**Spring Boot 高级编程**

# 常用地址

* 常用工具
  + JDK1.8：https://pan.baidu.com/s/1Oy-7ySjjsxt2vdfDAOpDwg
  + Tomcat8.0：https://pan.baidu.com/s/1W3RtsdQ0GJbZwTsV-nkxCQ
  + MySql5.5：https://pan.baidu.com/s/1YQ8iT7J30oNCeon5iheQ4g
  + 开发工具：https://pan.baidu.com/s/104yMHaujSAUC-MYXca6vmg
  + HBuilder：https://pan.baidu.com/s/1xjdeos-G1dtHmTQnbCx0xQ
  + PD：https://pan.baidu.com/s/1VhvjSfx-pHYvwkTXY4GZqw
  + EditPlus：https://pan.baidu.com/s/1w-iArkuTny5HE8H6JWH7Bw
  + 亿图：https://pan.baidu.com/s/1uscCuRKrr5v2YrV97b5MlA
* 共享目录：https://pan.baidu.com/s/1IbRfdQkPFTPftgJc-\_ZNog
* maven中央库：https://mvnrepository.com/
* spring boot文档： https://docs.spring.io/spring-boot/docs/2.0.0.RC1/reference/htmlsingle/
* toYml： https://www.toyaml.com/index.html

# Spring Boot基础介绍

## 1.1 什么是Spring Boot（特性、核心）

* spring boot并不是新的技术框架，像maven整合了jar一样，它整合了主流的第三方框架，形成的一套快速开发框架；
* 核心

1. 快速搭建项目；
2. Starter依赖，将常用的依赖按组聚集在一起，形成单条依赖，能够非常方便地进行包管理（maven继承的方式）；
3. 遵循“习惯优于配置”的原则，简化配置，无web.xml；
4. Bean自动配置，完全采用注解化，对jar包对象的引用，无需再配置bean对象，可直接注入；
5. 内嵌servlet容器，无需外部tomcat；
6. 提供运行时的应用监控；

## 1.2 为什么要用Spring Boot

* 能够使配置、编码、部署变得简单；

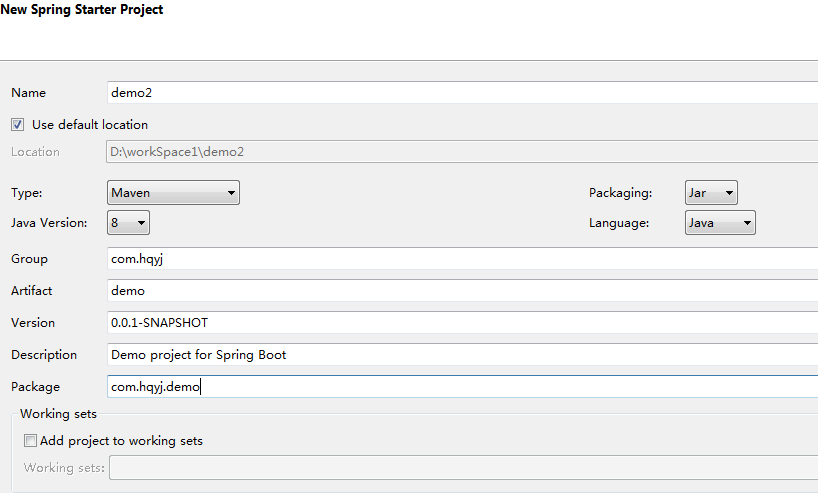
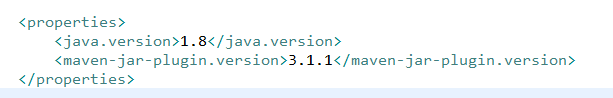
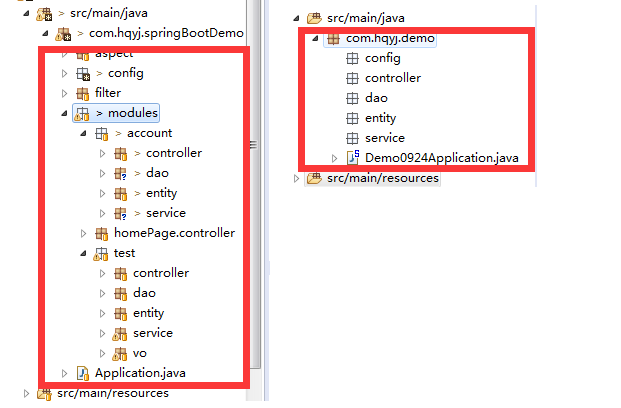
## 1.3 mvc、boot、cloud区别与联系

* spring mvc：web应用中的mvc框架；
* spring boot：垂直应用框架的大集成，web组件基于mvc；
* spring cloud：分布式应用框架，包含注册中心、服务治理、网关中心、断路器……基于spring boot快速开发单个微服务，boot + cloud = 分布式开发；

## 1.4 核心注解

* @SpringBootApplication ---- 启动spring boot工程注解；
* @Configuration ---- 配置类注解，相当于<Beans></Beans>；
* @PropertySource(value={"classpath:config/applicationTest.properties"},ignoreResourceNotFound=true) ---- 加载指定属性文件;
* @Component ---- 注册为 Spring bean，注解表明一个类会作为组件类，并告知Spring要为这个类创建bean；
* @Bean(name = "test1SqlSessionFactory") ---- 生成bean放入Spring容器，告诉Spring这个方法将会返回一个对象，这个对象要注册为Spring应用上下文中的bean；
* @Resource(name="dynamicDataSource") ---- 按照 name 注入对象；
* @Autowired ---- 按照 type 注入对象；
* @Aspect ---- aop注解;
* @Transactional ---- 事物注解；
* @RestController（@Controller + @ResponseBody） ---- 控制器注解;
  + @RequestMapping(value="/city/save",method=RequestMethod.POST, produces="application/json")
    - consumes ---- 进入方法的数据类型
    - produces ---- 方法返回数据类型
    - 常见类型（application/json、application/x-www-form-urlencoded）
  + @RequestBody ---- 接受 json 数据
  + @ModelAttribute("conciergeItem") ---- 接受 form 数据
* @Service ---- service注解;
* @Repository ---- dao层数据仓库注解;

## 1.5 基础框架搭建

* 创建web工程(eclipse)
  + new ---- project ---- spring start project
    - 
    - 
  + 完成创建后，pom报错，错误信息unknow，有些IDE插件不兼容造成，添加以下插件，再进行maven ---- update project的操作即可
    - 
  + 启动项目；
* 创建简单接口（业务开发）；
  + 注意分包规则
    - 
  + 创建TestController

* + 
  + 访问：http://127.0.0.1:8080/test/info

## 1.6 banner

* 简介：SpringBoot Logo 以及版本信息；
* 自定义banner信息
  + src/main/resources ---- 创建banner.txt文件；
    - http://www.network-science.de/ascii/ ---- 制作banner文件；
    - ${application.version} ---- #这个是MANIFEST.MF文件中的版本号
    - ${application.formatted-version}
    - ${spring-boot.version} ---- #这个是springboot的版本号
    - ${spring-boot.formatted-version}
* 关闭banner ---- 修改启动类

* + 

## 1.7 问题回顾

* 环境问题
  + 相关问题
    - 相关jar无法下载到本地;
    - eclipse maven 插件无法升级等问题;
  + 解决思路
    - 统一eclipse开发环境；
    - 更换全新的工作空间；
    - 重新指定maven setting 文件，更换远程仓库地址；
    - 更换本地仓库地址；
  + 相关方案
    - 指定相关版本

* + - * 
    - 报错：org.apache.maven.archiver.mavenarchiver.getmanifest
      * maven配置文件不是最新的，需要升级maven插件
      * help ---- Install New Software ---- add（任选一个）
        + https://otto.takari.io/content/sites/m2e.extras/m2eclipse-mavenarchiver/0.17.2/N/LATEST
        + http://repo1.maven.org/maven2/.m2e/connectors/m2eclipse-mavenarchiver/0.17.2/N/LATEST/
      * 点击 next 前，将 eclipse Install 窗口右下角的 Contact all update sites during install to find required software 选项；
        + 安装成功后，重启eclipse 然后再右键项目maven update project；

# 配置文件

## 2.1 种类、优先级、读取（类型安全配置）

* 种类：全局配置文件（application.yml or application.properties）、其他配置文件（log……）；
* 优先级 ---- 优先级由高到低，相同位置 application.properties 高于 application.yml；
  + 外置：xx.jar同目录下/config；
  + 外置：xx.jar同目录；
  + 内置：src/main/resources/config；
  + 内置：src/main/resources；
  + ------------------------------测试------------------------
  + application.properties书写第一条配置：server.port=8086
  + 准备application.yml文件，同样的位置、不同的端口配置，启动项目看谁的配置生效；
  + 不同配置的application.properties，分别放在src/main/resources和src/main/resources/config下，启动项目看谁生效；
* 读取
  + 针对application.properties
    - application.properties书写如下配置

* + - * 
    - TestController.java ---- 直接用@Value注解获得属性值

* + - * 
    - 启动项目访问接口：http://127.0.0.1:8086/test/config
  + 针对其他配置文件
    - 在application.properties相同的位置创建applicationTest.properties；

* + - * 
    - 创建配置类ApplicationConfigTestBean.java（modules.test.vo包下，虽然是配置对象，但此处用于展现给页面做测试用，所以放在vo包下），自行添加get、set方法；

* + - * 
    - TestController.java ---- 注入ApplicationConfigTestBean.java，并进行调用；

* + - * 
    - 启动项目访问接口：http://127.0.0.1:8086/test/config
  + 注意事项：key命名不能用com.thornBird驼峰法命名，需用.或者-间隔；

## 2.2 log配置

* 介绍：spring boot使用的是Logback日志系统，特点：高效、实现slf4j接口、多格式支持……
* 创建log配置文件logback-dev.xml；

* + 
* application.properties ---- 指定log配置文件位置，有以下三种，我们采用第三种方式；
  + logging.config=logback-dev.xml ---- D:\workSpace\personalArchives\logback-dev.xml
  + logging.config=${user.home}/logback-dev.xml ---- C:\Users\hujiang\logback-dev.xml
  + logging.config=classpath:config/logback-dev.xml ---- D:\workSpace\personalArchives\target\classes\config\logback-dev.xml
* TestController ---- 添加log测试接口；

* + 
  + 注意：我们提供了三个接口，URL中有公共的部分/test，将公共部分提到类注解上，方法注解上写不同的部分；
* 测试方法：修改logback-dev.xml日志输出级别，重启工程，访问http://127.0.0.1:8086/test/logTest，在控制台和日志文件查看日志是否输出正确；

## 2.3 多环境配置

* 需求：当应用程序需要部署到不同的环境时，一些配置细节会有不同，比如日志级别，数据源等，解决办法：方法一：每次发布的时候修改日志文件；方法二：使用 Spring boot 的多环境配置；
* 步骤
  + 创建不同环境的配置文件 ---- application-{profile}.properties，{profile}为环境标识，我们创建application-dev.properties、application-live.properties两个环境的配置文件，分别配置：

* + - 
  + 创建live环境的log配置文件：logback-live.xml，并将日志输出级别修改为info；
  + application.properties
    - 删除原有的log配置logging.config=classpath:config/logback-dev.xml ；
    - 新增环境配置：

* + - 
  + 测试方法：修改spring.profiles.active=dev，指向不同的环境，重启应用，访问http://127.0.0.1:8086/test/logTest，查看日志是否输出正确；

## 2.4 热部署配置

* pom ---- 添加spring-boot-devtools依赖；

* + 
* application.properties ---- # dev tools
  + # dev tools
  + #热部署生效
  + spring.devtools.restart.enabled=true
  + #设置重启的目录，添加哪个目录的文件需要重启
  + spring.devtools.restart.additional-paths=src/main/java
* 测试方法：启动应用，修改TestController中logTest方法返回内容，查看应用是否自动重启，调用http://127.0.0.1:8086/test/logTest，查看返回结果是否更新；

## 2.5 http && https 配置

* 为什么要使用https
  + http叫超文本传输协议，使用TCP端口80，默认情况下数据是明文传送的，数据可以通过抓包工具捕获到；
  + https叫安全的超文本传输协议，使用TCP端口443，数据会用PKI（公钥基础设施）中的公钥进行加密，这样抓包工具捕获到的数据包也没有办法看包中的内容，安全性大大提高，要解密数据的话就要用到PKI中的私钥；
* 步骤
  + ssl证书：
    - 阿里云购买；
    - keytool -genkey -alias tomcat -keyalg RSA ---- 创建别名为tomcat的一条证书，指定使用RSA算法，生成文件C:\Users\hujiang\.keystore；
    - 注意：如果该目录已经有了.keystore，创建不成功，删除后再次创建；
    - 将生成好的证书放在resources文件夹下；
  + application.properties ---- 做相应配置;

* + - 
  + 在config\web包下，创建web server相关的配置类WebServerConfig；

* + - 
  + 在WebServerConfig中，注册ServletWebServerFactory；
    - TomcatServletWebServerFactory实现，重写postProcessContext方法；
    - 设置安全收集器SecurityCollection，匹配所有URL
    - 设置安全约束SecurityConstraint，加入约束类型为CONFIDENTIAL(秘密的)
    - 注入上下文
    - 如果需要同时支持 http 和 https，还需要添加 http 连接器
  + 在WebServerConfig中，注册Connector；
    - application.properties，我们做如下修改，新增了http对应的端口，并且将http、https的端口分别改为自己默认的端口，这样我们在访问的时候不必输入端口号；

* + - * 
  + WebServerConfig代码如下：

* + - 
  + 启动项目，打印日志
    - Starting ProtocolHandler ["https-jsse-nio-443"]
    - Starting ProtocolHandler ["http-nio-80"]
  + 测试：分别用https://127.0.0.1/test/logTest、http://127.0.0.1/test/logTest访问；

# Spring Boot 企业级开发

## 3.1 引入mybatis

* 简介：持久层框架，支持自定义sql、存储过程和高级映射，源于ibatis，2010年，apache的Ibatis框架停止更新，并移交给了google团队，同时更名为MyBatis，从2010年后Ibatis在没更新过，彻底变成了一个孤儿框架，一个没人维护的框架注定被mybatis拍在沙滩上；
* 使用步骤
  + pom引入jar：
    - mysql-connector-java ---- 注意，在此我们需要指定MySQL版本，MySQL5用的驱动url是com.mysql.jdbc.Driver，MySQL6以后用的是com.mysql.cj.jdbc.Driver；
    - mybatis-spring-boot-starter
    - pagehelper-spring-boot-starter；

* + - 
  + application.properties配置dataSource、hikari、mybatis；

* + - 

## 3.2 定义映射实体

* 常用规则
  + 数据库以下划线连接，java以驼峰法命名；
  + 映射实体放在com.hqyj.springBootDemo.modules.\*.entuty包下；
  + 时间格式化：@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
* 数据库有m\_city、m\_country两个表，分别创建对应的实体bean ---- 自行创建get、set；

* + 

## 3.3 定义数据访问接口（Repository开发）

* 相关说明说明
  + 没有实现类，无需Mapper.xml；
  + @Repository ---- dao仓库注解；
  + @Mapper ---- 通过xml里面的namespace接口地址，生成bean，相当于@Repository+@MapperScane；

* + 

## 3.4 定义service、controller

* service定义规则
  + 在相应的模块包下创建service、serviceImpl包；
  + 创建CountryAndCitiesService、CountryAndCitiesServiceImpl；

* + 

* + 
* 创建CountryAndCitiesController

* + 

## 3.5 动态生成表

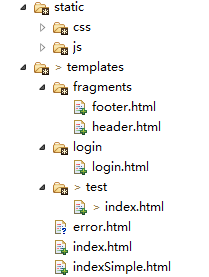
* jpa
  + pom ---- 引入spring-boot-starter-data-jpa；

* + - 
  + application.properties
    - spring.jpa.hibernate.ddl-auto=update ----系统会自动判断哪些表是新建的，哪些字段要修改类型，哪些字段要删除，哪些字段要新增，该操作不会破坏原有数据；
    - spring.jpa.show-sql=true
    - ------------------------------
    - validate：加载时，验证创建数据库表结构；
    - create：每次加载重新创建数据库表结构；
    - create-drop：加载时创建，退出是删除表结构；
  + 创建实实体bean
    - 自行添加get、set

* + - 
  + 启动系统，数据库会生成对应的表；

## 3.6 引入thymeleaf

* thymeleaf简介
  + XML/XHTML/HTML5模版引擎，它提供标准和spring标准两种方言，可以直接套用模板实现JSTL（JSP标准标签库）、OGNL表达式（Struts2默认表达式）效果，可以快速的实现表单绑定、属性编辑器、国际化等功能；
* thymeleaf使用步骤
  + pom引入：spring-boot-starter-thymeleaf；

* + - 
  + application.properties相关配置
    - # 实现页面热部署，生产环境为true
    - spring.thymeleaf.cache=false
    - spring.thymeleaf.enabled=true
    - spring.thymeleaf.encoding=UTF-8
    - spring.thymeleaf.mode=HTML5
    - spring.thymeleaf.prefix=classpath:/templates/
    - spring.thymeleaf.suffix=.html
  + 创建视图、静态文件包（创建项目时候已经生成）
    - SpringBoot 默认模版页面（jsp、html）：classpath:/templates/\*.html
    - SpringBoot 默认静态文件（css、js）：classpath:/static/
  + templates、static分包规则；
    - templates根目录放置index.html（header、footer、container）、indexSimple.html（container） ---- 我们只需要在container中指定template的地址，则可以加载不同的页面；
    - templates/fragments：放置公共模版，header、footer、spinner等；
    - templates/error：放置error页面；
    - templates/welcome：放置welcome页面；
    - templates/login：放置login页面；
    - templates/xx：放置其他业务页面；
    - css：放置css文件；
    - js：放置js文件；
    - 
  + 创建公共页面
    - index.html

* + - * 
    - header.html

* + - * 
    - footer.html

* + - * 
    - resources文件夹下创建error.html
  + 创建公共的controller

## 3.7 页面数据处理

* controller

* + 
* test.html

* + 

## 3.8 过滤器、拦截器

* 简介
  + 过滤器：在javaweb中，对传入的request、response 进行一些处理，滤掉一些信息，或者提前设置一些参数；
  + 拦截器：切面编程，在业务流中增强某些功能，比如日志记录、权限检查、性能监控、Session和cookie处理等；
* 区别
  + 拦截器是基于Java的反射机制的，而过滤器是基于函数回调；
  + 拦截器不依赖与servlet容器，依赖于web框架，在SpringMVC中就是依赖于SpringMVC框架，过滤器依赖于servlet容器；
  + 拦截器只能对action（也就是controller）请求起作用，而过滤器则可以对几乎所有的请求起作用，并且可以对请求的资源进行起作用，但是缺点是一个过滤器实例只能在容器初始化时调用一次；
  + 拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问；
  + 在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次；
  + 拦截器可以获取IOC容器中的各个bean，而过滤器就不行；
  + 拦截器功能更强大些，Filter能做的事情，他都能做，而且可以在请求前，请求后执行，比较灵活；
* 过滤器使用步骤
  + 在基础包下创建filter包，并创建过滤器UrlFilter（使用@WebFilter注解，实现Filter）

* + - 
  + 在config.web包下创建mvc配置类WebMvcConfig，并注册过滤器；

* + - 
* 拦截器使用步骤
  + 基础包下创建子包interceptor，创建自定义拦截器MyInterceptor，实现HandlerInterceptor；

* + - 
  + 配置类 WebMvcConfig，实现 WebMvcConfigurer，重写addInterceptors方法，注册拦截器；

* + - 

## 3.9 文件上传、下载

* test页面书写上传文件form、下载按钮；

* + 
* TestController类书写相应的控制器：下载控制器、单文件上传控制器、多文件上传控制器；

* + 

# Spring Boot实战（用户权限控制系统）

## 4.1敏捷开发 && scrum

* 敏捷开发: 客户需求为核心，高效，迭代增量，大项目分成小项目，分别完成独立运行，控制时间周期，任务优先级，交付功能模块，根据客户反馈持续集成；
* scrum：一种敏捷开发流程；
  + 一个大的需求，预估工作量和时间，比如2个月时间，形成一个sprint，划分成任务列表，比如前端、service 和 存储过程任务等；
  + 计划会议，将任务细化成issue，每个issue的开发周期控制在1到2天，分派到个人头上；
  + 根据每个任务创建父类分支，细化的issue提交到对应的分支上，每日进行standing meeting，工作进度以及遇到的问题，每日做dailymerge，确保dev分支能正常集成编译；
  + 一个sprint完成，进行演示和总结会议，回顾优缺点；

## 4.2 准备工作

* 准备实体bean（User.java、Role.java、Resource.java、RoleResource.java、UserRole.java），自行添加get、set方法；

* + - 
* 准备实体bean对应的dao、service、controller ---- 增删改查方法；
* 准备login页面、error页面、index页面和相应的控制器；
  + error.html ---- main/resources/templates/error.html

* + - 
  + /error ---- spring boot默认定义了该URL，所以我们无需书写controller；
  + login.html ---- main/resources/templates/ account/login.html

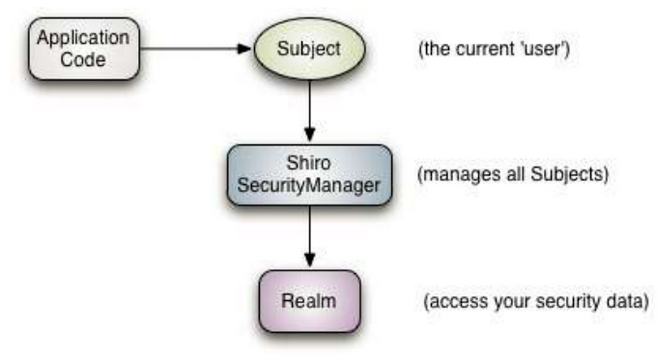
* + - 
  + AccountController .java---- modules.account.controller

* + - 
  + 访问：http://127.0.0.1/login
  + index.html（homePage） ---- main/resources/templates/ homePage/index.html

* + - 
  + HomePageController.java ---- modules.homePage.controller

* + - 
  + 访问：http://127.0.0.1/home

## 4.3 shiro介绍

* shiro 和 security 的区别
  + Spring Security 是一个重量级的安全管理框架；Shiro 则是一个轻量级的安全管理框架；
  + Spring Security 概念复杂，配置繁琐；Shiro 概念简单、配置简单；
  + Spring Security 功能强大；Shiro 功能简单；
* shiro核心组件
  + Subject ---- 是与程序进行交互的对象，可以是人也可以是服务或者其他，通常就理解为用户；
  + SecurityManager ---- Shiro的核心，初始化时协调各个模块运行；
  + Realms ---- Shiro中作为应用程序和安全数据之间的“桥梁”或“连接器”。他获取安全数据来判断subject是否能够登录，subject拥有什么权限。他有点类似DAO。在配置realms时，需要至少一个realm；
  + ShiroFilterFactoryBean ---- 指定路径拦截规则；
* 操作流程
  + 
  + 应用代码通过subject来进行认证和授权，Subject委托给SecurityManager，我们需要给shiro的SecurityManager注入Realm，从而让SecurityManager能得到合法的用户及其权限进行判断；

## 4.4 用户处理

* 引入shiro
  + pom引入jar ---- shiro-spring-boot-web-starter；
  + application.properties
    - shiro.web.enabled=true ----开启shiro
    - shiro.sessionManager.sessionIdCookieEnabled=true ---- 是否允许将sessionId放入cookie
    - shiro.sessionManager.sessionIdUrlRewritingEnabled=true ---- 是否允许将sessionId放入url
    - shiro.unauthorizedUrl=/error ---- 验证失败页面；
    - shiro.successUrl=/index ---- 验证成功页面
    - shiro.loginUrl=/login ---- 登录页面
  + 配置shiro核心组件
    - MyRealm ---- config.web.shiro ---- 核心组件，继承AuthorizingRealm，进行验证操作；

* + - * 
    - ShiroConfig ---- config.web.shiro ---- 配置DefaultWebSecurityManager 和ShiroFilterFactoryBean；

* + - * 
    - AccountController ---- 用户登录控制器

* + - * 
    - 测试
      * 访问 http://127.0.0.1/home ---- 跳转到http://127.0.0.1/login
      * 输入错误用户名和密码，依旧在http://127.0.0.1/home，提示错误信息；
      * 输入正确用户名和密码，跳转到http://127.0.0.1/home

## 4.5 登录权限设置