项目的一些说明：

建议开发工具：**IntelliJ IDEA**，Spring Tool Suite (STS)

建议开发环境：**JDK1.8**，Gradle 2.3+ or **Maven 3.0+**

项目使用maven导入依赖包，右键Application运行，或者发布到外部Tomcat运行（发布到外部tomcat需要把pom.xml下面的几个包的provided注释去掉），建议开发时使用内嵌tomcat，到预发布时再放到外部tomcat运行。

可以使用resources目录下的mybatis的生成工具generator，进行逆向工程生成相应的dao层和mapping文件。

调用顺序：JSP/Html -> Service -> ServiceImpl -> Dao ->（DaoImpl）-> Mapper

目录结构：

JAVA类：

Config：数据库配置和系统配置

Controller：控制层

Dao：持久化层

Mapper：dao层调用的mapping文件，写sql语句

Domain: 实体层

Global: 全局控制层

Interceptor: 拦截器层

Service: 服务层

Util: 工具类层

Application：入口类

Resource:

Generator:：mybatis生成工具

Static：静态文件

Application.properties: 配置文件（可以使用yaml文件格式**application.yml**）

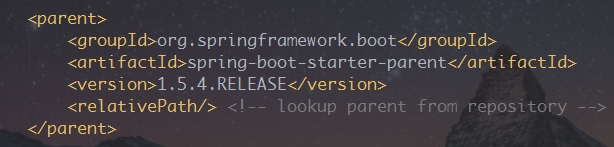
Logback-spring.xml：logback日志控制

Spring.sql：案例数据库初始化文件

Webapp:

存放jsp文件

Pom.xml依赖包文件



这块配置就是Spring Boot父级依赖，有了这个，当前的项目就是Spring Boot项目了，spring-boot-starter-parent是一个特殊的starter,它用来提供相关的Maven默认依赖，使用它之后，常用的包依赖可以省去version标签。

开发规范说明：

View-> Controller -> Service -> ~~ServiceIml~~ -> Dao

（中和面向接口编程和项目开发需求，为了不过度设计，去掉区分service接口层和实现层）

视图层(controller)：

选用JSP或者Thymeleaf，页面中尽量使用EL表达式，避免出现JAVA代码；

控制层(service)：

每个类需要加上@RestController，@RequestMapping()，RestController注解相当于@ResponseBody ＋ @Controller合在一起的作用，返回JSON类型数据。如需要返回视图则通过类似new ModelAndView(“index”)直接返回。

类名添加RequestMapping是为了避免不同controller类中出现相同的映射导致启动报错。

数据访问层(dao):

~~单表操作使用spring boot jpa框架进行数据操作，dao层继承CrudRepository或者JpaRepository，里面有包含现成的常用方法可以调用。~~

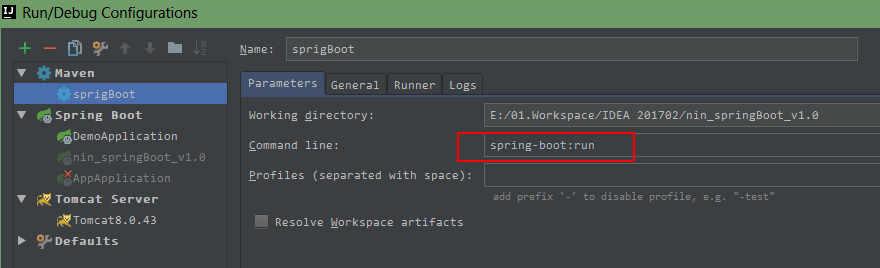
Mybatis有注解方式以及XML配置方式两种方式实现。

不涉及动态SQL语句的简单的语句只需要使用@Insert、@Update、@Delete、@Select这4个注解

涉及表关联的多表操作使用mybatis框架，使用mybatis generator工具generatorConfig.xml生成相关表的dao,domain,mapper.xml文件，再写相关的sql语句。

运行发布：

可以使用内置tomcat，也可以使用外部tomcat。运行内部tomcat统一使用mvn指令springBoot:run，这样就可以同时运行内部tomcat和外部tomcat，不用去修改pom.xml文件的依赖包。部署到外部tomcat，如果再直接右键运行会报找不到内置tomcat容器，但是使用mvn指令springBoot:run就可以同时部署到内部容器。



配置文件application.yml中的server.context-path: /prefix/，中的prefix可以配置成项目名，这样部署到内置tomcat和外部tomcat就能一致，访问都要加上项目名。

关于热部署：

目前测试可用的有两种热部署工具：spring-boot-devtools和JRebel插件

[spring-boot-devtools安装](http://412887952-qq-com.iteye.com/blog/2300313)

[Intellij IDEA使用Spring-boot-devTools无效解决办法](http://blog.csdn.net/a332472810/article/details/52524166)

[Idea下Jrebel插件安装](http://www.jianshu.com/p/d177316890e3)

[Idea下Jrebel最新激活方法](http://blog.csdn.net/lixuanshengchao/article/details/72599021)

[Eclipse下jrebel插件安装](http://blog.csdn.net/xiaopihai86/article/details/50762386)

[Myeclipse下jrebel最新激活方式](http://www.cnblogs.com/oskyhg/p/6667016.html)

缓存功能：

集成Redis实现**缓存机制**，使用Spring cache + redis，实现缓存。

集成Spring session实现session集群部署，使用spring session + redis，实现了session的共享。将session保存到redis集中管理，登录系统在不同的app之间跳转的时候，session都是一致。

缓存的应用，具体实现参照nin\_springBoot\_v1.0中的DemoInfoServiceImpl类

**登录信息等数据存储到java的内置对象session中，频繁操作数据的方法要加上缓存注解。**

缓存的注解介绍：

@EnableCaching会为每个bean中被 @Cacheable, @CachePut and @CacheEvict修饰的public方法进行缓存操作。

缓存都是通过key-value进行储存的，value或cacheNames必须指定(value是cacheNames的别名)，指定多个value用(value = {"value1", "value2"})如果没有指定key，spring会提供一个默认的KeyGenerator，这个KeyGenerator根据参数生成key，如果方法没有参数返回KeyGenerator.EMPTY，如果有一个参数返回这个实例，如果有多个参数返回包含这些参数的SimpleKey。可以通过继承CachingConfigurerSupport自己指定KeyGenerator，类上加@Configuration注解。也可以像上面那样自己指定key，需要了解SPEL表达式。

@Cacheable该注解可以用在方法或者类级别。当他应用于方法级别的时候，就是如上所说的缓存返回值了。当应用在类级别的时候，这个类的所有方法的返回值都将被缓存

支持如下几个参数：

value：缓存位置名称，不能为空，如果使用EHCache，就是ehcache.xml中声明的cache的name

key：缓存的key，默认为空，既表示使用方法的参数类型及参数值作为key，支持SpEL

condition：触发条件，只有满足条件的情况才会加入缓存，默认为空，既表示全部都加入缓存，支持SpEL

@CachePut 这个注释可以确保方法被执行，同时方法的返回值也被记录到缓存中，实现缓存与数据库的同步更新，理解为update语句。

[@CachEvict](mailto:-@CachEvict) 支持如下几个参数：

value：缓存位置名称，不能为空，同上

key：缓存的key，默认为空，同上

condition：触发条件，只有满足条件的情况才会清除缓存，默认为空，支持SpEL

allEntries：true表示清除value中的全部缓存，默认为false

Redis的介绍：

[Redis在windows下安装过程](http://www.cnblogs.com/M-LittleBird/p/5902850.html)

统一异常处理功能：

异常处理分为全局异常处理和局部异常处理；

全局异常处理类GlobalErrorController实现ErrorController类，会捕获由spring boot捕获的全局异常映射到/error，主要为捕获404错误和权限异常

局部异常处理类GlobalExceptionHandler使用spring的controllerAdvice注解捕获controller层异常。可以捕获运行时异常，或者抛出用户自定义异常信息。开发阶段使用/error/debug方便搜索查找，发布阶段使用/error/500返回统一的500错误页面。

统一http请求返回的数据格式：

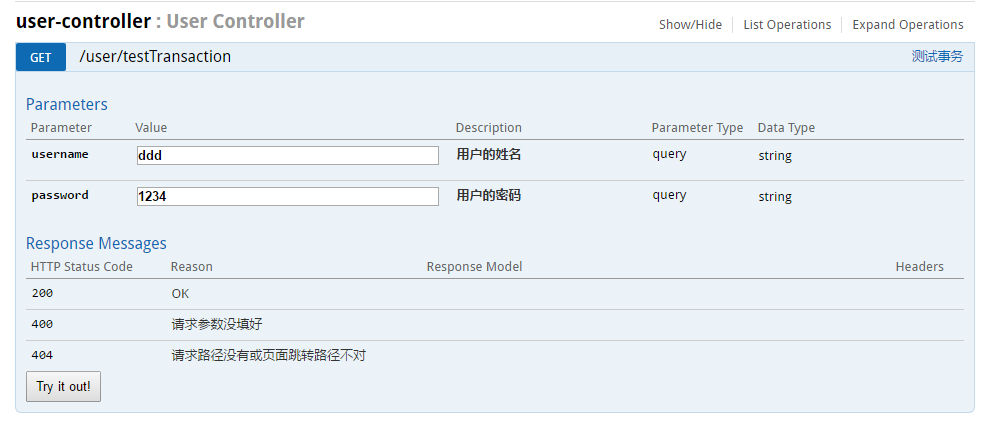
使用serveltUtil类的createSuccessResponse和createErrorResponse返回正确和异常信息

使用Swagger2构建restful api

具体示例，见UserController类

访问http://localhost:8080/mySpringBoot/swagger-ui.html，便可以查看构建的所有服务的API文档。其中mySpringBoot是当前项目名（根据实际项目修改）。

wagger UI不仅仅是文档，还提供了在线API调试的功能



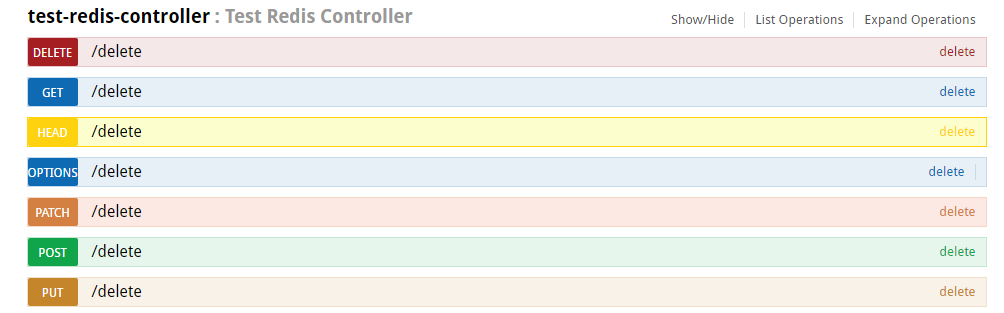
点击Try it out！后就可以调用相应的服务，并看到相应的返回状态。

必须在@RequestMapping注明method是POST还是Get，否则SwaggerUI中将生成该接口的所有方法（包括PUT,DELETE,POST等）

@GetMapping,@PostMapping,@PutMapping,@DeleteMapping 等注解是Spring MVC 4.3X版本添加的新注解，它们是对@RequestMapping注解的简化封装。

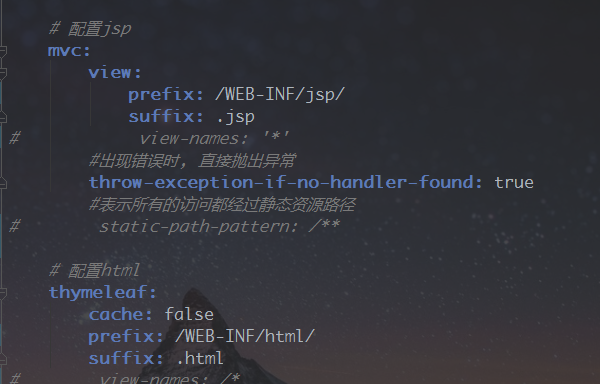
以@GetMapping为例，Spring官方文档说：@GetMapping是一个组合注解，是@RequestMapping(method = RequestMethod.GET)的缩写。该注解将HTTP Get 映射到 特定的处理方法上。

实际开发中各个controller的具体动作需要使用具体的注解，这样生成的api才不会各个类型的动作都重复一遍。例如：



多视图支持

目前支持thymeleaf模板和jsp模板页面



控制器根据返回的页面请求，先去验证是否存在html页面，存在则返回不存在则返回jsp

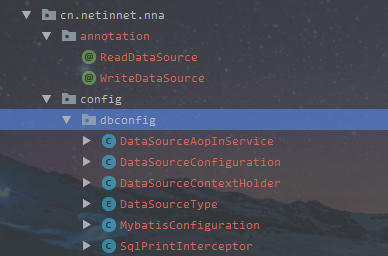
因此html和jsp页面不能同名

配置数据源读写分离

为了缓解大量的并发访问，利用MySQL主从配置，实现读写分离，减轻数据库压力。

原理：主服务器（Master）负责网站NonQuery操作，从服务器负责Query操作，用户可以根据网站功能模特性块固定访问Slave服务器，或者自己写个池或队列，自由为请求分配从服务器连接。

配置：



主要配置文件在config目录下的dbconfig目录里，测试案例参照controller层的TestReadAndWrite类，将DataSourceAopInService、DataSourceConfiguration、

\* MybatisConfiguration的注解去掉注释

访问http://localhost:8080/mySpringBoot/swagger-ui.html进行测试

可以现在本地同一台服务器创建多个数据库进行测试。（测试数据库见doc目录下的test\_write、test\_read01、test\_read02）

常见报错：

org.springframework.web.HttpRequestMethodNotSupportedException: Request method 'GET' not supported

原因是非法使用@PostMapping，应该使用@GetMapping

Get和Post请求的区别，贴在这里

1. 哪一些情况下，浏览器会发送get请求

a. 直接在浏览器地址栏输入某个地址

b. 点击链接

c. 表单默认的提交方式

2. 哪一些情况下，浏览器会发送post请求？

a. 设置表单method = "post"