreen=true requestedOrientation=1
          hiddenRequested=true clientHidden=true
reportedDrawn=false reportedVisible=false
          mAppStopped=true
          numInterestingWindows=1 numDrawnWindows=1
inPendingTransaction=false allDrawn=true (animator=true)
          startingData=null removed=false firstWindowDrawn=true
mIsExiting=false


    DimLayerController
      Task=55
        dimLayer=shared, animator=null, continueDimming=false
        mDimSurface=Surface(name=DimLayerController/Stack=0)
mLayer=110999 mAlpha=0.0
        mLastBounds=[-384,-512][1920,2560] mBounds=[-384,-512]
[1920,2560]
```

```
        Last animation:  mDuration=200 mStartTime=7877723
curTime=9020147
         mStartAlpha=0.6 mTargetAlpha=0.0
      Task=56
        dimLayer=shared, animator=null, continueDimming=false
        mDimSurface=Surface(name=DimLayerController/Stack=0)
mLayer=110999 mAlpha=0.0
        mLastBounds=[-384,-512][1920,2560] mBounds=[-384,-512]
[1920,2560]
        Last animation:  mDuration=200 mStartTime=7877723
curTime=9020147
         mStartAlpha=0.6 mTargetAlpha=0.0
      Stack=1
        dimLayer=shared, animator=null, continueDimming=false
        mDimSurface=Surface(name=DimLayerController/Stack=0)
mLayer=110999 mAlpha=0.0
        mLastBounds=[-384,-512][1920,2560] mBounds=[-384,-512]
[1920,2560]
        Last animation:  mDuration=200 mStartTime=7877723
curTime=9020147
         mStartAlpha=0.6 mTargetAlpha=0.0
      Stack=0
        dimLayer=shared, animator=null, continueDimming=false
        mDimSurface=Surface(name=DimLayerController/Stack=0)
mLayer=110999 mAlpha=0.0
        mLastBounds=[-384,-512][1920,2560] mBounds=[-384,-512]
[1920,2560]
        Last animation:  mDuration=200 mStartTime=7877723
curTime=9020147
         mStartAlpha=0.6 mTargetAlpha=0.0

    DockedStackDividerController
      mLastVisibility=false
      mMinimizedDock=false
      mAdjustedForIme=false
      mAdjustedForDivider=false
```

查看 **dpi**

dpi

```
rk3568_r:/ $ wm size
Physical size: 1536x2048

rk3568_r:/ $ wm density
Physical density: 240


rk3566_rgo:/ # wm size
Physical size: 1536x2048

rk3566_rgo:/ # wm density
Physical density: 680
```

设置 dpi

```
RK3566:/ $ wm density 480
RK3566:/ $ wm density 320
```

6.11. dump 系统信息

```
neo@MacBook-Pro-M2 ~> adb shell dumpsys
```

电池信息

```
neo@MacBook-Pro-M2 ~> adb shell dumpsys battery
Current Battery Service state:
  AC powered: true
```

```
  USB powered: false
  Wireless powered: false
  Max charging current: 0
  Max charging voltage: 0
  Charge counter: 0
  status: 2
  health: 2
  present: true
  level: 100
  scale: 100
  voltage: 0
  temperature: 424
  technology:
```

6.12. 解锁

```
neo@MacBook-Pro-M2 ~ % adb root
neo@MacBook-Pro-M2 ~ % adb reboot bootloader
neo@MacBook-Pro-M2 ~ % fastboot flashing unlock
neo@MacBook-Pro-M2 ~ % fastboot getvar unlocked
neo@MacBook-Pro-M2 ~ % adb disable-verity
neo@MacBook-Pro-M2 ~ % adb reboot
neo@MacBook-Pro-M2 ~ % adb root
neo@MacBook-Pro-M2 ~ % adb remount
```

6.13. 蓝牙管理

蓝牙相关adb命令

查看蓝牙信息

```
neo@MacBook-Pro-M2 ~ % adb shell dumpsys bluetooth_manager
Bluetooth Status
  enabled: true
```

```
  state: ON
  address: 22:22:9D:4D:03:00
  name: Bluetooth
  time since enabled: 00:12:52.801
```

获取蓝牙开关状态

```
adb shell settings get global bluetooth_on
```

返回1表示开启，0表示关闭

```
neo@MacBook-Pro-M2 ~ % adb shell settings get global
bluetooth_on
1
```

打开蓝牙

```
adb shell service call bluetooth_manager 6
```

关闭蓝牙

```
adb shell service call bluetooth_manager 8
```

允许被发现，此时屏幕会弹出，需要按键确认

```
adb shell am start -a
android.bluetooth.adapter.action.REQUEST_DISCOVERABLE
```

获取蓝牙MAC地址

```
adb shell settings get secure bluetooth_address
```

第 29 章 AndroidManifest.xml

1. SDK 版本配置

```
<uses-sdk
        android:minSdkVersion="26"
        android:targetSdkVersion="28" />
```

2. 开启网络

```
<uses-permission android:name="android.permission.INTERNET" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.android.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />

            </intent-filter>
        </activity>

    </application>
    <uses-permission android:name="android.permission.INTERNET"
/>
</manifest>
```

3. 文件存储权限

```
        <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"
/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

sdcard

```
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />
```

4. 相机权限

```
        <uses-permission
android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
```

5. GPS 定位权限

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.INTERNET"
/>
```

6. 全屏-无标题

首先定义主题

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Theme.Main.Fullscreen"
parent="Theme.AppCompat.Light">
        <item name="windowActionBar">false</item>//无ActionBar
        <item name="windowNoTitle">true</item> //无标题
        <item name="android:windowFullscreen">true</item> //全
屏
        <item name="android:navigationBarColor"
tools:targetApi="lollipop">@android:color/transparent</item>
        <item
name="android:windowBackground">@drawable/fo2</item>//背景图
        <item name="android:windowDrawsSystemBarBackgrounds"
tools:targetApi="lollipop">true</item>
    </style>
</resources>
```

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:context="cn.netkiller.demo.MainActivity">

    <ImageView
        android:id="@+id/imageView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="matrix"
        app:srcCompat="@drawable/fo2"
        tools:ignore="MissingConstraints" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

AndroidManifest.xml 切换主题

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android">

    <uses-permission
android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.RECORD_AUDIO" />

    <application
        android:name="cn.netkiller.demo.ContextHolder"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Demo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

```
        android:theme="@style/Theme.Main.Fullscreen"
        android:usesCleartextTraffic="true">
        <activity
            android:name="cn.netkiller.demo.MainActivity"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>


    </application>

</manifest>
```

7. 设置为默认开机启动

```
<category android:name="android.intent.category.HOME" />
<category android:name="android.intent.category.DEFAULT" />
```

```
        <activity
            android:name="cn.netkiller.album.MainActivity"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
                <category
android:name="android.intent.category.HOME" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
```

8. 开机启动

AndroidManifest.xml 配置

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET"
/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="android.permission.RECORD_AUDIO" />
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:name=".ContextHolder"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Demo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Main.Fullscreen"
```

```
        android:usesCleartextTraffic="true">
        <service
            android:name=".service.VoiceInteractionService"
            android:enabled="true"
            android:exported="true"></service>
        <service
            android:name=".service.InfraredSensorService"
            android:enabled="true"
            android:exported="true" />

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
                <category
android:name="android.intent.category.HOME" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

        <receiver
            android:name=".BootReceiver"
            android:enabled="true"
            android:exported="true"
            tools:ignore="WrongManifestParent">
            <intent-filter android:priority="1000">
                <action
android:name="android.intent.action.BOOT_COMPLETED" />

                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>

        <service
            android:name=".service.MessageQueueService"
            android:enabled="true"
            android:exported="true" />
    </application>
```

```
</manifest>
```

receiver 配置

```
        <receiver
            android:name=".BootReceiver"
            android:enabled="true"
            android:exported="true"
            tools:ignore="WrongManifestParent">
            <intent-filter android:priority="1000">
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>
```

```
package cn.netkiller.album;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

public class BootReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if
(intent.getAction().equals(Intent.ACTION_BOOT_COMPLETED)) {
            Log.e("BootReceiver","自启动了 ！！！！！ ");
            intent = new Intent(context, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent);
        }
    }
```

```
}
```

9. 默认横屏

android:screenOrientation="landscape"

```
            <activity
            android:name=".MainActivity"
            android:exported="true"
            android:screenOrientation="landscape">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
                <category
android:name="android.intent.category.HOME" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
```

10. 禁止屏幕旋转变化

landscape = 横向，portrait = 纵向

```
android:screenOrientation="landscape"
```

第 30 章 设备

1. 环境变量

1.1. 扩展存储

状态

```
Environment.getExternalStorageState().equals(Environment.MEDI
A_MOUNTED);
```

绝对路径

```
Environment.getExternalStorageDirectory().getAbsolutePath();
```

```
Log.i("Environment", "getExternalStorageDirectory(): " +
Environment.getExternalStorageDirectory().toString());

Log.i("Environment","getExternalStoragePublicDirectory(Enviro
nment.DIRECTORY_PICTURES): " +
Environment.getExternalStoragePublicDirectory(Environment.DIR
ECTORY_PICTURES).toString());
```

1.2. 下载缓存目录

```
Log.i("Environment", "getDownloadCacheDirectory(): "+
Environment.getDownloadCacheDirectory().toString());
```

1.3. 数据目录

```
Log.i("Environment", "getRootDirectory(): " +
Environment.getRootDirectory().toString());
Log.i("Environment", "getDataDirectory(): " +
Environment.getDataDirectory().toString());
```

2. 配置文件

2.1. *.properties 文件

创建 properties 文件 res/config/development.properties

```
api_url=https://api.netkiller.cn/v1/
api_key=123456
```

```
package cn.netkiller.app;

import android.content.Context;
import android.content.res.Resources;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public final class Config {
    private static final String TAG = "Config";

    public static String getKey(Context context, String name)
{
        Resources resources = context.getResources();

        try {
            InputStream rawResource =
resources.openRawResource(R.config.development);
            Properties properties = new Properties();
            properties.load(rawResource);
            return properties.getProperty(name);
        } catch (Resources.NotFoundException e) {
            Log.e(TAG, "Unable to find the config file: " +
e.getMessage());
```

```
        } catch (IOException e) {
            Log.e(TAG, "Failed to open config file.");
        }

        return null;
    }
}
```

```
String apiUrl = Config.getKey(this, "api_url");
String apiKey = Config.getKey(this, "api_key");
```

2.2. 再 AndroidManifest.xml 使用 meta-data element 定义

```
...
<application ...>
    ...
        ...
    <meta-data android:name="api_url"
android:value="https://api.netkiller.cn/v1/"/>
    <meta-data android:name="api_key"
android:value="123456"/>
</application>
```

```
public static String getMetaData(Context context, String
name) {
    try {
        ApplicationInfo ai =
context.getPackageManager().getApplicationInfo(context.getPac
kageName(), PackageManager.GET_META_DATA);
        Bundle bundle = ai.metaData;
```

```
        return bundle.getString(name);
    } catch (PackageManager.NameNotFoundException e) {
        Log.e(TAG, "Unable to load meta-data: " +
e.getMessage());
    }
    return null;
}
```

```
String apiUrl = getMetaData(this, "api_url");
String apiKey = getMetaData(this, "api_key");
```

2.3. 再 build.gradle 文件中配置 productFlavors

```
productFlavors {
    prod {
        buildConfigField 'String', 'API_URL',
'"https://api.netkiller.cn/v1/"'
        buildConfigField 'String', 'API_KEY', '"123456"'
    }
}
```

引用 config 方法

```
String apiUrl = BuildConfig.API_URL;
String apiKey = BuildConfig.API_KEY;
```

2.4. 从 assets 目录读取配置文件

```
import java.io.InputStream;
import java.util.Properties;

import android.content.Context;

public class Config {
        public static Properties getProperties(Context c){
                Properties properties = new Properties();
                try {
                        //方法一：通过activity中的context攻取
setting.properties的FileInputStream
                        InputStream in =
c.getAssets().open("appConfig.properties");
                //方法二：通过class获取setting.properties的
FileInputStream
                //InputStream in =
PropertiesUtill.class.getResourceAsStream("/assets/setting.pr
operties "));
                properties.load(in);
                } catch (Exception e1) {
                        e1.printStackTrace();
                }
                return props;
        }
}
```

```
Properties properties =
Config.getProperties(context.getApplicationContext());
serverUrl = properties.getProperty("serverUrl");
Log.i("URL", serverUrl);
```

配置文件例子

```
package cn.netkiller.album.config;

import cn.netkiller.album.MainApp;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
import java.util.Properties;

public class Config {
    private static final String TAG = "Config";
    private Properties properties = null;
    public Config() {
        try {
            properties = new Properties();
            InputStream inputStream =
MainApp.getContext().getAssets().open("config.properties");
            InputStreamReader isr = new
InputStreamReader(inputStream, StandardCharsets.UTF_8);
            properties.load(isr);
            isr.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    public String getServerURI() {
        return
properties.getProperty("mqtt.serveruri").trim();
    }

    public String getUsername() {
        return
properties.getProperty("mqtt.username").trim();
    }

    public String getPassword() {
        return
properties.getProperty("mqtt.password").trim();
    }
```

```
    public String getProjectName() {
        return properties.getProperty("project.name").trim();
    }

    public String getProjectScenes() {
        return
properties.getProperty("project.scenes").trim();
    }

    public String getProjectGroup() {
        return
properties.getProperty("project.group").trim();
    }

    public String getTopicBroadcast() {

        return
properties.getProperty("mqtt.topic.broadcast").trim();
    }

    public String getTopicRegister() {
        return
properties.getProperty("mqtt.topic.register").trim();
    }
    public String getTopicNotification(){
        return
properties.getProperty("mqtt.topic.notification").trim();
    }
}
```

3. 设备信息

设备 ID

```
String androidId =
Settings.System.getString(getContentResolver(),
"android_id");
```

4. Physical density

```
SystemProperties.getInt("qemu.sf.lcd_density",
SystemProperties.getInt("ro.sf.lcd_density",
DENSITY_DEFAULT));
```

5. 声卡

```
rk3288:/ $ cd /dev/snd/
rk3288:/dev/snd $ ls
controlC0 pcmC0D0c pcmC0D0p pcmC0D1p timer
```

5.1. 播放

播放测试

```
aiv8167sm3_bsp:/storage/emulated/0 # tinyplay zai.wav
Playing sample: 1 ch, 16000 hz, 16 bit
```

```
tinyplay netkiller.wav -D 0 -d 0 -r 48000 -c 2

aiv8167sm3_bsp:/storage/emulated/0 # tinyplay zai.wav -D 0 -d 0
-r 48000
Playing sample: 1 ch, 16000 hz, 16 bit
```

```
neo@MacBook-Pro-M2 tmp % adb shell tinyplay /sdcard/zai.wav
Playing sample: 1 ch, 16000 hz, 16 bit
```

5.2. 录音

录音测试

```
tinycap netkiller.wav -D 0 -d 0 -c 1 -r 48000
```

```
-D  card         声卡

-d  device      设备

-c  channels   通道

-r  rate    采样率

-b  bits    pcm 位宽

-p  period_size    一次中断的帧数

-n  n_periods      周期数

例子:  tinycap /sdcard/test.pcm -D 0 -d 0 -c 4 -r 48000 -b 32 -p
768 -n 10

声卡0；设备0；四通道；48K采样率；32位位宽；一帧数据存储大小；采样n次
```

查看录音设备

```
neo@MacBook-Pro-M2 tmp % adb shell ls "/dev/snd/pcmC*c"
/dev/snd/pcmC0D10c
/dev/snd/pcmC0D1c
/dev/snd/pcmC0D2c
/dev/snd/pcmC0D4c
/dev/snd/pcmC0D6c
/dev/snd/pcmC0D8c
/dev/snd/pcmC0D9c
```

默认录音参数 Capturing sample: 2 ch, 44100 hz, 16 bit

```
neo@MacBook-Pro-M2 tmp % adb shell tinycap /sdcard/file.pcm -D
0 -d 2 -T 5
Capturing sample: 2 ch, 44100 hz, 16 bit
Captured 0 frames
```

指定参数

```
neo@MacBook-Pro-M2 tmp % adb pull /sdcard/file.pcm
/sdcard/file.wav: 1 file pulled, 0 skipped. 14.6 MB/s (1851436
bytes in 0.121s)
neo@MacBook-Pro-M2 tmp % adb shell tinycap /sdcard/file.wav -D
0 -d 2 -c 2 -r 48000 -b 16 -T 5
Capturing sample: 2 ch, 48000 hz, 16 bit
Captured 0 frames
```

下载录音文件

```
neo@MacBook-Pro-M2 tmp % adb pull /sdcard/file.pcm
/sdcard/file.wav: 1 file pulled, 0 skipped. 7.0 MB/s (966700
bytes in 0.132s)
```

5.3. 查看声卡信息

device 0 表示录音设备

```
aiv8167sm3_bsp:/storage/emulated/0 # tinypcminfo -D 0 -d 0
Info for card 0, device 0:

PCM out:
      Access:   0x000009
   Format[0]:   0x000444
   Format[1]:   00000000
 Format Name:   S16_LE, S24_LE, S32_LE
   Subformat:   0x000001
        Rate:   min=8000Hz      max=192000Hz
    Channels:   min=1           max=2
 Sample bits:   min=16          max=32
 Period size:   min=32          max=32768
Period count:   min=2           max=256

PCM in:
cannot open device '/dev/snd/pcmC0D0c'
Device does not exist.
```

device 1 表示录音设备

```
aiv8167sm3_bsp:/storage/emulated/0 # tinypcminfo -D 0 -d 1
Info for card 0, device 1:

PCM out:
cannot open device '/dev/snd/pcmC0D1p'
Device does not exist.

PCM in:
      Access:   0x000009
   Format[0]:   0x000444
   Format[1]:   00000000
 Format Name:   S16_LE, S24_LE, S32_LE
   Subformat:   0x000001
        Rate:   min=8000Hz      max=192000Hz
    Channels:   min=1           max=2
 Sample bits:   min=16          max=32
```

```
 Period size:   min=32          max=8192
Period count:   min=2           max=256
```

如果不知道设备编号，可以使用 /proc/asound/cards 替代

```
aiv8167sm3_bsp:/storage/emulated/0 # tinypcminfo -D
/proc/asound/cards
Info for card 0, device 0:

PCM out:
      Access:   0x000009
   Format[0]:   0x000444
   Format[1]:   00000000
 Format Name:   S16_LE, S24_LE, S32_LE
   Subformat:   0x000001
        Rate:   min=8000Hz      max=192000Hz
    Channels:   min=1           max=2
 Sample bits:   min=16          max=32
 Period size:   min=32          max=32768
Period count:   min=2           max=256

PCM in:
cannot open device '/dev/snd/pcmC0D0c'
Device does not exist.
```

5.4. /proc/asound 设备信息

```
aiv8167sm3_bsp:/storage/emulated/0 # ls -1 /proc/asound
card0
cards
devices
hwdep
mtsndcard
oss
```

```
pcm
seq
timers
version
```

查看当前的声卡

```
aiv8167sm3_bsp:/storage/emulated/0 # cat /proc/asound/cards
 0 [mtsndcard      ]: mt-snd-card - mt-snd-card
                      mt-snd-card
```

```
aiv8167sm3_bsp:/storage/emulated/0 # cat /proc/asound/pcm
00-00: MultiMedia1_PLayback (*) :  : playback 1
00-01: MultiMedia_Capture (*) :  : capture 1
00-02: TDM_Capture (*) :  : capture 1
00-03: HMDI_PLayback (*) :  : playback 1
00-04: DL1_AWB_Record (*) :  : capture 1
00-05: MultiMedia2_PLayback (*) :  : playback 1
00-06: VOIP_Call_BT_Capture (*) :  : capture 1
00-07: MRGRX_PLayback (*) :  : playback 1
00-08: MRGRX_CAPTURE (*) :  : capture 1
00-09: BTCVSD_Capture snd-soc-dummy-dai-9 :  : playback 1 :
capture 1
00-10: BTCVSD_Playback snd-soc-dummy-dai-10 :  : playback 1 :
capture 1
```

```
aiv8167sm3_bsp:/storage/emulated/0 # cat /proc/asound/pcm
00-00: MultiMedia1_PLayback (*) :  : playback 1
00-01: MultiMedia_Capture (*) :  : capture 1
00-02: TDM_Capture (*) :  : capture 1
00-03: HMDI_PLayback (*) :  : playback 1
00-04: DL1_AWB_Record (*) :  : capture 1
```

```
00-05: MultiMedia2_PLayback (*) :  : playback 1
00-06: VOIP_Call_BT_Capture (*) :  : capture 1
00-07: MRGRX_PLayback (*) :  : playback 1
00-08: MRGRX_CAPTURE (*) :  : capture 1
00-09: BTCVSD_Capture snd-soc-dummy-dai-9 :  : playback 1 :
capture 1
00-10: BTCVSD_Playback snd-soc-dummy-dai-10 :  : playback 1 :
capture 1
```

5.5. 查看声卡当前占用设备

```
aiv8167sm3_bsp:/storage/emulated/0 # lsof |grep pcm
audio@2.0   253 audioserve   16u      CHR              116,3
0t0       8644 /dev/snd/pcmC0D0p
omx@1.0-s   316 mediacodec  mem       REG             179,14
57968        603 /vendor/lib/libMtkOmxAdpcmDec.so
m.netkiller  2956     u0_a51  mem   unknown
/dev/ashmem//data/user/0/cn.netkiller.ui/files/msclib/1691060
275989tts.pcm (deleted)
m.netkiller  2956     u0_a51   54u      CHR
116,5       0t0       9451 /dev/snd/pcmC0D2c
```

5.6. tinymix 设置声卡参数

查看所有参数

```
rk3288:/storage/emulated/0 # tinymix
Mixer name: 'rockchip,rt5640-codec'
Number of controls: 125
ctl     type    num     name
value
0       BOOL    1       Mono Playback Switch
```

```
Off
1       INT     2       Mono DAC Playback Volume
175 175
2       BOOL    2       Speaker Channel Switch
Off Off
3       INT     2       Speaker Playback Volume
31 31
4       BOOL    2       HP Channel Switch
Off Off
5       INT     2       HP Playback Volume
31 31
6       BOOL    2       OUT Playback Switch
Off Off
7       BOOL    2       OUT Channel Switch
Off Off
8       INT     2       OUT Playback Volume
31 31
9       BOOL    2       DAC2 Playback Switch
On On
10      INT     2       DAC1 Playback Volume
175 175
11      INT     1       IN1 Boost
0
12      INT     1       IN2 Boost
0
13      INT     1       IN3 Boost
0
14      INT     2       IN Capture Volume
23 23
15      BOOL    2       ADC Capture Switch
On On
16      INT     2       ADC Capture Volume
47 47
17      INT     2       Mono ADC Capture Volume
47 47
18      INT     2       ADC Boost Gain
0 0
19      ENUM    1       Class D SPK Ratio Control
2.77x
20      ENUM    1       ADC IF1 Data Switch
Normal
21      ENUM    1       DAC IF1 Data Switch
Normal
22      ENUM    1       ADC IF2 Data Switch
Normal
```

```
23      ENUM    1       DAC IF2 Data Switch
Normal
24      BYTE    128     ELD
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000
25      INT     1       AUDIO MODE
0
26      BOOL    1       RECMIXL HPOL Switch
Off
27      BOOL    1       RECMIXL INL Switch
Off
28      BOOL    1       RECMIXL BST3 Switch
Off
29      BOOL    1       RECMIXL BST2 Switch
Off
30      BOOL    1       RECMIXL BST1 Switch
Off
31      BOOL    1       RECMIXL OUT MIXL Switch
Off
32      BOOL    1       RECMIXR HPOR Switch
Off
33      BOOL    1       RECMIXR INR Switch
Off
34      BOOL    1       RECMIXR BST3 Switch
Off
35      BOOL    1       RECMIXR BST2 Switch
Off
36      BOOL    1       RECMIXR BST1 Switch
Off
37      BOOL    1       RECMIXR OUT MIXR Switch
Off
38      ENUM    1       Stereo ADC2 Mux
DMIC1
39      ENUM    1       Stereo ADC1 Mux
ADC
40      ENUM    1       Mono ADC L2 Mux
DMIC L1
41      ENUM    1       Mono ADC L1 Mux
ADCL
42      ENUM    1       Mono ADC R1 Mux
ADCR
43      ENUM    1       Mono ADC R2 Mux
```

```
DMIC R1
44      BOOL    1       Stereo ADC MIXL ADC1 Switch
Off
45      BOOL    1       Stereo ADC MIXL ADC2 Switch
Off
46      BOOL    1       Stereo ADC MIXR ADC1 Switch
Off
47      BOOL    1       Stereo ADC MIXR ADC2 Switch
Off
48      BOOL    1       Mono ADC MIXL ADC1 Switch
Off
49      BOOL    1       Mono ADC MIXL ADC2 Switch
Off
50      BOOL    1       Mono ADC MIXR ADC1 Switch
Off
51      BOOL    1       Mono ADC MIXR ADC2 Switch
Off
52      ENUM    1       DAI select
1:2|2:1
53      ENUM    1       SDI select
IF1
54      BOOL    1       DAC MIXL Stereo ADC Switch
Off
55      BOOL    1       DAC MIXL INF1 Switch
On
56      BOOL    1       DAC MIXR Stereo ADC Switch
Off
57      BOOL    1       DAC MIXR INF1 Switch
On
58      BOOL    1       Mono DAC MIXL DAC L1 Switch
Off
59      BOOL    1       Mono DAC MIXL DAC L2 Switch
On
60      BOOL    1       Mono DAC MIXL DAC R2 Switch
Off
61      BOOL    1       Mono DAC MIXR DAC R1 Switch
Off
62      BOOL    1       Mono DAC MIXR DAC R2 Switch
On
63      BOOL    1       Mono DAC MIXR DAC L2 Switch
Off
64      BOOL    1       DIG MIXL DAC L1 Switch
Off
65      BOOL    1       DIG MIXL DAC L2 Switch
Off
```

```
66      BOOL    1       DIG MIXR DAC R1 Switch
Off
67      BOOL    1       DIG MIXR DAC R2 Switch
Off
68      BOOL    1       SPK MIXL REC MIXL Switch
Off
69      BOOL    1       SPK MIXL INL Switch
Off
70      BOOL    1       SPK MIXL DAC L1 Switch
Off
71      BOOL    1       SPK MIXL DAC L2 Switch
Off
72      BOOL    1       SPK MIXL OUT MIXL Switch
Off
73      BOOL    1       SPK MIXR REC MIXR Switch
Off
74      BOOL    1       SPK MIXR INR Switch
Off
75      BOOL    1       SPK MIXR DAC R1 Switch
Off
76      BOOL    1       SPK MIXR DAC R2 Switch
Off
77      BOOL    1       SPK MIXR OUT MIXR Switch
Off
78      BOOL    1       SPOL MIX DAC R1 Switch
Off
79      BOOL    1       SPOL MIX DAC L1 Switch
Off
80      BOOL    1       SPOL MIX SPKVOL R Switch
Off
81      BOOL    1       SPOL MIX SPKVOL L Switch
Off
82      BOOL    1       SPOL MIX BST1 Switch
Off
83      BOOL    1       SPOR MIX DAC R1 Switch
Off
84      BOOL    1       SPOR MIX SPKVOL R Switch
Off
85      BOOL    1       SPOR MIX BST1 Switch
Off
86      BOOL    1       LOUT MIX DAC L1 Switch
Off
87      BOOL    1       LOUT MIX DAC R1 Switch
Off
88      BOOL    1       LOUT MIX OUTVOL L Switch
```

```
Off
89      BOOL    1       LOUT MIX OUTVOL R Switch
Off
90      BOOL    1       Speaker L Playback Switch
Off
91      BOOL    1       Speaker R Playback Switch
Off
92      BOOL    1       HP L Playback Switch
On
93      BOOL    1       HP R Playback Switch
On
94      ENUM    1       DAC L2 Mux
IF2
95      ENUM    1       DAC R2 Mux
IF2
96      BOOL    1       Stereo DAC MIXL DAC L1 Switch
Off
97      BOOL    1       Stereo DAC MIXL DAC L2 Switch
Off
98      BOOL    1       Stereo DAC MIXL ANC Switch
Off
99      BOOL    1       Stereo DAC MIXR DAC R1 Switch
Off
100     BOOL    1       Stereo DAC MIXR DAC R2 Switch
Off
101     BOOL    1       Stereo DAC MIXR ANC Switch
Off
102     BOOL    1       OUT MIXL SPK MIXL Switch
Off
103     BOOL    1       OUT MIXL BST1 Switch
Off
104     BOOL    1       OUT MIXL INL Switch
Off
105     BOOL    1       OUT MIXL REC MIXL Switch
Off
106     BOOL    1       OUT MIXL DAC R2 Switch
Off
107     BOOL    1       OUT MIXL DAC L2 Switch
Off
108     BOOL    1       OUT MIXL DAC L1 Switch
Off
109     BOOL    1       OUT MIXR SPK MIXR Switch
Off
110     BOOL    1       OUT MIXR BST2 Switch
Off
```

```
111     BOOL    1       OUT MIXR BST1 Switch
Off
112     BOOL    1       OUT MIXR INR Switch
Off
113     BOOL    1       OUT MIXR REC MIXR Switch
Off
114     BOOL    1       OUT MIXR DAC L2 Switch
Off
115     BOOL    1       OUT MIXR DAC R2 Switch
Off
116     BOOL    1       OUT MIXR DAC R1 Switch
Off
117     BOOL    1       HPO MIX DAC2 Switch
On
118     BOOL    1       HPO MIX DAC1 Switch
Off
119     BOOL    1       HPO MIX HPVOL Switch
Off
120     BOOL    1       Mono MIX DAC R2 Switch
Off
121     BOOL    1       Mono MIX DAC L2 Switch
Off
122     BOOL    1       Mono MIX OUTVOL R Switch
Off
123     BOOL    1       Mono MIX OUTVOL L Switch
Off
124     BOOL    1       Mono MIX BST1 Switch
Off
```

查看指定参数

```
rk3288:/ $ tinymix 33
RECMIXR INR Switch: Off
```

设置参数

```
# 当前位 Off 状态
rk3288:/ $ tinymix 33
RECMIXR INR Switch: Off

# 修改位 On 状态
rk3288:/ $ tinymix 33 1

rk3288:/ $ tinymix 33
RECMIXR INR Switch: On

# 修改回 Off 状态
rk3288:/ $ tinymix 33 0

rk3288:/ $ tinymix 33
RECMIXR INR Switch: Off
```

5.7. 麦克风阵列调试

USB-Audio - Yundea M1051

```
rk3568_r:/ # cat /proc/asound/cards
 0 [rockchiphdmi   ]: rockchip_hdmi - rockchip,hdmi
                      rockchip,hdmi
 1 [rockchiprk809co]: rockchip_rk809- - rockchip,rk809-codec
                      rockchip,rk809-codec
 2 [M1051          ]: USB-Audio - Yundea M1051
                      Yundea Technology Yundea M1051 at usb-
xhci-hcd.5.auto-1, full speed
```

```
rk3568_r:/ # cat /proc/asound/card2/usbmixer
USB Mixer: usb_id=0x4c4a3135, ctrlif=1, ctlerr=0
Card: Yundea Technology Yundea M1051 at usb-xhci-hcd.5.auto-1,
full speed
```

```
  Unit: 5
    Control: name="Auto Gain Control", index=0
    Info: id=5, control=7, cmask=0x0, channels=1,
type="BOOLEAN"
    Volume: min=0, max=1, dBmin=0, dBmax=0
  Unit: 5
    Control: name="Mic Capture Volume", index=0
    Info: id=5, control=2, cmask=0x0, channels=1, type="S16"
    Volume: min=-7264, max=-241, dBmin=-2837, dBmax=-94
  Unit: 5
    Control: name="Mic Capture Switch", index=0
    Info: id=5, control=1, cmask=0x0, channels=1,
type="INV_BOOLEAN"
    Volume: min=0, max=1, dBmin=0, dBmax=0
```

```
rk3568_r:/ # cat /proc/asound/card2/stream0
Yundea Technology Yundea M1051 at usb-xhci-hcd.5.auto-1, full
speed : USB Audio

Capture:
  Status: Stop
  Interface 2
    Altset 1
    Format: S16_LE
    Channels: 1
    Endpoint: 3 IN (ASYNC)
    Rates: 16000
```

```
rk3568_r:/ # cat /proc/asound/card2/pcm0c/info
card: 2
device: 0
subdevice: 0
stream: CAPTURE
id: USB Audio
name: USB Audio
subname: subdevice #0
```

```
class: 0
subclass: 0
subdevices_count: 1
subdevices_avail: 1
```

录音测试

尝试录音失败，参数设置不对

```
rk3568_r:/ # tinycap /sdcard/test.pcm -D 2 -d 0 -T 5
Unable to open PCM device (cannot set hw params: Invalid
argument)
Captured 0 frames

rk3568_r:/ # tinycap /sdcard/test.pcm -D 2 -d 0 -c 2 -T 5
Unable to open PCM device (cannot set hw params: Invalid
argument)
Captured 0 frames
```

查看麦克风参数

```
rk3568_r:/ # cat /proc/asound/card2/stream0
Yundea Technology Yundea M1051 at usb-xhci-hcd.5.auto-1, full
speed : USB Audio

Capture:
  Status: Stop
  Interface 2
    Altset 1
    Format: S16_LE
    Channels: 1
    Endpoint: 3 IN (ASYNC)
    Rates: 16000
```

这里可以看到 通道是 1，码率是 16000，调整录音参数之后，正常录音

```
rk3568_r:/ # tinycap /sdcard/test.pcm -D 2 -d 0 -c 1 -r 16000 -
T 5
Capturing sample: 1 ch, 16000 hz, 16 bit
Captured 81920 frames
```

下载录音文件

```
neo@MacBook-Pro-M2 ~ % cd tmp
neo@MacBook-Pro-M2 tmp % adb pull /sdcard/test.pcm
/sdcard/test.pcm: 1 file pulled, 0 skipped. 21.1 MB/s (163884
bytes in 0.007s)
```

第 31 章 Activity

1. 定义 UI

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
     <application android:label="Test">

                  ...
                  ...
        <activity android:name=".WriteActivity"></activity>

    </application>

</manifest>
```

```
setContentView(R.layout.view);
```

2. 隐藏虚拟键

```
        int uiOptions = View.SYSTEM_UI_FLAG_FULLSCREEN |
View.SYSTEM_UI_FLAG_HIDE_NAVIGATION |
View.SYSTEM_UI_FLAG_IMMERSIVE;

getWindow().getDecorView().setSystemUiVisibility(uiOptions);

        View decorView = getWindow().getDecorView();
        int uiOptions = View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY
                | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                | View.SYSTEM_UI_FLAG_FULLSCREEN
                | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY;
        decorView.setSystemUiVisibility(uiOptions);
```

Android API 30

```
        WindowInsetsController controller =
getWindow().getDecorView().getWindowInsetsController();
        controller.hide(WindowInsets.Type.statusBars());
        controller.hide(WindowInsets.Type.navigationBars());
        controller.hide(WindowInsets.Type.systemBars());
```

3. 显式四种跳转方式

```
    Intent intent = new
Intent(MainActivity.this,HomeActivity.class);
    startActivity(intent);

    Intent intent = new Intent();
    intent.setClass(MainActivity.this,HomeActivity.class);
    startActivity(intent);

    Intent intent = new Intent();
    ComponentName componentName = new
ComponentName(MainActivity.this,HomeActivity.class);
    intent.setComponent(componentName);
    startActivity(intent);


        startActivity(new
Intent(MainActivity.this,HomeActivity.class));
```

3.1. startActivity()

```
                Button button = (Button)
findViewById(R.id.writeButton);

        button.setOnClickListener(new View.OnClickListener()
{
            public void onClick(View v) {
                setContentView(R.layout.activity_write);
                Intent intent = new
Intent(MainActivity.this,WriteActivity.class);
                startActivity(intent);
            }
```

```
        });
```

4. 定时关闭

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Toast.makeText(getApplicationContext(), "5秒后关闭",
Toast.LENGTH_SHORT).show();
        final Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            public void run() {
                //结束本界面并跳转到收派员列表的界面
                finish();
            }
        }, 5000);

    }
```

```
        new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            view.close();
        }
    }, 10000);
```

5. 恢复触发

程序回到桌面，例如设置WI-FI，让步在回到程序，安卓会调用onResume()

```
@Override
public void onResume() {
    super.onResume();
    this.other();
}
```

6. 返回触发

```
    @Override
    public void onBackPressed() {
        // code here to show dialog
        super.onBackPressed();  // optional depending on your
needs
        ...
    }
```

7. 保持屏幕常开

```
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCR
EEN_ON);
```

取消设置

```
getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_S
CREEN_ON);
```

8. 标题栏添加返回按钮

onCreate 中添加

```
    @Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.settings_activity);

    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
        actionBar.setDisplayHomeAsUpEnabled(true);
    }
    }
```

Activity 中添加

```
    @Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            return true;
        default:
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

在AndroidManifest.xml中标明Activity的主题

```
<activity
            android:name=".ui.ShareActivity"
            android:exported="false"
            android:theme="@style/AppTheme"/>
```

values/styles.xml 文件中增加

```
        <style name="AppTheme"
parent="Base.Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item
name="colorPrimary">@color/colorThemeBackGround</item>
        <item
name="colorPrimaryDark">@color/colorThemeBackGround</item>
        <item
name="colorAccent">@color/colorThemeBackGround</item>
    </style>
```

9. Activity 间数据传递

9.1. Intent 方式

设置数据

```
Intent intent= new Intent();
intent.putExtra("name","zhangsan");
```

取出数据

```
Intent intent = getIntent();
String name=intent.getStringExtra("name");
```

9.2. Bundle 方式

```
Intent it = new Intent(Activity.Main.this, Activity2.class);
Bundle bundle=new Bundle();
bundle.putString("name", "This is from MainActivity!");
it.putExtras(bundle);
startActivity(it);
```

获取数据

```
Bundle bundle=getIntent().getExtras();
String name=bundle.getString("name");
```

9.3. Flag 属性

Flag属性用来设定Activity的启动模式

```
Intent intent = new Intent();
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

与清单文件中的设置launchMode属性值相同

```
Intent.FLAG_ACTIVITY_CLEAR_TOP = singleTask
Intent.FLAG_ACTIVITY_SINGLE_TOP = singleTop
Intent.FLAG_ACTIVITY_NEW_TASK = singleInstance
```

在 Service，BroadcastReceiver 中切换 View

FLAG_ACTIVITY_NEW_TASK

```
context.startActivity(new Intent(context,
PictureBookFullscreenActivity.class).addFlags(Intent.FLAG_ACT
IVITY_NEW_TASK));
```

在非Activity（比如Service，BroadcastReceiver）中startActivity需要添加flag Intent.FLAG_ACTIVITY_NEW_TASK

9.4. 返回值

有返回值的跳转

```
Intent intent = new
Intent(MainActivity.this,HomeActivity.class);
intent.putExtra("nickname","netkiller");
// 第一个参数Intent对象, 第二个参数 RequestCode
startActivityForResult(intent,REQUSET_CODE);
```

第一个参数 是不是我要的返回结果 第二个参数 是谁返回给我的 第三个参数 返回的附加信息

```
        @Override
    protected void onActivityResult(int requestCode, int
resultCode, @Nullable Intent intent) {
        super.onActivityResult(requestCode, resultCode,
intent);

        if(requestCode == REQUSET_CODE && resultCode ==
HomeActivity.RESULT_CODE){
            String msg = data.getStringExtra("msg");

Toast.makeText(MainActivity.this,msg,Toast.LENGTH_SHORT).show
();
        }
    }
```

返回结果

```
Intent intent = new Intent();
Intent oldIntent = getIntent();
String nickname = oldIntent.getStringExtra("nickname");
if(TextUtils.isEmpty(nickname)){
    intent.putExtra("msg",nickname);
}else{
    intent.putExtra("msg","Neo");
}

setResult(RESULT_CODE,intent);
//关闭页面
finish();
```

10. intentActivityResultLauncher 跳转

```
// 定义跳转
ActivityResultLauncher<Intent> intentActivityResultLauncher =
        registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(), result -> {
    Intent data = result.getData();
    if (result.getResultCode() == RESULT_OK && data != null)
{
       // 一些逻辑
    }

});

// 使用时
Intent intent = new Intent(this,跳转到的.class);

// 执行跳转
intentActivityResultLauncher.launch(intent);
```

11. startActivityForResult 替代方案

startActivityForResult 即将废弃

```
    private void dispatchTakePictureIntent() {
        Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        try {
            startActivityForResult(takePictureIntent, 1);
        } catch (ActivityNotFoundException e) {
            // display error state to the user
        }
    }
```

替代方案是

```
   //拍照
    private final ActivityResultLauncher<Void>
mLauncherCamera = registerForActivityResult(
    new ActivityResultContracts.TakePicturePreview(), result
-> {
        //result为拍摄照片Bitmap格式
    });

    //开启拍照, 返回结果Bitmap
    private void launchCamera() {
        mLauncherCamera.launch(null);
    }
```

11.1. 返回值

```
package cn.netkiller;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class SubActivity extends Activity{
        private TextView tv1;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                this.setContentView(R.layout.sub);
                tv1 = (TextView)this.findViewById(R.id.tv1);
                Intent intent = new Intent();
                intent.putExtra("response", "返回码为200");
                setResult(200,intent);
                finish();
        }

}
```

```
package cn.netkiller;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class IntentActivity extends Activity {
        private Button btn1;
        private TextView tv1;
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn1 = (Button)this.findViewById(R.id.brn1);
        tv1 = (TextView)this.findViewById(R.id.tv2);
        btn1.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                Intent intent = new Intent();

intent.setClass(IntentActivity.this, SubActivity.class);

startActivityForResult(intent, 100); //requestcode=100

                        }
                });
    }
    //回调函数，会根据requesrCode进行不同的响应
        @Override
        protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
                super.onActivityResult(requestCode,
resultCode, data);
                if(requestCode==100){
                        // TODO: 响应逻辑
                }
                if(resultCode==200){
                        Bundle bundle = data.getExtras();
                        String response =
bundle.getString("response");
                        tv1.setText(response);
                }
        }

}
```

12. Activity 关闭

```
package cn.netkiller.album;

import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class HotelActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hotel);

        TextView hotelClose = (TextView)
findViewById(R.id.hotelClose);

        hotelClose.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

    }
}
```

12.1. 退出 App

AndroidManifest.xml 中 activity 添加 android:launchMode="singleTask"

```
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:launchMode="singleTask">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.HOME" />
                <category
android:name="android.intent.category.DEFAULT" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

MainActivity 中添加 onNewIntent(Intent intent)

```
        @Override
    protected void onNewIntent(Intent intent) {
        super.onNewIntent(intent);
        if (intent != null) {
            boolean isExit = intent.getBooleanExtra("QUIT",
false);
            if (isExit) {
                this.finish();
            }
        }
    }
```

调用 quit 方法即可正常退出主程序

```
    public void quit(View v) {
        Intent intent = new Intent(this, MainActivity.class);
        intent.putExtra("QUIT", true);
        startActivity(intent);
    }
```

13. App 间跳转

```
    Intent intent = new Intent(Intent.ACTION_MAIN);
    //前提：知道要跳转应用的包名、类名
    ComponentName componentName = new
ComponentName("cn.netkiller.album.hotel",
"cn.netkiller.album.hotel.MainActivity");
    intent.setComponent(componentName);
    startActivity(intent);
```

14. Res 资源

14.1. 通过名称查找 layout ID

查找 layout 资源

```
        // 用法
context.getResources().getIdentifier("test_layout", "layout",
context.getPackageName());
        int test =
context.getResources().getIdentifier("test", "layout",
"cn.netkiller.album ");
        Log.d(TAG, test + " id ");
```

14.2. 查找 drawable 资源 ID

查找 drawable 资源

```
                // 用法
context.getResources().getIdentifier("ic_launcher","drawable"
,context.getPackageName())

        int identifier =
context.getResources().getIdentifier("hotel1", "drawable",
context.getPackageName());
        Log.d(TAG, "Resource identifier: " + identifier + "
");
```

14.3. 获取 color 颜色 ID

获取res文件夹下的color.xml文件下某个颜色字段的id

```
context.getResources().getIdentifier("yellow","color",context
.getPackageName())
```

14.4. 获取 array.xml 文件下某个字段的 ID

获取array.xml文件里名为“my_array”的id

```
context.getResources().getIdentifier("my_array","array",conte
xt.getPackageName())
```

14.5. 获取 style.xml 文件下的某个样式的 id

获取value下style.xml文件下的某个样式的id

```
context.getResources().getIdentifier("dialog_style","style",con
text.getPackageName())
```

第 32 章 Fragment

1. 启动 Fragment

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="?attr/fullscreenBackgroundColor"
    android:theme="@style/ThemeOverlay.裸眼3D产
品.FullscreenContainer"
    tools:context=".ui.PictureBookFullscreenActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</FrameLayout>
```

```
        getSupportFragmentManager()
        .beginTransaction()
        .add(R.id.fragmentContainerView, new
PictureBookStoryFullscreenFragment(bookId))
        .commit();
```

2. 关闭 Fragment

```
getActivity().onBackPressed()

getSupportFragmentManager().beginTransaction().remove(fragmen
t).commit();
```

3. 在 Fragment 中使用 findViewById

提示

使用 `getView()` 方法返回当前 `fragment` 的根视图。

```
Button btn = getView().findViewById(R.id.btn);
```

4. 在 Fragment 中使用 Intent 跳转

```
Intent intent = new Intent(getActivity(), MyService.class);
startActivity(intent);
```

5. Fragment 中调用 getPackageManager()

```
ResolveInfo resolveInfo =
getActivity().getPackageManager().resolveActivity(intent, 0);
```

6. 在 Fragment 中使用 runOnUiThread

```
private void showResponse(final String response) {
    //在子线程中更新UI
    getActivity().runOnUiThread(new Runnable() {
        @Override
        public void run() {
            text_dashboard.setText(response);
        }
    });
}
```

7. Fragment 中调用 findViewById

```
package cn.netkiller.album;

import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.fragment.app.Fragment;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link ClockFragment#newInstance} factory method
to
 * create an instance of this fragment.
 */
public class ClockFragment extends Fragment {
    private static final String TAG =
ClockFragment.class.getSimpleName();
    // TODO: Rename parameter arguments, choose names that
match
    // the fragment initialization parameters, e.g.
ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public ClockFragment() {
        // Required empty public constructor
```

```
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment ClockFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static ClockFragment newInstance(String param1,
String param2) {
        ClockFragment fragment = new ClockFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_clock,
container, false);
        TextView textViewDate =
view.findViewById(R.id.textViewDate);

        Handler handler = new Handler();
        handler.postDelayed(() -> {
            SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("M月d日 EEEE", Locale.CHINESE);
```

```
            String date = simpleDateFormat.format(new
Date());
            textViewDate.setText(date);
            Log.d(TAG, date);
        }, 30000);

        return view;
    }
}
```

8. 替换 FrameLayout

屏某个区域布局

```
<FrameLayout
    android:id="@+id/controller"
    android:layout_width="409dp"
    android:layout_height="681dp"
    tools:layout_editor_absoluteX="1dp"
    tools:layout_editor_absoluteY="49dp"
    tools:ignore="MissingConstraints">

</FrameLayout>
```

点击事件 android:onClick="hotelMenuClick"

```
          <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:onClick="hotelMenuClick"
        app:cardCornerRadius="10dp"
        app:cardElevation="0dp">

        <ImageView
            android:id="@+id/imageView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:scaleType="fitXY"
            app:srcCompat="@drawable/bg_tavern_menu" />
    </androidx.cardview.widget.CardView>
```

事件响应

```
    public void hotelMenuClick(View v) {

        getSupportFragmentManager()
                .beginTransaction()
                .replace(R.id.controller, new
HotelFragment(), null)
                .addToBackStack(null)
                .commit();

    }
```

9. Fragment 接收 BroadcastReceiver 广播

定义广播接收类

```
        private BroadcastReceiver receiver = new
BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent)
{
            //这里写需要的业务逻辑
        }
    };
```

注册广播

```
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater
inflater,
                             @Nullable ViewGroup container,
                             @Nullable Bundle
savedInstanceState) {

        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction("main.screen");

getActivity().getApplicationContext().registerReceiver(receiv
er, intentFilter);

        binding =
FragmentPictureBookStoryFullscreenBinding.inflate(inflater,
container, false);
        return binding.getRoot();
    }
```

广播销毁

```
    @Override
    public void onDestroy() {
        super.onDestroy();

getActivity().getApplicationContext().unregisterReceiver(rece
iver);

    }
```

第 33 章 Resources

1. strings.xml

1.1.

字符串

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="nickname">netkiller</string>
    <string name="play" formatted="false">缓冲进度为%d%%, 播放进度为%d%%</string>
<resources>
```

1.2.

定义数组

```
    <string-array name="language">
        <item>普通话</item>
        <item>粤语</item>
        <item>英语</item>
    </string-array>
```

1.3. 获取 Resource

```
context.getString(R.string.app_id)
```

第 34 章 Palette 视觉设计

1. 父容器定位

在相对布局中，可以通过以下的属性让的组合让控件处于父容器左上角、右上角、左下角、右下角、上下左右居中，正居中等九个位置。属性如下：

```
android:layout_alignParentLeft="true"    父容器左边
android:layout_alignParentRight="true"   父容器右边
android:layout_alignParentTop="true"     父容器顶部
android:layout_alignParentBottom="true" 父容器底部
android:layout_centerHorizontal="true"   水平方向居中
android:layout_centerVertical="true"     垂直方向居中
android:layout_centerInParent="true"     水平垂直都居中
```

```
android:layout_toLeftOf="@+id/button1"   在button1控件左方
android:layout_toRightOf="@+id/button1" 在button1控件右方
android:layout_above="@+id/button1"      在button1控件上方
android:layout_below="@+id/button1"      在button1控件下方

android:layout_alignLeft="@+id/button1" 与button1控件左边平齐
android:layout_alignRight="@+id/button1"          与button1控件右
边平齐
android:layout_alignTop="@+id/button1"   与button1控件上边平齐
android:layout_alignBottom="@+id/button1"         与button1控件下
边平齐
```

2. 样式布局

2.1. 对齐布局

垂直/水平剧中

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

2.2. LinearLayout

外边距设置

```
android:layout_marginLeft="10dp"
android:layout_marginTop="10dp"
android:layout_marginRight="10dp"
android:layout_marginBottom="10dp"
```

内边距设置

```
```

```
    android:padding="10dp"
```

水平剧中

android:gravity="center_horizontal"

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dp"
        android:text="唤醒词"
        android:textColor="#70ffffff"
        android:textSize="14sp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:text="小梅小梅"
        android:textColor="#00F8C1"
        android:textSize="24sp" />
</LinearLayout>
```

2.3. FrameLayout

FrameLayout 事件穿透

FrameLayout 浮动在另其他 UI 上方，点击 FrameLayout 某些问之，触发了下面的事件，解决方法是增加:

android:clickable="true"

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:RoundedCornerImageView="http://schemas.android.com/apk/
res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/transparent"
    tools:ignore="MissingDefaultResource"
    android:clickable="true">
```

方法二

```
        mFrameLayout.setOnTouchListener(new
View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            return true;
        }
    });
```

叠加层

```
```

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:foregroundGravity="left|top"
    tools:context=".BlankFragment">

    <TextView
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:background="@color/colorPrimary"
        android:gravity="bottom|right"
        android:text="第一层" />

    <TextView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:background="@color/colorAccent"
        android:gravity="bottom|right"
        android:text="第二层" />

    <TextView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="@color/colorPrimaryDark"
        android:gravity="bottom|right"
        android:text="第三层" />

</FrameLayout>
```

2.4. 动画

参数

- android:oneshot 代表播放次数 true 只展示一遍，设置为false会不停的循环播放动画
- android:duration 表示展示所用的该图片的时间长度

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">

    <item
        android:drawable="@drawable/hotel1"
        android:duration="150" />
    <item
        android:drawable="@drawable/hotel2"
        android:duration="150" />
    <item
        android:drawable="@drawable/hotel3"
        android:duration="150" />
    <item
        android:drawable="@drawable/hotel4"
        android:duration="150" />
    <item
        android:drawable="@drawable/hotel5"
        android:duration="150" />
</animation-list>
```

```
        <ImageView
    android:id="@+id/load_image"
    android:layout_width="25dp"
    android:layout_height="25dp"
    android:layout_gravity="center_vertical"
    android:scaleType="centerCrop"
    android:src="@drawable/loading_anim_image" />
```

```
    ImageView imageView = findViewById(R.id.imageView);
    animationDrawable = (AnimationDrawable)
```

```
imageView.getDrawable();
    //直接就开始执行，性能不是最佳的。
    animationDrawable.start();
```

2.5. 声音波形图

```
package cn.netkiller.album.view;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;

public class SoundWaveView extends View {
    private Paint paint;
    private float[] amplitudes;

    public SoundWaveView(Context context) {
        super(context);
        init();
    }

    public SoundWaveView(Context context, AttributeSet attrs)
{
        super(context, attrs);
        init();
    }

    private void init() {
        paint = new Paint();
        paint.setColor(Color.BLUE);
        paint.setStrokeWidth(2);
    }

    public void setAmplitudes(float[] amplitudes) {
        this.amplitudes = amplitudes;
        invalidate(); // 刷新视图
```

```
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        if (amplitudes == null) {
            return;
        }

        int width = getWidth();
        int height = getHeight();
        int centerY = height / 2;

        for (int i = 0; i < amplitudes.length; i++) {
            float x = width * i / amplitudes.length;
            float y = centerY + amplitudes[i] * centerY;
            canvas.drawLine(x, centerY, x, y, paint);
        }
    }
}
```

```
public class MainActivity extends AppCompatActivity {
    private SoundWaveView soundWaveView;
    private AudioRecord audioRecord;
    private boolean isRecording = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        soundWaveView = findViewById(R.id.sound_wave_view);
        int bufferSize = AudioRecord.getMinBufferSize(44100,
AudioFormat.CHANNEL_IN_MONO, AudioFormat.ENCODING_PCM_16BIT);
        audioRecord = new
AudioRecord(MediaRecorder.AudioSource.MIC, 44100,
AudioFormat.CHANNEL_IN_MONO, AudioFormat.ENCODING_PCM_16BIT,
bufferSize);
```

```
        startRecording();
    }

    private void startRecording() {
        isRecording = true;
        audioRecord.startRecording();

        new Thread(new Runnable() {
            @Override
            public void run() {
                short[] buffer = new short[1024];
                while (isRecording) {
                    int read = audioRecord.read(buffer, 0,
buffer.length);
                    if (read > 0) {
                        float[] amplitudes = new float[read];
                        for (int i = 0; i < read; i++) {
                            amplitudes[i] = buffer[i] /
32768f; // 归一化为[-1, 1]
                        }

soundWaveView.setAmplitudes(amplitudes);
                    }
                }
                audioRecord.stop();
            }
        }).start();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        isRecording = false;
    }
}
```

```
<com.example.myapplication.SoundWaveView
    android:id="@+id/sound_wave_view"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" />
```

第 35 章 UI 界面

1. Toast

```
Toast.makeText(getApplicationContext(), "默认Toast样式",
Toast.LENGTH_SHORT).show();
```

自定义样式

```
toast = Toast.makeText(getApplicationContext(),"自定义位置
Toast", Toast.LENGTH_LONG);
   toast.setGravity(Gravity.CENTER, 0, 0);
   toast.show();
```

带有图片的样式

```
toast = Toast.makeText(getApplicationContext(),"带图片的Toast",
Toast.LENGTH_LONG);
   toast.setGravity(Gravity.CENTER, 0, 0);
   LinearLayout toastView = (LinearLayout) toast.getView();
   ImageView imageView = new
ImageView(getApplicationContext());
   imageView.setImageResource(R.drawable.icon);
   toastView.addView(imageView, 0);
   toast.show();
```

2. Button

```
public class MainActivity extends AppCompatActivity {

    //我们需要自己写一个常量作为requestCode，在请求result时传递进去
    public static final int REQUEST_CODE_NORMAL = 100;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button button = (Button) findViewById(R.id.Button);

        button.setOnClickListener(new View.OnClickListener()
{
            public void onClick(View view) {
                                startActivityForResult(new
Intent(this,SecondActivity.class),REQUEST_CODE_NORMAL);
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode,
data);
        if (requestCode == REQUEST_CODE_NORMAL) {
                //获得Result数据并处理
            ...
            ...
        }
    }
}
```

```
public class SecondActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.save);

        Button button = (Button)
findViewById(R.id.SaveButton);

        button.setOnClickListener(new View.OnClickListener()
{
            public void onClick(View view) {
                        Intent intent = new
Intent(this,MainResultActivity.class);

intent.putExtra("content",etContent.getText().toString());
                        setResult(1,intent);
                        //发送Result数据给请求方，然后finish ()
                        finish();
            }
        });
    }
}
```

启用禁用

```
myButton.setEnabled(false);
```

实现 OnClickListener 接口

```
package cn.netkiller.video;
```

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    private Button buttonVideoView;
    private Button buttonSurfaceView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        buttonVideoView = (Button)
findViewById(R.id.buttonVideoView);
        buttonVideoView.setOnClickListener(this);

        buttonSurfaceView = (Button)
findViewById(R.id.buttonSurfaceView);
        buttonSurfaceView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        Intent intent;
        switch (v.getId()) {
            case R.id.buttonVideoView:
                startActivity(new Intent(MainActivity.this,
VideoViewActivity.class));
                break;
            case R.id.buttonSurfaceView:

                break;
            default:
                break;
        }
    }
}
```

Fragment 中使用 Button

```
                Button buttonWifi =
root.findViewById(R.id.buttonWifi);
        buttonWifi.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent();
                ComponentName componentName = new
ComponentName("com.android.settings",
"com.android.settings.wifi.WifiSettings");
                intent.setComponent(componentName);
                ResolveInfo resolveInfo =
getActivity().getPackageManager().resolveActivity(intent, 0);
                if (resolveInfo != null) {
                    startActivity(intent);
                }
            }
        });
```

圆形按钮

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:shape="oval"
    android:useLevel="false"
    tools:ignore="ResourceName">
    <solid android:color="#FFa500" />
</shape>
```

```
            <Button
                android:id="@+id/imeButton"
                android:layout_width="100dp"
                android:layout_height="100dp"
                android:layout_gravity="right"
                android:background="@drawable/rond"
                android:text="+"
                android:textColor="#ffffff"
                android:textSize="50dp" />
```

3. ListView

Array

```
        String[] list = Arrays.asList("Apple", "Banana",
"Orange", "Watermelon",
                "Pear", "Grape", "Pineapple", "Strawberry",
"Cherry", "Mango";
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(MainActivity.this,
android.R.layout.simple_list_item_1,data);
        ListView listView = (ListView)
findViewById(R.id.history);
        listView.setAdapter(adapter);
```

```
    <ListView
        android:id="@+id/history"
        android:layout_width="368dp"
        android:layout_height="444dp"
        android:scrollbars="horizontal"
        tools:layout_editor_absoluteX="8dp"
        tools:layout_editor_absoluteY="59dp" />
```

List

```
                List<String> list = Arrays.asList("Apple",
"Banana", "Orange", "Watermelon","Pear", "Grape",
"Pineapple", "Strawberry", "Cherry", "Mango");
        ArrayAdapter<String> adapter = new
```

```
ArrayAdapter<String>(MainActivity.this,
android.R.layout.simple_list_item_1,list);
        ListView listView = (ListView)
findViewById(R.id.history);
        listView.setAdapter(adapter);
```

setOnItemClickListener()

```
        List<String> list = Arrays.asList("Text 文本", "URL 网
址", "电话号码", "短信","开启应用", "地址", "日历", "图片", "邮箱",
"GPS 坐标");
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1,list);
        final ListView listView = (ListView)
findViewById(R.id.schemaList);
        listView.setAdapter(adapter);


        listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent,
View view, int position, long id) {

                String text =
listView.getItemAtPosition(position)+"";
                Log.e("WRITE","position="+position+",
text="+text);
            }
        });
```

用接口方法实现

```
public class MainActivity extends Activity implements
OnItemClickListener, OnScrollListener
```

```
                List<String> list = Arrays.asList("Text 文本",
"URL 网址", "电话号码", "短信","开启应用", "地址", "日历", "图片",
"邮箱", "GPS 坐标");
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1,list);
        final ListView listView = (ListView)
findViewById(R.id.schemaList);
        listView.setAdapter(adapter);

        listView.setOnItemClickListener(this);
        listView.setOnScrollListener(this);
```

```
        @Override
    public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
        String text =
listView.getItemAtPosition(position)+"";
        Log.e("WRITE","position="+position+", text="+text);
    }
```

```
        @Override
    public void onScrollStateChanged(AbsListView view, int
scrollState) {
```

```
        switch (scrollState) {
        case SCROLL_STATE_FLING:
            Log.i("tag", "用户手指离开屏幕后，因惯性继续滑动");
            Map<String,Object>map = new
HashMap<String,Object>();
            map.put("icon", R.drawable.ic_launcher);
            map.put("text", "新增加项目");
            dataList.add(map);
            adapter.notifyDataSetChanged();
            break;
        case SCROLL_STATE_IDLE:
            Log.i("tag","已经停止滑动");
            break;
        case SCROLL_STATE_TOUCH_SCROLL:
            Log.i("tag", "手指未离开屏幕，屏幕继续滑动");
            break;
        }
    }
```

4. Switch

```
    <Switch
        android:id="@+id/switchWrite"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="NDEF Message write"
        android:textOff="NDEF Message write Off"
        android:textOn="NDEF Message write On"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/writeButton"
        app:layout_constraintHorizontal_bias="0.076"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ndefWrite"
        app:layout_constraintVertical_bias="0.087" />
```

```
                final Switch switchWrite = (Switch)
findViewById(R.id.switchWrite);

        switchWrite.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {

                if(isChecked) {

status.setText(switchWrite.getTextOn().toString());
                } else {

status.setText(switchWrite.getTextOff().toString());
                }
```

```
            }
        });
```

```
        if(switchWrite.isChecked()){

    }
```

5. TextView

圆角边框

圆角，胶囊形状

```
        <TextView
            android:id="@+id/hotelClose"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="end"

android:background="@drawable/TextViewCornerRadius"
            android:paddingStart="12dp"
            android:paddingTop="2dp"
            android:paddingEnd="12dp"
            android:paddingBottom="2dp"
            android:text="关闭"
            android:textColor="@color/white"
            android:textSize="16sp" />
```

drawable/TextViewCornerRadius.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list
xmlns:android="http://schemas.android.com/apk/res/android" >
<item >
    <shape android:shape="rectangle">
        <stroke android:width="1.5dp"
android:color="@color/white" />
        <solid android:color="#00000000" />
        <corners android:radius="15dp" />
    </shape>
</item>
```

```
</layer-list>
```

阴影设置

```
android:shadowColor:      阴影的颜色
android:shadowDx:                 水平方向上的偏移量
android:shadowDy:                 垂直方向上的偏移量
android:shadowRadius:     是阴影的的半径大少
```

6. ProgressBar

7. Dialog

3. Widgets

3.1. ImageView

全屏显示

```
android:scaleType="matrix"
```

水平居中显示 android:layout_gravity="center"

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    app:srcCompat="@drawable/niboer"
    tools:ignore="MissingConstraints" />
```

剧中效果

android:scaleType="fitCenter"

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="fitCenter"
    app:srcCompat="@drawable/fo2"
```

```
        tools:ignore="MissingConstraints" />
```

ImageView 显示 URL 图片

方法一

```
   private Drawable getImageDrawableFromUrl(String url) {
        try {
            InputStream inputStream = (InputStream) new
URL(url).getContent();
            Drawable drawable =
Drawable.createFromStream(inputStream, "image.jpg");
//            Drawable drawable =
Drawable.createFromStream(new URL(url).openStream(),
"image.jpg");
//            Drawable drawable =
Drawable.createFromStream(getContext().getContentResolver().o
penInputStream(Uri.parse(url)), null);

            return drawable;
        } catch (IOException e) {
            Log.d("VoicePopup", e.getMessage());

        } catch (Exception e) {
            Log.d("VoicePopup", e.getMessage());
        }
        return null;
    }

    Drawable drawable = getImageDrawableFromUrl(image);
    imageView = findViewById(R.id.imageView);
    imageView.setImageDrawable(drawable);
```

方法二

```
        imageView = findViewById(R.id.imageView);

    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                InputStream is = (InputStream) new
URL(image).getContent();
                final Drawable d =
Drawable.createFromStream(is, null);
                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        imageView.setImageDrawable(d);
                    }
                });
            } catch (Exception e) {
            }
        }
    }).start();


        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    InputStream inputStream = new
URL("https://img.netkiller.cn/2d/f4873238-7860-11ee-8344-
0242ac12000c.png").openStream();
                    final Drawable drawable =
Drawable.createFromStream(inputStream, null);
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {

imageView.setImageDrawable(drawable);
                        }
                    });
                } catch (Exception e) {
                }
            }
        }).start();
```

方法三

```
        try {
            URL url = new
URL("https://img.netkiller.cn/2d/f4873238-7860-11ee-8344-
0242ac12000c.png");

            new Thread(new Runnable() {
                @Override
                public void run() {
                    Bitmap bitmap = null;
                    try {
                        bitmap =
BitmapFactory.decodeStream(url.openStream());
                        showImg(bitmap);

                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                }
            }).start();
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }


    private void showImg(final Bitmap bitmap) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                imageView.setImageBitmap(bitmap);
            }
        });
    }
```

唱片播放效果（旋转PNG图片）

旋转 PNG 动画效果，用于显示唱片播放效果

UI 布局

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:context="cn.netkiller.album.MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        app:srcCompat="@drawable/fo2"
        tools:ignore="MissingConstraints" />

    <cn.netkiller.album.view.SoundWaveView
        android:id="@+id/sound_wave_view"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_alignBottom="@id/imageView"

app:layout_constraintBottom_toBottomOf="@+id/imageView" />

    <ImageView
        android:id="@+id/imageViewWan"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_gravity="center"

app:layout_constraintBottom_toTopOf="@+id/sound_wave_view"

app:layout_constraintEnd_toEndOf="@+id/sound_wave_view"
```

```
app:layout_constraintStart_toStartOf="@+id/sound_wave_view"
        app:srcCompat="@drawable/wan"
        tools:ignore="MissingConstraints" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

旋转动画效果文件

创建旋转动画效果 res/anim/rotate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<rotate
xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:duration="2000"
        android:fromDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:repeatCount="-1"
        android:toDegrees="359"></rotate>
</rotate>
```

启动旋转效果

```
        Animation rotateAnimation =
AnimationUtils.loadAnimation(this, R.anim.rotate);
        LinearInterpolator linearInterpolator = new
LinearInterpolator();
        rotateAnimation.setInterpolator(linearInterpolator);
        View imageViewWan = findViewById(R.id.imageViewWan);
        imageViewWan.startAnimation(rotateAnimation);
```

3.2. TextClock

```
<TextClock
          android:id="@+id/textClockTime"
          android:layout_width="match_parent"
          android:layout_height="0dp"
          android:layout_weight="3"
          android:format12Hour="hh:mm"
          android:format24Hour="HH:mm"
          android:gravity="center"
          android:textColor="@android:color/black"
          android:textSize="40sp"
          android:textStyle="bold" />

      <TextClock
          android:id="@+id/textViewDate"
          android:layout_width="match_parent"
          android:layout_height="0dp"
          android:layout_weight="1"
          android:format12Hour="yyyy/MM/dd E"
          android:format24Hour="yyyy/MM/dd E"
          android:gravity="center"
          android:textColor="@android:color/black"
          android:textSize="16sp" />
```

```
              <LinearLayout
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_weight="1"
                  android:gravity="center"
                  android:orientation="vertical">

                  <TextClock
                      android:id="@+id/textClockTime"
                      android:layout_width="match_parent"
                      android:layout_height="wrap_content"
                      android:background="#00000000"
```

```
                android:format24Hour="HH:mm"
                android:gravity="center"
                android:textColor="@color/white"
                android:textSize="48sp" />

            <TextClock
                android:id="@+id/textViewDate"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:format12Hour="MM月dd日 EEEE"
                android:format24Hour="MM月dd日 EEEE"
                android:gravity="center"
                android:textColor="#70ffffff"
                android:textSize="16sp" />
        </LinearLayout>
```

3.3. 进度条

style属性:

```
@android:style/Widget.ProgressBar.Horizontal：水平进度条
@android:style/Widget.ProgressBar.Inverse：普通大小的进度条
@android:style/Widget.ProgressBar.Large：大环形进度条
@android:style/Widget.ProgressBar.Large.Inverse：大环形进度条
@android:style/Widget.ProgressBar.Small：小环形进度条
@android:style/Widget.ProgressBar.Small.Inverse：小环形进度条
```

案例

```
    <ProgressBar
        android:id="@+id/progressBar"

style="@style/Widget.AppCompat.ProgressBar.Horizontal"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="13dp"
        android:progress="0"
        android:visibility="invisible"
        app:layout_constraintBottom_toTopOf="@+id/textView"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="179dp" />
```

4. Containers

4.1. CardView

实现圆角 ImageView

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.cardview.widget.CardView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:cardCornerRadius="10dp"
        app:cardElevation="0dp"
        tools:ignore="MissingConstraints">

        <ImageView
            android:id="@+id/imageView7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:scaleType="fitXY"
            app:srcCompat="@drawable/bg_repast_menu" />
    </androidx.cardview.widget.CardView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

5. Legacy

5.1. GardView

5.2. GridView

6. 渐变背景色

实现界面背景颜色渐变效果

background_gradient.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android">
    <!--实现应用背景颜色渐变-->
    <gradient
        android:startColor="#F5736287"
        android:endColor="#FA7E7162"
        android:angle="1"/>
</shape>
```

设置背景

```
android:background="@drawable/background_gradient"
```

android:angle 角度参数

```
android:angle="0"        //效果是：是从左到右，按照开始颜色到结束颜色来渲染的
android:angle="90"       //效果是：是从下到上，按照开始颜色到结束颜色来渲染的
android:angle="180"      //效果是：是从右到左，按照开始颜色到结束颜色来渲染的
android:angle="270"      //效果是：是从上到下，按照开始颜色到结束颜色来
```

渲染的

设置圆角

```
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
            <gradient android:startColor="#00FFEA"
                android:endColor="#DA00FF"
                android:angle="45"/>
            <corners android:radius="10dp"/>
        </shape>
    </item>
</selector>
```

三色渐变

```
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
            <gradient
                android:startColor="#FF9800"
                android:centerColor="#11A5E8"
                android:endColor="#5C00FF"
                android:angle="45"/>
            <corners android:radius="10dp"/>
        </shape>
    </item>
</selector>


<selector
```

```
xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
            <gradient
                android:startColor="#FF9800"
                android:centerColor="#11A5E8"
                android:endColor="#5C00FF"
                android:angle="45"/>
            <corners android:radius="10dp"/>
        </shape>
    </item>
</selector>
```

7. 屏幕

7.1. 尺寸

```
        Resources resource = this.getResources();
        DisplayMetrics displayMetrics =
resource.getDisplayMetrics();

        Log.d(TAG, "getCurrentWindowMetrics: " +
displayMetrics.toString());
```

```
getCurrentWindowMetrics: DisplayMetrics{density=1.5,
width=1536, height=1964, scaledDensity=1.9499999, xdpi=240.0,
ydpi=240.0}
```

7.2. 屏幕触摸事件 onTouch(View view, MotionEvent motionEvent)

```
    private final View.OnTouchListener
mDelayHideTouchListener = new View.OnTouchListener() {
        @Override
        public boolean onTouch(View view, MotionEvent
motionEvent) {
            switch (motionEvent.getAction()) {
                case MotionEvent.ACTION_DOWN:

                    break;
                case MotionEvent.ACTION_UP:
                    view.performClick();
```

```
                    break;
                default:
                    break;
            }
            return false;
        }
    };
```

屏幕触摸事件 onTouchEvent(MotionEvent event)

```
    private float x;
    private float y;

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                x = event.getX();
                y = event.getY();
                Log.d("Motion", "ACTION_DOWN-> X: " + x + ","
+ "Y: " + y);
                break;
            case MotionEvent.ACTION_UP:
                Log.d("Motion", "ACTION_UP-> X: " +
event.getX() + ", Y: " + event.getY());
                if (event.getX() - x > 60) {
                    Log.d("Motion", "从左至右滑动: ➡️");
                }
                if (x - event.getX() > 60) {
                    Log.d("Motion", "从右至左滑动: ⬅️");
                }

                if (y - event.getY() > 60) {
                    Log.d("Motion", "从下至上滑动: ⬆️");
                }

                if (event.getY() - y > 60) {
                    Log.d("Motion", "从上至下滑动: ⬇️");
                }
```

```
                break;
            case MotionEvent.ACTION_MOVE:
                Log.d("Motion", "ACTION_MOVE-> X: " +
event.getX() + ", Y: " + event.getY());
                break;
        }
        return false;
    }
```

7.3. 手势事件

```

        gridView = (GridView) findViewById(R.id.gridView);

        GestureDetector gestureDetector = new
GestureDetector(this, new MyGesture());
        gridView.setOnTouchListener(new
View.OnTouchListener() {

            @Override
            public boolean onTouch(View v, MotionEvent event)
{
                // Log.e("MainActivity", event.getX()+"");
                return gestureDetector.onTouchEvent(event);
            }
        });
```

```


    public class MyGesture implements
GestureDetector.OnGestureListener {


        public MyGesture() {

        }
```

```
        // 用户轻触触摸屏，由1个MotionEvent ACTION_DOWN触发
        public boolean onDown(MotionEvent arg0) {
            Log.i("MyGesture", "onDown");

Toast.makeText(PictureBookFullscreenActivity.this, "onDown",
Toast.LENGTH_SHORT).show();
            return true;
        }

        /*
         * 用户轻触触摸屏，尚未松开或拖动，由一个1个MotionEvent
ACTION_DOWN触发
         * 注意和onDown()的区别，强调的是没有松开或者拖动的状态
         */
        public void onShowPress(MotionEvent e) {
            Log.i("MyGesture", "onShowPress");

Toast.makeText(PictureBookFullscreenActivity.this,
"onShowPress", Toast.LENGTH_SHORT).show();
        }

        // 用户（轻触触摸屏后）松开，由一个1个MotionEvent ACTION_UP
触发
        public boolean onSingleTapUp(MotionEvent e) {
            Log.i("MyGesture", "onSingleTapUp");

Toast.makeText(PictureBookFullscreenActivity.this,
"onSingleTapUp", Toast.LENGTH_SHORT).show();
            return true;
        }

        // 用户按下触摸屏、快速移动后松开，由1个MotionEvent
ACTION_DOWN, 多个ACTION_MOVE, 1个ACTION_UP触发
        public boolean onFling(MotionEvent e1, MotionEvent
e2, float velocityX, float velocityY) {
            Log.i("MyGesture", "onFling");

Toast.makeText(PictureBookFullscreenActivity.this, "onFling",
Toast.LENGTH_LONG).show();
            if (velocityX > -50) {
                startActivity(new
Intent(PictureBookFullscreenActivity.this,
ShareFullscreenActivity.class));
            }
```

```
            if (velocityX > 50) {
                startActivity(new
Intent(PictureBookFullscreenActivity.this,
FavoritesFullscreenActivity.class));
            }
            return true;
        }

        // 用户按下触摸屏，并拖动，由1个MotionEvent ACTION_DOWN,
多个ACTION_MOVE触发
        public boolean onScroll(MotionEvent e1, MotionEvent
e2, float distanceX, float distanceY) {
            Log.i("MyGesture", "onScroll e1: " + e1.getX() +
" e2: " + e2.getX() + " distanceX: " + distanceX + "
distanceY: " + distanceY);

Toast.makeText(PictureBookFullscreenActivity.this,
"onScroll", Toast.LENGTH_LONG).show();
            return true;
        }

        // 用户长按触摸屏，由多个MotionEvent ACTION_DOWN触发
        public void onLongPress(MotionEvent e) {
            Log.i("MyGesture", "onLongPress");

Toast.makeText(PictureBookFullscreenActivity.this,
"onLongPress", Toast.LENGTH_LONG).show();
        }
    }
```

第 36 章 Schedule 计划任务

1. 延迟执行

```
        new Handler().postDelayed(() -> {
        picture.browsePictureFolder();
    }, 30000);
```

```
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            MainActivity.this.finish();
        }
    }, 1800);
```

Android 11

```
        new Handler(Looper.getMainLooper()).postDelayed(new
Runnable() {
        public void run() {
            progressBar.setVisibility(View.INVISIBLE);
        }
    }, 3000);
```

2. Time 和 TimerTask 定时刷新

```
new Timer().schedule(new TimerTask() {
   @Override
   public void run() {
    //TODO: 定时做某件事情
   }
}, 5 * 1000, 5 * 1000);
```

```
package cn.netkiller.okhttp;

import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;

public class ScheduleActivity extends AppCompatActivity {

    private TextView clock;

    final Handler handler = new Handler() {
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            switch (msg.what) {
                case 1:
                    update(msg.obj.toString());
                    break;
```

```
            }
        }

        void update(String c) {

            clock.setText(c);
        }
    };

    Timer timer = new Timer();
    TimerTask task = new TimerTask() {
        DateFormat dateFormat = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

        public void run() {
            Message message = new Message();
            message.what = 1;
            message.obj = dateFormat.format(new Date());
            handler.sendMessage(message);
        }
    };


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_schedule);

        clock = (TextView) findViewById(R.id.clock);
        clock.setText("Today is ...");
        timer.schedule(task, 1000 * 5, 1000); //启动timer

    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (timer != null) {
            timer.cancel();
            timer = null;
        }

    }

}
```



3. 使用 Runnable 和 Handler 实现定时执行

原理是使用 handler.postDelayed 延迟 Runnable 的运行时间

```
package cn.netkiller.okhttp;

import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class RunnableActivity extends AppCompatActivity {

    private Handler handler = new Handler();
    private Runnable runnable = new Runnable() {
        public void run() {
            this.update();
            handler.postDelayed(this, 1000);// 1000 ms = 1s 间
隔1秒
        }

        void update() {
            DateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            time.setText(dateFormat.format(new Date()));
        }
    };
    private TextView time;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_runnable);

        time = (TextView) findViewById(R.id.time);
        time.setText("Start...");
```

```
        handler.postDelayed(runnable, 1000 * 5); // 5 秒后开始
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        handler.removeCallbacks(runnable);
    }

}
```

4. 循环执行

```
//首先创建一个Handler对象
Handler handler=new Handler();
//然后创建一个Runnable对像
Runnable runnable=new Runnable(){
   @Override
   public void run() {
    // TODO Auto-generated method stub
    //要做的事情，这里再次调用此Runnable对象，以实现每两秒实现一次的定时器操作
    handler.postDelayed(this, 2000);
   }
};
```

提示

请使用单例模式，否则每次都拿到 handler 都是新对象，无法管理已经运行的对象。

```
//使用PostDelayed方法，调用此Runnable对象
handler.postDelayed(runnable, 2000);

//关闭此定时器，可以这样操作
handler.removeCallbacks(runnable);

//移除所有的消息
handler.removeCallbacksAndMessages(null);
```

5. TimerTask 实现循环播放

```
    static class LoopPlayer {
        private static Timer timer;
        private static LoopPlayer loopPlayer;

        public synchronized static LoopPlayer getInstance() {
            if (loopPlayer == null) {
                loopPlayer = new LoopPlayer();
            }
            return loopPlayer;
        }

        public void schedule(TimerTask timerTask) {
            if (timer == null) {
                timer = new Timer();
            }
            timer.schedule(timerTask, 1000, 30000);
        }

        public void cannel() {
            if (timer != null) {
                timer.cancel();
                timer = null;
            }
        }
    }
```

例 **36.1.**

```
package cn.netkiller.album.skill;

import android.content.Context;
import android.content.Intent;
```

```
import android.util.Log;

import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

import cn.netkiller.album.R;

public class AlbumSkillComponent {
    private static final String TAG =
AlbumSkillComponent.class.getSimpleName();
    private final Timer timer = new Timer();
    boolean status = false;
    Context context;
    Map<Integer, String> map = new HashMap<Integer, String>()
{{
        put(R.drawable.kouhong, "娇兰KissKiss系列口红：这款口红以
其立方形的金色包装而闻名，提供多种饱满且持久的颜色。其配方含有透明度调节剂
和金色反射颗粒，可以为嘴唇带来光滑且饱满的效果。");
        put(R.drawable.xiangshui, "Miss Dior是Dior的另一款经典香
水，首次推出是在1947年，与Dior的第一款时装系列同时推出。这款香水的香调包
括粉红胡椒、柑橘、玫瑰、茉莉、香根草等，散发出一种浪漫、活力的气息。");
        put(R.drawable.xiezhuangshui, "美宝莲的卸妆水能有效地去除
脸部和眼部的彩妆，包括防水和长效彩妆。它的配方温和，不会对皮肤造成刺
激。");
        put(R.drawable.fendi, "这是香奈儿的一款经典粉底，以其轻薄的
质地和自然的妆效而受到喜爱。Vitalumiere粉底能提供中等的遮瑕力，同时给肌
肤带来光泽感，使肌肤看起来更加健康。");
        put(R.drawable.sfs, "说神仙水是最适合油皮痘肌的爽肤水绝对不
夸大！主要在于它的神奇成分Pitera，专业术语是半乳糖酵母样菌发酵产物滤液，
含有维生素、氨基酸、矿物质、有机酸这些对皮肤有益的成分，可以很好地帮助皮肤
调整水油平衡，改善肤质。如果是因为出油长痘的话，一定要试试它！");
    }};

    private TimerTask timerTask;

    public AlbumSkillComponent(Context context, String
question) {
        this.context = context;
        LoopPlayer loopPlayer = LoopPlayer.getInstance();
        if (question.contains("口红")) {
            this.play(R.drawable.kouhong, "娇兰KissKiss系列口
红：这款口红以其立方形的金色包装而闻名，提供多种饱满且持久的颜色。其配方含
```

```
有透明度调节剂和金色反射颗粒，可以为嘴唇带来光滑且饱满的效果。\n");
            this.status = true;
        } else if (question.contains("香水")) {
            this.play(R.drawable.xiangshui, "Miss Dior是Dior的另一款经典香水，首次推出是在1947年，与Dior的第一款时装系列同时推出。这款香水的香调包括粉红胡椒、柑橘、玫瑰、茉莉、香根草等，散发出一种浪漫、活力的气息。\n");
            this.status = true;
        } else if (question.contains("卸妆水")) {
            this.play(R.drawable.xiezhuangshui, "美宝莲的卸妆水能有效地去除脸部和眼部的彩妆，包括防水和长效彩妆。它的配方温和，不会对皮肤造成刺激。\n");
            this.status = true;
        } else if (question.contains("粉底")) {
            this.play(R.drawable.fendi, "这是香奈儿的一款经典粉底，以其轻薄的质地和自然的妆效而受到喜爱。Vitalumiere粉底能提供中等的遮瑕力，同时给肌肤带来光泽感，使肌肤看起来更加健康。\n");
            this.status = true;
        } else if (question.contains("爽肤水")) {
            this.play(R.drawable.sfs, "说神仙水是最适合油皮痘肌的爽肤水绝对不夸大！主要在于它的神奇成分Pitera，专业术语是半乳糖酵母样菌发酵产物滤液，含有维生素、氨基酸、矿物质、有机酸这些对皮肤有益的成分，可以很好地帮助皮肤调整水油平衡，改善肤质。如果是因为出油长痘的话，一定要试试它！");
            this.status = true;
        } else if (question.contains("停止")) {
            loopPlayer.cannel();
        } else if (question.contains("轮播")) {
            timerTask = new TimerTask() {
                final Integer[] keys =
map.keySet().toArray(new Integer[0]);
                final Random random = new Random();

                @Override
                public void run() {

                    Integer randomKey =
keys[random.nextInt(keys.length)];
                    String value = map.get(randomKey);
                    play(randomKey, value);
                    Log.d(TAG, value);

                }
            };
            loopPlayer.schedule(timerTask);
```

```
            this.status = true;
        } else {

            loopPlayer.cannel();
            this.play(R.drawable.logo, "没有找到产品，你可以这样对
我说，小美小美，介绍一下口红");
            this.status = true;
        }
    }

    public boolean hit() {
        return this.status;
    }

    public boolean play(int resource, String message) {
        Intent intent = new Intent();
        intent.setAction("album.broadcast.change");
        intent.putExtra("image", resource);
        intent.putExtra("message", message);
        context.sendBroadcast(intent);
        SkillMatching.say(message);
        return this.status;
    }

    static class LoopPlayer {
        private static Timer timer;
        private static LoopPlayer loopPlayer;

        public synchronized static LoopPlayer getInstance() {
            if (loopPlayer == null) {
                loopPlayer = new LoopPlayer();
            }
            return loopPlayer;
        }

        public void schedule(TimerTask timerTask) {
            if (timer == null) {
                timer = new Timer();
            }
            timer.schedule(timerTask, 1000, 30000);
        }

        public void cannel() {
            if (timer != null) {
                timer.cancel();
```

```
                    timer = null;
                }
            }
        }
    }
```

6. TimerTask 更新 UI

```
        Timer timer = new Timer();

        private void Clock() {

        TextView textViewTime =
findViewById(R.id.textViewTime);
        TimerTask task = new TimerTask() {
            DateFormat dateFormat = new
SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

            public void run() {
                String current = dateFormat.format(new
Date());
                textViewTime.post(new Runnable() {
                    @Override
                    public void run() {
                        textViewTime.setText(current);
                    }
                });
                Log.d(TAG, current);
            }
        };
        timer.schedule(task, 1000 * 5, 1000);
//        timer.cancel();
    }
```

第 37 章 Internationalization i18n with Android (国际化)

1. 创建国际化文件

进入 Android Studio 文件菜单 File -> New -> New Resource File



在左侧列表中找到 Locale 点击 “>>” 按钮



选择国家后，点击 OK 按钮即可。



资源文件夹中已经显示出国际化文件，上面并有对应的国旗。

查看项目文件夹

```
neo@MacBook-Pro ~/AndroidStudioProjects/locale % find
app/src/main/res | grep values
app/src/main/res/values-zh-rCN
app/src/main/res/values-zh-rCN/strings.xml
app/src/main/res/values
app/src/main/res/values/colors.xml
app/src/main/res/values/dimens.xml
app/src/main/res/values/styles.xml
app/src/main/res/values/strings.xml
```

2. strings.xml 文件

```
<resources>
    <string name="app_name">Netkiller</string>
    <string name="title_home">Home</string>
    <string name="title_dashboard">Dashboard</string>
    <string name="title_notifications">Notifications</string>
</resources>
```

3. 翻译语言

再 res/values/strings 目录上面单击鼠标右键，打开 Open Translations Editor 翻译编辑器。



单击地球图标，添加 zh-cn 语言



现在就可以对照翻译语言包文件了。

4. 引用国际化文件

```
String test = "Sign Up";

String test = getResources().getString(R.string.sign_up);
```

```
R.string.browserSentence = "You are using $1%s to browse the
Internet.";

String browser = getString(R.string.browserSentence,
browser.getBrowser());
```

```
TextView textView = new TextView(this);
TextView.setText(“Sign Up”);

TextView textView = new TextView(this);
textView.setText(R.string.sign_up);
```

```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!" />

<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
```

```
    android:text="@string/hello_world" />
```

5. 切换语言

```
        DisplayMetrics metrics =
getResources().getDisplayMetrics();
        Configuration configuration =
getResources().getConfiguration();
        configuration.setLocale(locale);
        getResources().updateConfiguration(configuration,
metrics);

        //重新启动 Activity
        Intent intent = new Intent(this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
```

第 38 章 存储

1. 存储目录

```
Log.d(TAG,
Environment.getExternalStoragePublicDirectory(Environment.DIREC
TORY_PICTURES).toString());
Log.d(TAG,
getExternalFilesDir(Environment.DIRECTORY_PICTURES).toString())
;

/storage/emulated/0/Pictures

/storage/emulated/0/Android/data/cn.netkiller.album.education/f
iles/Pictures
```

2. SharedPreferences

SharedPreferences是Android中的数据存储技术，常用来存储一些轻量级的数据。

实际上SharedPreferences是 NoSQL 数据库，用于处理的key-value键值对存储，类似Windows 系统的注册表，Linux 系统里的 Berkeley DB (bdb) 以及衍生出的 dba,mdb 这类 hash 表数据库。

2.1. 操作模式

```
Context.MODE_PRIVATE：为默认操作模式,代表该文件是私有数据,只能被应用本身访问,在该模式下,写入的内容会覆盖原文件的内容
Context.MODE_APPEND：模式会检查文件是否存在,存在就往文件追加内容,否则就创建新文件.
Context.MODE_WORLD_READABLE和Context.MODE_WORLD_WRITEABLE用来控制其他应用是否有权限读写该文件.
MODE_WORLD_READABLE：表示当前文件可以被其他应用读取.
MODE_WORLD_WRITEABLE：表示当前文件可以被其他应用写入.
```

2.2. 保存数据

```
        Button buttonPut = (Button)
findViewById(R.id.buttonPut);

        buttonPut.setOnClickListener(new
View.OnClickListener() {
            public void onClick(View view) {

                //实例化SharedPreferences对象
                SharedPreferences sharedPreferences =
getSharedPreferences("test", Activity.MODE_PRIVATE);
```

```
                //实例化SharedPreferences.Editor对象
                SharedPreferences.Editor editor =
sharedPreferences.edit();

                //用putString的方法保存数据
                editor.putString("name", "Neo");
                editor.putString("nickname", "netkiller");
                editor.putBoolean("sex", true);
                editor.putInt("age", 30);
                editor.putFloat("tall", 180.23f);
                Set<String> books = new HashSet<String>();
                books.add("Netkiller Linux 手札");
                books.add("Netkiller Java 手札");
                books.add("Netkiller Android 手札");
                editor.putStringSet("books", books);

                //提交当前数据
                editor.commit();

            }
        });
```

2.3. 读取数据

```
        Button buttonGet = (Button)
findViewById(R.id.buttonGet);

        buttonGet.setOnClickListener(new
View.OnClickListener() {
            public void onClick(View view) {

                //实例化SharedPreferences对象
                SharedPreferences sharedPreferences =
getSharedPreferences("test", Activity.MODE_PRIVATE);

                //使用getString方法获得value,
                String name =
```

```
sharedPreferences.getString("name", "");
                String nickname =
sharedPreferences.getString("nickname", "");
                boolean sex =
sharedPreferences.getBoolean("sex", false);
                int age = sharedPreferences.getInt("age", 0);
                float tall =
sharedPreferences.getFloat("tall", 0f);
                Set<String> books =
sharedPreferences.getStringSet("books", null);

                Log.i("SharedPreferences",
String.format("%s,%s,%s,%s,%s,%s", name, nickname, sex, age,
tall, books.toString()));

            }
        });
```

2.4. 通过 key 查询数据是否存在

```
                SharedPreferences sharedPreferences =
getSharedPreferences("test", Activity.MODE_PRIVATE);
                if (sharedPreferences.contains("nickname")) {
            Log.i("SharedPreferences",
sharedPreferences.getString("nickname", ""));
        }else{
            Log.i("SharedPreferences", "key: nickname 不存
在");
        }
```

2.5. 删除数据

```
        SharedPreferences sharedPreferences =
getSharedPreferences("test", Activity.MODE_PRIVATE);
        SharedPreferences.Editor editor =
sharedPreferences.edit();

    editor.remove("nickname");
    editor.commit();

    Log.i("SharedPreferences",
sharedPreferences.getAll().toString());
```

2.6. 清空数据

```
        SharedPreferences sharedPreferences =
getSharedPreferences("test", Activity.MODE_PRIVATE);
        SharedPreferences.Editor editor =
sharedPreferences.edit();
    editor.clear();
    editor.commit();

    Log.i("SharedPreferences",
sharedPreferences.getAll().toString());
```

2.7. 对象存储

对象存储，需要将对象序列化，然后反序列化

2.8. SharedPreferences 读取物理存储文件

SharedPreferences 的数据存储再一个 xml 文件中，他的地址是:

```
//<package name>应替换成应用的包名, <name>
File xmlFile = new File("/data/data/<package
name>/shared_prefs/<name>.xml");
```

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
   <string name="name">陈景峰</string>
   <string name="nickname">netkiller</string>
   <int name="age" value="30" />
</map>
```

3. SD Card

3.1. SD Card 状态

```
        if
(!Environment.getExternalStorageState().equals(Environment.ME
DIA_MOUNTED)) {
                    TextView textView = (TextView)
findViewById(R.id.textView);
                    textView.setText("SD 卡不存在，请插入 SD
卡！");
                }
```

3.2. Android 11 申请 sdcard 权限

```
        private static final int REQUEST_CODE = 1024;

    private void requestPermission() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            // 先判断有没有权限
            if (!Environment.isExternalStorageManager()) {
                Intent intent = new
Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION
);
                intent.setData(Uri.parse("package:" +
getPackageName()));
                startActivityForResult(intent, REQUEST_CODE);
            }
        } else if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.M) {
            // 先判断有没有权限
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED &&
```

```
                    ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED) {
//                  写入文件
            } else {
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_CODE);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode,
@NonNull String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode,
permissions, grantResults);
        if (requestCode == REQUEST_CODE) {
            if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED &&
                    ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_GRANTED) {
//                  写入文件
            } else {
//                  ToastUtils.show("存储权限获取失败");
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int
resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode,
data);
        if (requestCode == REQUEST_CODE &&
Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            if (Environment.isExternalStorageManager()) {
//                  写入文件
            } else {
//                  ToastUtils.show("存储权限获取失败");
            }
        }
    }
```

第 39 章 网络

1. Wifi 配置

方法一

```
        context.startActivity(new
Intent(Settings.ACTION_WIFI_SETTINGS));
```

方法二

```
        Button buttonWifi =
root.findViewById(R.id.buttonWifi);
        buttonWifi.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent();
                ComponentName componentName = new
ComponentName("com.android.settings",
"com.android.settings.wifi.WifiSettings");
                intent.setComponent(componentName);
                ResolveInfo resolveInfo =
getActivity().getPackageManager().resolveActivity(intent, 0);
                if (resolveInfo != null) {
                    startActivity(intent);
                }
            }
        });
```

2. OkHttp - An HTTP & HTTP/2 client for Android and Java applications

http://square.github.io/okhttp/

2.1. Gradle

再 app/build.gradle 文件中增加依赖包

```
implementation "com.squareup.okhttp3:okhttp:4.10.0"
```

app/build.gradle

```
neo@MacBook-Pro ~/AndroidStudioProjects/okhttp % cat
app/build.gradle

apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "cn.netkiller.okhttp"
        minSdkVersion 28
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
```

```
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support:design:28.0.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation
'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'com.squareup.okhttp3:okhttp:3.11.0'
}
```

2.2. AndroidManifest.xml 开启网络访问权限

在 app/src/main/AndroidManifest.xml 文件中增加

```
<uses-permission android:name="android.permission.INTERNET" />
```

否则 okhttp 无法访问网络，添加后的效果如下。

```
neo@MacBook-Pro ~/AndroidStudioProjects/okhttp % cat
app/src/main/AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.okhttp">
    <uses-permission android:name="android.permission.INTERNET"
/>
```

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2.3. okhttp 默认是 HTTPS 开启 HTTP

如果你尝试使用 http 链接 OkHttp3 就抛出异常: CLEARTEXT communication to " + host + " not permitted by network security policy

开发过程中由于 https ssl 需要购买证书，费用较高，通常测试环境我们仍然使用 http 下面方法是开启 http 模式，

创建文件 res/xml/network_security_config.xml 内容如下

```
neo@MacBook-Pro ~/AndroidStudioProjects/okhttp % cat
app/src/main/res/xml/network_security_config.xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">localhost</domain>
```

```
    </domain-config>
</network-security-config>
```

再 app/src/main/AndroidManifest.xml 文件中增加 android:networkSecurityConfig="@xml/network_security_config"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.okhttp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"

android:networkSecurityConfig="@xml/network_security_config">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

    <uses-permission android:name="android.permission.INTERNET"
/>
</manifest>
```

2.4. GET

```
        OkHttpClient client = new OkHttpClient();

    Request request = new Request.Builder()
            .url("http://www.netkiller.cn")
            .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) {
        throw new IOException("服务器端错误: " + response);
    }

    Headers responseHeaders = response.headers();
    for (int i = 0; i < responseHeaders.size(); i++) {
        System.out.println(responseHeaders.name(i) + ": " +
responseHeaders.value(i));
    }

    System.out.println(response.body().string());
```

URL 组装

```
                HttpUrl.Builder urlBuilder =
HttpUrl.parse("https://www.netkiller.cn").newBuilder();

urlBuilder.addPathSegments("search/cache_vector_chatgpt");
                urlBuilder.addQueryParameter("question",
"Helloworld!!!");
                String url = urlBuilder.build().toString();

                Log.d("API", url);
                OkHttpClient client = new OkHttpClient();
                try {
                    Request request = new
Request.Builder().url(url).build();
```

```
                    Response response =
client.newCall(request).execute();
                    if (response.isSuccessful()) {
                        Log.d("API",
response.body().string());
                    }
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
```

URL 输出

```
https://www.netkiller.cn/search/cache_vector_chatgpt?
question=Helloworld%21%21%21
```

Get 异步调用

```
        OkHttpClient client = new OkHttpClient();

        Request request = new
Request.Builder().url(url).build();
        Call call = client.newCall(request);
        call.enqueue(new Callback() {
            @Override
            public void onFailure(@NotNull Call call,
@NotNull IOException e) {

            }

            @Override
            public void onResponse(@NotNull Call call,
@NotNull Response response) throws IOException {
                if (response.isSuccessful()) {
                    Log.i("TAG", "Async: " +
```

```
response.body().string());
                }
            }
        });
```

2.5. POST

POST Form Data

from 数据如下

```
key=value&other=data
```

```
                String url =
"https://api.netkiller.cn/oauth/token";

                OkHttpClient client = new OkHttpClient();

        RequestBody formBody = new FormBody.Builder()
                .add("grant_type", "password")
                .add("username", "netkiller")
                .add("password", "123456")
                .build();

        Request request = new Request.Builder()
                .url(url)
                .post(formBody)
                .build();

        Response response = client.newCall(request).execute();
        if (!response.isSuccessful()) {
            throw new IOException("服务器端错误: " + response);
        }
```

POST RAW JSON

POST RAW Json 数据，数据的形态这样的

```
{"key":"value"}
```

```
public static final MediaType JSON =
MediaType.parse("application/json; charset=utf-8");

OkHttpClient client = new OkHttpClient();

String post(String url, String json) throws IOException {
  RequestBody body = RequestBody.create(JSON, json);
  Request request = new Request.Builder()
      .url(url)
      .post(body)
      .build();
  Response response = client.newCall(request).execute();
  return response.body().string();
}
```

数据流提交

```
OkHttpClient client = new OkHttpClient();
final MediaType MEDIA_TYPE_TEXT =
MediaType.parse("text/plain");
final String postBody = "Hello World";
```

```
RequestBody requestBody = new RequestBody() {
    @Override
    public MediaType contentType() {
        return MEDIA_TYPE_TEXT;
    }
    @Override
    public void writeTo(BufferedSink sink) throws IOException
{
        sink.writeUtf8(postBody);
    }
    @Override
    public long contentLength() throws IOException {
        return postBody.length();
    }
};

Request request = new Request.Builder()
        .url("https://www.google.com")
        .post(requestBody)
        .build();
Response response = client.newCall(request).execute();
if (!response.isSuccessful()) {
    throw new IOException("服务器端错误: " + response);
}
System.out.println(response.body().string());
```

2.6. HTTP PUT 请求

```


import java.util.concurrent.TimeUnit;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class Test {
  public static void main(String[] args) throws Exception {
    String url = "http://****";
```

```
    String json = "{\"param\":\"value2\",\"key\":\"value\"}";
    OkHttpClient client = new OkHttpClient().newBuilder() //
        .readTimeout(60, TimeUnit.SECONDS) // 设置读取超时时间
        .writeTimeout(60, TimeUnit.SECONDS) // 设置写的超时时间
        .connectTimeout(60, TimeUnit.SECONDS) // 设置连接超时时
间
        .build();

    MediaType JSON = MediaType.parse("application/json;
charset=utf-8");
    RequestBody body = RequestBody.create(json, JSON);
    Request request = new
Request.Builder().url(url).put(body).build();
    try (Response response =
client.newCall(request).execute()) {
      System.out.println(response.body().string());
    }
  }
}
```

2.7. http header 相关设置

设置 HTTP 头

添加Http头

```
Request request = new Request.Builder()
        .url(url)
        .addHeader("CLIENT", "AD")
        .addHeader("USERID", "343")
        .build();
```

覆盖 HTTP 头

```
        Request request = new Request.Builder()
                .header("Authorization", "replace this text
with your token")
                .url(url)
                .build();
```

```
public class Headers {
   public static void main(String[] args) throws IOException
{
    OkHttpClient client = new OkHttpClient();

    Request request = new Request.Builder()
            .url("http://www.netkiller.cn")
            .header("User-Agent", "Apple Mac")
            .addHeader("Accept", "text/html")
            .build();

    Response response = client.newCall(request).execute();
    if (!response.isSuccessful()) {
        throw new IOException("服务器端错误: " + response);
    }

    System.out.println(response.header("Server"));
    System.out.println(response.headers("Set-Cookie"));
   }
}
```

Cookie 管理

```
OkHttpClient mHttpClient = new
OkHttpClient.Builder().cookieJar(new CookieJar() {
```

```
    private final HashMap<String, List<Cookie>> cookieStore =
new HashMap<>();
    @Override
    public void saveFromResponse(HttpUrl url, List<Cookie>
cookies) {
        cookieStore.put(url.host(), cookies);
    }
    @Override
    public List<Cookie> loadForRequest(HttpUrl url) {
        List<Cookie> cookies = cookieStore.get(url.host());
        return cookies != null ? cookies : new
ArrayList<Cookie>();
    }
}).build();
```

禁用缓存

```
Request request = new Request.Builder()
    .cacheControl(new
CacheControl.Builder().noCache().build())
    .url(url)
    .build();
```

设置缓存 max-age

```
// 等同于 nocache
Request request = new Request.Builder()
    .cacheControl(new CacheControl.Builder()
        .maxAge(0, TimeUnit.SECONDS)
        .build())
    .url("https://www.netkiller.cn")
    .build();
```

```
// 设置缓存 365 天
Request request = new Request.Builder()
    .cacheControl(new CacheControl.Builder()
        .maxStale(365, TimeUnit.DAYS)
        .build())
    .url("https://www.netkiller.cn")
    .build();
```

强制缓存

```
Request request = new Request.Builder()
    .cacheControl(new CacheControl.Builder()
        .onlyIfCached()
        .build())
    .url("https://www.netkiller.cn/helloworld.txt")
    .build();
Response forceCacheResponse =
client.newCall(request).execute();
if (forceCacheResponse.code() != 504) {
  // The resource was cached! Show it.
} else {
  // The resource was not cached.
}
```

2.8. HTTP Base Auth

```
        OkHttpClient client = new
OkHttpClient.Builder().authenticator(
                new Authenticator(){
                    public Request authenticate(Route route,
Response response) {
                        String credential =
Credentials.basic("api","secret");
```

```
                        return
response.request().newBuilder().header("Authorization",
credential).build();
                    }
                }
        ).build();
```

2.9. HttpUrl.Builder 组装 URL 地址参数

使用字符串拼接 URL地址特别容易出错

```
String url = "https://www.netkiller.cn/article?username="+
username + "&category="+ category;
```

较好的处理方式是使用 HttpUrl.Builder

```
                HttpUrl.Builder builder =
HttpUrl.parse("https://www.netkiller.cn/article").newBuilder(
);
        builder.addQueryParameter("username", "netkiller");
        builder.addQueryParameter("category", "android");
        String url = builder.build().toString();

        Log.d("okhttp", url);
```

输出结果

```
https://www.netkiller.cn/article?
```

```
username=netkiller&category=android
```

2.10. Android Activity Example

```
package cn.netkiller.okhttp;

import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import java.io.IOException;

import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class HttpActivity extends AppCompatActivity {

    TextView textView;
    private Handler mHandler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_http);

        textView = (TextView) findViewById(R.id.textView);

        mHandler = new Handler(Looper.getMainLooper()) {
            @Override
            public void handleMessage(Message msg) {
                textView.setText((String) msg.obj);
            }
```

```
        };

        OkHttpClient client = new OkHttpClient();
        Request request = new
Request.Builder().url("https://api.netkiller.cn/member/json")
.build();
        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                Log.d("okhttp", "Connect timeout. " +
e.getMessage());
            }

            @Override
            public void onResponse(Call call, Response
response) throws IOException {
                String text = response.body().string();
                Log.d("okhttp", "HTTP Code " +
response.code() + " TEXT " + text);

                Message msg = new Message();
                msg.what = 0;
                msg.obj = text;
                mHandler.sendMessage(msg);

            }
        });

    }
}
```

2.11. Android Oauth2 + Jwt example

Oauth 返回数据，通过 Gson 的 fromJson 存储到下面类中。

```
package cn.netkiller.okhttp.pojo;

public class Oauth {
    private String access_token;
```

```
    private String token_type;
    private String refresh_token;
    private String expires_in;
    private String scope;
    private String jti;

    public String getAccess_token() {
        return access_token;
    }

    public void setAccess_token(String access_token) {
        this.access_token = access_token;
    }

    public String getToken_type() {
        return token_type;
    }

    public void setToken_type(String token_type) {
        this.token_type = token_type;
    }

    public String getRefresh_token() {
        return refresh_token;
    }

    public void setRefresh_token(String refresh_token) {
        this.refresh_token = refresh_token;
    }

    public String getExpires_in() {
        return expires_in;
    }

    public void setExpires_in(String expires_in) {
        this.expires_in = expires_in;
    }

    public String getScope() {
        return scope;
    }

    public void setScope(String scope) {
        this.scope = scope;
    }
```

```
    public String getJti() {
        return jti;
    }

    public void setJti(String jti) {
        this.jti = jti;
    }

    @Override
    public String toString() {
        return "Oauth{" +
                "access_token='" + access_token + '\'' +
                ", token_type='" + token_type + '\'' +
                ", refresh_token='" + refresh_token + '\'' +
                ", expires_in='" + expires_in + '\'' +
                ", scope='" + scope + '\'' +
                ", jti='" + jti + '\'' +
                '}';
    }
}
```

Activity 文件

```
package cn.netkiller.okhttp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import com.google.gson.Gson;

import java.io.IOException;

import cn.netkiller.okhttp.pojo.Oauth;
import okhttp3.Authenticator;
import okhttp3.Call;
import okhttp3.Callback;
```

```
import okhttp3.Credentials;
import okhttp3.FormBody;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import okhttp3.Route;

public class Oauth2jwtActivity extends AppCompatActivity {

    private TextView token;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_oauth2jwt);

        token = (TextView) findViewById(R.id.token);

        try {
            get();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }


    public static Oauth accessToken() throws IOException {


        OkHttpClient client = new
OkHttpClient.Builder().authenticator(
                new Authenticator() {
                    public Request authenticate(Route route,
Response response) {
                        String credential =
Credentials.basic("api", "secret");
                        return
response.request().newBuilder().header("Authorization",
credential).build();
                    }
                }
        ).build();
```

```
        String url = "https://api.netkiller.cn/oauth/token";

        RequestBody formBody = new FormBody.Builder()
                .add("grant_type", "password")
                .add("username", "blockchain")
                .add("password", "123456")
                .build();

        Request request = new Request.Builder()
                .url(url)
                .post(formBody)
                .build();

        Response response =
client.newCall(request).execute();
        if (!response.isSuccessful()) {
            throw new IOException("服务器端错误: " + response);
        }

        Gson gson = new Gson();
        Oauth oauth = gson.fromJson(response.body().string(),
Oauth.class);
        Log.i("oauth", oauth.toString());
        return oauth;
    }

    public void get() throws IOException {


        OkHttpClient client = new
OkHttpClient.Builder().authenticator(
                new Authenticator() {
                    public Request authenticate(Route route,
Response response) throws IOException {
                        return
response.request().newBuilder().header("Authorization",
"Bearer " + accessToken().getAccess_token()).build();
                    }
                }
        ).build();

        String url =
"https://api.netkiller.cn/misc/compatibility";

        Request request = new Request.Builder()
```

```
                .url(url)
                .build();

        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                call.cancel();
            }

            @Override
            public void onResponse(Call call, Response
response) throws IOException {

                final String myResponse =
response.body().string();

                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Log.i("oauth", myResponse);
                        token.setText(myResponse);
                    }
                });

            }
        });
    }

        public void post() throws IOException {


        OkHttpClient client = new
OkHttpClient.Builder().authenticator(
                new Authenticator() {
                    public Request authenticate(Route route,
Response response) throws IOException {
                        return
response.request().newBuilder().header("Authorization",
"Bearer " + accessToken().getAccess_token()).build();
                    }
                }
        ).build();

        String url =
"https://api.netkiller.cn/history/create";
```

```
        String json = "{\n" +
                "  \"assetsId\":
\"5bced71c432c001c6ea31924\",\n" +
                "  \"title\": \"添加信息\",\n" +
                "  \"message\": \"信息内容\",\n" +
                "  \"status\": \"录入\"\n" +
                "}";

        final MediaType JSON =
MediaType.parse("application/json; charset=utf-8");
        RequestBody body = RequestBody.create(JSON, json);

        Request request = new Request.Builder()
                .url(url)
                .post(body)
                .build();

        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                call.cancel();
            }

            @Override
            public void onResponse(Call call, Response
response) throws IOException {

                final String myResponse =
response.body().string();

                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {

//                          Gson gson = new Gson();
//                          Oauth oauth =
gson.fromJson(myResponse, Oauth.class);
//                          Log.i("oauth", oauth.toString());

                        token.setText(myResponse);
                    }
                });

            }
```

```
        });
    }

}
```

上面的例子演示了，怎样获取 access token 以及 HTTP 基本操作 GET 和 POST，再Restful接口中还可能会用到 PUT, DELETE 等等，原来相同，这里就不在演示。

2.12. HTTP/2

首先确认你的服务器是支持 HTTP2，HTTP2配置方法可以参考《Netkiller Web 手札》

下面是我的例子仅供参考，Nginx 开启 http2 代理后面的 Spring Cloud 接口。

```
server {
    listen       80;
    listen 443 ssl http2;
    server_name  api.netkiller.cn;

    ssl_certificate ssl/netkiller.cn.crt;
    ssl_certificate_key ssl/netkiller.cn.key;
    ssl_session_cache shared:SSL:20m;
    ssl_session_timeout 60m;

    charset utf-8;
    access_log  /var/log/nginx/api.netkiller.cn.access.log;

    error_page  497              https://$host$uri?$args;

    if ($scheme = http) {
        return 301 https://$server_name$request_uri;
    }

    location / {
```

```
                proxy_pass   http://127.0.0.1:8000;
        proxy_http_version 1.1;
                proxy_set_header    Host    $host;
                proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

安卓程序如下

```
                String url =
"https://api.netkiller.cn/member/json";

        OkHttpClient client = new OkHttpClient();
        Request request = new
Request.Builder().url(url).build();
        client.newCall(request).enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                Log.d("okhttp", "Connect timeout. " +
e.getMessage());
            }

            @Override
```

```
            public void onResponse(Call call, Response
response) throws IOException {
                String text = response.body().string();
                Log.d("okhttp", "HTTP Code " +
response.code() + " TEXT " + text);
                Log.d("okhttp", "Protocol: " +
response.protocol());
            }
        });
```

运行结果

```
D/okhttp: HTTP Code 200 TEXT
{"status":false,"reason":"","code":0,"data":
{"id":null,"username":"12322228888","mobile":"12322228888","pas
sword":"123456","wechat":"微信
ID","role":"Organization","captcha":null,"createDate":"2018-10-
25 09:25:23","updateDate":null}}
    Protocol: h2
```

输出 h2 表示当前接口与服务器连接是通过 http2 完成。

2.13. 异步更新 UI

```
                buttonSend.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {

                HttpUrl.Builder urlBuilder =
HttpUrl.parse(apiUrl).newBuilder();

urlBuilder.addPathSegments("search/cache_vector_chatgpt");
                urlBuilder.addQueryParameter("question",
```

```
editTextMessage.getText().toString());
                String url = urlBuilder.build().toString();
                Log.d("API", url);
                OkHttpClient client = new OkHttpClient();

                Request request = new
Request.Builder().url(url).build();
                Call call = client.newCall(request);

                call.enqueue(new Callback() {
                    @Override
                    public void onFailure(@NotNull Call call,
@NotNull IOException e) {

                    }

                    @Override
                    public void onResponse(@NotNull Call
call, @NotNull Response response) throws IOException {
                        if (response.isSuccessful()) {
                            answer =
response.body().string();
                            updateResult(answer);
                            Log.i("TAG", "Async: " + answer);
                        }
                    }
                });
            }
        });
```

```
    private void updateResult(final String response) {
        //在子线程中更新UI
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // 在这里进行UI操作，将结果显示到界面上
                text_dashboard.setText(response);
            }
        });
    }
```

Lambda 表达式

```
        private void showResponse(final String response) {
        //在子线程中更新UI
        runOnUiThread(() -> {
                        // 在这里进行UI操作，将结果显示到界面上
                        responseText.setText(response);
        });
    }
```

在 Fragment 中需这样使用 getActivity().runOnUiThread()

```
        buttonSend.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {

                HttpUrl.Builder urlBuilder =
HttpUrl.parse(apiUrl).newBuilder();

urlBuilder.addPathSegments("search/cache_vector_chatgpt");
                urlBuilder.addQueryParameter("question",
editTextMessage.getText().toString());
                String url = urlBuilder.build().toString();
                Log.d("API", url);
                OkHttpClient client = new OkHttpClient();

                Request request = new
Request.Builder().url(url).build();
                Call call = client.newCall(request);

                call.enqueue(new Callback() {
                    @Override
```

```
                    public void onFailure(@NotNull Call call,
@NotNull IOException e) {

                    }

                    @Override
                    public void onResponse(@NotNull Call
call, @NotNull Response response) throws IOException {
                        if (response.isSuccessful()) {
                            answer =
response.body().string();
                            getActivity().runOnUiThread(() ->
{

text_dashboard.setText(answer);
                            });
                            Log.i("TAG", "Async: " + answer);
                        }
                    }
                });
            }
        });
```

2.14. WebSocket Client

```
package cn.netkiller.service;

import lombok.extern.slf4j.Slf4j;
import okhttp3.*;
import org.reactivestreams.Publisher;
import org.springframework.stereotype.Component;
import
org.springframework.web.reactive.socket.WebSocketHandler;
import
org.springframework.web.reactive.socket.WebSocketMessage;
import
org.springframework.web.reactive.socket.WebSocketSession;
import
org.springframework.web.reactive.socket.client.StandardWebSoc
```

```
ketClient;
import
org.springframework.web.reactive.socket.client.WebSocketClien
t;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.concurrent.TimeUnit;

@Component
@Slf4j
public class WebSocketService {
    private final OkHttpClient webSocketClient;
    private final Request request;
    String url = "wss://socketsbay.com/wss/v2/1/demo/";
    private WebSocket webSocket;

    public WebSocketService() {
        webSocketClient = new
OkHttpClient.Builder().pingInterval(30,
TimeUnit.SECONDS).build();
        log.info("WebSocket url: {}", url);
        // 构造请求对象
        request = new Request.Builder().url(url).build();
        connect();
    }

    public void send(String text) {

        if (webSocket == null) {
            connect();
        }
        webSocket.send(text);
        log.info("Send: " + text);
    }

    public void connect() {
        // 调用websocket服务端
        webSocket = webSocketClient.newWebSocket(request, new
MyWebSocketListener());
        log.info(webSocket.toString());

    }
```

```
    class MyWebSocketListener extends WebSocketListener {

        /**
         * websocket连接建立后来到这个方法
         *
         * @param webSocket
         * @param response
         */
        @Override
        public void onOpen(WebSocket webSocket, Response
response) {
            super.onOpen(webSocket, response);
            try {
                log.info("onOpen: " +
response.body().string());
            } catch (IOException e) {

            }
        }

        @Override
        public void onMessage(WebSocket webSocket, String
text) {
            super.onMessage(webSocket, text);
            log.info("Receive: " + text);
        }

//        @Override
//        public void onMessage(WebSocket webSocket,
ByteString bytes) {
//            super.onMessage(webSocket, bytes);
//        }

        @Override
        public void onClosing(WebSocket webSocket, int code,
String reason) {
            super.onClosing(webSocket, code, reason);
            log.info("onClosing code: " + code + " reason: "
+ reason);
        }

        @Override
        public void onClosed(WebSocket webSocket, int code,
String reason) {
            super.onClosed(webSocket, code, reason);
```

```
            log.info("onClosed code: " + code + " reason: " +
reason);
        }

        @Override
        public void onFailure(WebSocket webSocket, Throwable
t, Response response) {
            super.onFailure(webSocket, t, response);
                        if (response == null) {
                    log.error("onFailure, response is
null.");
                    return;
                }
                try {
                    log.error("onFailure, code: {}, errmsg:
{}", response.code(), response.body().string());
                } catch (IOException e) {
                    log.warn("onFailure failed, error: {}",
e.getMessage());
                }
        }
    }

}
```

第 40 章 相机与相册

1. manifest 文件

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.camera">

    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <provider

android:name="android.support.v4.content.FileProvider"

android:authorities="cn.netkiller.camera.provider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data

android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths" />
```

```
        </provider>
    </application>

</manifest>
```

provider_paths.xml 文件

```
<?xml version="1.0" encoding="utf-8"?>
<paths
xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path name="external_files" path="."/>
</paths>
```

2. layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <Button
        android:id="@+id/buttonOpenCamera"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="拍照立即显示"

app:layout_constraintBottom_toTopOf="@+id/buttonAlbum"

app:layout_constraintEnd_toStartOf="@+id/buttonSavePhoto"
        app:layout_constraintHorizontal_bias="0.283"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/imageViewPicture"
        app:layout_constraintVertical_bias="0.0" />
```

```
    <Button
        android:id="@+id/buttonSavePhoto"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="20dp"
        android:layout_marginBottom="8dp"
        android:text="拍照存储后显示"

app:layout_constraintBottom_toTopOf="@+id/buttonAlbum"
        app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@+id/imageViewPicture"
        app:layout_constraintVertical_bias="0.0" />

    <ImageView
        android:id="@+id/imageViewPicture"
        android:layout_width="338dp"
        android:layout_height="366dp"
        android:layout_centerHorizontal="true"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
/>

    <Button
        android:id="@+id/buttonAlbum"
        android:layout_width="352dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Album"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />


</android.support.constraint.ConstraintLayout>
```

3. Activity

```
package cn.netkiller.camera;

import android.Manifest;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Environment;
import android.os.StrictMode;
import android.provider.MediaStore;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.FileProvider;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    private ImageView imageViewPicture;
    private Button buttonOpenCamera;
    private Button buttonSavePhoto;
    private Button buttonAlbum;
```

```
    private static int REQUEST_CAMERA = 1;
    private static int REQUEST_PHOTO = 2;
    private static int REQUEST_ALBUM = 3;
    private File imageFile;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageViewPicture = (ImageView)
findViewById(R.id.imageViewPicture);
        buttonOpenCamera = (Button)
findViewById(R.id.buttonOpenCamera);
        buttonSavePhoto = (Button)
findViewById(R.id.buttonSavePhoto);
        buttonAlbum = (Button)
findViewById(R.id.buttonAlbum);

        buttonOpenCamera.setOnClickListener(this);
        buttonSavePhoto.setOnClickListener(this);
        buttonAlbum.setOnClickListener(this);

        StrictMode.VmPolicy.Builder newbuilder = new
StrictMode.VmPolicy.Builder();
        StrictMode.setVmPolicy(newbuilder.build());

    }

    @Override
    public void onClick(View view) {
        Intent intent;
        switch (view.getId()) {
            case R.id.buttonOpenCamera:
                // 拍照并显示图片
                intent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);// 启动系统相机
                startActivityForResult(intent,
REQUEST_CAMERA);
                break;
            case R.id.buttonSavePhoto:

                if
(!Environment.getExternalStorageState().equals(Environment.ME
DIA_MOUNTED)) {
```

```
                    TextView textView = (TextView)
findViewById(R.id.textView);
                    textView.setText("SD 卡不存在，请插入 SD
卡！");
                }

                if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 1);
                    return;
                } else {

                    String dir =
Environment.getExternalStorageDirectory().getPath();
                    new File(dir).mkdirs();

                    imageFile = new File(dir, "tmp.png");

                    Uri imageUri =
FileProvider.getUriForFile(MainActivity.this,
"cn.netkiller.camera.provider", imageFile);
                    Log.d("album", imageFile.getPath());

                    intent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);// 启动系统相机
                    intent.putExtra(MediaStore.EXTRA_OUTPUT,
imageUri);
                    startActivityForResult(intent,
REQUEST_PHOTO);


                }

                break;
            case R.id.buttonAlbum:

                intent = new
Intent(Intent.ACTION_GET_CONTENT);
                intent.setType("image/*");

startActivityForResult(Intent.createChooser(intent, "Select
Picture"), 3);
```

```
                break;
            default:
                break;
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode,
data);
        if (resultCode == RESULT_OK) { // 如果返回数据
            if (requestCode == REQUEST_CAMERA) { // 判断请求码
是否为REQUEST_CAMERA,如果是代表是这个页面传过去的，需要进行获取
                Bundle bundle = data.getExtras(); // 从data中
取出传递回来缩略图的信息，图片质量差，适合传递小图片
                Bitmap bitmap = (Bitmap) bundle.get("data");
// 将data中的信息流解析为Bitmap类型
                imageViewPicture.setImageBitmap(bitmap);// 显
示图片
            } else if (requestCode == REQUEST_PHOTO) {
                Log.i("photo", imageFile.getPath());
                try {
//                    InputStream inputStream =
getContentResolver().openInputStream(imageUri);
                    String dir =
Environment.getExternalStorageDirectory().getPath();
                    FileInputStream fileInputStream = new
FileInputStream(imageFile);
                    Bitmap bitmap =
BitmapFactory.decodeStream(fileInputStream);
                    fileInputStream.close();

imageViewPicture.setImageBitmap(bitmap);// 显示图片
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else if (requestCode == REQUEST_ALBUM) {

                Bitmap bitmap = null;

                //外界的程序访问ContentProvider所提供数据 可以通过
ContentResolver接口
                ContentResolver resolver =
```

```
getContentResolver();
                Uri originalUri = data.getData();        //获
得图片的uri

                try {
                    bitmap =
MediaStore.Images.Media.getBitmap(resolver, originalUri);
//显得到bitmap图片
                } catch (IOException e) {
                    e.printStackTrace();
                }

                //获取图片的路径：
                Log.i("album", String.valueOf(originalUri));

                imageViewPicture.setImageBitmap(bitmap);
            }
        }
    }

}
```

4. LED flash 做手电筒

```
<uses-permission android:name="android.permission.FLASHLIGHT"
/>
```

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FlashLightActivity">

    <Button
        android:id="@+id/buttonLight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="On"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

```
package cn.netkiller.camera;
```

```
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import java.security.Policy;

public class FlashLightActivity extends AppCompatActivity {

    private Button buttonLight;
    private CameraManager cameraManager;
    private String cameraId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_flash_light);

        buttonLight = (Button)
findViewById(R.id.buttonLight);

        Boolean isFlashAvailable =
getApplicationContext().getPackageManager().hasSystemFeature(
PackageManager.FEATURE_CAMERA_FLASH);

        if (!isFlashAvailable) {

            AlertDialog alert = new
AlertDialog.Builder(FlashLightActivity.this).create();
            alert.setTitle("Error");
            alert.setMessage("Your device doesn't support
flash light!");
            alert.setButton(DialogInterface.BUTTON_POSITIVE,
"OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
int which) {
```

```
                    // closing the application
                    finish();
                    System.exit(0);
                }
            });
            alert.show();
            return;
        }

        cameraManager = (CameraManager)
getSystemService(Context.CAMERA_SERVICE);
        try {
            cameraId = cameraManager.getCameraIdList()[0];
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }

        buttonLight.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

                try {
                    light();
                } catch (CameraAccessException e) {
                    e.printStackTrace();
                }

            }
        });
    }

    public void light() throws CameraAccessException {


        if (buttonLight.getText().equals("On")) {
            Toast.makeText(getApplicationContext(), "打开手电
筒", Toast.LENGTH_SHORT).show();
            cameraManager.setTorchMode(cameraId, true);
            buttonLight.setText("Off");
        } else {
            Toast.makeText(getApplicationContext(), "关闭手电
筒", Toast.LENGTH_SHORT).show();
            cameraManager.setTorchMode(cameraId, false);
            buttonLight.setText("On");
```

```
            }
        }
}
```

第 41 章 麦克风与录音

1. 开启麦克风和SD卡权限

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.voice">

    <uses-permission
android:name="android.permission.RECORD_AUDIO" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2. layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/status"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="http://www.netkiller.cn"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/record"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginBottom="1dp"
        android:onClick="onClick"
        android:text="Record"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/play"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="7dp"
        android:text="Play"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
```

```
</android.support.constraint.ConstraintLayout>
```

3. Activity

```
package cn.netkiller.voice;

import android.Manifest;
import android.content.pm.PackageManager;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Environment;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    private static final int RECORD_AUDIO = 10;
    private Button record;
    private Button play;
    private MediaRecorder mediaRecorder;
    private TextView status;

    private MediaPlayer mediaPlayer;
    private String filename;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        status = (TextView) findViewById(R.id.status);
```

```
        record = (Button) findViewById(R.id.record);
        play = (Button) findViewById(R.id.play);

        record.setOnClickListener(this);
        play.setOnClickListener(this);


        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
PackageManager.PERMISSION_GRANTED ||
ActivityCompat.checkSelfPermission(this,
Manifest.permission.RECORD_AUDIO) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.RECORD_AUDIO}, RECORD_AUDIO);
        }

    }

    @Override
    public void onClick(View v) {

        switch (v.getId()) {
            case R.id.record:
                if (record.getText().equals("Record")) {
                    this.start();
                } else {
                    this.stop();
                }
                break;
            case R.id.play:
                play();
                status.setText("播放录音");
                break;
        }
    }

    private void start() {

        record.setText("Stop");
        status.setText("开启录音，请对准话筒讲话");

        mediaRecorder = new MediaRecorder();
```

```
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THRE
E_GPP);

mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_
NB);

        String dir =
Environment.getExternalStorageDirectory().getPath();
        String date = new SimpleDateFormat("yyyy-MM-
ddhhmmss").format(new Date());

        filename = String.format("%s/%s.3gp", dir, date);

        Log.e("Voice", "voice path " + filename);

        mediaRecorder.setOutputFile(filename);

        try {
            mediaRecorder.prepare();
        } catch (IOException e) {
            e.printStackTrace();
        }
        mediaRecorder.start();

    }

    private void stop() {
        record.setText("Record");
        status.setText("录音停止");

        if (mediaRecorder != null) {
            mediaRecorder.stop();
            mediaRecorder.release();
            mediaRecorder = null;
        }
    }

    private void play() {
        this.stop();
        if (filename == null) {
            Toast.makeText(getApplicationContext(), "请先录音",
Toast.LENGTH_SHORT).show();
            return;
```

```
        }
        try {

            if (mediaPlayer != null &&
mediaPlayer.isPlaying()) {
                mediaPlayer.reset();
            } else {
                mediaPlayer = new MediaPlayer();
                mediaPlayer.setDataSource(filename);
                mediaPlayer.prepare();
                mediaPlayer.start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mediaPlayer != null) {
            mediaPlayer.stop();
            mediaPlayer.release();
        }
    }

}
```

第 42 章 多媒体开发

1. MediaPlayer

1.1. 播放Raw下的元数据

播放一段特效声音，例如铃音，可以在点击按钮德时候调用 playSound()

```
        private void playSound(){

        MediaPlayer mediaPlayer =
MediaPlayer.create(FlashLightActivity.this, R.raw.alert);
        mediaPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {

            @Override
            public void onCompletion(MediaPlayer mp) {
                // TODO Auto-generated method stub
                mediaPlayer.release();
            }
        });
        mediaPlayer.start();
    }
```

播放 res 中的资源文件

```
        AssetFileDescriptor assetFileDescriptor =
context.getResources().openRawResourceFd(R.raw.music);

    public void play(AssetFileDescriptor assetFileDescriptor)
{
```

```
        if (mediaPlayer != null) {
            try {
                this.reset();

mediaPlayer.setDataSource(assetFileDescriptor);
                // 或使用这种方式
mediaPlayer.setDataSource(assetFileDescriptor.getFileDescript
or(), assetFileDescriptor.getStartOffset(),
assetFileDescriptor.getLength());
                mediaPlayer.prepare();
                //注意不能使用异步，异步方式是用来播放网络流音乐
                                    //
mediaPlayer.prepareAsync();
                assetFileDescriptor.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }

            mediaPlayer.start();
        }
    }
```

播放 assets 文件夹中的资源

```
//实例化播放器
MediaPlayer mediaPlayer = new MediaPlayer();
AssetManager am = getAssets();
try {
    AssetFileDescriptor afd = am.openFd("assets_video.mp4");
    mediaPlayer.setDataSource(afd.getFileDescriptor(),
afd.getStartOffset(), afd.getLength());
} catch (IOException e) {
    e.printStackTrace();
}
//设置准备就绪状态监听
mediaPlayer.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mp) {
```

```
            // 开始播放
            mediaPlayer.start();
        }
});
//准备播放
mediaPlayer.prepareAsync();
```

其他方式获得 assets 文件夹中的资源

```
                String url = "file:///android_asset/" +
"video.mp3";
        AssetFileDescriptor assetFileDescriptor = null;
        try {
            assetFileDescriptor =
context.getResources().getAssets().openFd("nepal.mp3");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
```

1.2. 播放assets文件夹中的音乐

```
 //需将资源文件放在assets文件夹
 AssetFileDescriptor fd = getAssets().openFd("samsara.mp3");
 mMediaPlayer.setDataSource(fd)
 mMediaPlayer.prepare()

 AssetFileDescriptor fd = getAssets().openFd("samsara.mp3");
 mMediaPlayer.setDataSource(fd, fd.getStartOffset(),
fd.getLegth())
 mMediaPlayer.prepare();
```

1.3. 播放互联网音乐

```
package cn.netkiller.demo.skill;

import android.media.MediaPlayer;
import android.util.Log;

import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;

import java.io.IOException;

public class Bible {
    private static final String TAG =
Bible.class.getSimpleName();

    public Bible(JSONArray array) {
        Log.i(TAG, array.toString());
        JSONObject item = (JSONObject) array.get(1);
        String url = item.getString("url");
        Log.i(TAG, "Audio url: " + url);
        MediaPlayer mediaPlayer = new MediaPlayer();
        try {
            mediaPlayer.setDataSource(url);//设置播放来源
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        mediaPlayer.prepareAsync();//异步准备
        mediaPlayer.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
            //异步准备监听
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {

                Log.i(TAG, "Voice异步文件时长: " +
mediaPlayer.getDuration() / 1000 + "");
                mediaPlayer.start();//播放
            }
        });
        mediaPlayer.setScreenOnWhilePlaying(true);// 设置播放的
时候一直让屏幕变亮
```

```
        mediaPlayer.setOnBufferingUpdateListener(new
MediaPlayer.OnBufferingUpdateListener() {
            //文件缓冲监听
            @Override
            public void onBufferingUpdate(MediaPlayer
mediaPlayer, int i) {

                Log.i(TAG, "Voice进度: " + i + "% Voice文件长
度" + mediaPlayer.getDuration() / 1000 + "");
            }
        });
    }
}
```

1.4. 使用单例模式

```
package cn.netkiller.demo.skill;

import android.media.MediaPlayer;
import android.util.Log;

import androidx.annotation.NonNull;

import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;

import java.io.IOException;

public class Bible {
    private static final String TAG =
Bible.class.getSimpleName();

    private static MediaPlayer mediaPlayer;

    public synchronized static MediaPlayer getInstance() {
        if (mediaPlayer == null)
            mediaPlayer = new MediaPlayer();
        return mediaPlayer;
```

```
    }

    public Bible(@NonNull JSONArray array) {
//        Log.i(TAG, array.toString());
        JSONObject item = (JSONObject) array.get(1);
        String url = item.getString("url");
//        Log.i(TAG, "Audio url: " + item.toString());
        MediaPlayer mediaPlayer = Bible.getInstance();
        try {
            mediaPlayer.reset();
            mediaPlayer.setDataSource(url);//设置播放来源
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        mediaPlayer.prepareAsync();//异步准备
        mediaPlayer.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
            //异步准备监听
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {
                Log.i(TAG, "Voice 播放异步文件, 时长: " +
mediaPlayer.getDuration() / 1000 + " Audio: " +
item.toString());
                mediaPlayer.start();//播放
            }
        });
        mediaPlayer.setScreenOnWhilePlaying(true);// 设置播放的
时候一直让屏幕变亮
        mediaPlayer.setOnBufferingUpdateListener(new
MediaPlayer.OnBufferingUpdateListener() {
            //文件缓冲监听
            @Override
            public void onBufferingUpdate(MediaPlayer
mediaPlayer, int i) {
                Log.i(TAG, "Voice 缓冲区加载进度: " + i + "%");
            }
        });
    }
}
```

1.5. 设置速度，快进播放

```
                PlaybackParams params = new PlaybackParams();
//                params.setSpeed(1.5f);
                params.setSpeed(5f);
                mediaPlayer.setPlaybackParams(params);
```

2. VideoView 开发

VideoView，用于播放一段视频媒体，它继承了SurfaceView，位于"android.widget.VideoView"，是一个视频控件。

```
VideoView也为开发人员提供了对应的方法，这里简单介绍一些常用的：

int getCurrentPosition()：获取当前播放的位置。
int getDuration()：获取当前播放视频的总长度。
isPlaying()：当前VideoView是否在播放视频。
void pause()：暂停
void seekTo(int msec)：从第几毫秒开始播放。
void resume()：重新播放。
void setVideoPath(String path)：以文件路径的方式设置VideoView播放的
视频源。
void setVideoURI(Uri uri)：以Uri的方式设置VideoView播放的视频源，可
以是网络Uri或本地Uri。
void start()：开始播放。
void stopPlayback()：停止播放。
setMediaController(MediaController controller)：设置
MediaController控制器。
setOnCompletionListener(MediaPlayer.onCompletionListener l)：监
听播放完成的事件。
setOnErrorListener(MediaPlayer.OnErrorListener l)：监听播放发生错
误时候的事件。
setOnPreparedListener(MediaPlayer.OnPreparedListener l)：：监听视
频装载完成的事件。
```

上面的一些方法通过方法名就可以了解用途。和MediaPlayer配合SurfaceView播放视频不同，VideoView播放之前无需编码装载视频，它会在start()开始播放的时候自动装载视频。并且VideoView在使用完之后，无需编码回收资源。

2.1. 播放网络视频

加入 android.permission.INTERNET 允许访问网络

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.video">

    <uses-permission
android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".VideoViewActivity">
</activity>
    </application>

</manifest>
```

最简洁的布局，只有一个 VideoView

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".VideoViewActivity">

    <VideoView
        android:id="@+id/videoView"
        android:layout_width="match_parent"
        android:layout_height="236dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

播放的文件来自 IPFS 星际文件系统

```
package cn.netkiller.video;

import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

public class VideoViewActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_video_view);

        Uri uri =
Uri.parse("http://ipfs.netkiller.cn/ipfs/QmcA1Fsrt6jGTVqAUNZB
qaprMEdFaFkmkzA5s2M6mF85UC");
        VideoView videoView = (VideoView)
this.findViewById(R.id.videoView);
        videoView.setMediaController(new
```

```
MediaController(this));
        videoView.setVideoURI(uri);
        videoView.start();
        videoView.requestFocus();

    }
}
```

运行程序开始播放视频，点击视频会在屏幕下方弹出 MediaController 控制条

2.2. MediaController 添加翻页事件

```
package cn.netkiller.video;

import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;

public class VideoViewActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_video_view);

        final Uri uri =
Uri.parse("http://ipfs.netkiller.cn/ipfs/QmcA1Fsrt6jGTVqAUNZB
qaprMEdFaFkmkzA5s2M6mF85UC");
        final VideoView videoView = (VideoView)
this.findViewById(R.id.videoView);
        MediaController mediaController = new
MediaController(this);
        mediaController.setMediaPlayer(videoView);
```

```
        mediaController.setPrevNextListeners(
                new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {

Toast.makeText(VideoViewActivity.this, "下一曲",
Toast.LENGTH_SHORT).show();

videoView.setVideoURI(Uri.parse("http://ipfs.netkiller.cn/ipf
s/QmUaDftnPB7zCTwTASnSAWLiXWd1L5vNGEeU585rddfVTh"));
                    }
                },
                new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {

Toast.makeText(VideoViewActivity.this, "上一曲",
Toast.LENGTH_SHORT).show();

videoView.setVideoURI(Uri.parse("http://ipfs.netkiller.cn/ipf
s/QmbvKvj9X368WMtmkLYFuf59gSwLXYDLcdJuSiSHKPhTG4"));
                    }
                });

        videoView.setMediaController(mediaController);
        videoView.setVideoURI(uri);
        videoView.start();
        videoView.requestFocus();

    }
}
```

2.3. 静音播放视频

```
                videoView.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
```

```
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {
                mediaPlayer.setVolume(0f, 0f);
            }
        });
```

2.4. 更新进度条

```
            new Thread() {
                @Override
                public void run() {
                    try {
                        while (videoView.isPlaying()) {
                            // 如果正在播放，没0.5.毫秒更新一次进度
条
                            int current =
videoView.getCurrentPosition();
                            seekBar.setProgress(current);
                            sleep(500);
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }.start();
```

2.5. 完整的例子

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".VideoViewActivity">

    <VideoView
        android:id="@+id/videoView"
        android:layout_width="match_parent"
        android:layout_height="236dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/textViewTime"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="00:00" />

            <SeekBar
                android:id="@+id/seekBar"
                android:layout_width="270dp"
                android:layout_height="wrap_content" />
```

```
            <TextView
                android:id="@+id/textViewCurrentPosition"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="00:00" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <Button
                android:id="@+id/buttonPlay"
                android:layout_width="183dp"
                android:layout_height="wrap_content"
                android:text="播放" />

            <Button
                android:id="@+id/buttonStop"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="停止" />

        </LinearLayout>

    </LinearLayout>

    <TextView
        android:id="@+id/textViewStatus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="        "
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/videoView"
/>
</android.support.constraint.ConstraintLayout>
```

```
package cn.netkiller.video;

import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.VideoView;

import java.text.SimpleDateFormat;
import java.util.Calendar;

public class VideoViewActivity extends AppCompatActivity
implements View.OnClickListener {

    private VideoView videoView;
    private SeekBar seekBar;
    private Button buttonPlay;
    private TextView textViewTime;
    private TextView textViewCurrentPosition;
    private TextView textViewStatus;

    private Handler handler = new Handler();
    private Runnable runnable = new Runnable() {
        public void run() {
            if (videoView.isPlaying()) {
                int current = videoView.getCurrentPosition();
                seekBar.setProgress(current);

textViewCurrentPosition.setText(time(videoView.getCurrentPosi
tion()));
            }
            handler.postDelayed(runnable, 500);
        }
```

```
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_video_view);

        final Uri uri =
Uri.parse("http://ipfs.netkiller.cn/ipfs/QmcA1Fsrt6jGTVqAUNZB
qaprMEdFaFkmkzA5s2M6mF85UC");
        videoView = (VideoView)
this.findViewById(R.id.videoView);
        videoView.setVideoURI(uri);
        videoView.requestFocus();

        videoView.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {

            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {

textViewTime.setText(time(videoView.getDuration()));
                textViewStatus.setText("视频加载完毕");
                buttonPlay.setEnabled(true);

            }
        });

        // 在播放完毕被回调
        videoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                Toast.makeText(VideoViewActivity.this, "播放完
成", Toast.LENGTH_SHORT).show();
            }
        });

        videoView.setOnErrorListener(new
MediaPlayer.OnErrorListener() {

            @Override
            public boolean onError(MediaPlayer mp, int what,
int extra) {
                // 发生错误重新播放
```

```
                play();
                Toast.makeText(VideoViewActivity.this, "播放出
错", Toast.LENGTH_SHORT).show();
                return false;
            }
        });


        textViewStatus = (TextView)
findViewById(R.id.textViewStatus);
        textViewStatus.setText("玩命加载中");

        textViewTime = (TextView)
findViewById(R.id.textViewTime);

        seekBar = (SeekBar) findViewById(R.id.seekBar);
        // 为进度条添加进度更改事件

seekBar.setOnSeekBarChangeListener(onSeekBarChangeListener);

        textViewCurrentPosition = (TextView)
findViewById(R.id.textViewCurrentPosition);

        buttonPlay = (Button) findViewById(R.id.buttonPlay);
        buttonPlay.setEnabled(false);
        final Button buttonStop = (Button)
findViewById(R.id.buttonStop);

        buttonPlay.setOnClickListener(this);
        buttonStop.setOnClickListener(this);

    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.buttonPlay:
                play();
                break;
            case R.id.buttonStop:
                stop();
                break;
            default:
                break;
        }
```

```
    }

    private SeekBar.OnSeekBarChangeListener
onSeekBarChangeListener = new
SeekBar.OnSeekBarChangeListener() {
        // 当进度条停止修改的时候触发
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            // 取得当前进度条的刻度
            int progress = seekBar.getProgress();
            if (videoView.isPlaying()) {
                // 设置当前播放的位置
                videoView.seekTo(progress);
            }
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onProgressChanged(SeekBar seekBar, int
progress,
                                      boolean fromUser) {

        }
    };

    protected void play() {

        if (buttonPlay.getText().equals("播放")) {
            buttonPlay.setText("暂停");
            textViewStatus.setText("请您欣赏");
            // 开始线程，更新进度条的刻度
            handler.postDelayed(runnable, 0);
            videoView.start();
            seekBar.setMax(videoView.getDuration());

        } else

        {
            buttonPlay.setText("播放");
            if (videoView.isPlaying()) {
                videoView.pause();
```

```
            }
        }

    }

    protected void stop() {
        if (videoView.isPlaying()) {
            videoView.stopPlayback();
        }
    }

    protected String time(long millionSeconds) {

        SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("mm:ss");
        Calendar c = Calendar.getInstance();
        c.setTimeInMillis(millionSeconds);
        return simpleDateFormat.format(c.getTime());
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        handler.removeCallbacks(runnable);
    }

}
```

2.6. 循环播放

```
        String uri = "android.resource://" + getPackageName()
+ "/" + R.raw.vfo1;
        VideoView videoView = (VideoView)
this.findViewById(R.id.videoView);
        videoView.setMediaController(new
MediaController(this));
        videoView.setVideoURI(Uri.parse(uri));
        videoView.start();
        videoView.requestFocus();
```

```
        videoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                Log.d(TAG, "我播完了");
                // 下一个视频
//                videoView.setVideoURI(Uri.parse(uri));
                videoView.seekTo(10);
                videoView.start();
            }
        });
```

2.7. 静音播放

静音播放，只需设置音量位 0，使用这个方法 mediaPlayer.setVolume(0f, 0f);

```
        String uri = "android.resource://" + getPackageName()
+ "/" + R.raw.vfo1;
        VideoView videoView = (VideoView)
this.findViewById(R.id.videoView);
        videoView.setMediaController(new
MediaController(this));
        videoView.setVideoURI(Uri.parse(uri));
        videoView.start();
        videoView.requestFocus();
        videoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                videoView.seekTo(2);
                videoView.start();
            }
        });
        videoView.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mediaPlayer) {
```

```
                mediaPlayer.setVolume(0f, 0f);
            }
        });
```

3. SoundPool

```
package cn.netkiller.sound;

import android.content.res.AssetManager;
import android.media.AudioManager;
import android.media.SoundPool;
import android.util.Log;

import cn.netkiller.demo.ContextHolder;

import java.io.IOException;
import java.util.HashMap;


public class SoundPoolUtil {
    private static final String TAG =
SoundPoolUtil.class.getSimpleName();
    private static SoundPool soundPool;
    private static HashMap<String, Integer> soundPoolMap =
new HashMap();

    public static void create() {
        if (soundPool != null) {
            return;
        }

        int maxStreams = 3;
        soundPool = new
SoundPool.Builder().setMaxStreams(maxStreams).build();
        //加载音频到内存
        try {
            AssetManager am =
ContextHolder.getContext().getAssets();
            soundPoolMap.put("唤醒提示音",
soundPool.load(am.openFd("audio/wakeResponse.mp3"), 1));
            soundPoolMap.put("未联网提示音",
soundPool.load(am.openFd("audio/offline.mp3"), 1));
            soundPoolMap.put("在",
soundPool.load(am.openFd("audio/zai.wav"), 1));
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        //资源加载完成回调
        soundPool.setOnLoadCompleteListener((soundPool,
sampleId, status) -> Log.i(TAG, "音频加载完毕，id=" +
sampleId));
    }

    public static void release() {
        if (soundPool != null) {
            soundPool.release();
            soundPool = null;
        }
    }

    public static void play(String audioName) {
        if (soundPool == null) {
            Log.e(TAG, "soundPool未初始化");
            return;
        }
        /**
         * 参数1：加载返回的声音Id
         * leftVolume：左声道音量，0.0-1.0f
         * rightVolume：右声道音量，0.0-1.0f
         * priority：优先级
         * loop：循环播放：  0(不循环)   -1(循环)
         * rate：播放速率  0.5--2.0f
         */
        soundPool.play(soundPoolMap.get(audioName), 1.0f,
1.0f, 1, 0, 1.0f);
    }

}
```

4. 音量控制

```
    private void volume(String control) {
        AudioManager audioManager = (AudioManager)
context.getSystemService(Context.AUDIO_SERVICE);
        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        int minVolume = 10;
        int stepVolume = 5;
        int currentMusicVolume =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
        int currentTTSVolume =
audioManager.getStreamVolume(AudioManager.STREAM_ALARM);


        switch (control) {
            case "VOLUME_MINUS": //步进减小
                currentMusicVolume -= stepVolume;
                if (currentMusicVolume < minVolume) {
                    currentMusicVolume = minVolume;
                }
                currentTTSVolume -= stepVolume;
                if (currentTTSVolume < minVolume) {
                    currentTTSVolume = minVolume;
                }
                break;
            case "VOLUME_PLUS": //步进累加
                currentMusicVolume += stepVolume;
                if (currentMusicVolume >= maxVolume) {
                    currentMusicVolume = maxVolume;
                }
                currentTTSVolume += stepVolume;
                if (currentTTSVolume > maxVolume) {
                    currentTTSVolume = maxVolume;
                }
                break;

            case "VOLUME_MAX": // 最大
                currentMusicVolume = currentTTSVolume =
maxVolume;
```

```
                break;
            case "VOLUME_MIN": //最小
                currentMusicVolume = currentTTSVolume =
minVolume;

                break;
            case "MUTE": //静音
                currentMusicVolume = currentTTSVolume =
minVolume;
                break;

        }

audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
currentMusicVolume, AudioManager.FLAG_SHOW_UI);

audioManager.setStreamVolume(AudioManager.STREAM_ALARM,
currentTTSVolume, AudioManager.FLAG_PLAY_SOUND);
        Log.d(TAG, String.format("volume:
currentMusicVolume=%s, currentTTSVolume=%s, maxVolume=%s",
currentMusicVolume, currentTTSVolume, maxVolume));
    }

    private void volume(double percent) {
        if (percent < 0.3) {
            return;
        }
        AudioManager audioManager = (AudioManager)
context.getSystemService(Context.AUDIO_SERVICE);
        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        int currentMusicVolume, currentTTSVolume;
        currentMusicVolume = currentTTSVolume = (int)
(maxVolume * percent);

audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
currentMusicVolume, AudioManager.FLAG_SHOW_UI);

audioManager.setStreamVolume(AudioManager.STREAM_ALARM,
currentTTSVolume, AudioManager.FLAG_PLAY_SOUND);
        Log.d(TAG, String.format("volume:
currentMusicVolume=%s, currentTTSVolume=%s, maxVolume=%s",
currentMusicVolume, currentTTSVolume, maxVolume));
    }
```

5. SurfaceView

6. Vitamio

第 43 章 定位

1. GPS + 网络 定位

1.1. manifest 权限配置

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.location">

    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission
android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
    </application>

</manifest>
```

1.2. layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TableLayout
        android:id="@+id/tableLayout"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="状态: " />

            <TextView
                android:id="@+id/status"
                android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:text="TextView" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="经度: " />

        <TextView
            android:id="@+id/textViewLatitude"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="纬度: " />

        <TextView
            android:id="@+id/textViewLongitude"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="海拔: " />
```

```
            <TextView
                android:id="@+id/textViewAltitude"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TextView" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textView4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="速度" />

            <TextView
                android:id="@+id/textViewSpeed"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TextView" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="时间: " />

            <TextView
                android:id="@+id/textViewTime"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TextView" />
        </TableRow>
    </TableLayout>

    <ListView
        android:id="@+id/listViewAddress"
        android:layout_width="368dp"
        android:layout_height="358dp"
```

```
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tableLayout" />
</android.support.constraint.ConstraintLayout>
```

1.3. Activity

```
package cn.netkiller.location;

import android.Manifest;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

```
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_PERMISSION_CODE = 12;
    private TextView textViewLatitude;
    private TextView textViewLongitude;
    private TextView textViewAltitude;
    private TextView textViewSpeed;
    private TextView textViewTime;
    private TextView status;

    private static final SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    private ListView listViewAddress;
    private ArrayAdapter<String> adapter;
    private ArrayList<String> list;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        textViewLatitude = (TextView)
findViewById(R.id.textViewLatitude);
        textViewLongitude = (TextView)
findViewById(R.id.textViewLongitude);
        textViewAltitude = (TextView)
findViewById(R.id.textViewAltitude);
        textViewSpeed = (TextView)
findViewById(R.id.textViewSpeed);
        textViewTime = (TextView)
findViewById(R.id.textViewTime);
        status = (TextView) findViewById(R.id.status);

        list = new ArrayList<String>();
        adapter = new ArrayAdapter<String>(MainActivity.this,
android.R.layout.simple_list_item_1, list);
        listViewAddress = (ListView)
findViewById(R.id.listViewAddress);
        listViewAddress.setAdapter(adapter);

        this.location();
    }
```

```
    private void loop() {

    }

    public void location() {

        //获取LocationManager对象
        LocationManager locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);

        boolean gpsStatus =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDE
R);
        Log.d("Location", "GPS Status: " + gpsStatus);

        boolean networkStatus =
locationManager.isProviderEnabled(LocationManager.NETWORK_PRO
VIDER);
        Log.d("Location", "Network Status: " +
networkStatus);

        //创建一个Criteria对象
        Criteria criteria = new Criteria();
        //设置粗略精确度
        criteria.setAccuracy(Criteria.ACCURACY_COARSE);
        //设置是否需要返回海拔信息
        criteria.setAltitudeRequired(true);
        //设置是否需要返回方位信息
        criteria.setBearingRequired(true);
        //设置是否允许付费服务
        criteria.setCostAllowed(false);
        //设置电量消耗等级
        criteria.setPowerRequirement(Criteria.POWER_HIGH);
        //设置是否需要返回速度信息
        criteria.setSpeedRequired(true);

        Log.d("Location", "Criteria: " +
criteria.toString());

        //获取最符合此标准的provider对象
//        String currentProvider =
locationManager.getProvider(LocationManager.GPS_PROVIDER).get
Name();
```

```
        //根据设置的Criteria对象，获取最符合此标准的provider对象
        String currentProvider =
locationManager.getBestProvider(criteria, true);

        Log.d("Location", "currentProvider: " +
currentProvider);
        status.setText(currentProvider);


        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION},
REQUEST_PERMISSION_CODE);
            return;
        } else {
            status.setText("正在获取GPS坐标请稍候...");
        }


locationManager.requestLocationUpdates(currentProvider, 0, 0,
locationListener);
        //根据当前provider对象获取最后一次位置信息
        Location location =
locationManager.getLastKnownLocation(currentProvider);

        //如果位置信息不为null，则请求更新位置信息
        if (location != null) {

            textViewLatitude.setText(location.getLatitude() +
"");
            textViewLongitude.setText(location.getLongitude()
+ "");
            textViewAltitude.setText(location.getAltitude() +
"");
            textViewSpeed.setText(location.getSpeed() + "");
            textViewTime.setText(location.getTime() + "");

            Log.d("Location", "Latitude: " +
```

```
location.getLatitude());
            Log.d("Location", "location: " +
location.getLongitude());

        } else {

            Log.d("Location", "Latitude: " + 0);
            Log.d("Location", "location: " + 0);

        }

    }

    //创建位置监听器
    private LocationListener locationListener = new
LocationListener() {

        //位置发生改变时调用
        @Override
        public void onLocationChanged(Location location) {
            status.setText("onLocationChanged");

            //位置信息变化时触发
            Log.e("Location", "定位方式: " +
location.getProvider());
            Log.e("Location", "纬度: " +
location.getLatitude());
            Log.e("Location", "经度: " +
location.getLongitude());
            Log.e("Location", "海拔: " +
location.getAltitude());
            Log.e("Location", "时间: " + location.getTime());


            textViewLatitude.setText(location.getLatitude() +
"");
            textViewLongitude.setText(location.getLongitude()
+ "");
            textViewAltitude.setText(location.getAltitude() +
"");
            textViewSpeed.setText(location.getSpeed() + "");
            textViewTime.setText(dateFormat.format(new
Date(location.getTime())) + "");

            //解析地址
```

```
            Geocoder geoCoder = new
Geocoder(MainActivity.this, Locale.getDefault());

            double latitude = location.getLatitude();
            double longitude = location.getLongitude();

            List<Address> locationList = null;
            try {
                locationList =
geoCoder.getFromLocation(latitude, longitude, 5);
            } catch (IOException e) {
                e.printStackTrace();
            }

//              Address address = locationList.get(0);//得到
Address实例第一个地址
//              status.setText(address.toString());
//              String countryName =
address.getCountryName();//得到国家名称，比如：中国
//              String locality = address.getLocality();//得到城
市名称，比如：北京市

            list.clear();

            for (Address address : locationList) {

                for (int n = 0; address.getAddressLine(n) !=
null; n++) {
                    String addressLine =
address.getAddressLine(n);//得到周边信息，包括街道等，i=0，得到街道
名称
                    list.add(addressLine);
                    Log.i("Location", "addressLine = " +
addressLine);
                    Log.d("Location",
address.getCountryName() + address.getAdminArea() +
address.getFeatureName());
                }
            }

            adapter.notifyDataSetChanged();

        }

        //provider失效时调用
```

```
        @Override
        public void onProviderDisabled(String provider) {

            Log.d("Location", "onProviderDisabled");
            status.setText("onProviderDisabled");

        }

        //provider启用时调用
        @Override
        public void onProviderEnabled(String provider) {

            Log.d("Location", "onProviderEnabled");
            status.setText("onProviderEnabled");

        }

        //状态改变时调用
        @Override
        public void onStatusChanged(String provider, int
status, Bundle extras) {

            Log.d("Location", "onStatusChanged");
            //GPS状态变化时触发
            switch (status) {
                case LocationProvider.AVAILABLE:
                    Log.e("Location", "当前GPS状态为可见状态");
                    break;
                case LocationProvider.OUT_OF_SERVICE:
                    Log.e("Location", "当前GPS状态为服务区外状
态");
                    break;
                case
LocationProvider.TEMPORARILY_UNAVAILABLE:
                    Log.e("Location", "当前GPS状态为暂停服务状
态");
                    break;
            }

        }

    };


    @Override
```

```
    public void onRequestPermissionsResult(int requestCode,
String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode,
permissions, grantResults);

        switch (requestCode) {
            case REQUEST_PERMISSION_CODE: {
                // If request is cancelled, the result arrays
are empty.
                if (grantResults.length > 0 &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                } else {
                    // permission denied, boo! Disable the
                    // functionality that depends on this
permission.
                }
                return;
            }

        }
    }


}
```

2. 只从 GPS 获取定位

默认安卓系统使用 GPS + 网络定位，网络定位速度非常快，GPS 需要一些搜星。但是网络定位没有海拔高度数据，所以有些场景需要 GPS 定位。

```
package cn.netkiller.ropeway;

import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.util.Log;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

import
com.google.android.material.bottomnavigation.BottomNavigation
View;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

```
import java.util.List;
import java.util.Locale;

import cn.netkiller.ropeway.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;

    private static final int REQUEST_PERMISSION_CODE = 12;
    private TextView textViewLatitude;
    private TextView textViewLongitude;
    private TextView textViewAltitude;
    private TextView textViewSpeed;
    private TextView textViewTime;
    private TextView status;

    private final ArrayList<String> loglist = new
ArrayList<String>();
    private ArrayAdapter<String> loggerArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding =
ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        BottomNavigationView navView =
findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level
destinations.
        AppBarConfiguration appBarConfiguration = new
AppBarConfiguration.Builder(R.id.navigation_home,
R.id.navigation_dashboard,
R.id.navigation_notifications).build();
        NavController navController =
Navigation.findNavController(this,
R.id.nav_host_fragment_activity_main);
        NavigationUI.setupActionBarWithNavController(this,
navController, appBarConfiguration);
        NavigationUI.setupWithNavController(binding.navView,
navController);
```

```
        textViewLatitude =
findViewById(R.id.textViewLatitude);
        textViewLongitude =
findViewById(R.id.textViewLongitude);
        textViewAltitude =
findViewById(R.id.textViewAltitude);
        textViewSpeed = findViewById(R.id.textViewSpeed);
        textViewTime = findViewById(R.id.textViewTime);
        status = findViewById(R.id.status);

        ListView listViewLogger =
findViewById(R.id.listViewLogger);
        loggerArrayAdapter = new ArrayAdapter<String>
(MainActivity.this, android.R.layout.simple_list_item_1,
loglist);
        listViewLogger.setAdapter(loggerArrayAdapter);


        this.location();
    }

    @SuppressLint("SetTextI18n")
    public void location() {

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION},
REQUEST_PERMISSION_CODE);
            return;
        } else {
            status.setText("正在获取GPS坐标请稍候...");
        }

        //获取LocationManager对象
        LocationManager locationManager = (LocationManager)
getSystemService(LOCATION_SERVICE);
```

```
        boolean gpsStatus =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDE
R);
//
locationManager.setTestProviderEnabled(LocationManager.NETWOR
K_PROVIDER,false);
        Log.d("Location", "GPS Status: " + gpsStatus);


        boolean networkStatus =
locationManager.isProviderEnabled(LocationManager.NETWORK_PRO
VIDER);
        Log.d("Location", "Network Status: " +
networkStatus);


locationManager.requestLocationUpdates(LocationManager.GPS_PR
OVIDER, 1000, 0, locationListener);
        //根据当前provider对象获取最后一次位置信息
        Location location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROV
IDER);

        loglist.add(String.format("GPS Status: %s, Network
Status: %s, Criteria: %s", gpsStatus, networkStatus,
criteria));

        //如果位置信息不为null，则请求更新位置信息
        if (location != null) {

            textViewLatitude.setText(location.getLatitude() +
"");
            textViewLongitude.setText(location.getLongitude()
+ "");
            textViewAltitude.setText(location.getAltitude() +
"");
            textViewSpeed.setText(location.getSpeed() + "");
            textViewTime.setText(location.getTime() + "");

            Log.d("Location", "Latitude: " +
location.getLatitude());
            Log.d("Location", "Location: " +
location.getLongitude());
            Log.d("Location", "Altitude: " +
location.getAltitude());
```

```
            loglist.add(String.format("Provider: %s,
Latitude: %s, Location: %s, Altitude: %s",
location.getProvider(), location.getLatitude(),
location.getLongitude(), location.getAltitude()));
        } else {

            Log.d("Location", "Latitude: " + 0);
            Log.d("Location", "location: " + 0);

        }

    }

    //创建位置监听器
    private final LocationListener locationListener = new
LocationListener() {

        //位置发生改变时调用
        @SuppressLint("SetTextI18n")
        @Override
        public void onLocationChanged(Location location) {
            status.setText("onLocationChanged");

            //位置信息变化时触发
            Log.e("Location", "定位方式: " +
location.getProvider());
            Log.e("Location", "纬度: " +
location.getLatitude());
            Log.e("Location", "经度: " +
location.getLongitude());
            Log.e("Location", "海拔: " +
location.getAltitude());
            Log.e("Location", "时间: " + location.getTime());

            SimpleDateFormat simpleDateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

            textViewLatitude.setText(location.getLatitude() +
"");
            textViewLongitude.setText(location.getLongitude()
+ "");
            textViewAltitude.setText(location.getAltitude() +
"");
            textViewSpeed.setText(location.getSpeed() + "");
            textViewTime.setText(simpleDateFormat.format(new
```

```
Date(location.getTime())) + "");

            loglist.add(String.format("Provider: %s,
Latitude: %s, Location: %s, Altitude: %s",
location.getProvider(), location.getLatitude(),
location.getLongitude(), location.getAltitude()));
            loggerArrayAdapter.notifyDataSetChanged();

        }

        //provider失效时调用
        @Override
        public void onProviderDisabled(String provider) {

            Log.d("Location", "onProviderDisabled");
            status.setText("onProviderDisabled");

        }

        //provider启用时调用
        @Override
        public void onProviderEnabled(String provider) {

            Log.d("Location", "onProviderEnabled");
            status.setText("onProviderEnabled");

        }

        //状态改变时调用
        @Override
        public void onStatusChanged(String provider, int
status, Bundle extras) {

            Log.d("Location", "onStatusChanged");
            //GPS状态变化时触发
            switch (status) {
                case LocationProvider.AVAILABLE:
                    Log.e("Location", "当前GPS状态为可见状态");
                    break;
                case LocationProvider.OUT_OF_SERVICE:
                    Log.e("Location", "当前GPS状态为服务区外状
态");
                    break;
                case
LocationProvider.TEMPORARILY_UNAVAILABLE:
```

```
                    Log.e("Location", "当前GPS状态为暂停服务状
态");
                    break;
            }

        }

    };


    @Override
    public void onRequestPermissionsResult(int requestCode,
String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode,
permissions, grantResults);

        switch (requestCode) {
            case REQUEST_PERMISSION_CODE: {
                // If request is cancelled, the result arrays
are empty.
                if (grantResults.length > 0 &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                } else {
                    // permission denied, boo! Disable the
                    // functionality that depends on this
permission.
                }
                return;
            }

        }
    }
}
```

第 44 章 电话

1. SIM 卡状态

```
                TelephonyManager telephonyManager =
(TelephonyManager)context.getSystemService(context.TELEPHONY_SE
RVICE);

        if(telephonyManager.getSimState() ==
TelephonyManager.SIM_STATE_READY){
            return true;
        }else{
            return false;
        }
```

2. 通信录与拨打电话

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.contacts">

    <uses-permission
android:name="android.permission.READ_CONTACTS"/>
    <uses-permission
android:name="android.permission.CALL_PHONE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="通信录与拨打电话"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/contact"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="60dp"
        android:layout_marginEnd="8dp"
        android:text="联系人"
        app:layout_constraintEnd_toStartOf="@+id/call"
        app:layout_constraintHorizontal_bias="0.478"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/phone" />

    <EditText
        android:id="@+id/phone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="64dp"
        android:layout_marginEnd="8dp"
        android:ems="10"
        android:inputType="phone"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.475"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2"
/>

    <Button
        android:id="@+id/call"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:layout_marginEnd="52dp"
        android:text="Call"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/phone" />

</android.support.constraint.ConstraintLayout>
```

```
package cn.netkiller.contacts;

import android.app.Activity;
import android.content.ContentResolver;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.provider.ContactsContract;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    private EditText phone;
    private Button contact;
    private Button call;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        contact = (Button) findViewById(R.id.contact);
        contact.setOnClickListener(this);

        call = (Button) findViewById(R.id.call);
        call.setOnClickListener(this);

        phone = (EditText) findViewById(R.id.phone);
        phone.setText("", TextView.BufferType.EDITABLE);


        if (ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.READ_CONTACTS) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions((Activity)
this, new String[]
{android.Manifest.permission.READ_CONTACTS}, 1);
        }
        if (ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions((Activity)
this, new String[]{android.Manifest.permission.CALL_PHONE},
1);
        }


    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.contact:
                Toast.makeText(MainActivity.this, "获取联系人手
机号码", Toast.LENGTH_SHORT).show();
                startActivityForResult(new
Intent(Intent.ACTION_PICK,
ContactsContract.Contacts.CONTENT_URI), 0);
                break;
            case R.id.call:
                Toast.makeText(MainActivity.this, "打电话",
Toast.LENGTH_SHORT).show();

                Intent intent = new Intent();
```

```
intent.setAction(Intent.ACTION_CALL);//ACTION_DIAL
                intent.setData(Uri.parse("tel:" +
phone.getText()));
                startActivity(intent);

                break;
        }

    }

    @Override
    protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode,
data);
        if (resultCode == Activity.RESULT_OK) {
            Uri contactData = data.getData();
            Cursor cursor =
getContentResolver().query(contactData, null, null, null,
null);
            cursor.moveToFirst();

            //条件为联系人ID
            String contactId =
cursor.getString(cursor.getColumnIndex(ContactsContract.Conta
cts._ID));
            //获得DATA表中的电话号码，条件为联系人ID,因为手机号码可能
会有多个
            Cursor contact =
getContentResolver().query(ContactsContract.CommonDataKinds.P
hone.CONTENT_URI, null,
ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=" +
contactId, null, null);
            while (contact.moveToNext()) {
                String contactNumber =
contact.getString(contact.getColumnIndex(ContactsContract.Com
monDataKinds.Phone.NUMBER));

                phone.setText(contactNumber);
            }
            cursor.close();

        }
    }
```

```
}
```

3. 发送短信

```
<uses-permission android:name="android.permission.SEND_SMS"
/>
```

```
        private void sendSMS(String phoneNumber,String
message){
        if(PhoneNumberUtils.isGlobalPhoneNumber(phoneNumber))
{
            Intent intent = new Intent(Intent.ACTION_SENDTO,
Uri.parse("smsto:"+phoneNumber));
            intent.putExtra("sms_body", message);
            startActivity(intent);
        }
    }
```

```
        public void sendSMS(String phoneNumber,String
message){
        android.telephony.SmsManager smsManager =
android.telephony.SmsManager.getDefault();
        //拆分短信内容，手机短信长度有限制
        List<String> divideContents =
smsManager.divideMessage(message);
        for (String text : divideContents) {
            smsManager.sendTextMessage(phoneNumber, null,
text, sentPI, deliverPI);
        }
    }
```

第 45 章 消息广播

安卓中有两种广播，一种是系统发出的广播信息，例如网络改变，电池的电量低等等，另一种是用户发出的广播信息。

Android 中的广播类型可以分为两种类型，标准广播和有序广播。

标准广播（Normal broadcasts）：标准广播是一种完全异步执行的广播，在广播发出之后，所有的广播接收器几乎会在同一时刻接收到这条广播消息。这种广播效率比较高，但同时也意味着它是无法被截断的。

有序广播（Ordered broadcasts）：有序广播则是一种同步执行的广播，在广播发出之后，同一时刻只会有一个广播接收器能够收到这条广播消息，当这个广播接收器中的逻辑执行完毕之后，广播才会继续传递。

```
android.intent.action.BATTERY_CHANGED    持久广播含充电状态，级别，以及其他相关的电池信息。
android.intent.action.BATTERY_LOW                显示设备的电池电量低。
android.intent.action.BATTERY_OKAY               指示电池正在低点后但没有问题。
android.intent.action.BOOT_COMPLETED     一次播出后，系统已完成启动。
android.intent.action.BUG_REPORT                 显示活动报告的错误。
android.intent.action.CALL                               执行呼叫由数据指定某人。
android.intent.action.CALL_BUTTON                用户按下“呼叫”按钮进入拨号器或其他适当的用户界面发出呼叫。
android.intent.action.DATE_CHANGED               日期改变。
android.intent.action.REBOOT                     有设备重启。
```

1. 动态注册

```
package cn.netkiller.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private IntentFilter intentFilter;
    private MyBroadcastReceiver myBroadcastReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        intentFilter = new IntentFilter();
        //为过滤器添加处理规则

intentFilter.addAction("android.net.conn.CONNECTIVITY_CHANGE"
);
        myBroadcastReceiver = new MyBroadcastReceiver();
        //注册广播接收器
        registerReceiver(myBroadcastReceiver, intentFilter);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //注销动态的广播接收器
        unregisterReceiver(myBroadcastReceiver);
    }

    //自定义内部类，继承 BroadcastReceiver
    public class MyBroadcastReceiver extends
BroadcastReceiver {

        @Override
```

```
        public void onReceive(Context context, Intent intent)
{
            Toast.makeText(context, "网络状态已改变",
Toast.LENGTH_SHORT).show();
        }
    }
}
```

现在尝试改变网络状态，例如开启或关闭飞行模式，程序会弹出"网络状态已改变"。

我的测试环境是 Android 9 Pie 没有加入下面的权限仍然能工作

```
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```

优化程序

```
package cn.netkiller.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.BatteryManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
```

```
    private IntentFilter intentFilter;
    private MyBroadcastReceiver myBroadcastReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        intentFilter = new IntentFilter();
        //为过滤器添加处理规则

intentFilter.addAction("android.net.conn.CONNECTIVITY_CHANGE"
);

intentFilter.addAction(ConnectivityManager.CONNECTIVITY_ACTIO
N);

intentFilter.addAction(Intent.ACTION_BATTERY_CHANGED);
        intentFilter.addAction(Intent.ACTION_BATTERY_LOW);

        myBroadcastReceiver = new MyBroadcastReceiver();
        //注册广播接收器
        registerReceiver(myBroadcastReceiver, intentFilter);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //注销动态的广播接收器
        unregisterReceiver(myBroadcastReceiver);
    }

    //自定义内部类，继承 BroadcastReceiver
    public class MyBroadcastReceiver extends
BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent)
{
            ConnectivityManager connectivityManager =
(ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo =
connectivityManager.getActiveNetworkInfo();
```

```
            //判断是否联网
            if (networkInfo != null &&
networkInfo.isConnected()) {
                Toast.makeText(context, "网络已连接",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(context, "网络不可用",
Toast.LENGTH_SHORT).show();
            }

            int status =
intent.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
            boolean isCharging = status ==
BatteryManager.BATTERY_STATUS_CHARGING ||
                    status ==
BatteryManager.BATTERY_STATUS_FULL;

            if (isCharging) {
                Toast.makeText(context, "正在充电",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(context, "电池已经充满",
Toast.LENGTH_SHORT).show();
            }

            int chargePlug =
intent.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);
            boolean usbCharge = chargePlug ==
BatteryManager.BATTERY_PLUGGED_USB;
            boolean acCharge = chargePlug ==
BatteryManager.BATTERY_PLUGGED_AC;
            if (usbCharge) {
                Toast.makeText(context, "USB 充电",
Toast.LENGTH_SHORT).show();
            }

        }
    }
}
```

2. 静态注册

Android Studio 选择 File - New - Other - Broadcast Receiver

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.broadcast">

    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver
            android:name=".MyReceiver"
            android:enabled="true"
            android:exported="true">

            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
```

```
        </receiver>
    </application>

</manifest>
```

MyReceiver 集成 BroadcastReceiver 在 onReceive 中写入你的业务逻辑。

```
package cn.netkiller.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "程序已启动，接收到系统启动广播",
Toast.LENGTH_SHORT).show();
    }
}
```

现在重启 Android 模拟器，启动后虽然 App 并没有进入，但是屏幕底部会看到 "程序已启动，接收到系统启动广播"

2.1. 电源管理

静态注册

```
        <receiver
            android:name=".receiver.StaticBroadcastReceiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter android:priority="1000">
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
                <action
android:name="android.intent.action.ACTION_BATTERY_CHANGED"
/>
                <action
android:name="android.intent.action.ACTION_BATTERY_LOW" />
                <action
android:name="android.intent.action.ACTION_BATTERY_OKAY" />
                <action
android:name="android.intent.action.ACTION_POWER_CONNECTED"
/>
                <action
android:name="android.intent.action.ACTION_POWER_DISCONNECTED
" />

                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </receiver>
```

动态注册

```
            IntentFilter filter = new IntentFilter();
            filter.addAction(Intent.ACTION_BATTERY_CHANGED);
            filter.addAction(Intent.ACTION_BATTERY_LOW);
            filter.addAction(Intent.ACTION_BATTERY_OKAY);
            filter.addAction(Intent.ACTION_POWER_CONNECTED);

filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
```

2.2. 接收不到消息

Android 8 以上，静态广播必须指定包名
intent.setPackage(context.getPackageName());

```
public static void broadcastTest(String message) {
    Log.d(TAG, "发送广播: " + message);
    Context context = ContextHolder.getContext();
    Intent intent = new Intent();
    intent.setAction("test.broadcast");
    intent.setPackage(context.getPackageName());
    intent.putExtra("message", message);
    context.sendBroadcast(intent);
}
```

3. 自定义用户消息广播

```
package cn.netkiller.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private IntentFilter intentFilter;
    private TextView textView;
    private MyBroadcastReceiver myBroadcastReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        intentFilter = new IntentFilter();

intentFilter.addAction("cn.netkiller.broadcast.MESSAGE");
        myBroadcastReceiver = new MyBroadcastReceiver();
        //注册广播接收器
        registerReceiver(myBroadcastReceiver, intentFilter);

        textView = (TextView) findViewById(R.id.textView);

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
```

```
                //把要发送的广播值传入Intent对象
                Intent intent = new
Intent("cn.netkiller.broadcast.MESSAGE");
                intent.putExtra("msg", "Helloworld");
                //调用Context的 sendBroadcast()方法发送广播
                sendBroadcast(intent);
                textView.setText("Send");

            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //注销动态的广播接收器
        unregisterReceiver(myBroadcastReceiver);
    }

    //自定义内部类，继承 BroadcastReceiver
    public class MyBroadcastReceiver extends
BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent)
{

            String data = intent.getStringExtra("msg");
            Toast.makeText(context, data,
Toast.LENGTH_SHORT).show();

        }
    }
}
```

4. 本地广播

注意：LocalBroadcastManager 已经被废弃

上面讲的系统广播是全局的，任何APP都能接收到你的广播，这样就很容易引起APP的安全性问题。很多时候我们只想接收来自本应用程序发出的广播。

```
package cn.netkiller.broadcast;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private TextView textView;
    private IntentFilter intentFilter;
    private LocalBroadcastManager localBroadcastManager;
    private MyBroadcastReceiver myBroadcastReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //获取LocalBroadcastManger 单li实例
        localBroadcastManager =
localBroadcastManager.getInstance(this);

        intentFilter = new IntentFilter();
```

```
intentFilter.addAction("cn.netkiller.broadcast.MESSAGE");
        myBroadcastReceiver = new MyBroadcastReceiver();
        //注册本地广播接收器

localBroadcastManager.registerReceiver(myBroadcastReceiver,
intentFilter);

        textView = (TextView) findViewById(R.id.textView);

        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {

                textView.setText("Send");

                Intent intent = new Intent();

intent.setAction("cn.netkiller.broadcast.MESSAGE");
                intent.putExtra("msg",
"http://www.netkiller.cn");
                //发送本地广播
                localBroadcastManager.sendBroadcast(intent);
            }
        });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //注销本地广播接收器

localBroadcastManager.unregisterReceiver(myBroadcastReceiver)
;
    }

    //自定义内部类，继承 BroadcastReceiver
    public class MyBroadcastReceiver extends
BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent)
{
```

```
            String data = intent.getStringExtra("msg");
            Toast.makeText(context, data,
Toast.LENGTH_SHORT).show();

        }
    }
}
```

5. 动态监听广播

```
        private void broadcast(final long Id) {

                // 注册广播监听系统的下载完成事件。
                IntentFilter intentFilter = new
IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE);
                broadcastReceiver = new BroadcastReceiver() {
                        @Override
                        public void onReceive(Context
context, Intent intent) {
                                long ID =
intent.getLongExtra(DownloadManager.EXTRA_DOWNLOAD_ID, -1);
                                if (ID == Id) {

Toast.makeText(getApplicationContext(), "任务:" + Id + " 下载完
成!", Toast.LENGTH_LONG).show();
                                }
                        }
                };

                registerReceiver(broadcastReceiver,
intentFilter);
        }
```

6. 广播重复接收

广播重复接收，解决方案，编辑 AndroidManifest.xml 文件，在所有 activity 中加入 singleTask

```
android:launchMode="singleTask"
```

7. 指定静态广播接收者

系统中注册了多个广播，需要发送给指定接收者。

```
    private void broadcastStory(String id, String image,
String story) {
        Intent intent = new Intent();
        intent.setAction("story");
        intent.putExtra("id", id);
        intent.putExtra("image", image);
        intent.putExtra("story", story);
        intent.setClassName(context.getPackageName(),
MainBroadcastReceiver.class.getName());
        context.sendBroadcast(intent);
    }

    private void broadcastShare(String id, String image,
String story) {
        Intent intent = new Intent();
        intent.setAction("share");
        intent.putExtra("id", id);
        intent.putExtra("image", image);
        intent.putExtra("story", story);
        intent.setClassName(context.getPackageName(),
ShareBroadcastReceiver.class.getName());
        context.sendBroadcast(intent);
    }
```

8. 异步执行广播

```
public class MyBroadcastReceiver extends BroadcastReceiver {
        private static final String TAG =
"MyBroadcastReceiver";

        @Override
        public void onReceive(Context context, Intent intent)
{
            final PendingResult pendingResult = goAsync();
            Task asyncTask = new Task(pendingResult, intent);
            asyncTask.execute();
        }

        private static class Task extends AsyncTask<String,
Integer, String> {

            private final PendingResult pendingResult;
            private final Intent intent;

            private Task(PendingResult pendingResult, Intent
intent) {
                this.pendingResult = pendingResult;
                this.intent = intent;
            }

            @Override
            protected String doInBackground(String...
strings) {
                StringBuilder sb = new StringBuilder();
                sb.append("Action: " + intent.getAction() +
"\n");
                sb.append("URI: " +
intent.toUri(Intent.URI_INTENT_SCHEME).toString() + "\n");
                String log = sb.toString();
                Log.d(TAG, log);
                return log;
            }

            @Override
```

```
            protected void onPostExecute(String s) {
                super.onPostExecute(s);
                // Must call finish() so the
BroadcastReceiver can be recycled.
                pendingResult.finish();
            }
        }
    }
```

第 46 章 Service 服务

手动调用方法

手动调用方法	作用
startService()	启动服务
stopService()	关闭服务
bindService()	绑定服务
unbindService()	解绑服务

自动调用的方法

自动调用方法	作用
onCreate()	创建服务
onStartCommand()	开始服务
onDestroy()	销毁服务
onBind()	绑定服务
onUnbind()	解绑服务

生命周期调用

```
1. 启动Service服务

单次：startService() –> onCreate() –> onStartCommand()
多次：startService() –> onCreate() –> onStartCommand() –>
onStartCommand()

2. 停止Service服务
```

```
stopService() –> onDestroy()

3. 绑定Service服务

bindService() –> onCreate() –> onBind()

4. 解绑Service服务

unbindService() –> onUnbind() –> onDestroy()

5. 启动绑定Service服务

startService() –> onCreate() –> onStartCommand() –>
bindService() –> onBind()

6. 解绑停止Service服务

unbindService() –> onUnbind() –> stopService() –> onDestroy()

7. 解绑绑定Service服务

unbindService() –> onUnbind(ture) –> bindService() –>
onRebind()
```

1. Service的基本用法

1.1. manifest 文件

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.service">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
```

```
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyService"
            android:enabled="true"
            android:exported="true"></service>
    </application>

</manifest>
```

这段代码不是手工加入的，只需在 Android Studio 中选择 File - New - Service - Service 创建 Service 会自动加入下面代码

```
<service
            android:name=".MyService"
            android:enabled="true"
            android:exported="true"></service>
```

1.2. 创建 Service

在 Android Studio 中选择 File - New - Service - Service 创建 Service

MyService继承自Service，并重写父类的onCreate()、onStartCommand()和onDestroy()方法

```
package cn.netkiller.service;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service {
    public MyService() {
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d("Service", "onCreate() executed");
        Log.d("Service", "MyService thread id is " +
Thread.currentThread().getId());
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {
        Log.d("Service", "onStartCommand() executed");
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d("Service", "onDestroy() executed");
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the
service.
        throw new UnsupportedOperationException("Not yet
implemented");
    }
}
```

```
```

onCreate() Service 创建的时候执行，已经创建的Service不会再执行

onStartCommand() 任何时候，只要执行 startService(intent); 便会执行

onDestroy() 停止的时候执行

1.3. Layout 代码

在布局文件中加入了两个按钮，一个用于启动Service，一个用于停止Service

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="368dp"
        android:layout_height="229dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0">
```

```
        <Button
            android:id="@+id/startService"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Start service" />

        <Button
            android:id="@+id/stopService"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Stop service" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

1.4. Activity 代码

```
package cn.netkiller.service;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    private Button startService;
    private Button stopService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        startService = (Button)
```

```
findViewById(R.id.startService);
        stopService = (Button)
findViewById(R.id.stopService);
        startService.setOnClickListener(this);
        stopService.setOnClickListener(this);

        Log.d("Service", "MainActivity thread id is " +
Thread.currentThread().getId());

    }

    @Override
    public void onClick(View v) {
        Intent intent;
        switch (v.getId()) {
            case R.id.startService:
                intent = new Intent(this, MyService.class);
                startService(intent);
                break;
            case R.id.stopService:
                intent = new Intent(this, MyService.class);
                stopService(intent);
                break;
            default:
                break;
        }
    }
}
```

2. Service 中启动线程

```
    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {

        Log.d("Service", "onStartCommand() begin");
        new Thread(new Runnable() {
            @Override
            public void run() {
                // 开始执行后台任务
                Log.d("Service", "onStartCommand()
executed");
            }
        }).start();

        Log.d("Service", "onStartCommand() end");

        return super.onStartCommand(intent, flags, startId);
    }
```

3. Service 和 Activity 通信

3.1. Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="368dp"
        android:layout_height="229dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0">

        <Button
            android:id="@+id/startService"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Start service" />

        <Button
            android:id="@+id/stopService"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Stop service" />
```

```
        <Button
            android:id="@+id/bindService"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Bind Service" />

        <Button
            android:id="@+id/unbindService"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Unbind Service" />

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

3.2. Service

```
package cn.netkiller.service;

import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service {

    private MyBinder myBinder = new MyBinder();

    public MyService() {
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d("Service", "onCreate() executed");
        Log.d("Service", "MyService thread id is " +
```

```
Thread.currentThread().getId());
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {

        Log.d("Service", "onStartCommand() begin");
        new Thread(new Runnable() {
            @Override
            public void run() {
                // 开始执行后台任务
                Log.d("Service", "onStartCommand()
executed");
            }
        }).start();

        Log.d("Service", "onStartCommand() end");

        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d("Service", "onDestroy() executed");
    }

    @Override
    public IBinder onBind(Intent intent) {
        return myBinder;
    }

    class MyBinder extends Binder {

        public void startTask() {
            new Thread(new Runnable() {
                @Override
                public void run() {
                    // 执行具体的任务
                    Log.d("Service", "startTask()");
                }
            }).start();
        }
```

```
    }
}
```

3.3. Activity

```
package cn.netkiller.service;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity
implements View.OnClickListener {

    private Button startService;
    private Button stopService;
    private Button bindService;
    private Button unbindService;

    private MyService.MyBinder myBinder;

    private ServiceConnection connection = new
ServiceConnection() {

        @Override
        public void onServiceDisconnected(ComponentName name)
{
        }

        @Override
        public void onServiceConnected(ComponentName name,
IBinder service) {
            myBinder = (MyService.MyBinder) service;
```

```
            myBinder.startTask();
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        startService = (Button)
findViewById(R.id.startService);
        stopService = (Button)
findViewById(R.id.stopService);
        startService.setOnClickListener(this);
        stopService.setOnClickListener(this);

        bindService = (Button)
findViewById(R.id.bindService);
        unbindService = (Button)
findViewById(R.id.unbindService);
        bindService.setOnClickListener(this);
        unbindService.setOnClickListener(this);

        Log.d("Service", "MainActivity thread id is " +
Thread.currentThread().getId());

    }

    @Override
    public void onClick(View v) {
        Intent intent;
        switch (v.getId()) {
            case R.id.startService:
                intent = new Intent(this, MyService.class);
                startService(intent);
                break;
            case R.id.stopService:
                intent = new Intent(this, MyService.class);
                stopService(intent);
                break;
            case R.id.bindService:
                Intent bindIntent = new Intent(this,
MyService.class);
                bindService(bindIntent, connection,
BIND_AUTO_CREATE);
```

```
                break;
            case R.id.unbindService:
                unbindService(connection);
                break;
            default:
                break;
        }
    }
}
```

4. Service 和 Toast

```
Handler handler=new Handler(Looper.getMainLooper());
handler.post(new Runnable(){
    public void run(){
        Toast.makeText(getApplicationContext() ,"显示Toast在屏
幕上! ",Toast.LENGTH_LONG).show();
    }
});
```

```
Handler handler = new Handler(Looper.getMainLooper());
handler.post(() -> {
    Toast.makeText(getApplicationContext(), "显示Toast在屏幕
上! ", Toast.LENGTH_LONG).show();
    notify1();
});
```

5. Service 中启动 Activity

```
        Intent intent = new Intent(getBaseContext(),
FullscreenActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    getApplication().startActivity(intent);
```

6. Service 中更新 UI

```
// 在 Service 中定义一个 Handler
private Handler mHandler = new
Handler(Looper.getMainLooper());

// 在 Service 中定义一个 Runnable 对象
private Runnable mRunnable = new Runnable() {
    @Override
    public void run() {
        // 在这里执行与 UI 相关的操作
    }
};

// 在 Service 中使用 Handler 将 Runnable 对象发送到 UI 线程的消息队
列中
mHandler.post(mRunnable);
```

```
private Handler handler = new
Handler(Looper.getMainLooper());
handler.post(() -> {
    musicSkillComponent.stop();
});
```

第 47 章 Notification 通知中心

1. 文本通知

```
    private int createNotification(String title, String text)
{
        String channelId = "channelId";
        String channelName = "channelName";
        String description = "description";
        String group = "group";
        int notificationId = new Random().nextInt(101);
        NotificationManagerCompat notificationManagerCompat =
NotificationManagerCompat.from(this);
        NotificationChannel channel = new
NotificationChannel(channelId, channelName,
NotificationManager.IMPORTANCE_HIGH);
        channel.setDescription(description);

notificationManagerCompat.createNotificationChannel(channel);

        NotificationCompat.Builder notification = new
NotificationCompat.Builder(this, channelId)
                .setContentTitle(title).setContentText(text)
                .setSmallIcon(R.mipmap.ic_launcher)

.setPriority(NotificationCompat.PRIORITY_HIGH)
                .setAutoCancel(true).setGroup(group);

        notificationManagerCompat.notify(notificationId,
notification.build());
        return notificationId;
    }
```

```
        CharSequence name = "test";
```

```
        String description = "test";
        NotificationChannel channel = new
NotificationChannel("test", name,
NotificationManager.IMPORTANCE_DEFAULT);
        channel.setDescription(description);


        NotificationManager notificationManager =
(NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

notificationManager.createNotificationChannel(channel);
        Notification.Builder builder = new
Notification.Builder(this, "test")
                .setContentTitle("XXX 门票打折")          //标
题
                .setContentText("参与 XXX 领取 XXX")      //内
容
                .setSubText("打折信息")                   //内
容下面的一小段文字
                .setWhen(System.currentTimeMillis())     //设
置通知时间
                .setSmallIcon(R.mipmap.ic_launcher)      //设
置小图标
                .setAutoCancel(false);                   //设
置点击后取消Notification

        notificationManager.notify(1, builder.build());
```

2. 添加点击操作

```
    private int createNotification(String title, String text)
{
        String channelId = "channelId";
        String channelName = "channelName";
        String description = "description";
        String group = "group";
        int notificationId = new Random().nextInt(101);

        Intent intent = new Intent(this,
FullscreenActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
        PendingIntent pendingIntent =
PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_IMMUTABLE);

        NotificationManagerCompat notificationManagerCompat =
NotificationManagerCompat.from(this);
        NotificationChannel channel = new
NotificationChannel(channelId, channelName,
NotificationManager.IMPORTANCE_HIGH);
        channel.setDescription(description);

notificationManagerCompat.createNotificationChannel(channel);

        NotificationCompat.Builder notification = new
NotificationCompat.Builder(this, channelId)
                .setContentTitle(title).setContentText(text)
                .setSmallIcon(R.mipmap.ic_launcher)

.setPriority(NotificationCompat.PRIORITY_HIGH)
                .setContentIntent(pendingIntent)
                .setAutoCancel(true);

        notificationManagerCompat.notify(notificationId,
notification.build());
        return notificationId;
    }
```

第 48 章 NFC (Near field communication)

1. AndroidManifest.xml 文件配置

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.nfc">
    <!--<uses-sdk android:minSdkVersion="14"/>-->
    <uses-permission android:name="android.permission.NFC" />
    <!-- 要求当前设备必须要有NFC芯片 -->
    <uses-feature
        android:name="android.hardware.nfc"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="NFC 初始化工具"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop">

            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>


            <intent-filter>
                <action
android:name="android.nfc.action.NDEF_DISCOVERED" />
                <category
```

```
android:name="android.intent.category.DEFAULT" />
                <data android:mimeType="text/plain" />

                <!--<data android:mimeType="text/plain" />-->
                <!--<data android:mimeType="*/*" />-->
            </intent-filter>
            <intent-filter>
                <action
android:name="android.nfc.action.TECH_DISCOVERED" />
            </intent-filter>

            <intent-filter>
                <action
android:name="android.nfc.action.TAG_DISCOVERED" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>

        </activity>

    </application>

</manifest>
```

2. Loyout 文件

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <TextView
        android:id="@+id/ndefMessage"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="8dp"
        android:text="message"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4"
/>

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        android:text="NDEF Message : "
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.03"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TableLayout
```

```
        android:id="@+id/tableLayout"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/ndefMessage">

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="NFC UID" />

            <TextView
                android:id="@+id/uid"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="UID"
                tools:layout_editor_absoluteX="146dp"
                tools:layout_editor_absoluteY="71dp" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="NFC Tag" />

            <TextView
                android:id="@+id/nfcTag"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

```
            android:text="tag"
            tools:layout_editor_absoluteX="179dp"
            tools:layout_editor_absoluteY="83dp" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="NFC Size" />

        <TextView
            android:id="@+id/size"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="size"
            tools:layout_editor_absoluteX="179dp"
            tools:layout_editor_absoluteY="150dp" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="NDEF Type" />

        <TextView
            android:id="@+id/schema"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="schema"
            tools:layout_editor_absoluteX="168dp"
            tools:layout_editor_absoluteY="257dp" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
```

```
            android:layout_height="match_parent">

        </TableRow>


        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textView7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="NDEF charset" />

            <TextView
                android:id="@+id/charset"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="charset"
                tools:layout_editor_absoluteX="163dp"
                tools:layout_editor_absoluteY="304dp" />


        </TableRow>


        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textView8"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="NDEF Lang" />

            <TextView
                android:id="@+id/language"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="lang"
                tools:layout_editor_absoluteX="163dp"
                tools:layout_editor_absoluteY="331dp" />
        </TableRow>
```

```
        <TableRow
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/textView6"
                android:layout_width="79dp"
                android:layout_height="wrap_content"
                android:text="Status" />

            <TextView
                android:id="@+id/status"
                android:layout_width="289dp"
                android:layout_height="wrap_content"
                android:text="status"
                tools:layout_editor_absoluteX="90dp"
                tools:layout_editor_absoluteY="404dp" />


        </TableRow>
    </TableLayout>


    <TextView
        android:id="@+id/textView9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="48dp"
        android:layout_marginEnd="8dp"
        android:text="NDEF Message write :"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tableLayout" />

    <TextView
        android:id="@+id/ndefWrite"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
```

```
        android:layout_marginEnd="8dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView9"
/>

    <Switch
        android:id="@+id/switchWrite"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="28dp"
        android:layout_marginEnd="8dp"
        android:text="NDEF Message write"
        android:textOff="NDEF Message write Off"
        android:textOn="NDEF Message write On"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ndefWrite"
/>

</android.support.constraint.ConstraintLayout>
```

3. Activity 文件

```
package cn.netkiller.nfc;


import android.nfc.FormatException;
import android.nfc.NdefRecord;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import android.app.PendingIntent;
import android.content.Intent;
import android.nfc.NdefMessage;
import android.nfc.NfcAdapter;

import android.nfc.Tag;
import android.nfc.tech.Ndef;
import android.os.Parcelable;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.Switch;
import android.widget.TextView;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private NfcAdapter nfcAdapter;
    private PendingIntent pendingIntent;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        final TextView status = (TextView)
findViewById(R.id.status);

        nfcAdapter = NfcAdapter.getDefaultAdapter(this);

        if (nfcAdapter == null) {
            System.out.println("**** NFC ERROR ****");
            status.setText("NFC is not available.");
            return;
        } else if (!nfcAdapter.isEnabled()) {
            status.setText("请开启系统NFC功能");
        }

        status.setText("Start...");

        pendingIntent = PendingIntent.getActivity(this, 0,
new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);


        final Switch switchWrite = (Switch)
findViewById(R.id.switchWrite);

        switchWrite.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton
buttonView, boolean isChecked) {

                if (isChecked) {

status.setText(switchWrite.getTextOn().toString());
                } else {

status.setText(switchWrite.getTextOff().toString());
                }
            }
        });
    }

    @Override
    protected void onResume() {
        super.onResume();
```

```
        nfcAdapter.enableForegroundDispatch(this,
pendingIntent, null, null);
    }

    @Override
    protected void onPause() {
        super.onPause();
        nfcAdapter.disableForegroundDispatch(this);
    }


    //当窗口的创建模式是singleTop或singleTask时调用，用于取代
onCreate方法
    //当NFC标签靠近手机，建立连接后调用
    @Override
    public void onNewIntent(Intent intent) {
        super.onNewIntent(intent);

        TextView status = (TextView)
findViewById(R.id.status);
        TextView type = (TextView) findViewById(R.id.nfcTag);
        TextView size = (TextView) findViewById(R.id.size);
        TextView uidTextView = (TextView)
findViewById(R.id.uid);
        TextView ndefMessage = (TextView)
findViewById(R.id.ndefMessage);
        TextView schema = (TextView)
findViewById(R.id.schema);
        TextView charset = (TextView)
findViewById(R.id.charset);
        TextView language = (TextView)
findViewById(R.id.language);
        TextView ndefWrite = (TextView)
findViewById(R.id.ndefWrite);
        Switch switchWrite = (Switch)
findViewById(R.id.switchWrite);


        Tag tag =
intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);

        if (switchWrite.isChecked()) {

            UUID uuid = UUID.randomUUID();
            try {
```

```
                write(uuid.toString(), tag);
                ndefWrite.setText(uuid.toString());
            } catch (IOException e) {
                e.printStackTrace();
            } catch (FormatException e) {
                e.printStackTrace();
            }

        }

        if
(NfcAdapter.ACTION_NDEF_DISCOVERED.equals(intent.getAction())
) {
            status.setText("Read NDEF Message...");

            byte[] uid = tag.getId();
            BigInteger n = new BigInteger(uid);
            String hex = n.toString(16);
            uidTextView.setText(hex);

            Ndef ndef = Ndef.get(tag);
            String log = ndef.getType() + "\n最大数据容量: " +
ndef.getMaxSize() + " bytes\n\n";
            System.out.println(log);
            type.setText(ndef.getType());
            size.setText(ndef.getMaxSize() + " bytes");

            Parcelable[] rawMessages =
intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES
);
            if (rawMessages != null) {
                NdefMessage[] messages = new
NdefMessage[rawMessages.length];
                for (int i = 0; i < rawMessages.length; i++)
{
                    messages[i] = (NdefMessage)
rawMessages[i];
                }
                byte[] payload = messages[0].getRecords()
[0].getPayload();

                try {

                    String tagId = new
String(messages[0].getRecords()[0].getType());
```

```
                    schema.setText(tagId);

                    String encoding = ((payload[0] & 128) ==
0) ? "UTF-8" : "UTF-16";
                    charset.setText(encoding);

                    int languageCodeLength = payload[0] &
0x3f;
                    String languageCode = new String(payload,
1, languageCodeLength, "US-ASCII");


//                    String lang = new String(payload, 1,
payload[0] & 0063, "US-ASCII");
                    language.setText(languageCode);

//                    String text = new
String(messages[0].getRecords()[0].getPayload());
                    String text = new String(payload,
languageCodeLength + 1, payload.length - languageCodeLength -
1, encoding);
                    ndefMessage.setText(text);

                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }

            }

        } else {
            status.setText("NOT NDEF Messages tag");
        }
    }


    private void write(String text, Tag tag) throws
IOException, FormatException {
        NdefRecord[] records = {createRecord(text)};
        NdefMessage message = new NdefMessage(records);
        // Get an instance of Ndef for the tag.
        Ndef ndef = Ndef.get(tag);
        // Enable I/O
        ndef.connect();
        // Write the message
        ndef.writeNdefMessage(message);
```

```
        // Close the connection
        ndef.close();
    }

    private NdefRecord createRecord(String text) throws
UnsupportedEncodingException {
        String lang = "en";
        byte[] textBytes = text.getBytes();
        byte[] langBytes = lang.getBytes("US-ASCII");
        int langLength = langBytes.length;
        int textLength = textBytes.length;
        byte[] payload = new byte[1 + langLength +
textLength];

        // set status byte (see NDEF spec for actual bits)
        payload[0] = (byte) langLength;

        // copy langbytes and textbytes into payload
        System.arraycopy(langBytes, 0, payload, 1,
langLength);
        System.arraycopy(textBytes, 0, payload, 1 +
langLength, textLength);

        NdefRecord recordNFC = new
NdefRecord(NdefRecord.TNF_WELL_KNOWN, NdefRecord.RTD_TEXT,
new byte[0], payload);

        return recordNFC;
    }

}
```

第 49 章 图形开发

1. Paint

```
paint = new Paint();
paint.setColor(Color.GREEN);
paint.setColor(0xFF00F8C1);
paint.setStrokeWidth(1);
```

2. AnimationDrawable

```
    private int getTotalDuration(AnimationDrawable
animationDrawable) {
        int totalDuration = 0;
        for (int i = 0; i <
animationDrawable.getNumberOfFrames(); i++) {
            totalDuration +=
animationDrawable.getDuration(i);
        }
        return totalDuration;
    }
```

第 50 章 下载管理

1. 从 URL 下来文件

```
    DownloadManager manager = (DownloadManager)
getSystemService(Context.DOWNLOAD_SERVICE);
    Uri uri =
Uri.parse("https://www.netkiller.cn/linux/images/cover.png");
    DownloadManager.Request request = new
DownloadManager.Request(uri);

request.setNotificationVisibility(DownloadManager.Request.VIS
IBILITY_VISIBLE);
    long reference = manager.enqueue(request);
```

2. 安装 APK

```
fun installApk(apkPath: Uri?) {
    val intent = Intent()
    intent.action = Intent.ACTION_VIEW
    intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK
    intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION)
    intent.setDataAndType(apkPath,
"application/vnd.android.package-archive")
    builder.context.startActivity(intent)
}
```

3. 下载后接收广播通知

```
    public void download(String url) {
        try {
            //下载路径，如果路径无效了，可换成你的下载路径
//            String url =
"https://www.netkiller.cn/linux/images/cover.png";
            String fileName = "test.jpg";
            if (!url.contains("https")) {
                url = url.replaceFirst("http", "https");
            }
            //创建下载任务,downloadUrl就是下载链接
            DownloadManager.Request request = new
DownloadManager.Request(Uri.parse(url));
            //指定下载路径和下载文件名
//
request.setDestinationInExternalPublicDir(Environment.getExte
rnalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
url.substring(url.lastIndexOf("/") + 1));
//
request.setNotificationVisibility(DownloadManager.Request.VIS
IBILITY_VISIBLE);
//            request.setTitle("Image Download");
//            request.setDescription("Image download using
DownloadManager.");
//
request.setDestinationInExternalPublicDir(getExternalFilesDir
(Environment.DIRECTORY_PICTURES) + "", "sample2.jpg");

request.setDestinationInExternalPublicDir(Environment.DIRECTO
RY_PICTURES, fileName);

request.setAllowedNetworkTypes(DownloadManager.Request.NETWOR
K_WIFI | DownloadManager.Request.NETWORK_MOBILE);

request.setNotificationVisibility(DownloadManager.Request.VIS
IBILITY_VISIBLE_NOTIFY_COMPLETED);
            //request.allowScanningByMediaScanner();
```

```
            //获取下载管理器
            DownloadManager downloadManager =
(DownloadManager) getSystemService(Context.DOWNLOAD_SERVICE);
            //将下载任务加入下载队列，否则不会进行下载
            long reference =
downloadManager.enqueue(request);
            broadcast(reference);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void broadcast(final long downloadId) {

        // 注册广播监听系统的下载完成事件。
        IntentFilter intentFilter = new
IntentFilter(DownloadManager.ACTION_DOWNLOAD_COMPLETE);
        BroadcastReceiver broadcastReceiver = new
BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent
intent) {
                try {
//                  long ID =
intent.getLongExtra(DownloadManager.EXTRA_DOWNLOAD_ID, -1);
                    long id =
intent.getLongExtra(DownloadManager.EXTRA_DOWNLOAD_ID, 0);
                    if (id == downloadId) {

Toast.makeText(getApplicationContext(), "任务:" + id + " 下载完
成!", Toast.LENGTH_LONG).show();
                        Log.d(TAG, "任务:" + id + " 下载完
成!");
                    }
                    if
(DownloadManager.ACTION_DOWNLOAD_COMPLETE.equals(intent.getAc
tion())) {
                        DownloadManager downloadManager =
(DownloadManager) getSystemService(Context.DOWNLOAD_SERVICE);
//                        Uri uri =
downloadManager.getUriForDownloadedFile(id);
//                        Log.d(TAG, uri.toString());
                        DownloadManager.Query query = new
```

```
DownloadManager.Query();
//                    //在广播中取出下载任务的id
                        query.setFilterById(id);
                        Cursor cursor =
downloadManager.query(query);
                        if (cursor.moveToFirst()) {

//                          int fileNameIdx =
cursor.getColumnIndex(DownloadManager.COLUMN_LOCAL_FILENAME);
                            //获取文件下载路径
                            int fileUriIdx =
cursor.getColumnIndex(DownloadManager.COLUMN_LOCAL_URI);

//                          String fileName =
cursor.getString(fileNameIdx);

                            String fileUri =
cursor.getString(fileUriIdx);
                            Log.d(TAG, "fileUri: " +
fileUri);


//                          //如果文件名不为空，说明已经存在了，拿到文
件名想干嘛都好
                            if (fileUri != null) {
//                            Intent intent = new Intent();
                              intent.setAction("image");
                              intent.putExtra("image",
fileUri);

context.sendBroadcast(intent);
                            }
                        }
                        cursor.close();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        };

        registerReceiver(broadcastReceiver, intentFilter);

    }
```

第 51 章 Android 多线程

1. GPIO

配置权限

```
   new Thread("画画线程") {
        @Override
        public void run() {
            picture(question);
        }
    }.start();

    new Thread("GPT线程") {
        @Override
        public void run() {
            if (question != null) {
                String sentence = cleaning();
                chatgpt(sentence);
            }
        }
    }.start();
```

```
// 步骤1：创建线程类，继承自Thread类
class MyThread extends Thread{

        // 步骤2：复写run（），内容 = 定义线程行为
    @Override
    public void run(){
    ... // 定义的线程行为
    }
}

// 步骤3：实例化线程类
```

```
MyThread mt=new MyThread(“线程名称”);

// 步骤4：通过线程对象控制线程的状态，如 运行、睡眠、挂起   / 停止
// start（）开启线程
mt.start();
```

第 52 章 EventBus

http://greenrobot.org/eventbus

在EventBus中主要有以下三个成员：

```
Event：事件，可以自定义为任意对象，类似Message类的作用；
Publisher：事件发布者，可以在任意线程、任意位置发布Event，已发布的Evnet则由EventBus进行分发；
Subscriber：事件订阅者，接收并处理事件，需要通过register(this)进行注册，而在类销毁时要使用unregister(this)方法解注册。每个Subscriber可以定义一个或多个事件处理方法，其方法名可以自定义，但需要添加@Subscribe的注解，并指明ThreadMode（不写默认为Posting）。
```

1. 添加 EventBus 依赖到项目Gradle文件

Gradle:

```
implementation 'org.greenrobot:eventbus:3.1.1'
```

完整的例子

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "cn.netkiller.eventbus"
        minSdkVersion 26
        targetSdkVersion 28
```

```
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation
'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'org.greenrobot:eventbus:3.1.1'
}
```

2. 快速开始一个演示例子

操作 EventBus 只需四个步骤

```
1. 注册事件

EventBus.getDefault().register( this );

2. 取消注册

EventBus.getDefault().unregister( this );

3. 订阅事件

        @Subscribe(threadMode = ThreadMode.MAIN)
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }

4. 发送数据

EventBus.getDefault().post(new MessageEvent("Helloworld"));
```

2.1. 创建 MessageEvent 类

```
package cn.netkiller.eventbus.pojo;

public class MessageEvent {
    public final String message;

    public MessageEvent(String message) {
        this.message = message;
    }
```

```
}
```

2.2. Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
/>

</android.support.constraint.ConstraintLayout>
```

2.3. Activity

```
package cn.netkiller.eventbus;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

import cn.netkiller.eventbus.pojo.MessageEvent;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EventBus.getDefault().register(this);

        findViewById(R.id.button).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                EventBus.getDefault().post(new
MessageEvent("Hello everyone!"));
            }
        });

    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        //取消注册 , 防止Activity内存泄漏
        EventBus.getDefault().unregister(this);
```

```
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }
}
```

3. Sticky Events

Sticky Events 粘性事件可以理解为Message做了持久化，直到Message被消费为止。无需注册即可发送Message。

下面的例子：在MainActivity发送事件，在StickyActivity里注册并且接收事件

```
A. MainActivity 发送事件：

EventBus.getDefault().postSticky(new
MessageEvent("http://www.netkiller.cn"));

B. StickyActivity 接收事件

1. 注册

EventBus.getDefault().register( this );

2. 事件接收

        @Subscribe(threadMode = ThreadMode.MAIN, sticky = true)
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }

3. 取消注册

EventBus.getDefault().unregister( this ) ;
```

3.1. MainActivity

Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
/>

</android.support.constraint.ConstraintLayout>
```

MainActivity

```
package cn.netkiller.eventbus;
```

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

import cn.netkiller.eventbus.pojo.MessageEvent;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        findViewById(R.id.button).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                EventBus.getDefault().postSticky(new
MessageEvent("Hello everyone!"));
                startActivity(new Intent(MainActivity.this,
StickyActivity.class));
            }
        });

    }

}
```

3.2. StickyActivity

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".StickyActivity">

</android.support.constraint.ConstraintLayout>
```

StickyActivity

```
package cn.netkiller.eventbus;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

import cn.netkiller.eventbus.pojo.MessageEvent;

public class StickyActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sticky);

        EventBus.getDefault().register(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }

    @Subscribe(threadMode = ThreadMode.MAIN, sticky = true)
```

```
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }

}
```

3.3. MessageEvent

```
package cn.netkiller.eventbus.pojo;

public class MessageEvent {
    public final String message;

    public MessageEvent(String message) {
        this.message = message;
    }
}
```

3.4. 删除粘性事件

```
MessageEvent stickyEvent =
EventBus.getDefault().getStickyEvent(MessageEvent.class);

// Better check that an event was actually posted before
if(stickyEvent != null) {
        // "Consume" the sticky event
        EventBus.getDefault().removeStickyEvent(stickyEvent);
        // Now do something with it
}
```

4. 线程模型

EventBus 有五种线程模型（ThreadMode）

Posting：直接在事件发布者所在线程执行事件处理方法；
Main：直接在主线程中执行事件处理方法（即UI线程），如果发布事件的线程也是主线程，那么事件处理方法会直接被调用，并且未避免ANR，该方法应避免进行耗时操作；
MainOrdered：也是直接在主线程中执行事件处理方法，但与Main方式不同的是，不论发布者所在线程是不是主线程，发布的事件都会进入队列按事件串行顺序依次执行；
BACKGROUND：事件处理方法将在后台线程中被调用。如果发布事件的线程不是主线程，那么事件处理方法将直接在该线程中被调用。如果发布事件的线程是主线程，那么将使用一个单独的后台线程，该线程将按顺序发送所有的事件。
Async：不管发布者的线程是不是主线程，都会开启一个新的线程来执行事件处理方法。如果事件处理方法的执行需要一些时间，例如网络访问，那么就应该使用该模式。为避免触发大量的长时间运行的事件处理方法，EventBus使用了一个线程池来有效地重用已经完成调用订阅者方法的线程以限制并发线程的数量。　后面会通过代码展示五种ThreadMode的工作方式。

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ThreadModeActivity">

    <Button
        android:id="@+id/buttonSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="Send"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/buttonThread"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:text="Send Thread"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/buttonSend"
/>
</android.support.constraint.ConstraintLayout>
```

```
package cn.netkiller.eventbus;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;

public class ThreadModeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_thread_mode);

        EventBus.getDefault().register(this);

        findViewById(R.id.buttonSend).setOnClickListener(new
View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                Log.d("EventBus Thread : ",
Thread.currentThread().getName());

EventBus.getDefault().post("http://www.netkiller.cn");
            }
        });


findViewById(R.id.buttonThread).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        Log.d("EventBus Thread : ",
Thread.currentThread().getName());

EventBus.getDefault().post("http://www.netkiller.cn");

                    }
                }).start();

            }
        });

    }

    @Subscribe(threadMode = ThreadMode.POSTING)
    public void onMessageEventPostThread(String event) {
        Log.d("EventBus PostThread", "Message:  " + event + "
thread: " + Thread.currentThread().getName());
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onMessageEventMainThread(String event) {
        Log.d("EventBus MainThread", "Message:  " + event + "
thread: " + Thread.currentThread().getName());
    }

    @Subscribe(threadMode = ThreadMode.MAIN_ORDERED)
    public void onEventMainOrdered(String event) {
        Log.d("EventBus MainOrdered", "Message: " + event + "
```

```
thread:" + Thread.currentThread().getName());
    }

    @Subscribe(threadMode = ThreadMode.BACKGROUND)
    public void onMessageEventBackgroundThread(String event)
{
        Log.d("EventBus BackgroundThread", "Message:  " +
event + "  thread: " + Thread.currentThread().getName());
    }

    @Subscribe(threadMode = ThreadMode.ASYNC)
    public void onMessageEventAsync(String event) {
        Log.d("EventBus Async", "Message:  " + event + "
thread: " + Thread.currentThread().getName());
    }


    @Override
    protected void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }
}
```

在 main 线程中发布消息

```
D/EventBus Thread :: main
D/EventBus MainThread: Message:  http://www.netkiller.cn
thread: main
D/EventBus PostThread: Message:  http://www.netkiller.cn
thread: main
D/EventBus Async: Message:  http://www.netkiller.cn  thread:
pool-1-thread-1
D/EventBus BackgroundThread: Message:  http://www.netkiller.cn
thread: pool-1-thread-2
D/EventBus MainOrdered: Message: http://www.netkiller.cn
thread:main
```

在线程中发布消息

```
D/EventBus Thread :: Thread-2
D/EventBus BackgroundThread: Message:  http://www.netkiller.cn
thread: Thread-2
D/EventBus PostThread: Message:  http://www.netkiller.cn
thread: Thread-2
D/EventBus Async: Message:  http://www.netkiller.cn  thread:
pool-1-thread-2
D/EventBus MainOrdered: Message: http://www.netkiller.cn
thread:main
D/EventBus MainThread: Message:  http://www.netkiller.cn
thread: main
```

5. 配置 EventBus

上面章节中的例子EventBus实例中采用默认方式

```
EventBus.getDefault().register(this);
```

这种方式的获取到的EventBus的都是默认属性，有时候并不能满足我们的要求，这时候我们可以通过EventBusBuilder来个性化配置EventBus的属性。

```
// 创建默认的EventBus对象，相当于EventBus.getDefault()。

EventBus installDefaultEventBus()：
// 添加由EventBus“注释预处理器生成的索引
EventBuilder addIndex(SubscriberInfoIndex index)：
// 默认情况下，EventBus认为事件类有层次结构（订户超类将被通知）
EventBuilder eventInheritance(boolean eventInheritance)：
// 定义一个线程池用于处理后台线程和异步线程分发事件
EventBuilder
executorService(java.util.concurrent.ExecutorService
executorService)：
// 设置忽略订阅索引，即使事件已被设置索引，默认为false
EventBuilder ignoreGeneratedIndex(boolean
ignoreGeneratedIndex)：
// 打印没有订阅消息，默认为true
EventBuilder logNoSubscriberMessages(boolean
logNoSubscriberMessages)：
// 打印订阅异常，默认true
EventBuilder logSubscriberExceptions(boolean
logSubscriberExceptions)：
// 设置发送的的事件在没有订阅者的情况时，EventBus是否保持静默，默认true
EventBuilder sendNoSubscriberEvent(boolean
sendNoSubscriberEvent)：
// 发送分发事件的异常，默认true
```

```
EventBuilder sendSubscriberExceptionEvent(boolean
sendSubscriberExceptionEvent)：
// 在3.0以前，接收处理事件的方法名以onEvent开头，方法名称验证避免不是以
此开头，启用严格的方法验证（默认：false）
EventBuilder strictMethodVerification(java.lang.Class<?>
clazz)
// 如果onEvent***方法出现异常，是否将此异常分发给订阅者（默认：false）
EventBuilder throwSubscriberException(boolean
throwSubscriberException)
```

我的实例参考

```
EventBus eventBus = EventBus.builder().eventInheritance(true)
    .ignoreGeneratedIndex(false)
    .logNoSubscriberMessages(true)
    .logSubscriberExceptions(false)
    .sendNoSubscriberEvent(true)
    .sendSubscriberExceptionEvent(true)
    .throwSubscriberException(false)
    .strictMethodVerification(true)
    .build();
eventBus.register(this);
```

6. 事件优先级

priority 数值越大优先级又高

```
        // MainActivity
        @Subscribe(priority = 2)
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }

        // SecondActivity
        @Subscribe(priority = 1)
    public void onMessageSecondEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }
```

时间拦截，MainActivity 收到信息后调用 EventBus.getDefault().cancelEventDelivery(event); 之后所有订阅将收不到信息。

```
        // MainActivity
        @Subscribe(priority = 2)
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
        EventBus.getDefault().cancelEventDelivery(event);
    }

        // SecondActivity
        @Subscribe(priority = 1)
    public void onMessageSecondEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
```

```
Toast.LENGTH_SHORT).show();
    }
```

7. 捕获异常事件

在 init() 中加入你的业务逻辑，根据需要，在特定的情况下使用 throw new Exception("异常信息"); 抛出异常。异常会被 hrowableFailureEvent(ThrowableFailureEvent event) 捕获到。

```
package cn.netkiller.eventbus;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;

import org.greenrobot.eventbus.EventBus;
import org.greenrobot.eventbus.Subscribe;
import org.greenrobot.eventbus.ThreadMode;
import org.greenrobot.eventbus.util.AsyncExecutor;
import org.greenrobot.eventbus.util.ThrowableFailureEvent;

import cn.netkiller.eventbus.pojo.MessageEvent;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EventBus.getDefault().register(this);

        findViewById(R.id.button).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

                AsyncExecutor.create().execute(
                        new AsyncExecutor.RunnableEx() {
```

```
                                @Override
                                public void run() throws
Exception {
                                    init();

EventBus.getDefault().post(new MessageEvent("Hello
everyone!"));
                                }
                            }
                    );
                }
            });
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        EventBus.getDefault().unregister(this);
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void onMessageEvent(MessageEvent event) {
        Toast.makeText(this, event.message,
Toast.LENGTH_SHORT).show();
    }

    public void init() throws Exception {
        // ...
        throw new Exception("实际发送异常");
    }

    @Subscribe(threadMode = ThreadMode.MAIN)
    public void hrowableFailureEvent(ThrowableFailureEvent
event) {
        Log.d("EventBus", "hrowableFailureEvent: " +
event.getThrowable().getMessage());
        Toast.makeText(this,
event.getThrowable().getMessage(),
Toast.LENGTH_SHORT).show();
    }

}
```

第 53 章 Android MQTT

1. build.gradle 添加依赖包

```
implementation group: 'org.eclipse.paho', name:
'org.eclipse.paho.mqttv5.client', version: '1.2.5'
implementation group: 'org.eclipse.paho', name:
'org.eclipse.paho.android.service', version: '1.1.1', ext:
'pom'
```

提示

2. AndroidManifest.xml

```
 <uses-permission android:name="android.permission.WAKE_LOCK"
/>
 <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
 <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
 <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
 <uses-permission android:name="android.permission.INTERNET" />
```

3. Android Mqtt v5 例子

```
package cn.netkiller.ropeway.service;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

import org.eclipse.paho.mqttv5.client.IMqttDeliveryToken;
import org.eclipse.paho.mqttv5.client.IMqttToken;
import org.eclipse.paho.mqttv5.client.MqttAsyncClient;
import org.eclipse.paho.mqttv5.client.MqttCallback;
import org.eclipse.paho.mqttv5.client.MqttConnectionOptions;
import org.eclipse.paho.mqttv5.client.MqttDisconnectResponse;
import
org.eclipse.paho.mqttv5.client.persist.MemoryPersistence;
import org.eclipse.paho.mqttv5.common.MqttException;
import org.eclipse.paho.mqttv5.common.MqttMessage;
import org.eclipse.paho.mqttv5.common.packet.MqttProperties;

public class MyService extends Service {
    MqttAsyncClient mqttAsyncClient;
    IMqttToken token;

    String topic = "/netkiller/test";
    String content = "Helloworld!!!";
    int qos = 2;
    String broker = "tcp://broker.emqx.io:1883";
    String clientId = "JavaSample" +
System.currentTimeMillis();

    public MyService() {
        try {
            MemoryPersistence persistence = new
MemoryPersistence();
            mqttAsyncClient = new MqttAsyncClient(broker,
clientId, persistence);
        } catch (MqttException me) {
            System.out.println("reason " +
```

```
me.getReasonCode());
            System.out.println("msg " + me.getMessage());
            System.out.println("loc " +
me.getLocalizedMessage());
            System.out.println("cause " + me.getCause());
            System.out.println("excep " + me);
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the
service.
        throw new UnsupportedOperationException("Not yet
implemented");
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d("Service", "onCreate() executed");
        try {
            MqttConnectionOptions mqttConnectionOptions = new
MqttConnectionOptions();
            mqttConnectionOptions.setCleanStart(false);

mqttConnectionOptions.setAutomaticReconnect(true);

            Log.d("Service", "Connecting to broker: " +
broker);

            token =
mqttAsyncClient.connect(mqttConnectionOptions);
            token.waitForCompletion();
            if (token.isComplete()) {
                Log.d("Service", "Connected");

mqttAsyncClient.subscribe("/netkiller/message", qos);
            }

        } catch (MqttException e) {
            throw new RuntimeException(e);
        }

        mqttAsyncClient.setCallback(new MqttCallback() {
```

```
            @Override
            public void disconnected(MqttDisconnectResponse
disconnectResponse) {

            }

            @Override
            public void mqttErrorOccurred(MqttException
exception) {

            }

            @Override
            public void messageArrived(String topic,
MqttMessage message) throws Exception {
                String msg = new
String(message.getPayload());
                Log.d("Service", String.format("接收消息 Id:%s,
Topic: %s, QoS: %s, Message: %s, ", message.getId(), topic,
message.getQos(), message.toString()));
            }

            @Override
            public void deliveryComplete(IMqttToken token) {

            }

            @Override
            public void connectComplete(boolean reconnect,
String serverURI) {
//                if (reconnect) {
//                    try {
//
mqttAsyncClient.subscribe("/netkiller/message", qos);
//                    } catch (MqttException e) {
//                        throw new RuntimeException(e);
//                    }
//                }
            }

            @Override
            public void authPacketArrived(int reasonCode,
MqttProperties properties) {
```

```
            }


            public void deliveryComplete(IMqttDeliveryToken
arg0) {
                try {
                    System.out.println(arg0.getMessage());
                } catch (MqttException e1) {
                    e1.printStackTrace();
                }

            }


            public void connectionLost(Throwable err) {
                System.out.println("连接丢失");
                System.out.println(err.getMessage());

            }
        });


    }

    @Override
    public int onStartCommand(Intent intent, int flags, int
startId) {
        Log.d("Service", "onStartCommand() executed");

        try {
            Log.d("Service", "Publishing message: " +
content);
            MqttMessage message = new
MqttMessage(content.getBytes());
            message.setQos(qos);
            token = mqttAsyncClient.publish(topic, message);
            token.waitForCompletion();
        } catch (MqttException e) {
            throw new RuntimeException(e);
        }

        return super.onStartCommand(intent, flags, startId);
    }

    @Override
```

```
    public void onDestroy() {
        super.onDestroy();
        try {
            if (mqttAsyncClient.isConnected()) {
                mqttAsyncClient.close();
                Log.d("Service", "Close client.");
            }
        } catch (MqttException e) {
            Log.d("Service", "Disconnected");
        }
        Log.d("Service", "onDestroy() executed");
    }
}
```

第 54 章 安卓开发版

1. rk3568

1.1. 声卡

```
rk3568_r:/storage/emulated/0 # cat /proc/asound/cards
 0 [rockchiphdmi   ]: rockchip_hdmi - rockchip,hdmi
                      rockchip,hdmi
 1 [rockchiprk809co]: rockchip_rk809- - rockchip,rk809-codec
                      rockchip,rk809-codec
 2 [UR22C          ]: USB-Audio - Steinberg UR22C
                      Yamaha Corporation Steinberg UR22C at
usb-xhci-hcd.5.auto-1, high speed
```

```
rk3568_r:/storage/emulated/0 # cat /proc/asound/devices
  2: [ 0- 0]: digital audio playback
  3: [ 0]   : control
  4: [ 1- 0]: digital audio playback
  5: [ 1- 0]: digital audio capture
  6: [ 1]   : control
  7: [ 2- 0]: digital audio playback
  8: [ 2- 0]: digital audio capture
  9: [ 2]   : control
 10: [ 2- 0]: raw midi
 33:        : timer
```

```
rk3568_r:/storage/emulated/0 # cat /proc/asound/pcm
00-00: fe400000.i2s-i2s-hifi i2s-hifi-0 : fe400000.i2s-i2s-hifi
```

```
i2s-hifi-0 : playback 1
01-00: fe410000.i2s-rk817-hifi rk817-hifi-0 : fe410000.i2s-
rk817-hifi rk817-hifi-0 : playback 1 : capture 1
02-00: USB Audio : USB Audio : playback 1 : capture 1
```

第 55 章 杂项

1. Sleep

```
try {
    Thread.sleep(10000);
} catch (InterruptedException e) {
    Log.e("Location", e.getMessage());
}
```

2. Caused by: java.net.UnknownServiceException: CLEARTEXT communication to 47.100.253.187 not permitted by network security policy

在AndroidManifest.xml的文件的application节点中增加 android:usesCleartextTraffic="true"

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET"
/>

    <application
        android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Ropeway"
        android:usesCleartextTraffic="true"
```

```
        tools:targetApi="31">
        <service
            android:name=".service.RopewayService"
            android:enabled="true"
            android:exported="true"></service>
        <service
            android:name=".service.BroadcastService"
            android:enabled="true"
            android:exported="true" />

        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <meta-data
        android:name="apiUrl"
        android:value="http://47.100.25.18:8000" />
</manifest>
```

3. 设计模式

3.1. 单例模式

```
package cn.netkiller.voice;

import android.media.MediaRecorder;
import android.os.Environment;
import android.util.Log;

import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Audio {

    private boolean isRecord = false;

    private MediaRecorder mediaRecorder;
    private String filename;

    private Audio() {

    }

    private static Audio instance;

    public synchronized static Audio getInstance() {
        if (instance == null)
            instance = new Audio();
        return instance;
    }

    public String getFilename() {
        return filename;
    }

    public void start() {
```

```
        if (mediaRecorder == null) {

            String path =
Environment.getExternalStorageDirectory().getPath();
            String folder = new SimpleDateFormat("yyyy-MM-
dd").format(new Date());
            String name = new
SimpleDateFormat("hhmmss").format(new Date());
            new File(path, folder).mkdirs();

            filename = String.format("%s/%s/%s.3gp", path,
folder, name);
            Log.e("Voice", "voice path " + filename);

            try {

                mediaRecorder = new MediaRecorder();

mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFA
ULT);

mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFA
ULT);
                mediaRecorder.setOutputFile(filename);
                mediaRecorder.prepare();
                mediaRecorder.start();

                isRecord = true;

            } catch (IOException ex) {
                ex.printStackTrace();

            }
        }

    }

    public void stop() {
        if (mediaRecorder != null && isRecord) {
            System.out.println("stopRecord");
            isRecord = false;
            mediaRecorder.stop();
```

```
            mediaRecorder.release();
            mediaRecorder = null;
        }
    }

}
```

4. Android OS 包

4.1. 进程ID

```
android.os.Process.myTid()
```

4.2. handler

```
        handler.postDelayed(() -> {
        }, 1000);
```

5. fastjson android

```
implementation 'com.alibaba:fastjson:2.0.20.android'
```

5.1. 对象转字符串

```
String json = JSON.toJSONString(user); //序列化
```

5.2. JsonObject 转对象

对象转JsonObject

```
User user =JSON.parseObject(json,User.class); //反序列化
JSONObject jsonObject=(JSONObject)JSON.toJSON(user);
jsonObject.getIntValue("id");
```

jsonObject 转 Java Object

```
User user=JSON.toJavaObject(jsonObject, User.class);
```

5.3. 字符串 与 json 互转

json 转 字符串

```
String jsonString=JSON.toJSONString(jsonObject);
```

字符串 转 json

```
JSONObject jsonObject=JSON.parseObject(jsonString);
jsonObject.getString("name");
```

5.4. json 转 数组

```
JSONArray
jArray=JSON.parseArray(JSON.toJSONString(userList));
```

5.5. JSON数组转List

```
List<Map> listMaps =
JSONArray.parseArray(JSON.toJSONString(data),Map.class);
List<Map> mapsList =
JSONObject.parseArray(JSON.toJSONString(data), Map.class);
```

5.6. Map 与 Json 互转

Json 转 map

```
Map<String,Object> maps =
JSONObject.parseObject(json2,Map.class);
```

Map转JSON

```
JSONObject jsonObject =
JSONObject.parseObject(JSON.toJSONString(maps));
```

6. Butter Knife

http://jakewharton.github.io/butterknife/

7. Android Things

7.1. GPIO

配置权限

```
<uses-permission
android:name="com.google.android.things.permission.USE_PERIPHER
AL_IO" />
<uses-permission
android:name="com.google.android.things.permission.MANAGE_INPUT
_DRIVERS" />
```

第 56 章 FAQ

1. java.net.UnknownServiceException: CLEARTEXT communication to 192.168.0.185 not permitted by network security policy

okhttp 默认使用 https 链接服务器，如果使用 http 会抛出现上面的异常

```
if (!Platform.get().isCleartextTrafficPermitted(host)) {
      throw new RouteException(new UnknownServiceException(
          "CLEARTEXT communication to " + host + " not
permitted by network security policy"));
}
```

创建文件 res/xml/network_security_config.xml 内容如下

```
neo@MacBook-Pro ~/AndroidStudioProjects/okhttp % cat
app/src/main/res/xml/network_security_config.xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

再 app/src/main/AndroidManifest.xml 文件中增加 android:networkSecurityConfig="@xml/network_security_config"

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.netkiller.okhttp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"

android:networkSecurityConfig="@xml/network_security_config">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

    <uses-permission android:name="android.permission.INTERNET"
/>
</manifest>
```

2. Caused by: android.os.NetworkOnMainThreadException

主线程不能访问网络，在访问网络的代码前面添加如下代码即可：

```
StrictMode.ThreadPolicy policy= new
StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
```

或者写在 setContentView(R.layout.activity_main); 后面

另一种方式是在线程中执行

```
       new Thread(new Runnable() {
            @Override
            public void run() {

                try {
                    String json =
get("http://192.168.0.185:8080/member/json");
                } catch (IOException e) {
                    e.printStackTrace();
                }

            }
        }).start();
```

3. java.lang.IllegalStateException: Player is accessed on the wrong thread.

```
// 在 Service 中定义一个 Handler
private Handler mHandler = new
Handler(Looper.getMainLooper());

// 在 Service 中定义一个 Runnable 对象
private Runnable mRunnable = new Runnable() {
    @Override
    public void run() {
        // 在这里执行与 UI 相关的操作
    }
};

// 在 Service 中使用 Handler 将 Runnable 对象发送到 UI 线程的消息队
列中
mHandler.post(mRunnable);
```

4. Manifest merger failed with multiple errors, see logs

通过 gradle processDebugManifest --stacktrace 定位问题

```
neo@MacBook-Pro-M2 album % gradle processDebugManifest --
stacktrace

> Task :test:copyTask
copyTask

> Task :test:preBuild
preBuild

> Task :test:preDebugBuild
preDebugBuild

> Task :test:createDebugCompatibleScreenManifests
createDebugCompatibleScreenManifests

> Task :test:generateDebugResValues
generateDebugResValues

> Task :test:extractDeepLinksDebug
extractDeepLinksDebug

> Task :test:processDebugMainManifest
[androidx.vectordrawable:vectordrawable-animated:1.0.0]
/Users/neo/.gradle/caches/transforms-
3/4ea0058e0b06b5f932e694b630ab8fbf/transformed/vectordrawable-
animated-1.0.0/AndroidManifest.xml Warning:
        Namespace 'androidx.vectordrawable' used in:
androidx.vectordrawable:vectordrawable-animated:1.0.0,
androidx.vectordrawable:vectordrawable:1.0.0.
processDebugMainManifest

> Task :test:processDebugManifest
processDebugManifest
```

```
Deprecated Gradle features were used in this build, making it
incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual
deprecation warnings and determine if they come from your own
scripts or plugins.

For more on this, please refer to
https://docs.gradle.org/8.4/userguide/command_line_interface.ht
ml#sec:command_line_warnings in the Gradle documentation.

BUILD SUCCESSFUL in 8s
26 actionable tasks: 24 executed, 2 up-to-date
neo@MacBook-Pro-M2 album %
```

5. 从 Android API 30 废弃 setSystemUiVisibility(uiOptions)

替代方案



第 57 章 讯飞云

1. AIUI

```
// 写入文本
//                byte[] content= "你好".getBytes();
//                String params = "data_type=text";
//                AIUIMessage msg = new
AIUIMessage(AIUIConstant.CMD_WRITE, 0, 0, "tag=write_data_1",
content);
//                mAIUIAgent.sendMessage(msg);

                AIUIMessage aiuiMessage = new AIUIMessage(0,
0, 0, "", null);
                aiuiMessage.msgType = AIUIConstant.CMD_WRITE;
                aiuiMessage.arg1 = 0;
                aiuiMessage.arg2 = 0;
                // 在输入参数中设置tag，则对应结果中也将携带该tag，可
用于关联输入输出
                aiuiMessage.params =
"data_type=text,tag=text-tag";
                aiuiMessage.data = "天
气".getBytes(StandardCharsets.UTF_8);
                mAIUIAgent.sendMessage(aiuiMessage);
```

1.1. AIUIPlayer

```
package cn.netkiller.aiui;

import android.content.Context;
import android.util.Log;

import androidx.annotation.NonNull;
```

```
import com.iflytek.aiui.player.common.data.MetaItem;
import com.iflytek.aiui.player.core.AIUIPlayer;
import com.iflytek.aiui.player.core.PlayState;
import com.iflytek.aiui.player.core.PlayerListener;
import com.iflytek.aiui.player.players.KuwoMusicRemote;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import kotlin.Unit;
import kotlin.jvm.functions.Function0;
import kotlin.jvm.functions.Function2;

public class MusicSkillComponent {
    private static final String TAG =
MusicSkillComponent.class.getSimpleName();
    private static MusicSkillComponent musicSkillComponent =
null;
    private static String kuwoParams = null;
    private static int playListIndex = 0; //当前播放歌曲序号
    private static JSONArray mPlayList = null; //播放列表
    private static AIUIPlayer mAIUIPlayer = null;
    private static KuwoMusicRemote kuwoRemote = null;
    private final Context context;
    private final PlayerListener mPlayListener = new
PlayerListener() {
        @Override
        public void onMediaUrl(@NonNull JSONObject
jsonObject) {
            Log.d(TAG, "onMediaUrl: " + jsonObject);
        }

        @Override
        public void onPlayerReady() {
            Log.i(TAG, "onPlayerReady()");
        }

        @Override
        public void onStateChange(@NonNull PlayState state) {
            Log.i(TAG, "onStateChange: PlayState is" +
state);
            switch (state) {
                case READY:
```

```
                    Log.d(TAG, "AIUIPlayer 播放准备就绪");
                    break;
                case PLAYING:
                    Log.d(TAG, "播放");
                    break;
                case PAUSED:
                    Log.d(TAG, "停止");
                    break;
                case LOADING:
                    Log.d(TAG, "音乐加载中……");
                    break;
                case COMPLETE:
                    Log.d(TAG, "继续");
                    startPlayMusic();
                    break;
                case IDLE:
                    long currentPosition =
mAIUIPlayer.getCurrentPosition();
                    Log.d(TAG,
String.format("CurrentPosition: %l", currentPosition));
                    break;
                case ERROR:
                    Log.d(TAG, "播放出错");
                    break;
                default:
                    break;
            }
        }

        @Override
        public void onMediaChange(@NonNull MetaItem metaItem)
{
            String songName = metaItem.getTitle();      //歌名
            String author = metaItem.getAuthor();      //作者
            playListIndex = getPlayIndex(songName);
            Log.d(TAG, "onMediaChange: " + metaItem);
        }

        @Override
        public void onError(int code, @NonNull String info) {
            Log.e(TAG, "onError 播放出错: " + code + " , 错误信息
为: " + info);
            // 真实错误码需要从 info 中解析   "track link failed
code: 40006 description:"
            if (info.isEmpty()) {
```

```
                if (code == 200001) {
                    Log.d(TAG, "歌曲\"" + "" + "\"播放出错: " +
code + "\nINFO: 产品未通过酷我验收,仅支持获取奇数id资源\n");
                    if (!mAIUIPlayer.next()) {
                    }
                }
            }

        }

        @Override
        public void onPlayerRelease() {

        }
    };

    public MusicSkillComponent(Context context) {
        this.context = context;
        initSDK();
    }

    public synchronized static MusicSkillComponent
getInstance(Context context) {
        if (musicSkillComponent == null) {
            musicSkillComponent = new
MusicSkillComponent(context);
        }
        return musicSkillComponent;
    }

    public boolean previous() {
        if (mAIUIPlayer != null) {
            return mAIUIPlayer.previous();
        }
        return false;
    }

    public boolean next() {
        if (mAIUIPlayer != null) {
            return mAIUIPlayer.next();
        }
        return false;
    }

    public void pause() {
```

```
        if (mAIUIPlayer == null) {
            return;
        }
        if (mAIUIPlayer.getCurrentState() ==
PlayState.PLAYING) {
            mAIUIPlayer.pause();
        }
    }

    public void resume() {
        if (mAIUIPlayer == null) {
            return;
        }
        if (mAIUIPlayer.getCurrentState() ==
PlayState.PAUSED) {
            mAIUIPlayer.resume();
        }
    }

    public boolean play(@NonNull JSONObject object) {

//        Log.d(TAG, object.toString());

        try {

            JSONArray musics = object.getJSONArray("result");
            if (null != musics) {
                mPlayList = musics;
                playListIndex = 0;
            }
            return startPlayMusic();

        } catch (JSONException e) {
            e.printStackTrace();
        }

        return false;
    }

    public void initSDK() {
        //TODO 开发者需要实现生成sn的代码，参考:
https://www.yuque.com/iflyaiui/zzoolv/tgftb5
        //注意事项1：sn每台设备需要唯一！！！！WakeupEngine的sn和
AIUI的sn要一致
        //注意事项2：获取的值要保持稳定，否则会重复授权，浪费授权量
```

```
        String serialNumber = "test";
        String appId = "c84e1ddb";
        String appKey = "7a1583c3190b83fe4d62573ee9cfbfc1";

        //TODO 设置酷我音乐SDK设置相关参数，appid和appkey请使用自己的进行开发，并且与aiui.cfg一致
        kuwoParams = "appId=" + appId + ",appKey=" + appKey + "," + "serialNumber=" + serialNumber + ",deviceModel=" + serialNumber + ",userId=" + serialNumber;


        if (null == kuwoRemote) {
            kuwoRemote = new KuwoMusicRemote(kuwoParams);
            // 酷我SDK日志开关 ：true 打开，false 关闭
            kuwoRemote.setDebug(false);

            if (null != kuwoRemote) {
                Log.i(TAG, "KuwoRemote 初始化成功");
            }

        }

        if (null == mAIUIPlayer) {
            mAIUIPlayer = new AIUIPlayer(context, kuwoParams);
            // 播放前焦点占用设置
            mAIUIPlayer.setParameter("customAudioFocus", "true");
            // 回调信息设置
            mAIUIPlayer.addListener(mPlayListener);
            // AIUIPlayer SDK调试日志设置 ：true 打开，false 关闭
            mAIUIPlayer.setDebug(false);
            // 初始化播放器
            mAIUIPlayer.initialize();
            Log.i(TAG, "AIUIPlayer 初始化成功");

        }
    }

    public void release() {
        mPlayList = null;
        if (null != mAIUIPlayer) {
            mAIUIPlayer.release();
            mAIUIPlayer = null;
        }
```

```
        if (null != kuwoRemote) {
            kuwoRemote.destroy();
            kuwoRemote = null;
        }
    }

    private void activate() {
//        if (isActivated) {
////            showToast("当前设备已激活酷我音乐");
//            return;
//        }
        kuwoRemote.active(() -> {
            Log.i(TAG, "激活成功");
            return null;
        }, (errCode, errInfo) -> {
            Log.i(TAG, "激活失败，错误码为：" + errCode + " ,信息为：  " + errInfo);
            return null;
        });
    }

    private void login() {
        //酷我不强制用户登陆，开发者自己实现登陆代码，可参考下方酷狗音乐代码，改一下接口
//        Intent intent = new Intent(KuwoDemo.this, LoginActivity.class);
//        startActivityForResult(intent, 3);
    }

    private void logout() {
        kuwoRemote.logout(new Function0<Unit>() {
            @Override
            public Unit invoke() {
                Log.i(TAG, "酷我账号退出成功");
//                LoginBtn.setText("手机登陆");
                return null;
            }
        }, new Function2<Integer, String, Unit>() {
            @Override
            public Unit invoke(Integer errCode, String errInfo) {
                Log.i(TAG, "酷我账号退出失败，错误码为：" + errCode + " ,信息为：  " + errInfo);
                return null;
            }
```

```
        });
    }

    public void stop() {
        if (mAIUIPlayer != null) {
            mAIUIPlayer.stop();
        }
    }

    private boolean startPlayMusic() {
        if (null == mPlayList || mPlayList.length() == 0) {
            Log.w(TAG, "播放列表为空");
            return false;
        }
        Log.i(TAG, "mPlayList is: " + mPlayList.toString());
        mAIUIPlayer.reset();
        return mAIUIPlayer.play(mPlayList, "musicX", "",
false, playListIndex);
    }

    //构建虚假播放信息测试AIUIPlayer SDK播放是否可以正常调用
//    private void mockPlayMusic() {
//        try {
//            JSONArray musiclist = new JSONArray();
//            JSONObject music = new JSONObject();
//            music.put("source", "kuwo");
//            music.put("songname", "天地龙鳞");
//            music.put("itemid", "353833243");
//            musiclist.put(0, music);
//            mPlayList = musiclist;
//        } catch (JSONException e) {
//            e.printStackTrace();
//        }
//
//        if (null == mPlayList || mPlayList.length() == 0) {
////            showToast("播放列表为空");
//            return;
//        }
//        Log.i(TAG, "mPlayList is:\n" +
mPlayList.toString());
//        mAIUIPlayer.reset();
//        mAIUIPlayer.play(mPlayList, "musicX", "", false,
playListIndex);
//    }
```

```
    // 获取当前播放下标
    private int getPlayIndex(String songName) {
        if (mPlayList != null) {
            try {
                String playData = null;
                for (int i = 0; i < mPlayList.length(); i++)
{
                    playData =
mPlayList.getJSONObject(i).toString();
                    if (playData.contains(songName)) {
                        return i;
                    }
                }
            } catch (JSONException e) {
                e.printStackTrace();
                return 0;
            }
        }
        return 0;
    }
}
```

1.2. 酷我音乐

获取音乐URL

通过 itemId 获取 audio URL

```
    kuwoRemote.getAudioUrl("26383685", "128kmp3", new
Function1<AudioUrl, Unit>() {
        @Override
        public Unit invoke(AudioUrl audioUrl) {
            Log.d(TAG, audioUrl.toString());
            return null;
        }
    }, new Function2<Integer, String, Unit>() {
        @Override
        public Unit invoke(Integer code, String msg) {
```

```
            Log.d(TAG, "Code: " + code + ", Msg: " + msg);
            return null;
        }
    });
```

日志输出结果

```
AudioUrl(expiretime=, itemid=26383685, source=kuwo,
audiopath=http://other.player.ri01.sycdn.kuwo.cn/6dcbdb125c72df
ff3978fe29ab50fdd4/64eeef6f/resource/n2/27/48/2060696053.mp3)
```

1.3. 控制技能

```
package cn.netkiller.skill;

import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.util.Log;

import com.alibaba.fastjson.JSONObject;

import java.text.NumberFormat;
import java.text.ParseException;

public class ControlSkillComponent {
    private static final String TAG =
ControlSkillComponent.class.getName();
    private final Context context;
    private MusicSkillComponent musicSkillComponent = null;

    public ControlSkillComponent(Context context, JSONObject
object) {
        this.context = context;
```

```
        musicSkillComponent =
MusicSkillComponent.getInstance(context);
        Log.d(TAG, "控制技能: " + object);
        String semanticIntent = object.getString("intent");
        switch (semanticIntent) {
            case "PAUSE":
                musicSkillComponent.pause();
                break;
            case "CHOOSE_NEXT":
                musicSkillComponent.next();
                break;
            case "CHOOSE_PREVIOUS":
                musicSkillComponent.previous();
                break;
            case "VOLUME_MINUS":
                this.volume("VOLUME_MINUS");
                break;
            case "VOLUME_PLUS":
                this.volume("VOLUME_PLUS");
                break;
            case "VOLUME_MIN":
                this.volume("VOLUME_MIN");
                break;
            case "VOLUME_MAX":
                this.volume("VOLUME_MAX");
                break;
            case "MUTE":
                this.volume("MUTE");
                break;
            case "VOLUME_SET":

                JSONObject slots = (JSONObject)
object.getJSONArray("slots").get(0);
                String stringPercent =
slots.getString("value");
                try {
                    double doublePercent = (Double)
NumberFormat.getPercentInstance().parse(stringPercent);
//                    int percent = floatPercent.intValue();
                    this.volume(doublePercent);
                    Log.d(TAG,
String.valueOf(doublePercent));
                } catch (ParseException e) {
                    Log.e(TAG, e.toString());
                }
```

```
                break;
            case "SETTING_OPEN":
                Log.e(TAG, "设置无线网络");
                this.wifi();
                break;
            case "SHUTDOWN":
                break;
            case "RESET":
                break;
        }


    }

    public void wifi() {
        //        context.startActivity(new
Intent(Settings.ACTION_WIFI_SETTINGS));
        Intent intent = new Intent();
        intent.setAction("android.network.wlan");
        intent.putExtra("message", "");
        context.sendBroadcast(intent);
    }

    private void volume(String control) {
        AudioManager audioManager = (AudioManager)
context.getSystemService(Context.AUDIO_SERVICE);
        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
//        int minVolume =
audioManager.getStreamMinVolume(AudioManager.STREAM_MUSIC);
        int minVolume = 10;
        int stepVolume = 5;
        int currentMusicVolume =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
        int currentTTSVolume =
audioManager.getStreamVolume(AudioManager.STREAM_ALARM);


        switch (control) {
            case "VOLUME_MINUS": //步进减小
                currentMusicVolume -= stepVolume;
                if (currentMusicVolume < minVolume) {
                    currentMusicVolume = minVolume;
                }
                currentTTSVolume -= stepVolume;
```

```
                if (currentTTSVolume < minVolume) {
                    currentTTSVolume = minVolume;
                }
                break;
            case "VOLUME_PLUS": //步进累加
                currentMusicVolume += stepVolume;
                if (currentMusicVolume >= maxVolume) {
                    currentMusicVolume = maxVolume;
                }
                currentTTSVolume += stepVolume;
                if (currentTTSVolume > maxVolume) {
                    currentTTSVolume = maxVolume;
                }
                break;

            case "VOLUME_MAX": // 最大
                currentMusicVolume = currentTTSVolume =
maxVolume;

                break;
            case "VOLUME_MIN": //最小
                currentMusicVolume = currentTTSVolume =
minVolume;

                break;
            case "MUTE": //静音
                currentMusicVolume = currentTTSVolume =
minVolume;
                break;

        }

audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
currentMusicVolume, AudioManager.FLAG_SHOW_UI);

audioManager.setStreamVolume(AudioManager.STREAM_ALARM,
currentTTSVolume, AudioManager.FLAG_PLAY_SOUND);
        Log.d(TAG, String.format("volume:
currentMusicVolume=%s, currentTTSVolume=%s, maxVolume=%s",
currentMusicVolume, currentTTSVolume, maxVolume));
    }

    private void volume(double percent) {
        if (percent < 0.3) {
            return;
```

```
        }
        AudioManager audioManager = (AudioManager)
context.getSystemService(Context.AUDIO_SERVICE);
        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        int currentMusicVolume, currentTTSVolume;
        currentMusicVolume = currentTTSVolume = (int)
(maxVolume * percent);

audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
currentMusicVolume, AudioManager.FLAG_SHOW_UI);

audioManager.setStreamVolume(AudioManager.STREAM_ALARM,
currentTTSVolume, AudioManager.FLAG_PLAY_SOUND);
        Log.d(TAG, String.format("volume:
currentMusicVolume=%s, currentTTSVolume=%s, maxVolume=%s",
currentMusicVolume, currentTTSVolume, maxVolume));
    }
}
```

1.4. 唤醒词

手工唤醒

```
AiuiEngine.MSG_wakeup(EngineConstants.WAKEUPTYPE_VOICE);
```

1.5. 汉字转拼音

```
implementation 'com.belerweb:pinyin4j:2.5.1'
```

```
package cn.netkiller.ai.utils;

import android.util.Log;

import net.sourceforge.pinyin4j.PinyinHelper;
import net.sourceforge.pinyin4j.format.HanyuPinyinCaseType;
import
net.sourceforge.pinyin4j.format.HanyuPinyinOutputFormat;
import net.sourceforge.pinyin4j.format.HanyuPinyinToneType;
import net.sourceforge.pinyin4j.format.HanyuPinyinVCharType;
import
net.sourceforge.pinyin4j.format.exception.BadHanyuPinyinOutpu
tFormatCombination;

public class Pinyin {
    private static final String TAG = Pinyin.class.getName();

    public static String toPinyin(String hanzi) {
        char[] chars = hanzi.trim().toCharArray();
        String hanyupinyin = "";

        //输出格式设置
        HanyuPinyinOutputFormat defaultFormat = new
HanyuPinyinOutputFormat();
        /**
         * 输出大小写设置
         *
         * LOWERCASE:输出小写
         * UPPERCASE:输出大写
         */

defaultFormat.setCaseType(HanyuPinyinCaseType.LOWERCASE);

        /**
         * 输出音标设置
         *
         * WITH_TONE_MARK:直接用音标符（必须设置WITH_U_UNICODE，否
则会抛出异常）
         * WITH_TONE_NUMBER：1-4数字表示音标
         * WITHOUT_TONE：没有音标
         */
//
```

```
defaultFormat.setToneType(HanyuPinyinToneType.WITH_TONE_MARK)
; //  必须设置WITH_U_UNICODE，否则会抛出异常

defaultFormat.setToneType(HanyuPinyinToneType.WITHOUT_TONE);

        /**
         * 特殊音标ü设置
         *
         * WITH_V：用v表示ü
         * WITH_U_AND_COLON：用"u:"表示ü
         * WITH_U_UNICODE：直接用ü
         */
//
defaultFormat.setVCharType(HanyuPinyinVCharType.WITH_U_UNICOD
E);

defaultFormat.setVCharType(HanyuPinyinVCharType.WITH_V);

        // 中文的正则表达式
        String hanziRegex = "[\\u4E00-\\u9FA5]+";

        try {
            for (int i = 0; i < chars.length; i++) {
                // 判断为中文,则转换为汉语拼音
                if
(String.valueOf(chars[i]).matches(hanziRegex)) {
                    hanyupinyin += PinyinHelper

.toHanyuPinyinStringArray(chars[i], defaultFormat)[0];
                } else {
                    // 不为中文,则不转换
                    hanyupinyin += chars[i];
                }
            }
        } catch (BadHanyuPinyinOutputFormatCombination e) {
            Log.e(TAG, "字符不能转成汉语拼音");
        }

        return hanyupinyin;
    }

}
```

2. 讯飞 TTS

2.1. 设置日志输出级别

```
Setting.setLogLevel(Setting.LOG_LEVEL.low);
```

2.2. 流式语音合成

```
package com.iflytek.mscv5plusdemo;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Window;
import android.widget.Toast;

import com.iflytek.cloud.ErrorCode;
import com.iflytek.cloud.InitListener;
import com.iflytek.cloud.SpeechConstant;
import com.iflytek.cloud.SpeechError;
import com.iflytek.cloud.SpeechEvent;
import com.iflytek.cloud.SpeechSynthesizer;
import com.iflytek.cloud.SynthesizerListener;
import com.iflytek.speech.setting.TtsSettings;

public class TestActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_test);
        tts();
```

```
    }

    private InitListener initListener = new InitListener() {
        @Override
        public void onInit(int code) {
            Log.d("TAG", "InitListener init() code = " +
code);
            if (code != ErrorCode.SUCCESS) {
                showTip("初始化失败,错误码：" + code + ",请点击网
址https://www.xfyun.cn/document/error-code查询解决方案");

            } else {
                // 初始化成功，之后可以调用startSpeaking方法
                // 注：有的开发者在onCreate方法中创建完合成对象之后
马上就调用startSpeaking进行合成,
                // 正确的做法是将onCreate中的startSpeaking调用移至
这里
            }
        }
    };
    private SpeechSynthesizer speechSynthesizer;

    private void setParam() {
//        Log.d("",
String.valueOf(speechSynthesizer.isSpeaking()));
        // 清空参数
        speechSynthesizer.setParameter(SpeechConstant.PARAMS,
null);
        //设置合成
//        if (mEngineType.equals(SpeechConstant.TYPE_CLOUD))
{
        //设置使用云端引擎

speechSynthesizer.setParameter(SpeechConstant.ENGINE_TYPE,
SpeechConstant.TYPE_CLOUD);
        //设置发音人

speechSynthesizer.setParameter(SpeechConstant.VOICE_NAME,
"xiaoyan");

//        } else if
(mEngineType.equals(SpeechConstant.TYPE_LOCAL)) {
//            //设置使用本地引擎
//
speechSynthesizer.setParameter(SpeechConstant.ENGINE_TYPE,
```

```
SpeechConstant.TYPE_LOCAL);
//            //设置发音人资源路径
//
speechSynthesizer.setParameter(ResourceUtil.TTS_RES_PATH,
getResourcePath());
//            //设置发音人
//
speechSynthesizer.setParameter(SpeechConstant.VOICE_NAME,
voicerLocal);
//        } else {
//
speechSynthesizer.setParameter(SpeechConstant.ENGINE_TYPE,
SpeechConstant.TYPE_XTTS);
//            //设置发音人资源路径
//
speechSynthesizer.setParameter(ResourceUtil.TTS_RES_PATH,
getResourcePath());
//            //设置发音人
//
speechSynthesizer.setParameter(SpeechConstant.VOICE_NAME,
voicerXtts);
//        }

//mTts.setParameter(SpeechConstant.TTS_DATA_NOTIFY,"1");//支持
实时音频流抛出，仅在synthesizeToUri条件下支持
        //设置合成语速
//
speechSynthesizer.setParameter(SpeechConstant.SPEED,
mSharedPreferences.getString("speed_preference", "50"));
//        //设置合成音调
//
speechSynthesizer.setParameter(SpeechConstant.PITCH,
mSharedPreferences.getString("pitch_preference", "50"));
//        //设置合成音量
//
speechSynthesizer.setParameter(SpeechConstant.VOLUME,
mSharedPreferences.getString("volume_preference", "50"));
//        //设置播放器音频流类型
//
speechSynthesizer.setParameter(SpeechConstant.STREAM_TYPE,
mSharedPreferences.getString("stream_preference", "3"));
        //      mTts.setParameter(SpeechConstant.STREAM_TYPE,
AudioManager.STREAM_MUSIC+"");

        // 设置播放合成音频打断音乐播放，默认为true
```

```
speechSynthesizer.setParameter(SpeechConstant.KEY_REQUEST_FOC
US, "true");

        // 设置音频保存路径，保存音频格式支持pcm、wav，设置路径为sd卡
请注意WRITE_EXTERNAL_STORAGE权限

speechSynthesizer.setParameter(SpeechConstant.AUDIO_FORMAT,
"wav");

speechSynthesizer.setParameter(SpeechConstant.TTS_AUDIO_PATH,
                getExternalFilesDir("msc").getAbsolutePath()
+ "/tts.pcm");
    }

    public void tts() {

        speechSynthesizer =
SpeechSynthesizer.createSynthesizer(this, initListener);


        // 设置参数
//        setParam();
//        Log.d(TAG, "准备点击：  " +
System.currentTimeMillis());
        String text = "你好小虎";


        setParam();
        speechSynthesizer.stopSpeaking();
        int code = speechSynthesizer.startSpeaking(text,
synthesizerListener);
//                        /**
//                         * 只保存音频不进行播放接口,调用此接口请注释
startSpeaking接口
//                         * text:要合成的文本，uri:需要保存的音频全
路径，listener:回调接口
//                        */
//                        String path =
getExternalFilesDir("msc").getAbsolutePath() + "/tts.pcm";
//                        int code = mTts.synthesizeToUri(text,
path, mTtsListener);

        if (code != ErrorCode.SUCCESS) {
            showTip("语音合成失败,错误码: " + code + ",请点击网址
```

```
https://www.xfyun.cn/document/error-code查询解决方案");
        }

//
//
//        mTts.pauseSpeaking();
//
//        mTts.resumeSpeaking();

    }

    SynthesizerListener synthesizerListener = new
SynthesizerListener() {

        @Override
        public void onSpeakBegin() {
            showTip("开始播放");
//                Log.d(TtsDemo.TAG, "开始播放: " +
System.currentTimeMillis());
        }

        @Override
        public void onSpeakPaused() {
            showTip("暂停播放");
        }

        @Override
        public void onSpeakResumed() {
            showTip("继续播放");
        }

        @Override
        public void onBufferProgress(int percent, int
beginPos, int endPos,
                                     String info) {
            // 合成进度
//                mPercentForBuffering = percent;
//
showTip(String.format(getString(R.string.tts_toast_format),mP
ercentForBuffering, mPercentForPlaying));
        }

        @Override
        public void onSpeakProgress(int percent, int
beginPos, int endPos) {
```

```
            // 播放进度
//                mPercentForPlaying = percent;
//
showTip(String.format(getString(R.string.tts_toast_format),
//                        mPercentForBuffering,
mPercentForPlaying));
        }

        @Override
        public void onCompleted(SpeechError error) {
            if (error == null) {
                showTip("播放完成");
            } else {
                showTip(error.getPlainDescription(true));
            }
        }

        @Override
        public void onEvent(int eventType, int arg1, int
arg2, Bundle obj) {
            // 以下代码用于获取与云端的会话id，当业务出错时将会话id提
供给技术支持人员，可用于查询会话日志，定位出错原因
            // 若使用本地能力，会话id为null
            if (SpeechEvent.EVENT_SESSION_ID == eventType) {
                String sid =
obj.getString(SpeechEvent.KEY_EVENT_AUDIO_URL);
//                    Log.d(TAG, "session id =" + sid);
            }

            //实时音频流输出参考
                        /*if (SpeechEvent.EVENT_TTS_BUFFER ==
eventType) {
                                byte[] buf =
obj.getByteArray(SpeechEvent.KEY_EVENT_TTS_BUFFER);
                                Log.e("MscSpeechLog", "buf is
=" + buf);
                        }*/
        }
    };

    private void showTip(final String str) {
        Toast mToast;
//        if (mToast != null) {
//            mToast.cancel();
//        }
```

```
        mToast = Toast.makeText(getApplicationContext(), str,
Toast.LENGTH_SHORT);
        mToast.show();
    }


}
```

3. 语音唤醒

3.1. 范例

```
package cn.netkiller.album.religion.ai;

import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import com.iflytek.cloud.RequestListener;
import com.iflytek.cloud.SpeechConstant;
import com.iflytek.cloud.SpeechError;
import com.iflytek.cloud.SpeechEvent;
import com.iflytek.cloud.VoiceWakeuper;
import com.iflytek.cloud.WakeuperListener;
import com.iflytek.cloud.WakeuperResult;
import com.iflytek.cloud.util.ResourceUtil;

import java.nio.charset.StandardCharsets;

import cn.netkiller.album.religion.ContextHolder;
import cn.assets.album.religion.R;

public class Wakeup {
    private static final String TAG =
Wakeup.class.getSimpleName();
    private final int curThresh = 1450;
    private final String threshStr = "门限值：";
    private final String keep_alive = "1";
    private final String ivwNetMode = "0";
    private final Context context;
    // 查询资源请求回调监听
    private final RequestListener requestListener = new
RequestListener() {
        @Override
        public void onEvent(int eventType, Bundle params) {
            // 以下代码用于获取查询会话id，当业务出错时将会话id提供给
```

```
技术支持人员，可用于查询会话日志，定位出错原因
            //if(SpeechEvent.EVENT_SESSION_ID == eventType) {
            //  Log.d(TAG,
"sid:"+params.getString(SpeechEvent.KEY_EVENT_SESSION_ID));
            //}
        }

        @Override
        public void onCompleted(SpeechError error) {
            if (error != null) {
                Log.d(TAG, "error:" + error.getErrorCode());
                showTip(error.getPlainDescription(true));
            }
        }

        @Override
        public void onBufferReceived(byte[] buffer) {
            try {
                String resultInfo = new String(buffer,
StandardCharsets.UTF_8);
                Log.d(TAG, "resultInfo:" + resultInfo);

//                JSONTokener tokener = new
JSONTokener(resultInfo);
//                JSONObject object = new
JSONObject(tokener);
//
//                int ret = object.getInt("ret");
//                if (ret == 0) {
//                    String uri = object.getString("dlurl");
//                    String md5 = object.getString("md5");
//                    Log.d(TAG, "uri:" + uri);
//                    Log.d(TAG, "md5:" + md5);
//                    showTip("请求成功");
//                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
    private VoiceWakeuper voiceWakeuper;
    // 唤醒结果内容
    private String resultString;
    private final WakeuperListener wakeuperListener = new
WakeuperListener() {
```

```
        @Override
        public void onResult(WakeuperResult result) {
            Log.d(TAG, "onResult");

            try {
                String text = result.getResultString();
                Log.d(TAG, text);
//                JSONObject object;
//                object = new JSONObject(text);
//                StringBuffer buffer = new StringBuffer();
//                buffer.append("【RAW】 " + text);
//                buffer.append("\n");
//                buffer.append("【操作类型】" +
object.optString("sst"));
//                buffer.append("\n");
//                buffer.append("【唤醒词id】" +
object.optString("id"));
//                buffer.append("\n");
//                buffer.append("【得分】" +
object.optString("score"));
//                buffer.append("\n");
//                buffer.append("【前端点】" +
object.optString("bos"));
//                buffer.append("\n");
//                buffer.append("【尾端点】" +
object.optString("eos"));
//                resultString = buffer.toString();
                showTip(text);
            } catch (Exception e) {
                resultString = "结果解析出错";
                e.printStackTrace();
            }

        }

        @Override
        public void onError(SpeechError error) {
            showTip(error.getPlainDescription(true));
        }

        @Override
        public void onBeginOfSpeech() {
        }
```

```
        @Override
        public void onEvent(int eventType, int isLast, int
arg2, Bundle obj) {
            // EVENT_RECORD_DATA 事件仅在 NOTIFY_RECORD_DATA 参
数值为 真 时返回
            if (eventType == SpeechEvent.EVENT_RECORD_DATA) {
                final byte[] audio =
obj.getByteArray(SpeechEvent.KEY_EVENT_RECORD_DATA);
                Log.i(TAG, "ivw audio length: " +
audio.length);
            }
        }

        @Override
        public void onVolumeChanged(int volume) {

        }
    };
    // 设置门限值 ： 门限值越低越容易被唤醒


    public Wakeup() {
        this.context = ContextHolder.getContext();
//        String param = "appid=" +
context.getString(R.string.app_id) + "," +
SpeechConstant.ENGINE_MODE + "=" + SpeechConstant.MODE_MSC;
//        SpeechUtility.createUtility(context, param);

        voiceWakeuper = VoiceWakeuper.createWakeuper(context,
null);
        if (null == voiceWakeuper) {
            // 创建单例失败，与 21001 错误为同样原因，参考
http://bbs.xfyun.cn/forum.php?mod=viewthread&tid=9688
            Toast.makeText(context
                    , "创建对象失败，请确认 libmsc.so 放置正确，\n
且有调用 createUtility 进行初始化"
                    , Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(context, "准备唤醒",
Toast.LENGTH_LONG).show();
            // 清空参数
            voiceWakeuper.setParameter(SpeechConstant.PARAMS,
null);
            // 唤醒门限值，根据资源携带的唤醒词个数按照“id:门限;id:门
限”的格式传入
```

```
voiceWakeuper.setParameter(SpeechConstant.IVW_THRESHOLD,
"0:1450");
            // 设置唤醒模式

voiceWakeuper.setParameter(SpeechConstant.IVW_SST, "wakeup");
            // 设置持续进行唤醒

voiceWakeuper.setParameter(SpeechConstant.KEEP_ALIVE, "1");
            // 设置闭环优化网络模式

voiceWakeuper.setParameter(SpeechConstant.IVW_NET_MODE, "1");
            // 设置唤醒资源路径

voiceWakeuper.setParameter(SpeechConstant.IVW_RES_PATH,
getResource());
            // 设置唤醒录音保存路径，保存最近一分钟的音频

voiceWakeuper.setParameter(SpeechConstant.IVW_AUDIO_PATH,

context.getExternalFilesDir("msc").getAbsolutePath() +
"/ivw.wav");

voiceWakeuper.setParameter(SpeechConstant.AUDIO_FORMAT,
"wav");
            // 如有需要，设置 NOTIFY_RECORD_DATA 以实时通过
onEvent 返回录音音频流字节
            //voiceWakeuper.setParameter(
SpeechConstant.NOTIFY_RECORD_DATA, "1" );
            // 启动唤醒
            /*
voiceWakeuper.setParameter(SpeechConstant.AUDIO_SOURCE,
"-1");*/
        }
    }

    public void startListening(WakeuperListener
wakeuperListener) {
        //非空判断，防止因空指针使程序崩溃
        voiceWakeuper = VoiceWakeuper.getWakeuper();
        if (voiceWakeuper != null) {
            resultString = "";
            voiceWakeuper.startListening(wakeuperListener);
        } else {
            showTip("唤醒未初始化");
```

```
        }
    }

    private void stopListening() {
        voiceWakeuper = VoiceWakeuper.getWakeuper();
        if (voiceWakeuper != null) {
            voiceWakeuper.stopListening();
        }
    }

    public void cancel() {
        // 销毁合成对象
        voiceWakeuper = VoiceWakeuper.getWakeuper();
        if (voiceWakeuper != null) {
            voiceWakeuper.cancel();
        }
    }

    public void destroy() {
        // 销毁合成对象
        voiceWakeuper = VoiceWakeuper.getWakeuper();
        if (voiceWakeuper != null) {
            voiceWakeuper.destroy();
        }
    }

    private String getResource() {
        final String resPath =
ResourceUtil.generateResourcePath(context,
ResourceUtil.RESOURCE_TYPE.assets, "ivw/" +
context.getString(R.string.app_id) + ".jet");
        Log.d(TAG, "resPath: " + resPath);
        return resPath;
    }

    /**
     * 查询闭环优化唤醒资源
     * 请在闭环优化网络模式1或者模式2使用
     */
//    public void queryResource() {
//        int ret =
voiceWakeuper.queryResource(getResource(), requestListener);
//        showTip("updateResource ret:" + ret);
//    }
    private void showTip(final String str) {
```

```
//         runOnUiThread(new Runnable() {
//             @Override
//             public void run() {

         Toast.makeText(context.getApplicationContext(), str,
Toast.LENGTH_SHORT).show();

//             }
//         });
    }
}
```

附录 A. 附录

1. 一致性算法

paxos raft