Springboot学习总结

目录

[基础部分 2](#_Toc518837171)

[简单的springboot项目 2](#_Toc518837172)

[适配springmvc 3](#_Toc518837173)

[集成mybatis 4](#_Toc518837174)

[使用springTest 7](#_Toc518837175)

[不涉及的web端的spring boot test 7](#_Toc518837176)

[设计web端的spring boot test 8](#_Toc518837177)

[springBoot 集成日志 10](#_Toc518837178)

# 基础部分

## 简单的springboot项目

1. 引入父级依赖

<**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>1.5.1.RELEASE</**version**>  
</**parent**>

1. 引入核心依赖

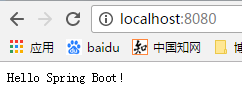
*<!-- spring boot 核心包 -->*<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
</**dependency**>

1. 创建启动入口

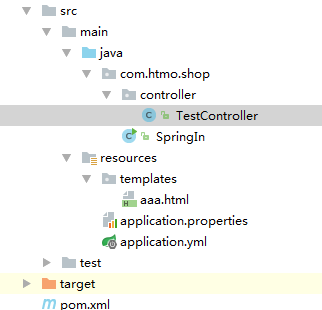
@RestController  
@SpringBootApplication  
**public class** SpringIn {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(SpringIn.**class**,args);  
 }  
  
 @RequestMapping(value = **"/"**,produces = **"text/plain;charset=UTF-8"**)  
 String index(){  
 **return "Hello Spring Boot!"**;  
 }

}

1. 运行截图：



1. 项目结构图



1. 注意事项
2. 启动类必须与项目包名的外层
3. application.yml的空格问题

## 适配springmvc

1. 引入依赖

不引入依赖modelAndView解析不出来对应的html文件，该依赖只适用于html

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

1. 创建模型视图对应：

@Controller  
**public class** TestController {  
*// @RequestMapping("/test")  
// public ModelAndView mvcTest(){  
// return new ModelAndView("aaa");  
// }* @RequestMapping(**"/test2"**)  
 **public** String mvcTest2(){  
 **return "/aaa"**;  
 }  
 @RequestMapping(**"/test"**)  
 **public** ModelAndView mvcTest(){  
 **return new** ModelAndView(**"aaa"**);  
 }  
}

1. Controller与RestController区别

其中注意到上文的请求方式的不同：

其中对于需要返回数据的使用RestController注解

其中对于需要使用springmvc返回一个model需要使用Cotroller注解

其中RestController注解= @Controller +@ResponseBody

## 集成mybatis

1. 首先引入依赖
   1. 其中需要引入数据库驱动
   2. 引入mybatis依赖包

<!-- mysql驱动-->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<!-- 引入spring boot mybatis依赖-->

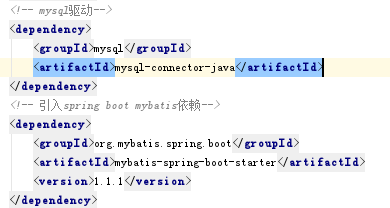
<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

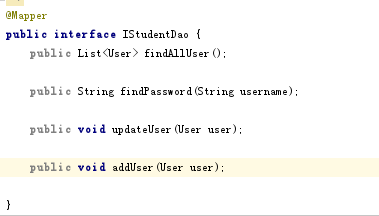
<version>1.1.1</version>

</dependency>



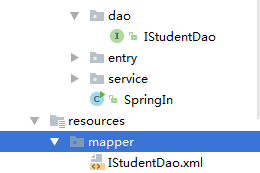
1. 创建dao层接口

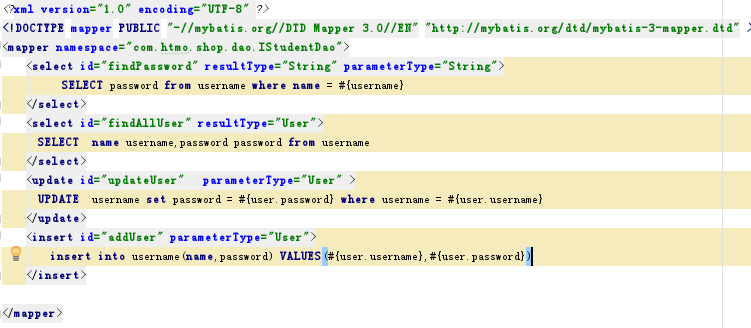
定义的dao层接口：注意dao层方法应遵循与开发手册相关的条例，然后dao层主要对Service层相关，对Service负责。注意在定义的dao层接口上加入Mapper注解这样才会被容器进行扫描，当然也可以不加在主启动类加入MapperScan也是可行的。



1. 创建mapper文件

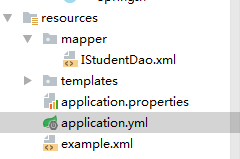
mapper 文件在Resource文件中定义，一般在resource文件中定义一个mapper文件夹，然后在mapper文件夹下定义对应的mapper文件且命名要是对应dao的具体实现。Namespace注意要指定对应的接口完整路径,对于各级参数最好是指定完整路径，但是也可以在主配置文件中采用设置别名的方式进行设置。

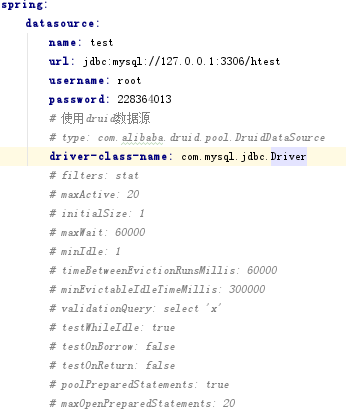


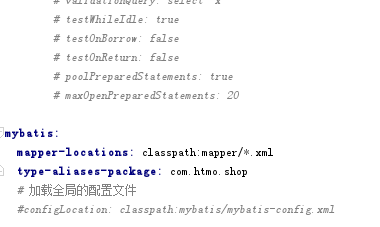


1. 在springboot主配置文件中进行配置

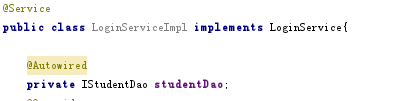
对于主配置文件的配置 与 配置连接池问题后面再进行研究







1. 在Service层中进行引用



## 使用springTest

### 不涉及的web端的spring boot test

1. 引入依赖

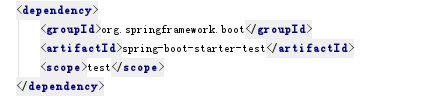
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

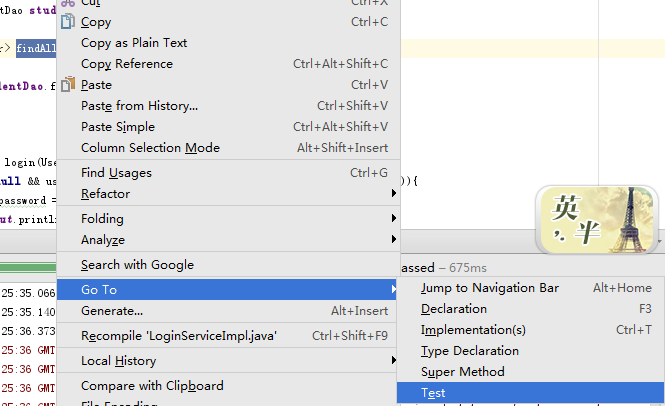
<scope>test</scope>

</dependency>



1. 创建对应Service的测试类

选中需要测试的方法，然后利用idea中的插件生成对应的测试类。



1. 进入到对应的测试类中进行设置

其中SpringBootTest注解表示测试类，springbootTest注解可以设置文件主配置文件参数，测试类优先获取test/resources里的配置文件其次是main里面的。@SpringBootTest(classes = {TestBeanConfiguration.class,Cat.class})使用测试bean。

@SpringBootTest(properties = {"app.version=1.0"}) 设置系统参数。

@RunWith注解表示JUnit将调用它所引用的类来运行该类中的测试而不是开发者去在junit内部去构建它。

SpringRunner.class提供了一些spring测试环境上下文去规范JUnit测试，意味着TestContextManager和支持相关的类和注解。

提供了以下一些特性：

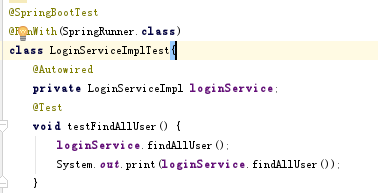
当没有特定的ContextConfiguration#loader()（@ContextConfiguration(loader=...)）被定义那么就是SpringBootContextLoader作为默认的ContextLoader。

自动搜索到SpringBootConfiguration注解的文件。

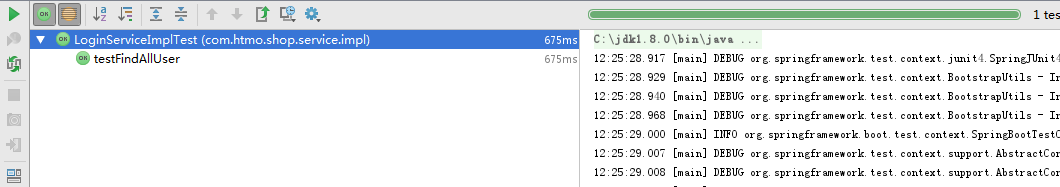
允许自动注入Environment类读取配置文件。

提供一个webEnvironment环境，可以完整的允许一个web环境使用随机的端口或者自定义的端口。

注册了TestRestTemplate类可以去做接口调用。



1. 运行测试结果

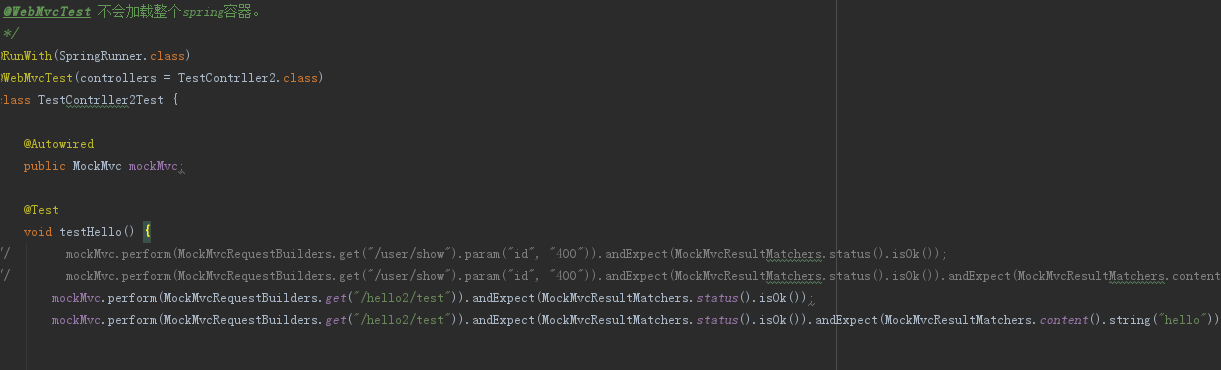


### 涉及web端的spring boot test

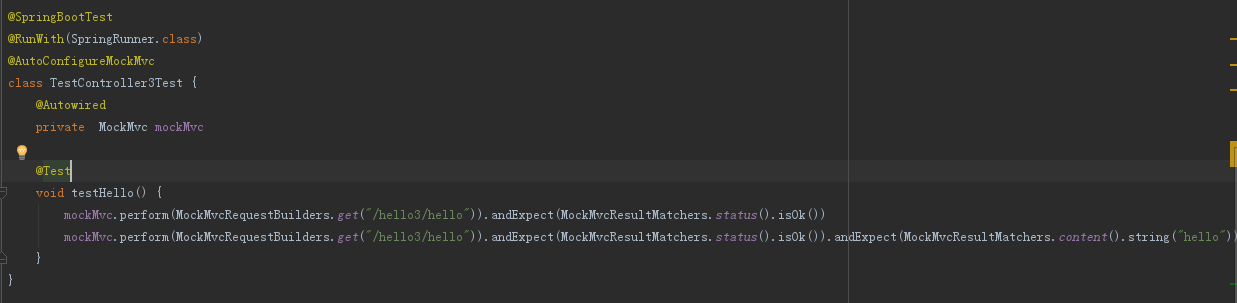
1. 使用restTemplate方式实现对controller层的验证



1. 使用@WebMvcTest注解进行测试



1. 使用@SpringBootTest()与@AutoConfigureMockMvc结合，@SpringBootTest使用@SpringBootTest加载测试的spring上下文环境，@AutoConfigureMockMvc自动配置MockMv



## springBoot 集成日志

对于SpringBoot日志，共分TRACE < DEBUG < INFO < WARN < ERROR < FATAL六个等级。如果设置为 WARN ，则低于 WARN 的信息都不会输出。 Spring Boot中默认配置ERROR、WARN和INFO级别的日志输出到控制台。其中在application.yml文件中设置debug: true会输出debug级别的日志在控制台。

### 简单的日志配置

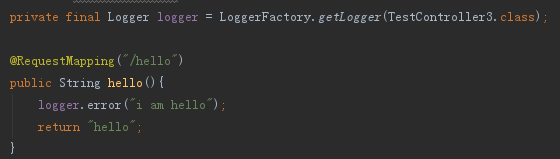
1. 引入依赖

其中依赖如下，但是该依赖包含在thymeleaf中故在引入thymeleaf后可以不显示引入

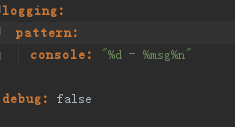
<!-- 集成日志 -->  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-logging</artifactId>  
</dependency>

1. 在类中使用Logger类

其中getLogger 中参数必须为本类的反射类

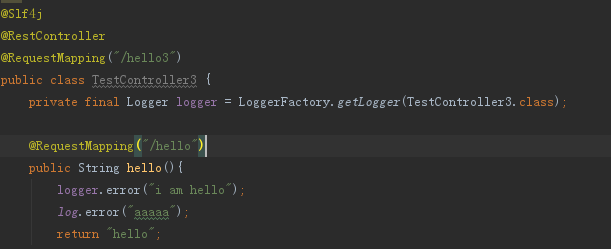


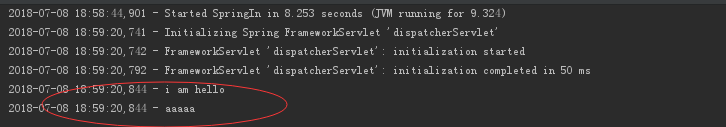
1. 在application.yml 中配置输出



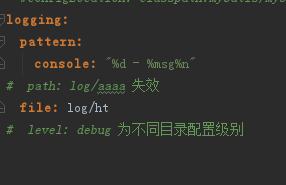
### 使用lombok简化日志

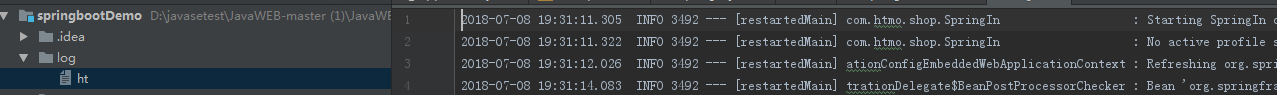
与上文实现的唯一区别是在实现类中用一个注解，取代了获取logger实例的过程。





### 使用logging.path与logging.file持久化日志





## 使用spring boot 集成spring security