spring

http://www.netkiller.cn

Netkiller 手札系列电子书

Netkiller Spring Cloud 手札

陈景峰 著

Spring Boot

Netkiller Spring Cloud 手札

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. 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
2. 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
            3.1.1. Linux systemd
            3.1.2. 传统 init.d 脚本
            3.1.3. 编译用于Tomcat的 War
        3.2. @SpringBootApplication
            3.2.1. 排除 @EnableAutoConfiguration 加载项

3.3. @EnableAutoConfiguration
3.4. @ComponentScan
3.5. @EntityScan 实体扫描
3.6. @EnableJpaRepositories
3.7. CharacterEncodingFilter
3.8. 隐藏 Banner
3.9. 实体与仓库扫描
3.10. 列出 Beans
3.11. Tomcat 端口
3.12. 配置项设定
3.13.
4. Properties 配置文件
4.1. application.properties 配置文件
4.1.1. application.properties 参考
4.1.2. 启动指定参数
4.1.2.1. --spring.config.location 指定配置文件
4.1.2.2. --spring.profiles.active 切换配置文件
4.1.3. 加载排除
4.1.4. PID FILE
4.1.5. server
4.1.5.1. 端口配置
4.1.5.2. Session 配置
4.1.5.3. cookie
4.1.5.4. error 路径
4.1.5.5. 压缩传输
4.1.5.6. ssl
4.1.6. logging
4.1.7. 内嵌 tomcat server
4.1.7.1.
4.1.7.2. server.tomcat.basedir
4.1.7.3. access.log
4.1.7.4. charset
4.1.8. servlet

4.1.8.1. 上传限制
4.1.8.2. server.servlet.context-path
4.1.9. JSON 输出与日期格式化
4.1.10. SMTP 相关配置
4.1.11. Redis
4.1.12. MongoDB
4.1.13. MySQL
4.1.14. Oracle
4.1.15. default_schema
4.1.16. datasource
4.1.17. velocity
4.1.18. Security 相关配置
4.1.19. MVC 配置
4.1.20. Kafka 相关配置
4.2. Properties 文件
4.2.1. @Value 注解
4.2.2. containsProperty 读取配置文件
4.2.3. @PropertySource 注解载入 properties 文件
4.2.4. @EnableConfigurationProperties 引用自定义 *.properties 配置文件
4.2.5. 手工载入 *.properties 文件
4.2.6. spring.profiles.active 参数切换配置文件
4.2.7. SpringApplicationBuilder.properties() 方法添加配置项
4.2.8. 参数引用
4.2.9. 产生随机数
4.2.10. List 列表类型
4.2.11. Map类型
4.2.12. Binder
5. Spring boot with Logging
5.1. 配置日志文件
5.1.1. 日志输出级别

        5.1.2. Spring boot 2.1 以后的版本不打印 Mapped 日志问题
        5.1.3. 禁止控制台输出日志
        5.1.4. 定制日志格式
    5.2. 打印日志
        5.2.1. lombok
    5.3. logback 配置详解
        5.3.1. 标准输出
        5.3.2. 分隔日志
        5.3.3. 按照文件尺寸分割日志
        5.3.4. 指定Class过滤日志
        5.3.5. 日志写入 MongoDB
        5.3.6. configuration 属性配置
        5.3.7. contextName 设置上下文名称
        5.3.8. property 设置变量
        5.3.9. encoder 日志格式设置
        5.3.10. RollingFileAppender
6. Spring boot with Jetty
7. Spring boot with HTTP2 SSL
    7.1. 生成自签名证书
    7.2. application.properties 配置文件
    7.3. 启动 Spring boot
    7.4. restTemplate 调用实例
    7.5. HTTP2
8. Spring boot with Webpage
    8.1. Maven
    8.2. application.properties
    8.3. Application
    8.4. IndexController
    8.5. src/main/webapp/WEB-INF/jsp/index.jsp
    8.6. 集成模板引擎
9. Spring boot with Velocity template
    9.1. Maven

    9.2. Resource
    9.3. Application
    9.4. RestController
    9.5. Test
10. Spring boot with Thymeleaf
    10.1. Maven
    10.2. application.properties
    10.3. Controller
    10.4. HTML5 Template
11. Spring boot with Session share
    11.1. Redis
        11.1.1. Maven
        11.1.2. application.properties
        11.1.3. Application
    11.2. 测试 Session
    11.3. JDBC
    11.4. Springboot 2.1
12. Spring boot with Caching
    12.1. maven
    12.2. 启用 Cache
    12.3. 测试 Controller
    12.4. @Cacheable 的用法
    12.5. @CachePut 用法
    12.6. 解决Expire 和 TTL 过期时间
    12.7. SpEL表达式
13. Spring boot with Email
    13.1. Maven
    13.2. Resource
    13.3. POJO
    13.4. RestController
    13.5. Test
14. Spring boot with Hessian
    14.1. Maven
    14.2. Application
    14.3. HessianServiceExporter
    14.4. Service

14.5. RestController
15. Spring boot with Git version
15.1. CommonRestController 公共控制器
15.2. VersionRestController 测试控制器
15.3. 创建 .gitattributes 文件
16. Spring boot with Data restful
16.1. Maven
17. Spring boot with ELK(Elasticsearch + Logstash + Kibana)
17.1. TCP 方案
17.2. Redis 方案
17.3. Kafka 方案
17.4. Other
18. Springboot with Ethereum (web3j)
18.1. Maven
18.2. application.properties
18.3. TestRestController
18.4. 测试
19. Spring boot with Async
19.1. 最简单的配置
19.2. 异步线程池
20. Spring boot with csv
21. Spring boot with Redis
21.1. Spring boot with Redis
21.1.1. maven
21.1.2. application.properties
21.1.3. JUnit
21.1.4. Controller
21.2. Redis Pub/Sub
21.2.1. Redis配置类
21.2.2. 订阅和发布类
21.2.3. 消息发布演示
22. Spring boot with MongoDB
22.1. Maven

    22.2. Application
    22.3. MongoTemplate
    22.4. Repository
23. Spring boot with MySQL
    23.1. Maven
    23.2. Resource
    23.3. Application
    23.4. JdbcTemplate
    23.5. CrudRepository
24. Spring boot with Oracle
    24.1. Maven
    24.2. application.properties
    24.3. Application
    24.4. CrudRepository
    24.5. JdbcTemplate
    24.6. Controller
25. Spring boot with PostgreSQL
    25.1. pom.xml
    25.2. application.properties
    25.3. Application
    25.4. CrudRepository
    25.5. JdbcTemplate
    25.6. Controller
    25.7. Test
26. Spring boot with Datasource
    26.1. Master / Slave 主从数据库数据源配置
        26.1.1. application.properties
        26.1.2. 配置主从数据源
        26.1.3. 选择数据源
    26.2. 多数据源配置
    26.3. JPA 多数据源
    26.4. Connection and Statement Pooling
        26.4.1. org.apache.tomcat.jdbc.pool.DataSource
        26.4.2. druid
        26.4.3. c3p0 - JDBC3 Connection and Statement Pooling

26.4.4. dbcp2
26.4.5. bonecp
26.4.6. dbcp2
27. Spring boot with Elasticsearch
27.1. Maven
27.2. Application
27.3. application.properties
27.4. Domain
27.5. ElasticsearchRepository
28. Spring boot with Elasticsearch TransportClient
28.1. Maven
28.2. Application
28.3. application.properties
28.4. ElasticsearchConfiguration
28.5. RestController
29. Spring boot with Apache Hive
29.1. Maven
29.2. application.properties
29.3. Configuration
29.4. CURD 操作实例
30. Spring boot with Phoenix
30.1. Maven
30.2. application.properties
30.3. Configuration
31. Spring boot with RabbitMQ(AMQP)
31.1. maven
31.2. RabbitMQConfig
31.3. 生产者
31.4. 消费者
32. Spring boot with Apache Kafka
32.1. 安装 kafka
32.2. maven
32.3. Spring boot Application
32.4. EnableKafka
32.5. KafkaListener
32.6. 测试

32.7. 完整的发布订阅实例
32.7.1. Consumer
32.7.2. Producer
32.7.3. Test
32.8. Spring cloud with Kafka
32.8.1. Application 主文件
32.8.2. 资源配置文件
32.8.2.1. application.properties
32.8.2.2. bootstrap.properties
32.8.2.3. Git 仓库
32.8.3. 启用 kafka
32.8.4. 消息发布主程序
33. Spring boot with Scheduling
33.1. Application.java
33.2. Component
33.3. 查看日志
33.4. 计划任务控制开关
33.5. @Scheduled 详解
33.5.1. 每3秒钟一运行一次
33.5.2. 凌晨23点运行
33.6. Timer 例子
33.7. ScheduledExecutorService 例子
34. Spring boot with Swagger
34.1. Maven 文件
34.2. SpringApplication
34.3. EnableSwagger2
34.4. RestController
34.5. @Api()
34.6. @ApiOperation()
34.7. @ApiResponses
34.8. @ApiModel 实体类
35. Spring boot with lombok
35.1. @Builder
35.2. @Slf4j 注解

36. Spring boot with Docker
    36.1. 通过 Docker 命令构建镜像
        36.1.1. 手工编译镜像
        36.1.2. Dockerfile 放在 src/main/docker/Dockerfile 下
        36.1.3. 通过参数指定 Springboot 文件
        36.1.4. SPRING_PROFILES_ACTIVE 指定配置文件
        36.1.5. 推送镜像到仓库
    36.2. 通过 Maven 构建 Docker 镜像
        36.2.1. Maven + Dockerfile 方案一
        36.2.2. Maven + Dockerfile 方案二
        36.2.3. Maven 不使用 Dockerfile 文件
        36.2.4. 推送镜像
    36.3. [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]
    36.4. curl: (35) LibreSSL SSL_connect: SSL_ERROR_SYSCALL in connection to localhost:8888
37. Spring boot with Docker stack
    37.1. 编译 Docker 镜像
    37.2.
38. Spring boot with Kubernetes
    38.1. Kubernetes 编排脚本
    38.2. 部署镜像
39. Spring boot with command line
    39.1. Maven
    39.2. CommandLineRunner 例子
    39.3. ApplicationRunner 例子

40. Spring Boot Actuator
    40.1. Maven 依赖
    40.2. 与 Spring Boot Actuator 有关的配置
        40.2.1. 禁用HTTP端点
        40.2.2. 安全配置
    40.3. actuator 接口
    40.4. 健康状态
        40.4.1. 健康状态
    40.5. 关机
    40.6. info 配置信息
    40.7. 计划任务
41. String boot with RestTemplate
    41.1. RestTemplate Example
        41.1.1. pom.xml
        41.1.2. web.xml
        41.1.3. springframework.xml
        41.1.4. RestController
        41.1.5. POJO
        41.1.6. 在控制器中完整实例
        41.1.7. 测试
    41.2. GET 操作
        41.2.1. 返回字符串
        41.2.2. 传递 GET 参数
    41.3. POST 操作
        41.3.1. postForObject
            41.3.1.1. 传递对象
            41.3.1.2. 传递数据结构 MultiValueMap
        41.3.2. postForEntity
    41.4. PUT 操作
    41.5. Delete 操作
    41.6. 上传文件
    41.7. HTTP Auth
        41.7.1. Client

41.8. PKCS12
41.9. Timeout 超时设置
41.9.1. JRE 启动参数设置超时时间
41.9.2. RestTemplate timeout with SimpleClientHttpRequestFactory
41.9.3. @Configuration 方式
42. SpringBootTest
42.1. Maven 依赖
42.2. 测试类
42.2.1. Junit基本注解介绍
42.3.
42.3.1. Assert.assertEquals 判断相等
42.3.2. Assert.assertTrue
42.4. JPA 测试
42.5.
42.6. Controller单元测试
42.7. WebTestClient
43. Spring boot with Aop
43.1. Aspect
43.1.1. Maven
43.1.2. Pojo 类
43.1.3. Service 类
43.1.4. Aspect 类
43.1.5. 控制器
43.1.6. Application
43.1.7. 测试
44. Spring boot with starter
44.1. 实现 starter
44.1.1. Maven pom.xml 依赖包
44.1.2. 配置文件处理
44.1.3. 自动配置文件
44.1.4. 启用 starter 的自定义注解
44.2. 引用 starter

44.2.1. Maven pom.xml 引入依赖
44.2.2. 通过注解配置 starter
44.2.3. 测试运行结果
45. Spring boot with Grafana
45.1. Springboot 集成 InfluxDB
45.2. InfluxDB
46. Spring Boot with Prometheus
46.1. Maven 依赖
46.2. application.properties 配置文件
46.3. 启动类
46.4. 测试
46.5. 控制器监控
46.6. 自定义埋点监控
46.6.1. 拦截器
46.6.2. 计数器元件
46.6.3. 配置类
46.6.4. 测试埋点效果
3. Spring MVC
1. @EnableWebMvc
1.1. CORS 跨域请求
1.2. Spring MVC CORS with WebMvcConfigurerAdapter
2. @Controller
2.1. @RequestMapping
2.1.1. @RequestMapping("/")
2.1.2. 映射多个URL
2.1.3. 匹配通配符
2.1.4. headers
2.1.5. @GetMapping
2.1.6. @PostMapping
2.1.7. RequestMapping with Request Parameters - @RequestParam
2.1.7.1. HTTP GET

2.1.7.2. HTTP POST
2.1.7.3. @RequestParam 传递特殊字符串
2.1.7.4. 传递日期参数
2.1.7.5. 上传文件
2.1.7.6. @RequestParam - POST 数组
2.2. @RequestBody
2.2.1. @RequestBody 传递 List
2.2.2. 传递 Map 数据
2.3. @RequestHeader - 获取 HTTP Header 信息
2.3.1. @RequestHeader 从 Http 头中获取变量
2.4. RequestMapping with Path Variables - @PathVariable
2.4.1. URL 参数传递
2.4.2. 默认值
2.4.3. URL 传递 Date 类型
2.4.4. 处理特殊字符
2.4.5. @PathVariable 注意事项
2.5. @ModelAttribute
2.6. @ResponseBody
2.6.1. 直接返回HTML
2.7. @ResponseStatus 设置 HTTP 状态
2.8. @CrossOrigin
2.8.1. maxAge
2.9. @CookieValue - 获取 Cookie 值
2.10. @SessionAttributes
2.11. ModelAndView
2.11.1. 变量传递
2.11.2. ModelMap 传递多个变量
2.11.3. redirect
2.11.4. ArrayList
2.11.5. HashMap
2.11.6. 传递对象
2.11.7.

        2.12. HttpServletRequest / HttpServletResponse
            2.12.1. HttpServletResponse
            2.12.2. HttpServletRequest
3. @RestController
    3.1. 返回实体
    3.2. JSON
    3.3. 处理原始 RAW JSON 数据
    3.4. 返回 JSON 对象 NULL 专为 "" 字符串
    3.5. XML
    3.6. 兼容传统 json 接口
    3.7. @PageableDefault 分页
    3.8. 上传文件
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
        4.3.1. css
            4.3.1.1. cssClass
            4.3.1.2. cssStyle
            4.3.1.3. cssErrorClass
        4.3.2. cssClass
    4.4. Thymeleaf
        4.4.1. Maven pom.xml
        4.4.2. Spring 配置
        4.4.3. controller
        4.4.4. HTML5 Template
        4.4.5. thymeleaf 渲染表格
        4.4.6. URL 链接
        4.4.7. 拆分字符串
        4.4.8. 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
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
7.1.1. 定义校验器
7.1.2. 获取 BindingResult 结果
7.1.3. 测试校验效果
7.2. 自定义注解
7.2.1. 定义校验器注解接口
7.2.2. 实现接口
7.2.3. 注解用法
7.2.4. 测试注解
8. Interceptor
8.1. WebMvcConfigurerAdapter
8.2. HandlerInterceptor
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"}

4. WebFlux framework
    1. Getting Started
        1.1. Maven
        1.2. Application
        1.3. RestController
        1.4. 测试
    2. WebFlux Router
        2.1. Component 原件
        2.2. 路由配置
        2.3. Thymeleaf
            2.3.1. 模板引擎 Thymeleaf 依赖
            2.3.2. application.properties 相关的配置
            2.3.3.
            2.3.4. Tymeleaf 视图
        2.4. Webflux Redis
            2.4.1. Maven Redis 依赖
            2.4.2. Redis 配置
            2.4.3. Config
            2.4.4. Service
            2.4.5.
        2.5. Webflux Mongdb
            2.5.1. Maven 依赖
            2.5.2. Repository
            2.5.3. Service
            2.5.4. 控制器
        2.6. SSE
        2.7. Mono/Flux
5. Spring Data
    1. Jackson
        1.1. @JsonIgnore 返回json是不含有该字段
        1.2. @JsonFormat 格式化 json 时间格式
            1.2.1. 日期格式化
            1.2.2. 时区
            1.2.3. 枚举

1.2.4. 枚举
1.3. @JsonComponent
2. Spring Data with Redis
2.1. 集成 Redis XML 方式
2.1.1. pom.xml
2.1.2. springframework-servlet.xml
2.1.3. Controller
2.1.4. index.jsp
2.1.5. 测试
2.2. RedisTemplate
2.2.1. stringRedisTemplate 基本用法
2.2.2. 设置缓存时间
2.2.3. 字符串截取
2.2.4. 追加字符串
2.2.5. 设置键的字符串值并返回其旧值
2.2.6. increment
2.2.7. 删除 key
2.2.8. 返回字符串长度
2.2.9. 如果key不存便缓存。
2.2.10. 缓存多个值 /获取多个值 multiSet / multiGet
2.2.11. List
2.2.11.1. rightPush
2.2.11.2. rightPushAll
2.2.11.3. rightPushIfPresent
2.2.11.4. leftPush
2.2.11.5. leftPushAll
2.2.11.6. range
2.2.12. SET 数据类型
2.2.12.1. 返回集合中的所有成员
2.2.12.2. 取出一个成员
2.2.12.3. 随机获取无序集合中的一个元素
2.2.12.4. 随机获取 n 个成员（存在重复数据）
2.2.12.5. 随机获取 n 个不重复成员

2.2.12.6. 在两个 SET 间移动数据
2.2.12.7. 成员删除
2.2.12.8. 返回集合数量
2.2.12.9. 判断元素是否在集合成员中
2.2.12.10. 对比两个集合求交集
2.2.12.11. 对比两个集合求交集，然后存储到新的 key 中
2.2.12.12. 合并两个集合，并去处重复数据
2.2.12.13. 合并两个集合去重复后保存到新的 key 中
2.2.12.14. 计算两个合集的差集
2.2.12.15. 计算两个合集的差集，然后保存到新的 key 中
2.2.12.16. 遍历 SET 集合
2.2.13. 有序的 set 集合
2.2.14. Hash
2.2.14.1. put
2.2.14.2. putAll
2.2.14.3. 从键中的哈希获取给定hashKey的值
2.2.14.4. delete
2.2.14.5. 确定哈希hashKey是否存在
2.2.14.6. 从哈希中获取指定的多个 hashKey 的值
2.2.14.7. 只有hashKey不存在时才能添加值
2.2.14.8. 获取整个Hash
2.2.14.9. 获取所有key
2.2.14.10. 通过 hashKey 获取所有值
2.2.14.11. 值加法操作
2.2.14.12. 遍历 Hash 表
2.2.15. 过期时间未执行
2.2.16. setBit / getBit 二进制位操作

2.2.17. 存储 Json 对象
2.2.17.1. 集成 RedisTemplate 定义新类 JsonRedisTemplate
2.2.17.2. 配置 Redis
2.2.17.3. 测试
2.3. Spring Data Redis - Repository Examples
2.3.1. @EnableRedisRepositories 启动 Redis 仓库
2.3.2. 定义 Domain 类
2.3.3. Repository 接口
2.3.4. 测试代码
3. Spring Data with MongoDB
3.1. Example Spring Data MongoDB
3.1.1. pom.xml
3.1.2. springframework-servlet.xml
3.1.3. POJO
3.1.4. Controller
3.1.5. 查看测试结果
3.1.6. 条件查询
3.2. @Document
3.2.1. 指定表名
3.2.2. @Id
3.2.3. @Version
3.2.4. @Field 定义字段名
3.2.5. @Indexed
3.2.5.1. 普通索引
3.2.5.2. 唯一索引
3.2.5.3. 索引排序方式
3.2.5.4. 稀疏索引
3.2.5.5. 索引过期时间设置
3.2.6. @CompoundIndex 复合索引
3.2.6.1. 普通复合索引
3.2.6.2. 唯一复合索引
3.2.7. @TextIndexed

        3.2.8. @GeoSpatialIndex 地理位置索引
        3.2.9. @Transient 丢弃数据，不存到 mongodb
        3.2.10. @DBRef 做外外键引用
            3.2.10.1. Article 类
            3.2.10.2. Hypermedia 类
            3.2.10.3. MongoRepository
            3.2.10.4. RestController
            3.2.10.5. 运行结果
        3.2.11. @DateTimeFormat
        3.2.12. @NumberFormat
        3.2.13. 在 @Document 中使用 Enum 类型
        3.2.14. 在 @Document 中定义数据结构 List/Map
        3.2.15. GeoJson 数据类型
    3.3. MongoRepository
        3.3.1. 扫描仓库接口
        3.3.2. findAll()
        3.3.3. deleteAll()
        3.3.4. save()
        3.3.5. count()
        3.3.6. exists() 判断是否存在
        3.3.7. existsById()
        3.3.8. findByXXXX
        3.3.9. findAll with OrderBy
            3.3.9.1. order by boolean 布尔型数据排序
        3.3.10. findAll with Sort
        3.3.11. FindAll with Pageable
            3.3.11.1. PageRequest - springboot 1.x 旧版本
        3.3.12. StartingWith 和 EndingWith
        3.3.13. Between
        3.3.14. Before / After
        3.3.15. @Query
    3.4. mongoTemplate
        3.4.1. Save 保存

3.4.2. Insert
3.4.3. updateFirst 修改符合条件第一条记录
3.4.4. updateMulti 修改符合条件的所有
3.4.5. 查找并保存
3.4.6. upsert - 修改符合条件时如果不存在则添加
3.4.7. 删除
3.4.8. 查找一条数据
3.4.9. 查找所有数据
3.4.10. Query
3.4.10.1. 翻页
3.4.10.2. between
3.4.11. Criteria
3.4.11.1. is
3.4.11.2. Regex 正则表达式搜索
3.4.11.3. lt 和 gt
3.4.11.4. exists()
3.4.11.5. 包含
3.4.12. Update
3.4.12.1. set
3.4.12.2. 追加数据
3.4.12.3. 更新数据
3.4.12.4. 删除数据
3.4.12.5. inc
3.4.12.6. update.addToSet
3.4.13. BasicUpdate
3.4.14. Sort
3.4.15. Query + PageRequest
3.4.16. newAggregation
3.4.17. 创建索引
3.4.18. 子对象操作
3.4.18.1. List 类型
3.5. GeoJson 反序列化
3.6. FAQ

3.6.1. location object expected, location array not in correct format; nested exception is com.mongodb.MongoWriteException: location object expected, location array not in correct format
4. Spring Data with MySQL
4.1. 选择数据库表引擎
4.2. 声明实体
4.2.1. @Entity 声明实体
4.2.2. @Table 定义表名
4.2.2.1. catalog
4.2.2.2. schema
4.2.2.3. uniqueConstraints
4.2.3. @Id 定义主键
4.2.4. @Column 定义字段
4.2.4.1. 字段长度
4.2.4.2. 浮点型
4.2.4.3. 创建于更新控制
4.2.4.4. TEXT 类型
4.2.4.5. 整形数据类型
4.2.5. @Lob 注解属性将被持久化为 Blog 或 Clob 类型
4.2.6. @NotNull 不能为空声明
4.2.7. @Temporal 日期定义
4.2.8. @DateTimeFormat 处理日期时间格式
4.2.9. 默认时间规则
4.2.9.1. CreatedDate
4.2.9.2. 与时间日期有关的 hibernate 注解
4.2.9.2.1. 设置默认时间
4.2.9.2.2. 创建时间
4.2.9.2.3. 更新时间
4.2.9.3. 数据库级别的默认创建日期时间定义

4.2.9.4. 数据库级别的默认创建日期与更新时间定义
4.2.9.5. 最后修改时间
4.2.10. Enum 枚举数据类型
4.2.10.1. 实体中处理 enum 类型
4.2.10.2. 数据库枚举类型
4.2.11. SET 数据结构
4.2.12. JSON 数据类型
4.2.13. 索引
4.2.13.1. 普通索引
4.2.13.2. 唯一索引
4.2.13.3. 复合索引
4.2.14. 创建复合主键
4.2.15. @JoinColumn
4.2.16. @OneToOne
4.2.17. OneToMany 一对多
4.2.18. ManyToMany 多对多
4.2.19. 外键级联删除
4.2.20. 其他
4.2.20.1. Cascade
4.2.20.2. @JsonIgnore
4.2.20.3. @EnableJpaAuditing 开启 JPA 审计功能
4.3. 实体继承
4.4. Repository
4.4.1. CrudRepository
4.4.2. JpaRepository
4.4.3. findByXXX
4.4.3.1. 传 Boolean 参数
4.4.3.2. Eunm 传递枚举参数
4.4.4. count 操作
4.4.5. OrderBy

            4.4.6. GreaterThan
            4.4.7. PageRequest 翻页操作
                4.4.7.1. PageRequest.of
                4.4.7.2. Pageable
            4.4.8. Sort 排序操作操作
            4.4.9. Query
                4.4.9.1. 参数传递
                4.4.9.2. 原生 SQL
                4.4.9.3. @Query 与 Pageagble
                4.4.9.4. 返回指定字段
                4.4.9.5. 返回指定的模型
            4.4.10. @Transactional
                4.4.10.1. 删除更新需要 @Transactional 注解
                4.4.10.2. 回滚操作
            4.4.11. 锁 @Lock
5. EntityManager
6. Spring Data with JdbcTemplate
    6.1. execute
    6.2. queryForInt
    6.3. queryForLong
    6.4. queryForObject
        6.4.1. 返回整形与字符型
        6.4.2. 查询 Double 类型数据库
        6.4.3. 返回日期
        6.4.4. 返回结果集
        6.4.5. 通过 "?" 向SQL传递参数
        6.4.6. RowMapper 记录映射
    6.5. queryForList
        6.5.1. Iterator 用法
        6.5.2. for 循环
        6.5.3. forEach 用法
    6.6. queryForMap
    6.7. query

6.7.1. ResultSet
6.7.2. ResultSetExtractor
6.7.3. RowMapper
6.8. queryForRowSet
6.9. update
6.10.
6.11. 实例参考
6.11.1. 参数传递技巧
7. Spring Data with Elasticsearch
7.1. 内嵌 Elasticsearch
7.1.1. Maven
7.1.2. src/main/resources/application.properties
7.1.3. Domain Class
7.1.4. ElasticsearchRepository
7.1.5. SearchRestController
7.1.6. 测试
7.2. 集群模式
7.3. Document
7.4. Elasticsearch 删除操作
7.5. FAQ
7.5.1. java.lang.IllegalStateException: Received message from unsupported version: [2.0.0] minimal compatible version is: [5.0.0]
8. Spring Data FAQ
8.1. No identifier specified for entity
8.2. Oracle Date 类型显示日期和时间
8.3. java.lang.ClassCastException: java.lang.Long cannot be cast to java.lang.Integer
8.4. Executing an update/delete query; nested exception is javax.persistence.TransactionRequiredException: Executing an update/delete query
6. 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
        2.7.1. Account
        2.7.2. AccountRepository
        2.7.3. 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
        4.6.1. 添加用户
        4.6.2. 添加多个用户，并指定角色
        4.6.3. 获取当前用户
5. Spring Authorization Server
    5.1. Oauth2 协议
        5.1.1. token
        5.1.2. grant_type

5.1.3. 授权码授权模式（Authorization Code Grant）
5.1.4. 密码模式（Resource Owner Password Credentials Grant）
5.1.5. 客户端凭证模式（Client Credentials Grant）
5.1.6. 刷新 TOKEN 方式
5.2. Maven 依赖
5.3. Spring cloud with Oauth2
5.3.1. authorization_code
5.3.1.1. 验证服务器器
5.3.1.1.1.
5.3.1.1.2.
5.3.1.1.3.
5.3.1.1.4.
5.3.1.1.5. 测试
5.3.2. Spring boot with Oauth2 - Password
5.3.2.1. Maven
5.3.2.2. Password tools
5.3.2.3. Server
5.3.2.3.1. Maven
5.3.2.3.2. application.properties
5.3.2.3.3. EnableAuthorizationServer
5.3.2.3.4. EnableResourceServer
5.3.2.3.5. Entity Table
5.3.2.3.6. UserRepository
5.3.2.3.7. UserService
5.3.2.3.8. TestRestController
5.3.2.3.9. 数据库初始化
5.3.2.3.10. Test
5.3.2.4. Spring boot with Oauth2 RestTemplate
5.3.2.4.1. Maven
5.3.2.4.2. OAuth2ClientConfiguration.java
5.3.2.4.3. Application.java
5.3.2.4.4. application.properties

5.3.2.4.5. Controller
5.3.2.4.6. Test
5.3.3. Spring boot with Oauth2 jwt
5.3.3.1. Maven
5.3.3.2. Authorization Server
5.3.3.3. Resource Server
5.3.3.4. Web Security
5.3.3.5. 插入数据
5.3.3.6. 使用 CURL 测试 JWT
5.3.3.7. 测试 Shell
5.3.3.8. refresh_token
5.3.4. Spring boot with Oauth2 jwt 非对称证书
5.3.4.1. 创建证书
5.3.4.2. Authorization Server
5.3.4.3. Resource Server
5.3.5. Apple iOS 访问 Oauth2
5.3.6. Oauth2 客户端
5.3.6.1.
5.3.6.2. application.yml
5.3.6.3. SpringApplication
5.3.6.4. WebSecurityConfigurer
5.3.6.5. TestController
5.3.7. Android Oauth2 + Jwt example
5.3.8. RestTemplate 使用 HttpClient
5.3.8.1. Maven
5.3.8.2. SpringBootApplication
5.3.8.3. ClientRestController
5.3.8.4. Test
5.3.9. 自签名证书信任问题
5.3.10. Principal
5.3.11. SecurityContextHolder 对象
5.3.12. 资源服务器配置
5.3.12.1. access()
5.3.12.1.1.
5.3.12.1.2.

5.3.13. Client
5.3.13.1. Overriding Spring Boot 2.0 Auto-configuration
5.3.14. Oauth2 常见问题
5.3.14.1. 修改 /oauth/token 路径
5.3.14.2. password 认证方式静态配置用户列表
7. Spring Cloud
1. Spring Cloud Config
1.1. Maven 项目 pom.xml 文件
1.2. Server
1.2.1. Maven config 模块
1.2.2. Application
1.2.3. application.properties
1.2.4. Git 仓库
1.2.5. 测试服务器
1.3. Client
1.3.1. Maven pom.xml
1.3.2. Application
1.3.3. bootstrap.properties
1.3.4. 测试 client
1.4. Config 高级配置
1.4.1. 仓库配置
1.4.1.1. 分支
1.4.1.2. basedir
1.4.1.3. HTTP Auth
1.4.1.4. 本地git仓库
1.4.1.5. native 本地配置
1.4.2. Config server 用户认证
1.4.2.1. Server 配置
1.4.2.1.1. application.properties
1.4.2.1.2. Maven
1.4.2.1.3. 测试是否生效

1.4.2.2. Client 配置
1.4.3. 加密敏感数据
1.4.4. Spring Cloud Config JDBC Backend
1.4.4.1. Maven pom.xml
1.4.4.2. 数据库表结构
1.4.4.3. Config 服务器
1.4.4.4. application.properties
1.5. Old
1.5.1. Server (Camden.SR5)
1.5.2. Client (Camden.SR5)
2. Spring Cloud Consol
2.1. Spring Cloud Consul 配置
2.2. Maven 父项目
2.3. Consul 服务生产者
2.3.1. Maven
2.3.2. application.properties
2.3.3. SpringApplication
2.3.4. TestController
2.4. Consul 服务消费者
2.4.1. Maven
2.4.2. application.properties
2.4.3. SpringApplication
2.4.4. TestController
2.5. Openfeign
2.5.1. Maven
2.5.2. application.properties
2.5.3. SpringApplication
2.5.4. Feign 接口
2.5.5. TestController
3. Spring Cloud Netflix
3.1. Eureka Server
3.1.1. Maven
3.1.2. Application
3.1.3. application.properties
3.1.4. 检查注册服务器

3.2. Eureka Client
    3.2.1. Maven
    3.2.2. Application
    3.2.3. RestController
    3.2.4. application.properties
    3.2.5. 测试
3.3. Feign client
    3.3.1. Maven
    3.3.2. Application
    3.3.3. interface
    3.3.4. application.properties
    3.3.5. 测试
    3.3.6. fallback
3.4. 为 Eureka Server 增加用户认证
    3.4.1. Maven
    3.4.2. application.properties
    3.4.3. Eureka Client
    3.4.4. Feign Client
3.5. Eureka 配置项
    3.5.1. /eureka/apps
    3.5.2. Eureka instance 配置项
    3.5.3. Eureka client 配置项
    3.5.4. Eureka Server配置项
3.6. ribbon
    3.6.1.
    3.6.2. LoadBalancerClient 实例
        3.6.2.1. application.properties
        3.6.2.2. LoadBalancerClient 获取服务器列表
    3.6.3. Ribbon 相关配置
        3.6.3.1. 内置负载均衡策略
3.7. 获取 EurekaClient 信息
3.8. Zuul
    3.8.1. Maven
    3.8.2. EnableZuulProxy

3.8.3. application.yml
3.8.4. 负载均衡配置
4. Openfeign
4.1. Openfeign 扫描包配置
4.2. 用户认证
4.3. 配置连接方式
4.4.
5. Spring Cloud Gateway
5.1. Gateway 例子
5.1.1. Maven
5.1.2. SpringApplication
5.1.3. application.yml
5.1.4. RouteLocator 方式
5.2. 路由配置
5.2.1. 转发操作
5.2.2. URL 参数
6. Spring Cloud Stream
7. Spring Cloud Bus
8. Spring Cloud Sleuth
9. Spring Cloud 相关的 application.properties 配置
9.1. 启用或禁用 bootstrap
9.2. bootstrap.properties 配置文件
10. Spring Cloud with Kubernetes
10.1. Config
10.1.1. Maven 依赖
10.1.2. Spring Cloud 配置文件
10.1.3. 程序文件
10.1.3.1. SpringBootApplication 启动文件
10.1.3.2. 配置类
10.1.3.3. 控制器
10.1.4. Kubernetes 编排脚本
10.1.5. 测试
10.2. 注册发现

10.2.1. Maven 父项目
10.2.2. provider
10.2.2.1. Maven 依赖
10.2.2.2. Springboot 启动类
10.2.2.3. 控制器
10.2.2.4. application.properties 配置文件
10.2.2.5. Kubernetes provider 编排脚本
10.2.3. consumer
10.2.3.1. Maven 依赖
10.2.3.2. Springboot 启动类
10.2.3.3. 控制器
10.2.3.4. FeignClient 接口
10.2.3.5. application.properties 配置文件
10.2.3.6. Kubernetes consumer 编排脚本
10.2.4. 测试
10.2.5.
11. FAQ
11.1. Cannot execute request on any known server
11.2. @EnableDiscoveryClient与@EnableEurekaClient区别
11.3. Feign请求超时
11.4. 已停止的微服务节点注销慢或不注销
11.5. Feign 启动出错 PathVariable annotation was empty on param 0.
11.6. Feign 提示 Consider defining a bean of type 'common.feign.Cms' in your configuration.
11.7. Load balancer does not have available server for client
11.8. Eureka Client (Dalston.SR1)
11.8.1. Maven
11.8.2. Application
11.8.3. RestController
11.8.4. application.properties

11.8.5. 测试
11.9. Config Server(1.3.1.RELEASE)
11.9.1. Server
11.9.1.1. Maven
11.9.1.2. Application
11.9.1.3. application.properties
11.9.1.4. Git 仓库
11.9.1.5. 测试服务器
11.9.2. Client
11.9.2.1. Maven pom.xml
11.9.2.2. Application
11.9.2.3. bootstrap.properties
11.9.2.4. 测试 client
8. 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
9. Miscellaneous
1. Object to Json
2. Json To Object
10. 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

A. 附录

1. Springboot example

范例清单

2.1. Spring boot with Velocity template (pom.xml)
2.2. Spring boot with Email (pom.xml)
2.3. RedisTemplate
2.4. Example Spring boot with Oracle

2.5. Spring boot with Apache kafka.
2.6. Spring boot with Apache kafka.
2.7. Test Spring Kafka
5.1. Spring Data Redis Example
5.2. Spring Data MongoDB - springframework-servlet.xml
7.1. Share feign interface.
8.1. MyBatis

Netkiller Spring Cloud 手札

Spring Cloud Cookbook

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>

电子书最近一次更新于 2020-11-15 03:24:53 .

http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net

微信订阅 号 netkiller-ebook (微信扫描二维码)

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

2020-11-15 03:24:53

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 系列电子书始于 2000 年，风风雨雨走过20年，将在 2020 年终结，之后不在更新。作出这种决定原因 很多，例如现在的阅 读习惯已 经转向短视频，我个人的时间，身体健康情况等等.....

感谢读者粉丝这20年的支持

虽 然电子书不再更新，后面我还会活跃在知乎社区和微信公众号

自述

《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
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。

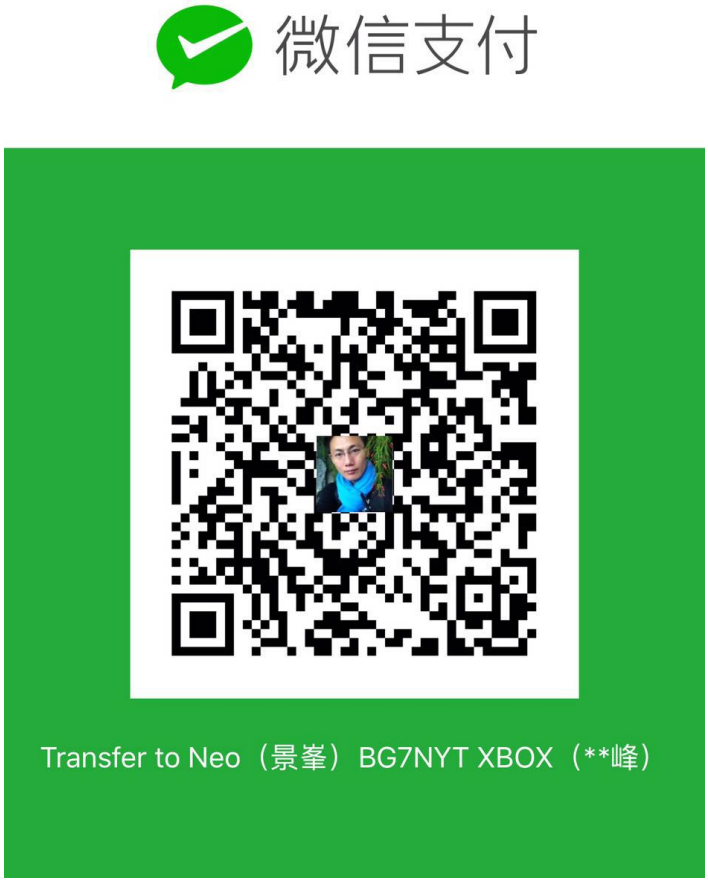
银行(Bank)

招商银行(China Merchants Bank)

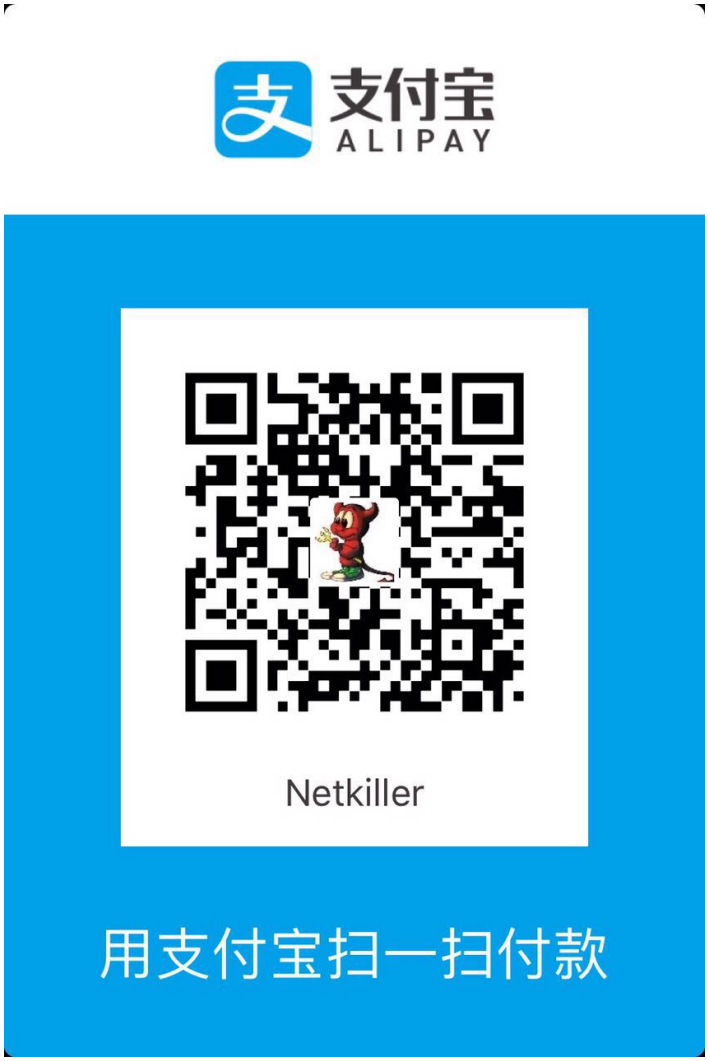
开户名：陈景峰

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 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...

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 2 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.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfig
uration;
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

3.1.1. 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
```

3.1.2. 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 $?
```

3.1.3. 编译用于Tomcat的 War

```
package demo;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.context.web.SpringBootServletInitialize
r;
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
```

3.2.1. 排除 @EnableAutoConfiguration 加载项

```
@SpringBootApplication(exclude =
DataSourceAutoConfiguration.class)
```

3.3. @EnableAutoConfiguration

exclude 排除配置，下面例子是排除 DataSource配置

```
@EnableAutoConfiguration(exclude=
{DataSourceAutoConfiguration.class})
```

3.4. @ComponentScan

@ComponentScan 注入会扫描 @Controller 与 @RestController

```
@ComponentScan
@ComponentScan({"cn.netkiller.controller"})
@ComponentScan({"cn.netkiller.controller", "cn.netkiller.rest"})
```

3.5. @EntityScan 实体扫描

```
@EntityScan("common.domain")
```

3.6. @EnableJpaRepositories

扫描 Jpa 仓库

```
@EnableJpaRepositories("common.domain")
```

3.7. CharacterEncodingFilter

```
        public @Bean Filter characterEncodingFilter() {
                CharacterEncodingFilter characterEncodingFilter
= new CharacterEncodingFilter();
                characterEncodingFilter.setEncoding("UTF-8");
                characterEncodingFilter.setForceEncoding(true);
                return characterEncodingFilter;
        }
```

3.8. 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);
}
```

3.9. 实体与仓库扫描

```
@EntityScan(basePackages = { "cn.netkiller.model" })
@EnableJpaRepositories(basePackages = {
"cn.netkiller.repository" })
```

3.10. 列出 Beans

```
package cn.netkiller;

import java.util.Arrays;
```

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposito
ries;
import
org.springframework.data.mongodb.repository.config.EnableMongoRe
positories;
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);
                }

        }
}
```

3.11. 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);
                                }

                        }
                }
        }
}
```

3.12. 配置项设定

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

```
    SpringApplication.run(Backend.class,
        "--spring.application.name=backend",
        "--server.port=9000"
    );
  }
```

3.13.

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();
```

4. Properties 配置文件

4.1. application.properties 配置文件

https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

4.1.1. 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/jdbcjobstore/tables_@@platfo
rm@@.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,application/javascr
ipt # 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/,classpath:/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/session/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.ExpiredTransactionStatusManagerScanner # 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/schema-
@@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/integration/jdbc/sc
hema-@@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_services,sun.java.comman
d # 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_services,sun.java.comman
d # 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/v1/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.wavefront.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.
```

4.1.2. 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。

4.1.2.1. --spring.config.location 指定配置文件

```
java -jar demo.jar --spring.config.location=/opt/config/application.properties
```

4.1.2.2. --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
```

4.1.3. 加载排除

```
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSou
rceAutoConfiguration
```

4.1.4. 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).
```

4.1.5. server

4.1.5.1. 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]}
```

4.1.5.2. 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
```

4.1.5.3. 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=/
```

4.1.5.4. error 路径

```
server.error.path=/error
```

4.1.5.5. 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
```

4.1.5.6. 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
```

4.1.6. logging

```
logging.path=/tmp # 日志目录默认为 /tmp
logging.file=your.log # 日志文件名称，默认为
spring.log
```

```
java -jar spring-boot-app.jar --
logging.file=/tmp/spring-boot-app.log
```

4.1.7. 内嵌 tomcat server

4.1.7.1.

连接数配置

```
server.tomcat.max-threads=2048 # 最大线程数
```

4.1.7.2. 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
```

4.1.7.3. 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
```

4.1.7.4. 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
```

4.1.8. servlet

4.1.8.1. 上传限制

```
spring.servlet.multipart.max-file-size=2MB
spring.servlet.multipart.max-request-size=10MB
spring.http.multipart.enabled=false
```

4.1.8.2. server.servlet.context-path

```
server.servlet.context-path=/your
```

4.1.9. 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
```

4.1.10. 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
```

4.1.11. 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
```

4.1.12. 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.cn/
                                spring.data.mongodb.repositories.enabled=true
```

4.1.13. 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
```

4.1.14. 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
```

4.1.15. default_schema

```
spring.jpa.properties.hibernate.default_schema=schema
```

4.1.16. 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
```

4.1.17. 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
```

4.1.18. 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

4.1.19. MVC 配置

是否支持favicon.ico，默认为: true

```
spring.mvc.favicon.enabled=false
```

4.1.20. 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.
```

4.2. Properties 文件

4.2.1. @Value 注解

application.properties

```
server.name=Linux
server.host=192.168.0.1,172.16.0.1
```

```
        @Value("${server.name}")
        private String name;
```

处理逗号分割得值

```
        @Value("#{'${server.host}'.split(',')}")
        private List<String> host;
```

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;
```

4.2.2. containsProperty 读取配置文件

```
this.environment.containsProperty("spring.jpa.database-platform")
```

4.2.3. @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;
        }

}
```

4.2.4. @EnableConfigurationProperties 引用自 定义 *.properties 配置文件

Application.java 涮锅配置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.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import
org.springframework.boot.context.properties.EnableConfigurationProperties;
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.EnableMongoRepositories;

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.ConfigurationProperties;
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})
```

4.2.5. 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();
                }
        }
```

4.2.6. 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
```

4.2.7. SpringApplicationBuilder.properties() 方法添加配置项

```
  public static void main(String[] args) {
    new
SpringApplicationBuilder(Application.class).properties("spring.config.name=clien
t").run(args);
  }
```

4.2.8. 参数引用

```
book.name=SpringCloud
book.author=netkiller
book.title=《${book.name}》作者 ${book.author}
```

4.2.9. 产生随机数

```
# 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]}
```

4.2.10. 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
```

4.2.11. 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
```

4.2.12. 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. Spring boot with Logging

5.1. 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)

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
```

5.1.1.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
```

5.1.2. Spring boot 2.1 以后的版本不打印 Mapped日 志问题

```
logging.level.org.springframework.web=trace
```

5.1.3. 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.path=/tmp
logging.file=/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

5.1.4. 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
```

5.2. 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。

5.2.1. lombok

```
            <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
```

```
@Slf4j
class DemoApplicationTests {
  @Test
  public void test(){
    log.debug("输出DEBUG日志.......");
  }
}
```

5.3. 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
```

5.3.1. 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>
```

5.3.2. 分隔日 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>
```

5.3.3. 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>
```

5.3.4. 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>
```

5.3.5.日 志写入 MongoDB

5.3.6. 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。

5.3.7. 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>

5.3.8. 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" />
```

5.3.9. 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

5.3.10. 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。

6. 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>
```

7. Spring boot with HTTP2 SSL

7.1. 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
```

7.2. 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
```

7.3. 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
```

7.4. 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>>() {});
```

7.5. 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%
```

8. Spring boot with Webpage

ViewResolver

8.1. 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>
```

8.2. application.properties

```
                        spring.mvc.view.prefix=/WEB-INF/jsp/
                        spring.mvc.view.suffix=.jsp
```

8.3. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
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);
        }
}
```

8.4. 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);
        }
}
```

8.5. 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>
```

8.6. 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.DefaultServle
tHandlerConfigurer;
import
org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;
import
org.springframework.web.servlet.view.InternalResourceViewResolv
er;

@Configuration
@EnableWebMvc
public class WebMvcConfig extends WebMvcConfigurerAdapter {

        @Override
        public void
configureDefaultServletHandling(DefaultServletHandlerConfigurer
configurer) {
                configurer.enable();
        }

        @Bean
        public InternalResourceViewResolver viewResolver() {
                InternalResourceViewResolver resolver = new
InternalResourceViewResolver();
                resolver.setPrefix("WEB-INF/jsp/");
                resolver.setSuffix(".jsp");
                return resolver;
        }

}
```

9. Spring boot with Velocity template

9.1. 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 2.1. 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>
```

9.2. 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>
```

9.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfiguration
Properties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
import
org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;

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);
        }

}
```

9.4. 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);
        }
}
```

9.5. 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
```

10. Spring boot with Thymeleaf

10.1. Maven

```
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
thymeleaf</artifactId>
        </dependency>
```

10.2. 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
```

10.3. 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";
        }

}
```

10.4. 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>
```

11. Spring boot with Session share

11.1. Redis

11.1.1. 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>
```

11.1.2. 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
```

11.1.3. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
import
org.springframework.scheduling.annotation.EnableScheduling;
import
org.springframework.session.data.redis.config.annotation.web.ht
tp.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.ht
tp.EnableRedisHttpSession;

@Configuration
@EnableRedisHttpSession
public class RedisHttpSessionConfig {

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

}
```

11.2. 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
```

11.3. JDBC

```
spring.session.store-type=jdbc
spring.session.jdbc.table-name=SESSIONS
```

11.4. 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
```

12. Spring boot with Caching

12.1. maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
cache</artifactId>
                </dependency>
```

12.2. 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);
    }

}
```

12.3. 测试 Controller

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);
}
```

12.4. @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) {

   returnnull;

}
```

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 []"
```

12.5. @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;

}
```

12.6. 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.RedisConnectionFactory
;
import
org.springframework.data.redis.serializer.Jackson2JsonRedisSeria
lizer;
import
org.springframework.data.redis.serializer.RedisSerializationCont
ext;

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(re
disConnectionFactory),
		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(RedisSerializationCo
ntext.SerializationPair.fromSerializer(jackson2JsonRedisSerializ
er)).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
```

12.7. 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.re
fresh:600}")
        public String expire() {
                return "Test";
        }
```

13. Spring boot with Email

13.1. Maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

例 2.2. 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>
```

13.2. 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
```

13.3. 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;
	}

}
```

13.4. 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);
        }
}
```

13.5. 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}
```

14. Spring boot with Hessian

14.1. Maven

```
<dependency>
        <groupId>com.caucho</groupId>
        <artifactId>hessian</artifactId>
        <version>4.0.38</version>
</dependency>
```

14.2. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
//import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
//import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
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);

        }
}
```

14.3. 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/HelloWorldService"
);

factory.setServiceInterface(HelloWorldService.class);
                return factory;
        }
}
```

14.4. 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;
        }
}
```

14.5. 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.");
        }
}
```

15. 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

15.1. 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.AuthenticationPrin
cipal;
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());
        }
}
```

15.2. 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.";
        }

}
```

15.3. 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 $
```

16. 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

16.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
rest</artifactId>
                </dependency>
```

17. 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。

17.1. 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.LogstashTcpSocketAppender"
>
        <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>
```

17.2. 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.xml
" />
        <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"
    }
}
```

17.3. Kafka 方案

17.4. Other

18. Springboot with Ethereum (web3j)

18.1. Maven

```
                <dependency>
                        <groupId>org.web3j</groupId>
                        <artifactId>web3j-spring-boot-
starter</artifactId>
                        <version>1.6.0</version>
                </dependency>
```

18.2. application.properties

```
web3j.client-
address=https://ropsten.infura.io/CsS9shwaAab0z7B4LP2d
web3j.admin-client=true
```

18.3. 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;
        }

}
```

18.4. 测试

```
neo@MacBook-Pro ~ % curl http://localhost:8080/version
Geth/v1.8.3-stable/linux-amd64/go1.10
```

19. Spring boot with Async

异步执行

19.1. 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
    public void doTaskThree() throws Exception {
        // 业务逻辑
    }

}
```

设置线程池，并且运行完成后推出

```
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.ThreadPoolTaskExecuto
r;

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;
    }


}
```

19.2. 异步线程池

```
package cn.netkiller.wallet.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableAsync;

@Configuration
@EnableAsync
public class ExecutorConfig {
        /** 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");
    }
```

```
}
```

20. 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();
        }
```

21. Spring boot with Redis

21.1. Spring boot with Redis

21.1.1. maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

21.1.2. 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
```

21.1.3. 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"));
    }
}
```

21.1.4. 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;
```

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.toExternalForm()
);
    }
}
```

例 2.3. 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;
        }
```

21.2. Redis Pub/Sub

21.2.1. 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.RedisConnectionFactory
;

import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.listener.ChannelTopic;
import
org.springframework.data.redis.listener.RedisMessageListenerCont
ainer;
import
org.springframework.data.redis.listener.adapter.MessageListenerA
```

```
dapter;

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;
        }

}
```

21.2.2. 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);
        }
}
```

21.2.3. 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;
        }
```

22. Spring boot with MongoDB

22.1. 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>
```

22.2. Application

Application.java

```
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfi
guration;
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.EnableMongoR
epositories;

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);
        }

}
```

22.3. 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;
        }

}
```

22.4. 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.EnableAutoConfiguration;
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.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;

@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@mdb
.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;
        }
```

```
}
```

23. Spring boot with MySQL

23.1. 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>
```

23.2. 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)
```

23.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfiguration
Properties;
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);
        }

}
```

23.4. 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));
                });
        }
}
```

23.5. 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";
        }
}
```

24. Spring boot with Oracle

24.1. 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 2.4. 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>
```

24.2. application.properties

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@//192.168.4.9:1521/orcl
```

```
.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))
                        )
                        )
```

24.3. Application

```
package api;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.context.properties.EnableConfiguration
Properties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
import
org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;

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);
        }

}
```

24.4. 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 + "]";
	}

}
```

24.5. 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));
		});
	}
```

24.6. 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));
                });

        }
}
```

25. Spring boot with PostgreSQL

25.1. 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>
```

25.2. 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
```

25.3. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
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);
        }
}
```

25.4. 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);
}
```

25.5. 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));
                });
        }
```

25.6. 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));
                });
        }
}
```

25.7. 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
```

26. Spring boot with Datasource

数据源配置

26.1. Master / Slave 主从数据库数据源配置

26.1.1. application.properties

```
spring.datasource.master.driverClassName = com.mysql.cj.jdbc.Driver
spring.datasource.master.url=jdbc:mysql://192.168.1.240:3306/test?
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/test?
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
```

26.1.2. 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.DataSourceProperties;
import
org.springframework.boot.context.properties.ConfigurationProperties;
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().build();
        }

        @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().build();
        }

        @Bean("slaveJdbcTemplate")
        public JdbcTemplate slaveJdbcTemplate(@Qualifier("Slave")
DataSource Master) {
                return new JdbcTemplate(Master);
        }

}
```

26.1.3. 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;
```

26.2. 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.ConfigurationProperties;
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.DataSourceTransactionManager;
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().build();
        }

        @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().build();
	}

	@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().build();
	}

	@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;
```

26.3. 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().createEntityManager();
    }

    @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).getObject());
    }

}
```

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().createEntityManager()
;
    }

    @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()
);
    }

}
```

26.4. Connection and Statement Pooling

注意：下面的实例仅限 Spring boot 2.0.2.RELEASE

26.4.1. 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
```

26.4.2. druid

pom.xml

```
                <dependency>
                        <groupId>com.alibaba</groupId>
                        <artifactId>druid</artifactId>
                        <version>1.0.24</version>
                </dependency>
```

application.properties

```
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
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=20
spring.datasource.filters=stat,wall,log4j
spring.datasource.connectionProperties=druid.stat.mergeSql=true;druid.st
at.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
spring.jpa.database=MYSQL
```

26.4.3. 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.ComboPooledDataSource
```

26.4.4. dbcp2

```
spring.datasource.type = org.apache.commons.dbcp2.BasicDataSource
```

26.4.5. bonecp

```
spring.datasource.type = com.jolbox.bonecp.BoneCPDataSource
```

26.4.6. dbcp2

```
spring.datasource.type = org.apache.commons.dbcp2.BasicDataSource
```

27. Spring boot with Elasticsearch

27.1. 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>
```

27.2. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
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);
	}
}
```

27.3. application.properties

```
spring.data.elasticsearch.repositories.enabled=true
spring.data.elasticsearch.cluster-nodes=127.0.0.1:9300
```

27.4. 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;
    }
}
```

27.5. 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);
```

```
}
```

28. 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。

28.1. 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/persistenc
e-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>
```

28.2. Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
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);
       }
}
```

28.3. 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_time
out=60s
```

28.4. 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;
        }
}
```

28.5. 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}
```

29. Spring boot with Apache Hive

29.1. 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>
```

29.2. 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
```

29.3. 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);
    }

}
```

29.4. 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";
        }
}
```

30. Spring boot with Phoenix

30.1. 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>
```

30.2. 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
```

30.3. 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.getProperty
("phoenix.default-auto-commit")));
        return dataSource;
    }

     @Bean(name = "phoenixJdbcTemplate")
     public JdbcTemplate
phoenixJdbcTemplate(@Qualifier("phoenixJdbcDataSource")
```

```
DataSource dataSource) {
          return new JdbcTemplate(dataSource);
     }
}
```

31. Spring boot with RabbitMQ(AMQP)

31.1. maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

31.2. 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);
    }

}
```

31.3. 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";
```

```
    }

}
```

31.4. 消费者

```
@Component
public class Consumer {

    @RabbitListener(queues = RabbitMQConfig.QUEUE_NAME)
    public void consumeMessage(String message) {
        System.out.println("consume message {}", message);
    }
}
```

32. Spring boot with Apache Kafka

Spring boot 1.5.1

32.1. 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
```

32.2. 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>
```

32.3. Spring boot Application

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
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);
```

```
        }
}
```

32.4. 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.ConcurrentKafkaListenerContain
erFactory;
import
org.springframework.kafka.config.KafkaListenerContainerFactory;
import org.springframework.kafka.core.ConsumerFactory;
import
org.springframework.kafka.core.DefaultKafkaConsumerFactory;
import
org.springframework.kafka.listener.ConcurrentMessageListenerCon
tainer;

import java.util.HashMap;
import java.util.Map;

@Configuration
@EnableKafka
public class KafkaConsumerConfig {

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

        @Bean

KafkaListenerContainerFactory<ConcurrentMessageListenerContaine
r<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();
```

```
        }

}
```

32.5. 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();
        }
}
```

32.6. 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

32.7. 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。

32.7.1. Consumer

例 2.5. Spring boot with Apache kafka.

SpringApplication

```
package cn.netkiller;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
//import
org.springframework.data.jpa.repository.config.EnableJpaReposit
ories;
//import
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
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.ConcurrentKafkaListenerContain
erFactory;
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;
    }
}
```

32.7.2. Producer

例 2.6. 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";
	}
```

```
}
```

32.7.3. Test

例 2.7. 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);
    }
}
```

32.8. Spring cloud with Kafka

32.8.1. 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);
        }
}
```

32.8.2. 资源配置文件

32.8.2.1. 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@172.1
6.0.10:8761/eureka/
# ==============================
```

32.8.2.2. 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
```

32.8.2.3. 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
```

32.8.3. 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 {
}
```

32.8.4. 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(cacheKey
));
                        } 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.siteId,
"delete");
                } else {
                        cmsTrashs =
cmsTrashRepository.findBySiteIdAndTypeAndCtimeGreaterThanOrderB
yCtime(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/post.
json", responseRestful, String.class);

                // logger.info(article.toString());
                if (response != null) {
                        logger.info(response);
                }
        }
}
```

33. 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。

33.1. 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.EnableAutoConfiguration;
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.EnableJpaReposito
ries;
import
org.springframework.data.mongodb.repository.config.EnableMongoRe
positories;
import
org.springframework.scheduling.annotation.EnableScheduling;
import
org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigur
er;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigur
erAdapter;

@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。

33.2. 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");
        }

}
```

33.3. 查看日 志

```
tail -f spring.log
```

33.4. 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.ConditionalOnPr
operty;
```

```
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
```

33.5. @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

33.5.1. 每3秒钟一运行一次

```
@Component
@EnableScheduling
public class MyTask {

    @Scheduled(cron="*/3 * * * * *")
    public void myTaskMethod(){
```

```
        //do something
    }
}
```

33.5.2. 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 );
        }
```

33.6. 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);
        }

}
```

33.7. 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);

        }

}
```

34. Spring boot with Swagger

34.1. 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>
```

34.2. 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);
        }
}
```

34.3. 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").apiInfo
(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").conta
ct(new Contact("Neo Chen", "http://www.netkiller.cn",
"netkiller@msn.com")).license("Mit
License").licenseUrl("").version("1.0").build();
        }

}
```

34.4. 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";
        }
}
```

34.5. @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 {

}
```

34.6. @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;
        }
```

```
}
```

34.7. @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;
        }
```

34.8. @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;
        }
}
```

35. 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

35.1. @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();
```

35.2. @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";
        }

}
```

36. Spring boot with Docker

36.1. 通过 Docker 命令构建镜像

36.1.1. 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
```

36.1.2. 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
```

36.1.3. 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
```

36.1.4. SPRING_PROFILES_ACTIVE 指定配置文件

```
% docker run -e "SPRING_PROFILES_ACTIVE=prod" -p 8080:8080
netkiller/docker
```

36.1.5. 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:
sha256:3078fea95c633f007be33b829efae0ff8e9d78ad463925af7d07752c95eb43
a3 size: 1165
```

36.2. 通过 Maven 构建 Docker 镜像

36.2.1. 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</dockerfile
>

<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:
sha256:38ec2c78a60ec4d5773c93534e433237be154ff5afa476965a68837b43ef2f
19
[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] ------------------------------------------------------------------
----------
```

36.2.2. 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>
```

36.2.3. 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] ------------------------------------------------------------------
----------
```

36.2.4. 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.versi
on}</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>
```

36.3. [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>
```

36.4. 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/libressl/libr
essl-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=localh
ost
```

```
i:/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkiller/CN=localh
ost
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDijCCAnKgAwIBAgIJAP/SjXit0rVsMA0GCSqGSIb3DQEBCwUAMHMxCzAJBgNV
BAYTAkNOMREwDwYDVQQIEwhTaGVuemhlbjESMBAGA1UEBxMJR3Vhbmdkb25nMRUw
EwYDVQQKEwxuZXRraWxsZXIuY24xEjAQBgNVBAsTCW5ldGtpbGxlcjESMBAGA1UE
AxMJbG9jYWxob3N0MB4XDTIwMDkwNzA4NTUzOVoXDTIwMTIwNjA4NTUzOVowczEL
MAkGA1UEBhMCQ04xETAPBgNVBAgTCFNoZW56aGVuMRIwEAYDVQQHEwlHdWFuZ2Rv
bmcxFTATBgNVBAoTDG5ldGtpbGxlci5jbjESMBAGA1UECxMJbmV0a2lsbGVyMRIw
EAYDVQQDEwlsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCTZUtf/siYQr3MBstphQsBceRxvflDm2C4ztZ8OemDzH2avhI7edD6rzrJQ0V2
1n1XlTRgwoYqoTgeIdQ1DbzgrCliBYy+3E9vcp8WWzYz9o2YZRphYUr37iWonP+b
ZkLqzmRLuASRNZ8sBrwD7Mvs5IXfJZQ8wruO0V4oJQ5NOzcxDmbA0WGJn/0QZDKN
/tR7Rw3g9B96fFYGI/T7g4nuteEiUqQ9GJ1gx3utBd31Z1m8cV59ZsWd+Y2P14LO
W+YxkpB56OZKWWr1ExxQdZmLIME+D0d40M8At6rCAvclMKa7dva6+ZRxPlizVkQS
L4JNT1WOMtVaUUhFX5xlhsBtAgMBAAGjITAfMB0GA1UdDgQWBBROMJrswZ37wsxV
sm0N9AHOE8ZiODANBgkqhkiG9w0BAQsFAAOCAQEAPIgc6ZcQueQTEym36gx2IRWT
wLVQEabyS4/xeu89aRfbGDOavBajNwStqGdWUE8PRb95bhfvziZ61c6gBO9IE23j
GOmIQTW5RvZL6HLJgqR3LngZUiV/Ugwuno5Uo8IN25duq993tNmdCG8YeBtfuy/j
OFRrn96OT/Trj04NfYmC7nqBThyNmLPY5Oeo0XkhIAqqcLJE8/SJ9zd16vmgVhPM
UlsFJcZoL1uhbNXQuLPv8id8tntH+Lli39RVwd56CgTW7k9YFfFNV0mCeWBsAYl3
74R8l4ClV15o31wH/qPLg0F6uE/M/xsz56WIu2e5Oa30issz0DjYrG9GiQ2kDA==
-----END CERTIFICATE-----
subject=/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkiller/CN=
localhost
issuer=/C=CN/ST=Shenzhen/L=Guangdong/O=netkiller.cn/OU=netkiller/CN=l
ocalhost
---
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:
856BA1E0CFE8AC65AEC838C0A4DA0503C7A05F0BA803B127D3B1EBBB8FF1A344
    Session-ID-ctx:
    Master-Key:
2DCA2747330C8008958B1A4F3EF340044FE69455EA730DA0E30DF97A13E6EB7BCABDF
DF5CA0FA5B278701EA25D694CAB
    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","password
":"123456"}}}
```

37. Spring boot with Docker stack

37.1. 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] ------------------------------------------------------------
----------------
```

37.2.

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
```

38. Spring boot with Kubernetes

首先你需要构建 docker 镜像，并且 push 到 registry 参考这里

38.1. 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
```

38.2. 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
```

39. Spring boot with command line

39.1. 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>
```

39.2. 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
```

39.3. 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);
```

```
            }
        }
```

40. Spring Boot Actuator

健康检查、审计、统计和监控

40.1. Maven 依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

40.2. 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
```

40.2.1. 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
```

40.2.2. 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
```

40.3. actuator 接口

```
neo@MacBook-Pro ~ % curl -s http://localhost:8080/actuator | jq
{
  "_links": {
    "self": {
      "href": "http://localhost:8080/actuator",
      "templated": false
    },
    "health": {
      "href": "http://localhost:8080/actuator/health",
      "templated": false
    },
    "health-component": {
      "href":
"http://localhost:8080/actuator/health/{component}",
      "templated": true
```

```
    },
    "health-component-instance": {
      "href":
"http://localhost:8080/actuator/health/{component}/{instance}",
      "templated": true
    },
    "info": {
      "href": "http://localhost:8080/actuator/info",
      "templated": false
    }
  }
}
```

40.4. 健康状态

curl localhost:8080/actuator/health

```
neo@MacBook-Pro ~ % curl -s
http://localhost:8080/actuator/health | jq
{
  "status": "UP"
}
```

40.4.1. 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://localhost:8080/actuator/health | jq
```

```
{
  "status": "UP",
  "details": {
    "diskSpace": {
      "status": "UP",
      "details": {
        "total": 250790436864,
        "free": 23556112384,
        "threshold": 10485760
      }
    }
  }
}
```

40.5. 关机

配置文件中加入

```
management.endpoint.shutdown.enabled=true
```

```
curl -X POST localhost:8080/actuator/shutdown
```

40.6. 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://localhost:8080/actuator/info
| jq
{
  "app": {
    "name": "spring-boot-example",
    "version": "v1.0.0"
  }
}
```

40.7. 计划任务

```
http://localhost:8080/actuator/scheduledtasks
```

41. 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

41.1. RestTemplate Example

41.1.1. 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>
```

41.1.2. 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>
```

41.1.3. 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.UrlBasedViewResolver">
                <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.PropertyPlaceholderConfi
gurer">
                <property name="location"
value="classpath:resources/development.properties" />
        </bean>

        <!-- Redis Connection Factory -->
        <bean id="jedisConnFactory"
class="org.springframework.data.redis.connection.jedis.JedisConnectionFa
ctory" 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>
```

41.1.4. 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;
        }
}
```

41.1.5. 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 +
"]";
        }

}
```

41.1.6. 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/html/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);
        }
}
```

41.1.7. 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","referrer":"h
ttp://www.netkiller.cn","href":"http://www.netkiller.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.netkiller.cn
","referrer":"http://www.netkiller.cn","href":"http://www.netkiller.cn"}
```

41.2. GET 操作

41.2.1. 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/html/2/2/42564
.html", String.class);
                return text;
        }
```

41.2.2. 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);
        }
```

41.3. POST 操作

41.3.1. postForObject

41.3.1.1. 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;
        }
```

41.3.1.2. 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;
        }
```

41.3.2. 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();
        }
```

41.4. 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);
        }
```

41.5. 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);
        }
```

41.6. 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);
    }
}
```

41.7. HTTP Auth

41.7.1. 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);
```

41.8. 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.HttpComponentsClientHttpRequestFactory;
import org.springframework.security.oauth2.client.OAuth2RestOperations;
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(60000);

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();

        }

}
```

41.9. Timeout 超时设置

41.9.1. JRE 启动参数设置超时时间

```
-Dsun.net.client.defaultConnectTimeout=<TimeoutInMiliSec>
-Dsun.net.client.defaultReadTimeout=<TimeoutInMiliSec>
```

41.9.2. 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;
}
```

41.9.3. @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.SimpleClientHttpRequestFactory;
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);
        }
}
```

42. SpringBootTest

42.1. Maven 依赖

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
</dependency>
```

42.2. 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);

        }
}
```

42.2.1. 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
```

42.3.

42.3.1. Assert.assertEquals 判断相等

42.3.2. Assert.assertTrue

42.4. 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());

    }
}
```

42.5.

```
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.WebEnvironment;
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"));
    }

}
```

42.6. 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")).andExpect(MockMvcResultMatche
rs.model().attributeExists("students"));

    }


     @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")).andReturn();

    }

}
```

42.7. 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().isOk().expectBody(String.cla
ss).isEqualTo("Hello world!");
        }

        @Test
        public void client() {

        }
}
```

43. Spring boot with Aop

43.1. Aspect

43.1.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
aop</artifactId>
                </dependency>
```

43.1.2. 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
        }

}
```

43.1.3. 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) {

	}
}
```

43.1.4. 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);
        }
}
```

43.1.5. 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";
	}

}
```

43.1.6. 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);
        }
}
```

43.1.7. 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
```

44. 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。

44.1. 实现 starter

44.1.1. 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>
```

44.1.2. 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.ConfigurationProper
ties;

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 + "]";
	}

}
```

44.1.3.自 动配置文件

```
package cn.netkiller.autoconfigure;

import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.context.properties.EnableConfiguration
Properties;
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);
        }
}
```

44.1.4. 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 {

}
```

44.2. 引用 starter

44.2.1. 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>
```

44.2.2. 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! ");
        }
}
```

44.2.3. 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!
```

45. Spring boot with Grafana

45.1. 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
```

45.2. 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
```

46. Spring Boot with Prometheus

46.1. 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>
```

46.2. application.properties 配置文件

开启 metrics

```
spring.application.name=springboot-with-prometheus
management.endpoints.web.exposure.include=*
management.metrics.tags.application=${spring.application.name}
```

46.3. 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.MeterReg
istryCustomizer;
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);
        }
```

```
}
```

46.4. 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
```

46.5. 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();
  }
}
```

46.6.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。

46.6.1. 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();
        }

}
```

46.6.2. 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();
        }
}
```

46.6.3. 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.InterceptorRe
gistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rer;

@Configuration
public class IntercepterConfiguration implements
WebMvcConfigurer {

        @Bean
        public PrometheusInterceptor prometheusInterceptor() {

                return new PrometheusInterceptor();
        }

        @Override
        public void addInterceptors(InterceptorRegistry
registry) {

registry.addInterceptor(prometheusInterceptor()).addPathPattern
s("/**");

        }
}
```

46.6.4. 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
```

第 3 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.DefaultServle
tHandlerConfigurer;
import
org.springframework.web.servlet.config.annotation.EnableWebMvc;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;
import
org.springframework.web.servlet.view.InternalResourceViewResolv
er;

@Configuration
@EnableWebMvc
public class WebMvcConfig extends WebMvcConfigurerAdapter {

        @Override
        public void
configureDefaultServletHandling(DefaultServletHandlerConfigurer
```

```
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)
```

2.1.1. @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 })
```

2.1.2. 映射多个URL

```
@RequestMapping({ "/news/zh-cn", "/news/zh-tw" })
@ResponseBody
public String getNewsByPath() {
        return "Hello";
}
```

2.1.3. 匹配通配符

```
@Controller
@RequestMapping("/test/*")

public class TestController {

        @RequestMapping
        public String default() {
                return "OK";
        }
}
```

2.1.4. 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
```

2.1.5. @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)
```

2.1.6. @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)
```

2.1.7. 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;
```

2.1.7.1. 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
```

2.1.7.2. 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>
```

2.1.7.3. @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(" ", "+");

        }
```

2.1.7.4. 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");
        }
```

2.1.7.5. 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
```

2.1.7.6. @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);
        }
```

2.2. @RequestBody

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);
}
```

2.2.1. @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"]
```

2.2.2. 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
```

2.3. @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;
        }
```

2.3.1. @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;
        }
```

2.4. 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。

2.4.1. 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>
```

2.4.2. 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){
...
}
```

2.4.3. 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;
```

```
        }
```

2.4.4. 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=RequestMethod.GET)
public void download(HttpSession
        session,@PathVariable("version")String version){
        return version;
}
```

2.4.5. @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;
        }
```

2.5. @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());
        }
```

2.6. @ResponseBody

```
import org.springframework.web.bind.annotation.ResponseBody;
```

2.6.1. 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;
        }

}
```

2.7. @ResponseStatus 设置 HTTP 状态

```
        @RequestMapping(value = "/", method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.CREATED)
    public String create(@RequestBody MultiValueMap<String, String> map) {
        return "OK";
    }
```

2.8. @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";
    }
}
```

2.8.1. 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;
        }
}
```

2.9. @CookieValue - 获取 Cookie 值

```
@RequestMapping("/sessionInfo")
public void sessionInfo(@CookieValue("JSESSIONID") String cookie)  {

  //...

}
```

2.10. @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 {
  ...
}
```

2.11. ModelAndView

2.11.1. 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;
}
```

2.11.2. 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>
```

2.11.3. redirect

```
        @RequestMapping("/testRedirect")
        public ModelAndView testRedirect(){
                    RedirectView view = new RedirectView("testMapString.html");
                    return new ModelAndView(view);
        }
```

2.11.4. 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>
```

2.11.5. 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>
```

2.11.6. 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>
```

2.11.7.

2.12. HttpServletRequest / HttpServletResponse

2.12.1. 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");
    }
}
```

2.12.2. 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. 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;
        }
```

3.2. 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);
    }

}
```

3.3. 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);
        }
```

3.4. 返回 JSON 对象 NULL 专为 "" 字符串

```
package api.config;

import java.io.IOException;

import
org.springframework.boot.autoconfigure.condition.ConditionalOnM
issingBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import
org.springframework.http.converter.json.Jackson2ObjectMapperBui
lder;

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(new
JsonSerializer<Object>() {
            @Override
            public void serialize(Object o, JsonGenerator
jsonGenerator, SerializerProvider serializerProvider) throws
IOException, JsonProcessingException {
                jsonGenerator.writeString("");
            }
        });
        return objectMapper;
    }
}
```

3.5. 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;
        }
}
```

3.6. 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);
        }
```

3.7. @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。

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.RedirectAttributes;

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;
        }
```

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.ResourceHandlerRegistr
y;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MyWebMvcConfigurer implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/images/**").addResourceLocations("classpat
h:/images/");
    }
}
```

4.3. Using Spring's form tag library

4.3.1. css

4.3.1.1. 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"/>
```

4.3.1.2. 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"/>
```

4.3.1.3. 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"/>
```

4.3.2. cssClass

4.4. Thymeleaf

http://thymeleaf.org/

4.4.1. Maven pom.xml

```
                    <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-
thymeleaf</artifactId>
            </dependency>
```

4.4.2. Spring 配置

```
        <!--
************************************************************** -->
        <!-- THYMELEAF-SPECIFIC ARTIFACTS -->
        <!-- TemplateResolver <- TemplateEngine <- ViewResolver -->
        <!--
************************************************************** -->

        <bean id="templateResolver"

class="org.thymeleaf.templateresolver.ServletContextTemplateResolver">
                <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>
```

4.4.3. 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 {

        @RequestMapping(value = "/{name}", method = RequestMethod.GET)
        public String getMovie(@PathVariable String name, ModelMap
model) {
                model.addAttribute("name", name);
                return "hello";
        }

}
```

4.4.4. 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>
```

4.4.5. 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>
```

4.4.6. 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" />
```

4.4.7. 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>
```

4.4.8.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.EnableAutoConfiguration;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfi
guration;
import
org.springframework.boot.context.properties.EnableConfiguration
Properties;
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.EnableMongoR
epositories;

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);
    }
}
```

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

7.1.1. 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 + "]";
        }

}
```

7.1.2. 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();
        }

}
```

7.1.3. 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

7.2.1. 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 {};
}
```

7.2.2. 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);
                }
        }
}
```

7.2.3. 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 + "]";
        }

}
```

7.2.4. 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. WebMvcConfigurerAdapter

```
package mis.config;

import mis.interceptor.SpringMVCInterceptor;
import
org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.InterceptorRe
gistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;

@Configuration
public class WebAppConfig extends WebMvcConfigurerAdapter {
    /**
     * 配置拦截器
     */
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new
SpringMVCInterceptor()).addPathPatterns("/**");
    }
}
```

8.2. 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 {

    }
}
```

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.UrlBasedViewResolver">
                <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.InternalResourceViewResolver">
                <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.ServletContextTemplateResolver">
                <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/**");
        }
}
```

第 4 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 Router

2.1. 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.ServerRequest;
import
org.springframework.web.reactive.function.server.ServerResponse;

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(BodyI
nserters.fromObject("Hello World!!!"));
        }
}
```

2.2. 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.RequestPredicat
es;
import
org.springframework.web.reactive.function.server.RouterFunction;
import
org.springframework.web.reactive.function.server.RouterFunctions
;
import
org.springframework.web.reactive.function.server.ServerResponse;

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(Reques
tPredicates.accept(MediaType.TEXT_PLAIN)),
helloWorldHandler::helloWorld);
        }
}
```

2.3. Thymeleaf

2.3.1. 模板引擎 Thymeleaf 依赖

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
```

2.3.2. 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.
```

2.3.3.

```
    @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";
    }
```

2.3.4. 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>
```

2.4. Webflux Redis

2.4.1. Maven Redis 依赖

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-
redis-reactive</artifactId>
                </dependency>
```

2.4.2. 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
```

2.4.3. Config

```
        @Bean
        public ReactiveRedisTemplate<String, String>
reactiveRedisTemplate(ReactiveRedisConnectionFactory factory) {
                ReactiveRedisTemplate<String, String>
reactiveRedisTemplate = new ReactiveRedisTemplate<>
(factory,RedisSerializationContext.string());
                return reactiveRedisTemplate;
        }
```

2.4.4. 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));
        }
}
```

2.4.5.

```
@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));
        }
}
```

2.5. Webflux Mongdb

2.5.1. Maven 依赖

```
        <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-mongodb-
reactive</artifactId>
        </dependency>
```

2.5.2. Repository

```
import
org.springframework.data.mongodb.repository.ReactiveMongoReposit
ory;

import cn.netkiller.entity.User;

public interface UserRepository extends
ReactiveMongoRepository<User, Long>{

}
```

2.5.3. 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();
        }
}
```

2.5.4. 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。

2.6. SSE

```
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("/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.getT1()
)).data(data.getT2()).build());
        }

        @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());
        }

}
```

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
```

2.7. Mono/Flux

Mono(返回0或1个元素)/Flux(返回0-n个元素)

第 5 章 Spring Data

1. Jackson

1.1. @JsonIgnore 返回json是不含有该字段

```
        @JsonIgnore
        private String entityName =
this.getClass().getSimpleName();
```

1.2. @JsonFormat 格式化 json 时间格式

1.2.1.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();
```

1.2.2. 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;
}
```

1.2.3. 枚举

```
public class Test {
   @JsonFormat(shape=JsonFormat.Shape.NUMBER)
   @JsonProperty("birthDate")
   private Date birthDate;
}
```

```
{
```

```
  "birthDate" : 1528702883858
}
```

1.2.4. 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
}
```

1.3. @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();
        }

}
```

2. Spring Data with Redis

2.1. 集成 Redis XML 方式

2.1.1. pom.xml

```
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-redis</artifactId>
                </dependency>
```

2.1.2. springframework-servlet.xml

```
        <!-- Redis Connection Factory -->
        <bean id="jedisConnFactory"

class="org.springframework.data.redis.connection.jedis.JedisConnectionFactory"
                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 5.1. 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.InternalResourceViewResolver">
```

```
                <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.PropertyPlaceholderConfigurer">
                <property name="location"
value="classpath:resources/development.properties" />
        </bean>

        <bean id="dataSource"
                class="org.springframework.jdbc.datasource.DriverManagerDataSource">
                <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.JedisConnectionFactory"
                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>
```

2.1.3. 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);
	}
}
```

2.1.4. 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>
```

2.1.5. 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

2.2. RedisTemplate

2.2.1. 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
```

2.2.2. 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

2.2.3. 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
```

2.2.4. 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
```

2.2.5. 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("name","Jerry"));
        // 结果 neo
```

2.2.6. 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
```

2.2.7. 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 );
        }
```

2.2.8. 返回字符串长度

```
redisTemplate.opsForValue().set("key","hello world");
System.out.println(redisTemplate.opsForValue().size("key"));
```

2.2.9. 如果key不存便缓存。

```
System.out.println(redisTemplate.opsForValue().setIfAbsent("name","neo"));      // name
之前已经存在 false
System.out.println(redisTemplate.opsForValue().setIfAbsent("age","11"));
// age 之前不存在 true
```

2.2.10. 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().multiSetIfAbsent(maps));         // 返回
true
System.out.println(redisTemplate.opsForValue().multiSetIfAbsent(maps2));        // 返回
false
```

2.2.11. List

2.2.11.1. 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"));
```

2.2.11.2. rightPushAll

```
                String[] stringarrays = new String[]{"1","2","3"};
        redisTemplate.opsForList().rightPushAll("listarrayright",stringarrays);
        System.out.println(redisTemplate.opsForList().range("listarrayright",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("listcollectionright",0,-1));
```

2.2.11.3. rightPushIfPresent

```
                System.out.println("========== KEY 不存在==========");

System.out.println(redisTemplate.opsForList().rightPushIfPresent("rightPushIfPresent","a
a"));

System.out.println(redisTemplate.opsForList().rightPushIfPresent("rightPushIfPresent","b
b"));
        System.out.println("========== KEY 已经存在==========");

System.out.println(redisTemplate.opsForList().rightPushIfPresent("rightPushIfPresent","a
a"));

System.out.println(redisTemplate.opsForList().rightPushIfPresent("rightPushIfPresent","b
b"));
```

2.2.11.4. leftPush

```
redisTemplate.opsForList().leftPush("list","java");
redisTemplate.opsForList().leftPush("list","python");
redisTemplate.opsForList().leftPush("list","c++");
```

2.2.11.5. 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",stringarrays);
```

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("listcollection",0,-1));
结果:[3, 2, 1]
```

2.2.11.6. range

```
    System.out.println(redisTemplate.opsForList().range("listarray",0,-1));
        // 结果:[3, 2, 1]
```

2.2.12. 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.valueOf(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);
        }

}
```

2.2.12.1. 返回集合中的所有成员

```
System.out.println(redisTemplate.opsForSet().members("setTest"));
```

2.2.12.2. 取出一个成员

```
System.out.println(redisTemplate.opsForSet().pop("setTest"));
```

2.2.12.3. 随机获取无序集合中的一个元素

```
System.out.println("Random member: " +
redisTemplate.opsForSet().randomMember("setTest"));
```

2.2.12.4. 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]
```

2.2.12.5. 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]
```

2.2.12.6. 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"));
```

2.2.12.7. 成员删除

```
String[] arrays = new String[]{"Java","PHP"};
System.out.println(redisTemplate.opsForSet().remove("setTest",arrays));
```

2.2.12.8. 返回集合数量

```
System.out.println(redisTemplate.opsForSet().size("setTest"));
```

2.2.12.9. 判断元素是否在集合成员中

```
ystem.out.println(redisTemplate.opsForSet().isMember("setTest","Linux"));
```

2.2.12.10. 对比两个集合求交集

```
        System.out.println(redisTemplate.opsForSet().members("key"));
    System.out.println(redisTemplate.opsForSet().members("otherKey"));
    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("library1",library2));
```

2.2.12.11. 对比两个集合求交集，然后存储到新的 key 中

```
System.out.println(redisTemplate.opsForSet().intersectAndStore("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().intersectAndStore("key",otherKey,"destKey")
);
```

2.2.12.12. 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));
```

2.2.12.13. 合并两个集合去重复后保存到新的 **key** 中

```
System.out.println(redisTemplate.opsForSet().unionAndStore("key","otherKey","destKey"));

System.out.println(redisTemplate.opsForSet().unionAndStore("key",otherKey,"destKey"));
```

2.2.12.14. 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));
```

2.2.12.15. 计算两个合集的差集，然后保存到新的 **key** 中

```
System.out.println(redisTemplate.opsForSet().differenceAndStore("key","otherKey","destKe
y"));
```

2.2.12.16. 遍历 **SET** 集合

```
        Cursor<Object> curosr = redisTemplate.opsForSet().scan("setTest",
ScanOptions.NONE);
    while(curosr.hasNext()){
        System.out.println(curosr.next());
    }
```

2.2.13. 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));
```

2.2.14. Hash

2.2.14.1. put

```
        redisTemplate.opsForHash().put("redisHash","name","neo");
    redisTemplate.opsForHash().put("redisHash","age",30);
    redisTemplate.opsForHash().put("redisHash","nickname","netkiller");
```

2.2.14.2. 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"));
```

2.2.14.3. 从键中的哈希获取给定hashKey的值

```
        System.out.println(redisTemplate.opsForHash().get("redisHash","age"));
```

2.2.14.4. delete

删除指定的哈希 hashKeys

```
        System.out.println(redisTemplate.opsForHash().delete("redisHash","name"));
```

2.2.14.5. 确定哈希hashKey是否存在

确定哈希hashKey是否存在

```
System.out.println(redisTemplate.opsForHash().hasKey("redisHash","age"));
```

2.2.14.6. 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("redisHash",keys))
```

2.2.14.7. 只有hashKey不存在时才能添加值

```
System.out.println(redisTemplate.opsForHash().putIfAbsent("redisHash","age",30));
```

2.2.14.8. 获取整个Hash

```
System.out.println(redisTemplate.opsForHash().entries("redisHash"));
```

2.2.14.9. 获取所有key

```
System.out.println(redisTemplate.opsForHash().keys("redisHash1"));
```

2.2.14.10. 通过 hashKey 获取所有值

```
System.out.println(redisTemplate.opsForHash().values("redisHash"));
```

2.2.14.11. 值加法操作

```
System.out.println(redisTemplate.opsForHash().increment("redisHash","age",1)
```

2.2.14.12. 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());
        }
```

2.2.15. 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);
        }
});
```

2.2.16. 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("bitTest",7));
```

2.2.17. 存储 Json 对象

2.2.17.1. 集成 RedisTemplate 定义新类 JsonRedisTemplate

```
package cn.netkiller.wallet.redis;

import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.Jackson2JsonRedisSerializer;
import org.springframework.data.redis.serializer.RedisSerializer;
import org.springframework.data.redis.serializer.StringRedisSerializer;

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();
        }
}
```

2.2.17.2. 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.RedisConnectionFactory;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.data.redis.listener.ChannelTopic;
import org.springframework.data.redis.listener.RedisMessageListenerContainer;
import org.springframework.data.redis.listener.adapter.MessageListenerAdapter;

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);
        }

}
```

2.2.17.3. 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;
        }
}
```

2.3. Spring Data Redis - Repository Examples

2.3.1. @EnableRedisRepositories 启动 Redis 仓库

```
package api.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.repository.configuration.EnableRedisRepositories;

@Configuration
@EnableRedisRepositories
public class CachingConfigurer {

}
```

2.3.2. 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;
        }

}
```

2.3.3. 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);
}
```

2.3.4. 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;
        }

}
```

3. Spring Data with MongoDB

https://docs.spring.io/spring-data/mongodb/docs/current/reference/html/

3.1. Example Spring Data MongoDB

3.1.1. 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>
```

3.1.2. 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 5.2. 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.UrlBasedViewResolver">
                <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.PropertyPlaceholderConfigurer">
                <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.JedisConnectionFactory" 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
```

3.1.3. 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 + "]";
	}

}
```

3.1.4. 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();
	}
```

```
}
```

3.1.5. 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" }
```

3.1.6. 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;
        }
```

3.2. @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 + "]";
        }

}
```

3.2.1. 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")
```

3.2.2. @Id

```
        @Id
        private String id;
```

3.2.3. @Version

```
@Version
private Long version;
```

3.2.4. @Field 定义字段名

```
@Field("url")
private String link;
```

3.2.5. @Indexed

https://docs.spring.io/spring-data/mongodb/docs/current/api/org/springframework/data/mongodb/core/index/Indexed.html

索引

3.2.5.1. 普通索引

```
@Indexed
```

3.2.5.2. 唯一索引

```
@Indexed(unique=true)
```

3.2.5.3. 索引排序方式

```
@Indexed(name = "first_name_index", direction = IndexDirection.DESCENDING)
```

3.2.5.4. 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;
```

3.2.5.5. 索引过期时间设置

```
@Indexed(name = "expire_after_seconds_index", expireAfterSeconds = 10)
private LocalDateTime updateDate;
```

3.2.6. @CompoundIndex 复合索引

3.2.6.1. 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
}
```

3.2.6.2. 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) })
```

3.2.7. @TextIndexed

```
@Document(language = "spanish")
class SomeEntity {

    @TextIndexed String foo;

    @Language String lang;

    Nested nested;
}

class Nested {

    @TextIndexed(weight=5) String bar;
    String roo;
}
```

3.2.8. @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)
```

3.2.9. @Transient 丢弃数据，不存到 mongodb

```
public class User {

    @Transient
    private Integer age;
```

```
    // standard getter and setter

}
```

3.2.10. @DBRef 做外外键引用

3.2.10.1. 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 + "]";
	}

}
```

3.2.10.2. 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
+ "]";
	}

}
```

3.2.10.3. 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> {

}
```

3.2.10.4. 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;
        }

}
```

3.2.10.5. 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"
}
```

3.2.11. @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();
```

3.2.12. @NumberFormat

```
@NumberFormat(style=Style.CURRENCY)
private double money;
```

3.2.13. 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;
```

3.2.14. 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;
```

3.2.15. GeoJson 数据类型

```
        @GeoSpatialIndexed
        private GeoJsonPoint location; // GPS 地址位置
```

```
location = new GeoJsonPoint(Double.valueOf(longitude), Double.valueOf(latitude));
```

3.3. MongoRepository

3.3.1. 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")
```

3.3.2. findAll()

```
        @RequestMapping(value = "read", method = RequestMethod.GET, produces = {
"application/xml", "application/json" })
        @ResponseStatus(HttpStatus.OK)
        public List<Withdraw> read() {
                return repository.findAll();
        }
```

3.3.3. deleteAll()

```
repository.deleteAll();
```

3.3.4. save()

```
repository.save(new City("Shenzhen", "China"));
```

3.3.5. count()

```
@RequestMapping("count")
public long count() {
        return repository.count();
}
```

3.3.6. exists() 判断是否存在

```
boolean isExists = userRepository.exists(user.getId());
```

3.3.7. existsById()

```
memberRepository.existsById(id);
```

3.3.8. findByXXXX

```
List<User> findByName(String name);

List<User> users = userRepository.findByName("Eric");
```

3.3.9. findAll with OrderBy

3.3.9.1. 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.findAllByMemberIdOrderByDefaultsDesc(memberId);
```

3.3.10. findAll with Sort

```
List<User> users = userRepository.findAll(new Sort(Sort.Direction.ASC, "name"));
```

3.3.11. FindAll with Pageable

```
Pageable pageable = PageRequest.of(0, 1);
Page<User> page = userRepository.findAll(pageable);
List<User> users = pages.getContent();
```

3.3.11.1. 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
```

3.3.12. 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");
```

3.3.13. 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);
```

3.3.14. Before / After

```
List<Assets> findAllByUpdateDateBefore(Date yesterday);
List<Assets> findAllByUpdateDateBeforeAndStatus(Date yesterday, String status);

List<Assets> findAllByUpdateDateAfter(Date yesterday);
```

3.3.15. @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);
```

3.4. 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;
```

3.4.1. 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");
```

3.4.2. 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}*
```

3.4.3. 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);
```

3.4.4. 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);
```

3.4.5. 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);
```

3.4.6. 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);
```

3.4.7. 删除

```
User user = new User();
user.setId("5bbf091efd9557069c4a25c5")
mongoTemplate.remove(user, "user");
```

3.4.8. 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);
}
```

3.4.9. 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);
}
```

3.4.10. Query

3.4.10.1. 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);
}
```

3.4.10.2. 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;
        }
```

3.4.11. Criteria

3.4.11.1. is

```
Query query = new Query();
query.addCriteria(Criteria.where("name").is("Neo"));
List<User> users = mongoTemplate.find(query, User.class);
```

3.4.11.2. 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);
```

3.4.11.3. 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);
```

3.4.11.4. 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);
}
```

3.4.11.5. 包含

```
public List<Person> findByFavoriteBooks(String favoriteBook) {
   Query query = new Query();
   query.addCriteria(Criteria.where("favoriteBooks").in(favoriteBook));
   return mongoTemplate.find(query, Person.class);
}
```

3.4.12. Update

3.4.12.1. set

```
Update update = new Update();
update.set("name", "Netkiller");
```

3.4.12.2. 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);
```

3.4.12.3. 更新数据

```
        Query query =
Query.query(Criteria.where("classId").is("1").and("Students.studentId").is("1"));
    Update update = Update.update("Students.$.name", "lisa");
    mongoTemplate.upsert(query, update, "class");
```

3.4.12.4. 删除数据

```
        Query query =
Query.query(Criteria.where("classId").is("1").and("Students.studentId").is("3"));
        Update update = new Update();
        update.unset("Students.$");
        mongoTemplate.updateFirst(query, update, "class");
```

3.4.12.5. inc

```
public void updateMultiplePersonAge() {
   Query query = new Query();
   Update update = new Update().inc("age", 1);
   mongoTemplate.findAndModify(query, update, Person.class);;
}
```

3.4.12.6. 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");
```

3.4.13. 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);
```

3.4.14. 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);
```

3.4.15. Query + PageRequest

```
final Pageable pageableRequest = new PageRequest(0, 2);
Query query = new Query();
query.with(pageableRequest);
```

3.4.16. newAggregation

```
                MultilevelDirectSellingAccountRewardsSettlementDetails
multilevelDirectSellingAccountRewardsSettlementDetails = new
MultilevelDirectSellingAccountRewardsSettlementDetails();

multilevelDirectSellingAccountRewardsSettlementDetails.setLoginname("111");
                multilevelDirectSellingAccountRewardsSettlementDetails.setPhone("111");

multilevelDirectSellingAccountRewardsSettlementDetails.setRecommenderLoginname("111");

multilevelDirectSellingAccountRewardsSettlementDetails.setRecommenderPhone("111");
```

```
multilevelDirectSellingAccountRewardsSettlementDetails.setRecommenderName("Neo");

multilevelDirectSellingAccountRewardsSettlementDetails.setRecommenderType("客户");
                multilevelDirectSellingAccountRewardsSettlementDetails.setAmount(5.02);

multilevelDirectSellingAccountRewardsSettlementDetails.setCreatedDate(new Date());

multilevelDirectSellingAccountRewardsSettlementDetailsRepository.save(multilevelDirectSe
llingAccountRewardsSettlementDetails);

                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<MultilevelDirectSellingAccountRewardsSettlementDetails> results =
mongoTemplate.aggregate(aggregation,
MultilevelDirectSellingAccountRewardsSettlementDetails.class,
MultilevelDirectSellingAccountRewardsSettlementDetails.class);
                System.out.println(results.getRawResults().get("result").toString());
```

3.4.17. 创建索引

```
mongoOps.indexOps(User.class).ensureIndex(new Index().on("name", Direction.ASC));
```

3.4.18. 子对象操作

3.4.18.1. 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);
```

3.5. 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
}
```

3.6. FAQ

3.6.1. 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。

4. Spring Data with MySQL

4.1. 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
```

4.2. 声明实体

4.2.1. @Entity 声明实体

声明 Class 即是数据库表

```
@Entity
@Table
public class Your_table {
        ...
        ...
}
```

4.2.2. @Table 定义表名

4.2.2.1. catalog

```
@Table(name="CUSTOMERS",catalog="hibernate")
```

4.2.2.2. schema

配置Schema

```
@Table(name="tabname", schema="public")
```

4.2.2.3. 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"})}
```

4.2.3. @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;
```

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
```

4.2.4. @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。

4.2.4.1. 字段长度

字段长度定义

```
@Column(name="name", length=80, nullable=true)
```

4.2.4.2. 浮点型

```
        @Column(precision=18, scale=5)
    private BigDecimal principal;

    @Column(name="Price", columnDefinition="Decimal(10,2) default '100.00'")
```

4.2.4.3. 创建于更新控制

```
        @Column(name = "ctime", nullable = false, insertable = false, updatable = false)
```

4.2.4.4. TEXT 类型

```
        private String subject;
        @Column(columnDefinition = "TEXT")
        private String content;
```

4.2.4.5. 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
```

4.2.5. @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;
}
```

4.2.6. @NotNull 不能为空声明

```
        @NotNull
        public String username;
```

4.2.7. @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;
}
```

4.2.8. @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;
```

4.2.9. 默认时间规则

4.2.9.1. 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;
```

4.2.9.2. 与时间日 期有关的 hibernate 注解

4.2.9.2.1. 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;
```

4.2.9.2.2. 创建时间

```
@Temporal(TemporalType.TIMESTAMP)
@Column(updatable = false)
@org.hibernate.annotations.CreationTimestamp
protected Date createdDate;
```

4.2.9.2.3. 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
)
```

4.2.9.3. 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;
```

4.2.9.4. 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
```

4.2.9.5. 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;
```

4.2.10. Enum 枚举数据类型

4.2.10.1. 实体中处理 enum 类型

@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;
```

@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;
```

4.2.10.2. 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
```

4.2.11. 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

4.2.12. 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)
```

4.2.13. 索引

4.2.13.1. 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 + "]";
	}

}
```

4.2.13.2. 唯一索引

针对字段做唯一索引

```
@Column(unique = true)
```

4.2.13.3. 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;
```

4.2.14. 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);
}
```

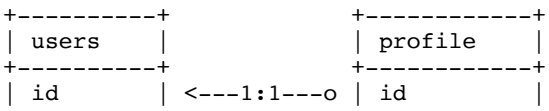
4.2.15. @JoinColumn

@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 "";

}
```

4.2.16. @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。



```
    | name     |             | sex        |
    | password |             | email      |
    +----------+             +------------+
```

```
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 + "]";
	}

}
```

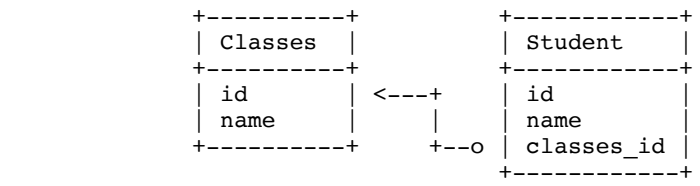
```
CREATE TABLE `users` (
        `id` INT(11) NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(255) NULL DEFAULT NULL,
        `password` VARCHAR(255) NULL DEFAULT NULL,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;

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")
```

4.2.17. OneToMany 一对多

我们要实现一个一对多实体关系，ER 图如下



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 = -54229057455199483l2L;
	@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

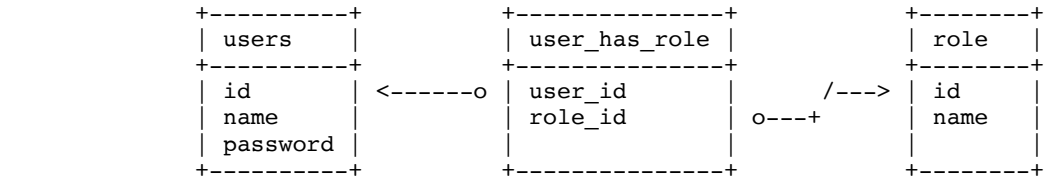
```
CREATE TABLE `classes` (
	`id` INT(11) NOT NULL AUTO_INCREMENT,
	`name` VARCHAR(255) NULL DEFAULT NULL,
	PRIMARY KEY (`id`)
)
```

```
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);
```

4.2.18. 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 = -24801941125970463 49L;
	@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 = 673703746567780032 6L;
	@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;
```

4.2.19. 外键级联删除

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 + "]";
	}

}
```

4.2.20. 其他

4.2.20.1. Cascade

CascadeType.PERSIST （级联新建）
CascadeType.REMOVE （级联删除）
CascadeType.REFRESH （级联刷新）
CascadeType.MERGE （级联更新）中选择一个或多个。
CascadeType.ALL

4.2.20.2. @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;
```

4.2.20.3. @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
}
```

4.3. 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{

}
```

4.4. 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
```

4.4.1. 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!
```

4.4.2. 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.
```

4.4.3. 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();
        }
```

4.4.3.1. 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);

}
```

4.4.3.2. 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.findByMemberIdAndStatisticsIdAndType(uid, id,
type);
```

4.4.4. count 操作

```
public interface UserRepository extends CrudRepository<User, Long> {

    Long countByFirstName(String firstName);

    @Transactional
    Long deleteByFirstName(String firstName);

    @Transactional
    List<User> removeByFirstName(String firstName);

}
```

4.4.5. OrderBy

```
public List<StudentEntity> findAllByOrderByIdAsc();
public List<StudentEntity> findAllByOrderByIdDesc();
List<RecentRead> findByMemberIdOrderByIdDesc(int memberId, Pageable pageable);
```

4.4.6. 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);

}
```

4.4.7. PageRequest 翻页操作

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

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.
```

4.4.7.1. 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;
        }
```

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));
```

4.4.7.2. Pageable

接口 实现 PagingAndSortingRepository

```
package api.repository.h5;

import org.springframework.data.repository.PagingAndSortingRepository;

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;
        }
```

排序 /browse?sort=id,desc
每页返回数量 /browse?size=10
返回第二页5条数据 /browse?size=5&page=1
返回第二页5条数据，ID倒序排序 /browse?size=5&page=1&sort=id,desc

4.4.8. 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);
```

4.4.9. Query

4.4.9.1. 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);
}
```

4.4.9.2. 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);
```

4.4.9.3. @Query 与 Pageagble

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#_native_queries

```
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 = 671099220657056861L;
        @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);

}
```

4.4.9.4. 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();
```

4.4.9.5. 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();
```

4.4.10. @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。

4.4.10.1. 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);

}
```

4.4.10.2. 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");
    }
```

4.4.11. 锁 @Lock

```
interface UserRepository extends Repository<User, Long> {

  // Plain query method
  @Lock(LockModeType.READ)
  List<User> findByLastname(String lastname);
```

5. 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();
  }
```

```
}
```

6. Spring Data with JdbcTemplate

6.1. execute

```
jdbcTemplate.execute("CREATE TABLE USER (id integer, name
varchar(100))");
```

6.2. queryForInt

```
int count = jdbcTemplate.queryForInt("SELECT COUNT(*) FROM
USER");
```

6.3. queryForLong

```
lang count = jdbcTemplate.queryForLong("SELECT COUNT(*) FROM
USER");
```

6.4. queryForObject

6.4.1. 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);
```

6.4.2. 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;
        }
```

6.4.3. 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));
```

```
        }
```

6.4.4. 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));
                });

        }
```

6.4.5. 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);
        }
}
```

6.4.6. 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;
    }
});
```

6.5. 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");
}
```

6.5.1. 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");
}
```

6.5.2. 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);
        }
```

6.5.3. forEach 用法

```
                jdbcTemplate.queryForList("select id from
public.contract").forEach(item -> {
                        logger.debug(item.toString());
                });
```

6.6. queryForMap

```
Map<String, Object> map = this.jdbcTemplate.queryForMap("SELECT
* FROM USERS WHERE USERNAME=?", "username");

System.out.println(map.get("USERNAME"));
```

6.7. query

6.7.1. 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;
});
```

6.7.2. 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;
    }
});
```

6.7.3. 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;
        }
```

6.8. queryForRowSet

```
SqlRowSet rs = jdbcTemplate.queryForRowSet("select * from
test");
```

6.9. 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);
        }
```

6.10.

```
new MapSqlParameterSource("symbol", symbol)
```

6.11. 实例参考

6.11.1. 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;
        }
```

7. Spring Data with Elasticsearch

7.1. 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。

7.1.1. 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/persistence
-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>
```

7.1.2. 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_timeo
ut=60s
spring.data.elasticsearch.properties.host=127.0.0.1
spring.data.elasticsearch.properties.port=9200
spring.data.elasticsearch.properties.path.home=/tmp
```

7.1.3. 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
+ "]";
        }

}
```

7.1.4. 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.ElasticsearchR
epository;
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);
}
```

7.1.5. 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.ArticleElasticsearchRep
ository;

@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);
        }
}
```

7.1.6. 测试

```
MacBook-Pro:~ neo$ curl
http://test:test@localhost:8443/restful/search/article/create.js
on
{"id":1,"title":"sssss","description":null,"author":null,"source
":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,"source
":null,"content":"test","ctime":null}
```

7.2. 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
```

7.3. Document

```
@Document(indexName = "customer", type = "external", shards = 1,
replicas = 0, refreshInterval = "-1")
```

7.4. 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.ElasticsearchTrashRepos
itory;

@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);
                        }
                }
        }
}
```

7.5. FAQ

7.5.1. 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

8. Spring Data FAQ

8.1. No identifier specified for entity

8.2. 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;
        }

}
```

8.3. 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。

8.4. 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 6 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.EnableAutoConfiguration;
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.EnableJpaReposit
ories;
import
```

```
org.springframework.data.mongodb.repository.config.EnableMongoR
epositories;
import
org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigu
rerAdapter;

@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.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 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。

2.7.1. 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 + "]";
        }
}
```

2.7.2. 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);

}
```

2.7.3. 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.b
uilders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.authentication.c
onfigurers.GlobalAuthenticationConfigurerAdapter;
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;
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.UserDetailsServic
e;
import
org.springframework.security.core.userdetails.UsernameNotFoundE
xception;

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().and(
).httpBasic().and().csrf().disable();

                // http.authorizeRequests().antMatchers("/",
"/index.html", "/css/**",
                //
"/js/**","/static/**","/setup.html").permitAll().anyRequest().a
uthenticated().and().formLogin().loginPage("/login.html").permi
tAll().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.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 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.b
uilders.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.Htt
pSecurity;
import
org.springframework.security.config.annotation.web.builders.Web
Security;
import
org.springframework.security.config.annotation.web.configuratio
n.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuratio
```

```
n.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("secre
t1").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();
        }

}
```

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").and
().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://example.
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.Htt
pSecurity;
import
org.springframework.security.config.annotation.web.configuratio
n.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuratio
n.WebSecurityConfigurerAdapter;
import
org.springframework.security.web.header.writers.StaticHeadersWr
iter;

@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/show/*","/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","/v1/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(SessionCreationPolicy.STATELE
SS);
}
```

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

4.6.1. 添加用户

```
        @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth)
throws Exception {
        auth
            .inMemoryAuthentication()
                .withUser("user").password("password").roles("USER");

    }
```

4.6.2. 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")
                ;
    }
```

4.6.3. 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）

5.1.1. 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。

5.1.2. 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.

5.1.3. 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_URL&scope=r
ead&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=AUTHORIZATION_COD
E&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。

5.1.4. 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。

5.1.5. 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

5.1.6. 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=CLIENT_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

5.3.1. authorization_code

5.3.1.1. 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>
```

5.3.1.1.1.

5.3.1.1.2.

5.3.1.1.3.

5.3.1.1.4.

5.3.1.1.5. 测试

```
http://localhost:8080/oauth/authorize?
response_type=code&client_id=sso&redirect_uri=http://localhost:8082/call
back&scope=read
http://localhost:8080/oauth/token?
client_id=sso&grant_type=authorization_code&redirect_uri=http://localhos
t:8082/callback&code=ZzLi3w

localhost:8082/admin&code=ZzLi3w

localhost:8080/oauth/authorize?
response_type=code&client_id=client&redirect_uri=http://www.netkiller.cn
&state=123

localhost:8080/oauth/authorize?
response_type=code&client_id=sso&redirect_uri=http://localhost:8082/test
&scope=app



http://localhost:8080/oauth/token?
client_id=sso&grant_type=authorization_code&redirect_uri=http://localhos
t:8082&code=8z5J9L

http://localhost:8080/oauth/authorize?
response_type=code&client_id=sso&redirect_uri=http://localhost:8080&scop
e=app

http://www.netkiller.cn/?code=eX6IMV&state=123
http://localhost:18084/oauth/token?
client_id=client&grant_type=authorization_code&redirect_uri=http://api.n
etkiller.cn&code=bzxoHn
```

5.3.2. 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

5.3.2.1. 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>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>
```

5.3.2.2. 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.BCryptPasswordEncoder;

/**
 * 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
```

5.3.2.3. Server

5.3.2.3.1. 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>
```

5.3.2.3.2. 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
```

5.3.2.3.3. 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.AuthenticationManager;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
```

```
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.config.annotation.configurers.Client
DetailsServiceConfigurer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
AuthorizationServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
EnableAuthorizationServer;
import
org.springframework.security.oauth2.config.annotation.web.configurers.Au
thorizationServerEndpointsConfigurer;

@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").accessTokenValidit
ySeconds(expiration).scopes("read",
"write").authorizedGrantTypes("password",
"refresh_token").resourceIds("resource");
        }

}
```

```
        @Override
        public void configure(ClientDetailsServiceConfigurer clients)
throws Exception {

clients.inMemory().withClient("api").secret(passwordEncoder().encode("se
cret")).accessTokenValiditySeconds(expiration).scopes("read",
"write").authorizedGrantTypes("password",
"refresh_token").resourceIds("resource");
        }
```

5.3.2.3.4. EnableResourceServer

```
package cn.netkiller.oauth.server.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.method.configuration.Enab
leGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity
;
import
org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableW
ebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
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().authenticate
d();
        }

        @Bean
        @Override
        public AuthenticationManager authenticationManagerBean() throws
Exception {
                return super.authenticationManagerBean();
        }
}
```

5.3.2.3.5. 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;
	}

}
```

5.3.2.3.6. 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);

}
```

5.3.2.3.7. 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.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
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);
        }
}
```

5.3.2.3.8. 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);
        }
}
```

5.3.2.3.9. 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。

5.3.2.3.10. 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=123456'
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 !
```

5.3.2.4. Spring boot with Oauth2 RestTemplate

5.3.2.4.1. Maven

```
                <dependency>

<groupId>org.springframework.security.oauth</groupId>
                        <artifactId>spring-security-oauth2</artifactId>
                </dependency>
```

5.3.2.4.2. 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.DefaultOAuth2ClientContext;
import org.springframework.security.oauth2.client.OAuth2RestOperations;
import org.springframework.security.oauth2.client.OAuth2RestTemplate;
import
org.springframework.security.oauth2.client.resource.OAuth2ProtectedResou
rceDetails;
import
org.springframework.security.oauth2.client.token.AccessTokenRequest;
import
org.springframework.security.oauth2.client.token.DefaultAccessTokenReque
st;
import
org.springframework.security.oauth2.client.token.grant.password.Resource
OwnerPasswordResourceDetails;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
EnableOAuth2Client;

@EnableOAuth2Client
@Configuration
public class OAuth2ClientConfiguration {

        @Value("${oauth.resource:http://localhost:8080}")
        private String baseUrl;

@Value("${oauth.authorize:http://localhost:8080/oauth/authorize}")
```

```
        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));
        }

}
```

5.3.2.4.3. 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);
        }
}
```

5.3.2.4.4. application.properties

```
security.basic.enabled=false
```

5.3.2.4.5. Controller

```
package cn.netkiller.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.oauth2.client.OAuth2RestOperations;
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;
        }
}
```

5.3.2.4.6. Test

```
neo@MacBook-Pro ~/deployment % curl http://localhost:8080
```

OK

5.3.3. Spring boot with Oauth2 jwt

5.3.3.1. 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>
```

5.3.3.2. 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.AuthenticationManager;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.oauth2.config.annotation.configurers.Client
DetailsServiceConfigurer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
AuthorizationServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
EnableAuthorizationServer;
import
org.springframework.security.oauth2.config.annotation.web.configurers.Au
thorizationServerEndpointsConfigurer;
import
org.springframework.security.oauth2.provider.token.DefaultTokenServices;
import org.springframework.security.oauth2.provider.token.TokenStore;
import
org.springframework.security.oauth2.provider.token.store.JwtAccessTokenC
onverter;
import
```

```
org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

@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_CLIENT')");
        // }

        @Override
        public void configure(ClientDetailsServiceConfigurer clients)
throws Exception {

clients.inMemory().withClient("api").secret(passwordEncoder().encode("se
cret")).accessTokenValiditySeconds(expiration).refreshTokenValiditySecon
ds(expiration).scopes("read", "write").authorizedGrantTypes("password",
"refresh_token").authorities("USER").resourceIds("blockchain");
        }

        @Override
        public void configure(AuthorizationServerEndpointsConfigurer
configurer) throws Exception {

configurer.tokenStore(tokenStore()).accessTokenConverter(accessTokenConv
erter());
                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;
        }

}
```

5.3.3.3. 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.configuration.
EnableResourceServer;
import
org.springframework.security.oauth2.config.annotation.web.configuration.
ResourceServerConfigurerAdapter;
import
org.springframework.security.oauth2.config.annotation.web.configurers.Re
sourceServerSecurityConfigurer;
import
org.springframework.security.oauth2.provider.token.DefaultTokenServices;
import org.springframework.security.oauth2.provider.token.TokenStore;
import
org.springframework.security.oauth2.provider.token.store.JwtAccessTokenC
onverter;
import
org.springframework.security.oauth2.provider.token.store.JwtTokenStore;

@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;
        }

}
```

5.3.3.4. 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.AuthenticationManager;
import
org.springframework.security.config.annotation.method.configuration.Enab
```

```
leGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity
;
import
org.springframework.security.config.annotation.web.builders.WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableW
ebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;

@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().authenticated();

http.authorizeRequests().antMatchers(HttpMethod.OPTIONS).permitAll().any
Request().authenticated().and().httpBasic().and().csrf().disable();
        }

        @Bean
        @Override
        public AuthenticationManager authenticationManagerBean() throws
Exception {
                return super.authenticationManagerBean();
        }

}
```

5.3.3.5. 插入数据

```
INSERT INTO `user` (username, password, enabled) VALUES
('netkiller@msn.com',
'$2a$10$Cyj3hM39V34r5pMeQ.Y9peuUqYMBSvsJ7GTBgp4.stWaTtWMboYGS', true);
```

5.3.3.6. 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.eyJhdWQiOlsicmVzb3
VyY2UiXSwiZXhwIjoxNTM1MTE4MzYxLCJ1c2VyX25hbWUiOiJuZXRraWxsZXJAbXNuLmNvbS
IsImp0aSI6ImM5YzA4ODczLWZlMTctNGM2My05MDA1LTIzZGJlNTE1NTQ0YiIsImNsaWVudF
9pZCI6ImFwaSIsInNjb3BlIjpbInJlYWQiLCJ3cml0ZSJdfQ.ydxJQKtdBNWHELL8_aXVoZE
TNMygXs8T1HQ5XWDBELA","token_type":"bearer","refresh_token":"eyJhbGciOiJ
IUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsicmVzb3VyY2UiXSwidXNlcl9uYW1lIjoibm
V0a2lsbGVyQG1zbi5jb20iLCJzY29wZSI6WyJyZWFkIiwid3JpdGUiXSwiYXRpIjoiYzljMD
g4NzMtZmUxNy00YzYzLTkwMDUtMjNkYmU1MTU1NDRiIiwiZXhwIjoxNTM1MTE4MzYxLCJqdG
kiOiI3ODJiNmY2NC0xYzdiLTQ0YWYtOThlZC1iZDk3Y2QzYzE1NjEiLCJjbGllbnRfaWQiOi
JhcGkifQ.16QUHOrVPUBF97E_972AcLA83zEK7UzaI424PeAmX6E","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.eyJhdWQiOlsiYmxvY2tjaGFpbiJdLCJleHA
iOjE1MzUxMTkyMTUsInVzZXJfbmFtZSI6Im5ldGtpbGxlckBtc24uY29tIiwianRpIjoiZTN
jYmIxNTItYmFkZS00MGM1LTkxOGItZjJjMzZiYTBiMTE2IiwiY2xpZW50X2lkIjoiYXBpIiw
ic2NvcGUiOlsicmVhZCIsIndyaXRlIl19.ophWAvIT1ZmWCIQyAgyuzmQ98TJsN49OeR3CBS
J_X54" -X GET http://localhost:8080/test/mongo
{"id":"5b800709e18a211e83c7f355","name":"Netkiller"}
```

5.3.3.7. 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
```

5.3.3.8. refresh_token

```
curl -s  -X POST --user 'api:secret' -d
'grant_type=refresh_token&client_id=api&client_secret=secret&refresh_tok
en=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOlsiYmxvY2tjaGFpbiJdLCJ1
c2VyX25hbWUiOiJibG9ja2NoYWluIiwic2NvcGUiOlsicmVhZCIsIndyaXRlIl0sImF0aSI6
ImU2OGE4ODUzLWJlODUtNGUwNy1hYzE0LTM0MDI1ZDBiZGY0ZiIsImV4cCI6MTU0MDM4MzA5
NCwianRpIjoiY2JjNTk2ZWMtOTkyZi00NmQ2LWFiZGYtMTcyYWFiZmEwNDFiIiwiY2xpZW50
X2lkIjoiYXBpIn0.PqsGl5Dm8WBqwbhHf-LqmUP1toCpsFS4gLeOP2bMwR4'
${URL}/oauth/token
```

5.3.4. Spring boot with Oauth2 jwt 非对称证书

5.3.4.1. 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-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAj6ePdDwBrHKX3kNFnbve
T1rTTbyA9GjaiZNwj2X4Y0In7RCFl8auXXBn2DxztQMGqHY2Ydc3/26Gu9Vri441
r8/RInA6UpzzDRl5SeYYTobcgfIVpfQ0hTX0xzuMDVLVoLibGfcvGy7ZkrJjQFX8
lIaO84K8KP/yzma5622XJ+f5hkXmTX5e0tXGDCPjVO1dSrouPWqhcbM0Kf6y3RdE
JkNRTHLky6afx8MNobakz1Ab9K7cjD8De6LwScwMQMFU46traN/3Fw0lZFxKkpay
+sEUHvHDUYWTuVovUmfiKMX8fj5QCm4imPdA3pF/jjM+xeeVcTID3qffDGOKrGTF
HQIDAQAB
-----END PUBLIC KEY-----
```

复制 jwt.jks 和 public.crt 到 src/main/resources 目录下

5.3.4.2. 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;
        }
```

5.3.4.3. Resource Server

```
        @Bean
        public JwtAccessTokenConverter accessTokenConverter() {
                JwtAccessTokenConverter converter = new
JwtAccessTokenConverter();

                String publicKey = "-----BEGIN PUBLIC KEY-----\n" +

"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAj6ePdDwBrHKX3kNFnbve\n" +

"T1rTTbyA9GjaiZNwj2X4Y0In7RCFl8auXXBn2DxztQMGqHY2Ydc3/26Gu9Vri441\n" +

"r8/RInA6UpzzDRl5SeYYTobcgfIVpfQ0hTX0xzuMDVLVoLibGfcvGy7ZkrJjQFX8\n" +

"lIaO84K8KP/yzma5622XJ+f5hkXmTX5e0tXGDCPjVO1dSrouPWqhcbM0Kf6y3RdE\n" +

"JkNRTHLky6afx8MNobakz1Ab9K7cjD8De6LwScwMQMFU46traN/3Fw0lZFxKkpay\n" +

"+sEUHvHDUYWTuVovUmfiKMX8fj5QCm4imPdA3pF/jjM+xeeVcTID3qffDGOKrGTF\n" +
                                "HQIDAQAB\n" +
                                "-----END PUBLIC KEY-----";

                converter.setVerifierKey(publicKey);
                return converter;
        }
```

5.3.5. 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);
}];
```

5.3.6. Oauth2 客户端

环境：Java11 + Springboot 2.1.3

5.3.6.1.

```
<?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>
```

5.3.6.2. 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
```

5.3.6.3. SpringApplication

```
package cn.netkiller.oauth2.client;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
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);
        }
}
```

5.3.6.4. WebSecurityConfigurer

```
package cn.netkiller.oauth2.client;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity
;
import
org.springframework.security.config.annotation.web.configuration.EnableW
ebSecurity;
```

```
import
org.springframework.security.config.annotation.web.configuration.WebSecu
rityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends WebSecurityConfigurerAdapter
{
        @Override
        protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().anyRequest().authenticated().and().oauth2Login(
);
        }
}
```

5.3.6.5. TestController

```
package cn.netkiller.oauth2.client;

import java.security.Principal;

import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.oauth2.client.registration.ClientRegistrati
on;
import
org.springframework.security.oauth2.client.registration.ClientRegistrati
onRepository;
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("netkiller");

                return clientRegistration;
        }
}
```

5.3.7. 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);
                    }
                });
```

```
            }
        });
    }

}
```

5.3.8. RestTemplate 使用 HttpClient

5.3.8.1. 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>
```

5.3.8.2. 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);
        }
}
```

5.3.8.3. 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.OAuth2RestOperations;
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.json",HttpMe
thod.GET,entity,String.class);
```

```
            System.out.println(response.getBody());
                return response.getStatusCode().toString() + " " +
response.getBody();
        }
}
```

5.3.8.4. 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 !
```

5.3.9.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.HttpComponentsClientHttpRequestFactory;
import org.springframework.security.oauth2.client.OAuth2RestOperations;
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(60000);
```

```
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();
        }
}
```

5.3.10. 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;
        }
}
```

5.3.11. SecurityContextHolder 对象

```
        @RequestMapping(value = "principal", method = RequestMethod.GET)
    public Object getPrincipal() {
        Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
        return principal;
    }

    // 查询用户角色
```

```
    @RequestMapping(value = "roles", method = RequestMethod.GET)
    public Object getRoles() {
        return
SecurityContextHolder.getContext().getAuthentication().getAuthorities();
    }
```

5.3.12. 资源服务器配置

ResourceServerConfigurerAdapter

5.3.12.1. 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()
        ;
    }
```

5.3.12.1.1.

```
        @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')");
    }
```

5.3.12.1.2.

```
                        @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"));
                        }
                    }
                });
            }
```

5.3.13. Client

5.3.13.1. Overriding Spring Boot 2.0 Auto-configuration

```
@Configuration
public class OAuth2LoginConfig {

        @Bean
        public ClientRegistrationRepository
clientRegistrationRepository() {
                return new
InMemoryClientRegistrationRepository(this.googleClientRegistration());
        }

        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/auth")

.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();
        }
}
```

5.3.14. Oauth2 常见问题

5.3.14.1. 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
        }
```

5.3.14.2. 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 7 章 Spring Cloud

1. Spring Cloud Config

1.1. 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>
```

1.2. Server

1.2.1. 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>
```

1.2.2. 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);
        }
}
```

1.2.3. application.properties

```
server.port=8888
spring.cloud.config.server.git.uri=https://github.com/netkiller/
config.git
```

1.2.4. 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
```

1.2.5. 测试服务器

```
neo@netkiller $ curl http://localhost:8888/server-
development.json
{"message":"Hello world","test":{"a":"KKOOKK"}}
```

1.3. Client

1.3.1. 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>
```

1.3.2. 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.RefreshScope
;
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

1.3.3. bootstrap.properties

```
spring.application.name=server-development
spring.cloud.config.uri=http://localhost:8888
management.security.enabled=false
```

1.3.4. 测试 client

```
neo@netkiller $ curl http://localhost:8080/message.json
Hello world
```

1.4. Config 高级配置

1.4.1. 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
```

1.4.1.1. 分支

label 是指 git 的分支

```
spring.cloud.config.label=mybranch
```

1.4.1.2. basedir

```
spring.cloud.config.server.git.basedir=api/configs
```

1.4.1.3. HTTP Auth

```
spring.application.name=config-server
spring.cloud.config.server.git.uri=https://netkiller:xxxxxx@gith
ub.com/xyz/microservices-configs.git
```

```
spring.application.name=config-server
spring.cloud.config.server.git.uri=https://github.com/xyz/micros
ervices-configs.git
spring.cloud.config.server.git.username=netkiller
spring.cloud.config.server.git.password=password
```

1.4.1.4. 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
```

1.4.1.5. 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"}
```

1.4.2. Config server 用户认证

1.4.2.1. Server 配置

1.4.2.1.1. 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
```

1.4.2.1.2. 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>
```

1.4.2.1.3. 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"}}
```

1.4.2.2. 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
```

1.4.3. 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
```

1.4.4. Spring Cloud Config JDBC Backend

1.4.4.1. 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>
```

1.4.4.2. 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;
```

1.4.4.3. 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')");
    }

}
```

1.4.4.4. 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
```

1.5. Old

1.5.1. 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>
```

1.5.2. 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>
```

2. Spring Cloud Consol

2.1. 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
```

2.2. 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>
```

2.3. Consul 服务生产者

2.3.1. 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>
```

2.3.2. 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
```

2.3.3. SpringApplication

```
package cn.netkiller.consul;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class Application {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                SpringApplication.run(Application.class, args);
        }
}
```

2.3.4. 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 + "!";
	}

}
```

2.4. Consul 服务消费者

2.4.1. 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>
```

2.4.2. 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
```

2.4.3. 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);
        }
}
```

2.4.4. 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.LoadBalancerClient;
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;
        }
}
```

2.5. Openfeign

2.5.1. 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>
```

2.5.2. 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
```

2.5.3. SpringApplication

```
package cn.netkiller.openfeign;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
```

```
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);
        }
}
```

2.5.4. 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";
        }

}
```

2.5.5. 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);
        }
}
```

3. Spring Cloud Netflix

3.1. Eureka Server

3.1.1. 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.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>
```

3.1.2. 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.EnableEurekaServ
er;

@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);
        }
}
```

3.1.3. application.properties

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
eureka.client.serviceUrl.defaultZone=http://localhost:${server.p
ort}/eureka/

logging.level.com.netflix.eureka=OFF
logging.level.com.netflix.discovery=OFF
```

3.1.4. 检查注册服务器

http://localhost:8761

3.2. Eureka Client

3.2.1. 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.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>
```

3.2.2. 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);
        }
}
```

3.2.3. 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";
        }
}
```

3.2.4. application.properties

```
spring.application.name=test-service
server.port=8080
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eurek
a/
```

3.2.5. 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
```

3.3. Feign client

3.3.1. 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.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>
```

3.3.2. 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!");
        }
}
```

3.3.3. 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

3.3.4. 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/eurek
a/
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
```

3.3.5. 测试

```
$ curl -s http://localhost:8088/get-greeting.json

GREETING
```

3.3.6. 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";
    }
}
```

3.4. 为 Eureka Server 增加用户认证

3.4.1. Maven

```
                <dependency>

<groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
security</artifactId>
                </dependency>
```

3.4.2. application.properties

```
                                security.user.name=eureka
                                security.user.password=s3cr3t
```

3.4.3. Eureka Client

```
                                spring.application.name=restful-
api-service

eureka.client.serviceUrl.defaultZone=http://eureka:s3cr3t@localh
ost:8761/eureka/
```

3.4.4. Feign Client

```
eureka.client.serviceUrl.defaultZone=http://eureka:s3cr3t@localh
ost:8761/eureka/
```

3.5. Eureka 配置项

3.5.1. /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.EurekaServerConfigBean

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

3.5.2. 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
```

3.5.3. 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=
```

3.5.4. 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
```

3.6. ribbon

3.6.1.

```
@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);
    }
}
```

3.6.2. LoadBalancerClient 实例

3.6.2.1. application.properties

```
web.ribbon.listOfServers=localhost:7900,localhost:7901,localhost
:7902
```

3.6.2.2. 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();
        }

}
```

3.6.3. Ribbon 相关配置

```
spring.cloud.loadbalancer.ribbon.enabled=false
```

3.6.3.1.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。

3.7. 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));
    }
}
```

3.8. Zuul

3.8.1. 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>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>
```

3.8.2. 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);
        }
}
```

3.8.3. 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
```

3.8.4. 负载均衡配置

```
zuul:
  routes:
    httpbin:
```

```
      path: /**
      serviceId: httpslb

httpslb:
  ribbon:
    listOfServers: api1.netkiller.org, api2.netkiller.cn
```

4. Openfeign

4.1. Openfeign 扫描包配置

```
@EnableFeignClients(basePackages = {"cn.netkiller.openfeign"})
```

4.2. 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");
    }
}
```

4.3. 配置连接方式

启用 okhttp

```
feign.okhttp.enabled=true
```

httpclient

```
feign.httpclient.enabled=true
```

4.4.

```
@FeignClient(name="myServiceName", url="localhost:8888")
public interface OpenfeignService {
        @RequestMapping("/")
        public String getName();
}
```

5. 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。

5.1. Gateway 例子

5.1.1. 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>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>
```

5.1.2. 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);

        }
}
```

5.1.3. 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
```

5.1.4. 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.RouteLocatorBui
lder;
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();
    }

}
```

5.2. 路由 配置

5.2.1. 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();
        }
```

5.2.2. 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

6. Spring Cloud Stream

7. Spring Cloud Bus

8. Spring Cloud Sleuth

分布式链路追踪工具

9. Spring Cloud 相关的 application.properties 配置

9.1. 启用或禁用 bootstrap

```
spring.cloud.bootstrap.enabled=false
spring.cloud.bootstrap.location=classpath:bootstrap.properties
```

9.2. 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

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。

10.1.1. 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>
```

10.1.2. 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
```

10.1.3. 程序文件

10.1.3.1. 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);
        }
}
```

10.1.3.2. 配置类

```
package cn.netkiller.config;

import org.springframework.boot.context.properties.ConfigurationProperties;
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;
        }
}
```

10.1.3.3. 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.EnableConfigurationProperties;
import org.springframework.cloud.context.config.annotation.RefreshScope;
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()) + "]";
        }
}
```

10.1.4. 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
```

10.1.5. 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:
sha256:3ef48e858254ee4d578fe1737fd948b2679c33d28d0dc573cf1e8076d0a054a1 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。

10.2.1. 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>
```

10.2.2. provider

10.2.2.1. 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>
```

10.2.2.2. 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.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class ProviderApplication {
        public static void main(String[] args) {
                System.out.println("Hello World!");
                SpringApplication.run(ProviderApplication.class, args);
        }
}
```

10.2.2.3. 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。

10.2.2.4. 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。

10.2.2.5. 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"
```

10.2.3. consumer

10.2.3.1. 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>
```

10.2.3.2. 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.EnableDiscoveryClient;
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);
        }
}
```

10.2.3.3. 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);
	}
}
```

10.2.3.4. 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";
	}
}
```

10.2.3.5. 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
```

10.2.3.6. 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
```

10.2.4. 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:
sha256:2591686cfc63888f3b97ca1e950205b58a47b3d3c618242465baa0796cf7e056 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:
sha256:42cdeb63cd67a5edf9e769538878a0c8f18048bcde1ef9ba22d58766afa2d52d 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
```

10.2.5.

```
spring.cloud.kubernetes.discovery.enabled=false
```

11. FAQ

11.1. 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);
        }
```

```
}
```

11.2. @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。

11.3. 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
```

11.4. 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
```

11.5. 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);
```

11.6. 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 7.1. 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);
        }
}
```

11.7. Load balancer does not have available server for client

```
com.netflix.client.ClientException: Load balancer does not have
available server for client: restful
```

11.8. Eureka Client (Dalston.SR1)

11.8.1. 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.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>
```

11.8.2. 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);
```

```
        }
}
```

11.8.3. 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";
        }
}
```

11.8.4. application.properties

```
spring.application.name=test-service
server.port=8080
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eure
ka/
```

11.8.5. 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
```

11.9. Config Server(1.3.1.RELEASE)

11.9.1. Server

11.9.1.1. 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>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>
```

11.9.1.2. 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);
        }

}
```

11.9.1.3. application.properties

```
server.port=8888
spring.cloud.config.server.git.uri=https://github.com/netkiller
/config.git
```

11.9.1.4. 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
```

11.9.1.5. 测试服务器

```
neo@netkiller $ curl http://localhost:8888/server-
development.json
{"message":"Hello world","test":{"a":"KKOOKK"}}
```

11.9.2. Client

11.9.2.1. 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>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>
```

11.9.2.2. 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

11.9.2.3. bootstrap.properties

```
spring.application.name=server-development
spring.cloud.config.uri=http://localhost:8888
management.security.enabled=false
```

11.9.2.4. 测试 client

```
neo@netkiller $ curl http://localhost:8080/message.json
Hello world
```

第 8 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.PropertyPlaceho
lderConfigurer">
                <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.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>
```

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 8.1. 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.WebApplicationContextUt
ils;
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 9 章 Miscellaneous

1. 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(user
);
```

2. 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);
```

第 10 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 =
7603772682950271321L;

	@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);
            }

        };
    }

}
```

附录 A. 附录

1. Springboot example

https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples