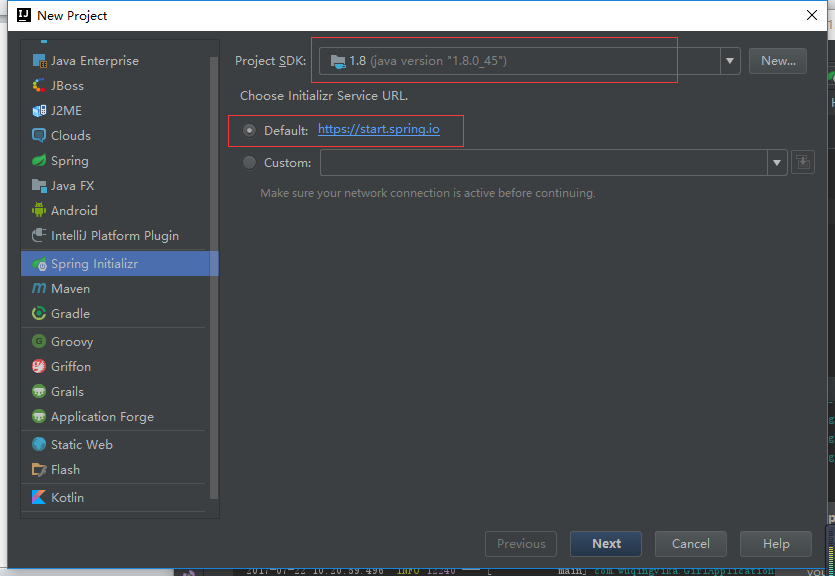
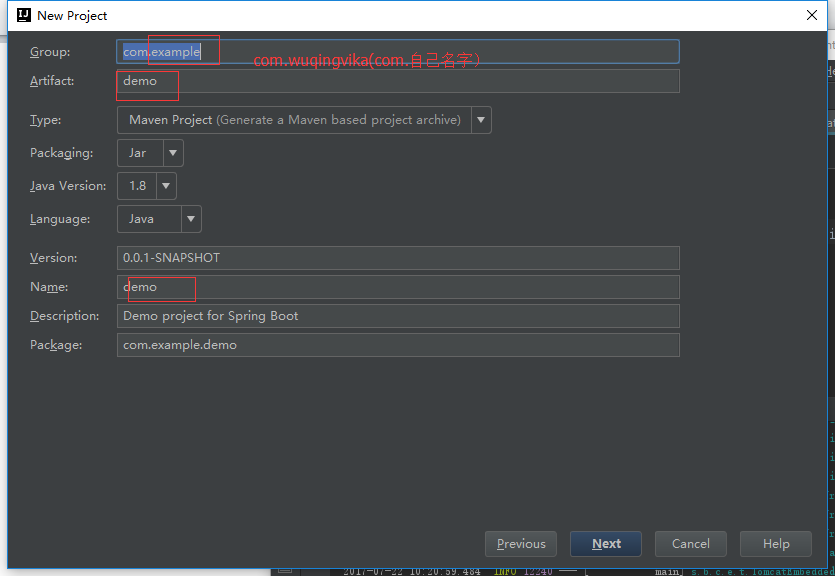
# 2017年7月22日星期六

## 1、初遇SpringBoot

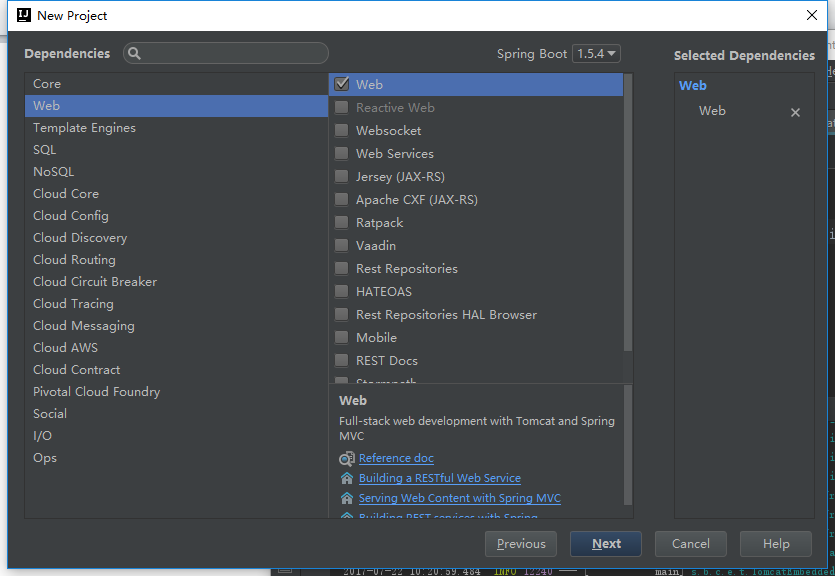
### 1.1 第一个SpringBoot应用



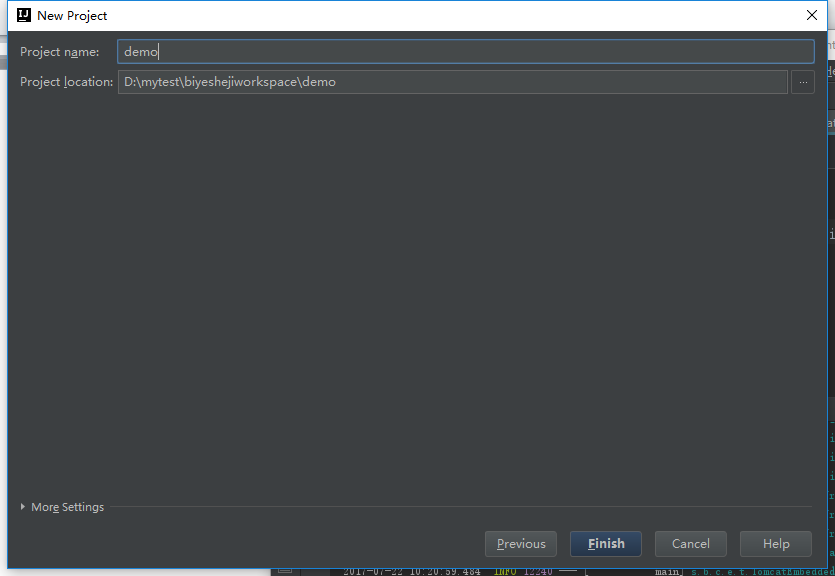
下一步：



然后 artifact随便 输入一个项目名，如girl..下面的name会自动变为girl.然后下一步吧。

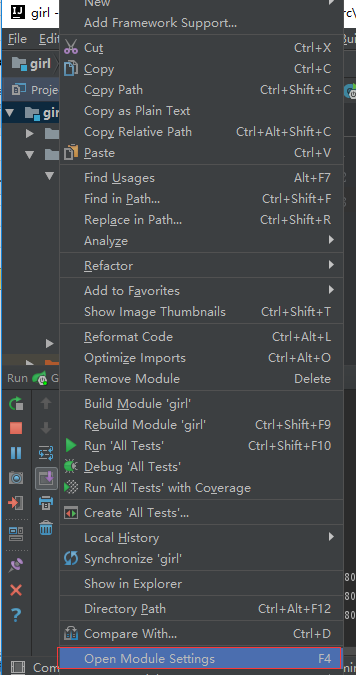


再下一步。

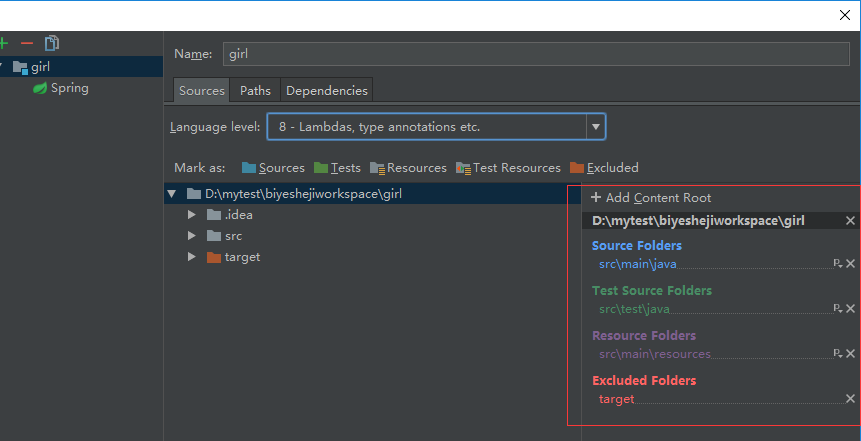


Finish。

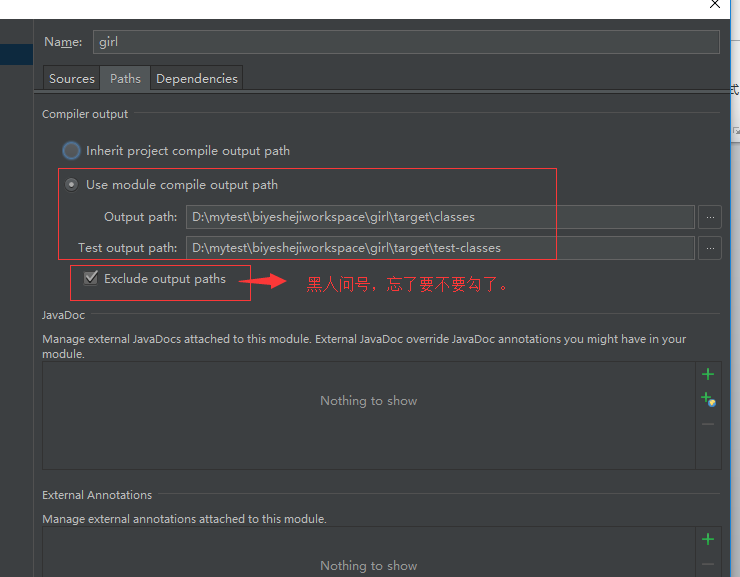
再右击你建的项目open module settings,如下：



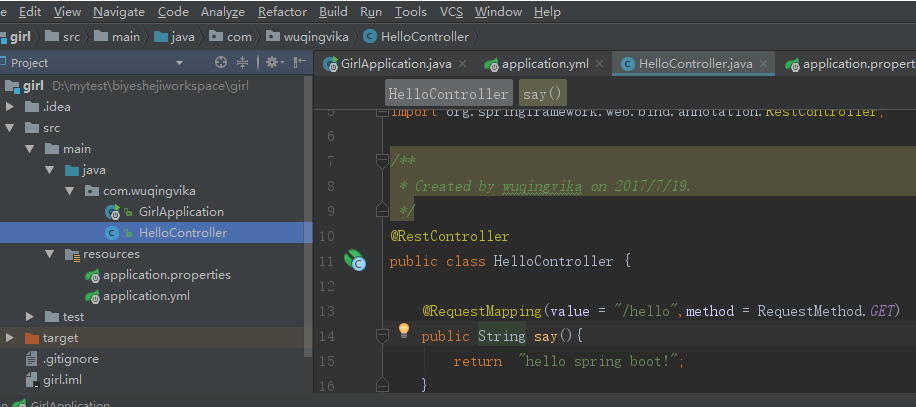
设置src(不设置不好右击run运行)



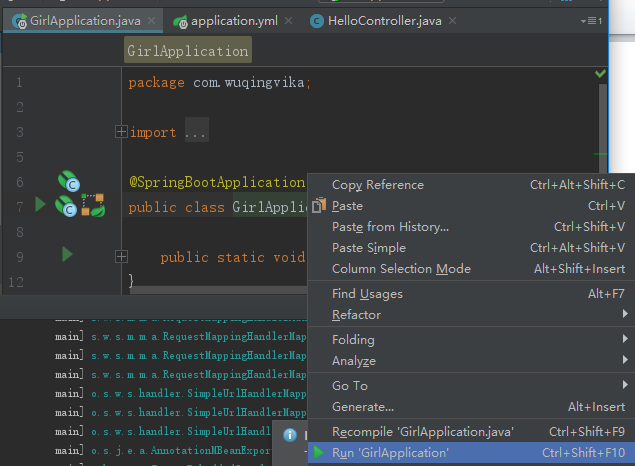
再点击Paths,设置



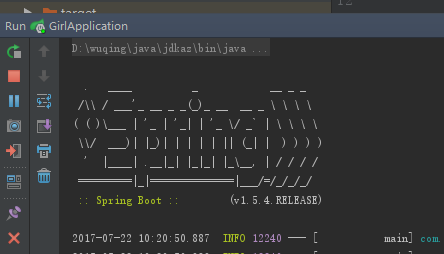
然后 添加 一个controller文件，



最后



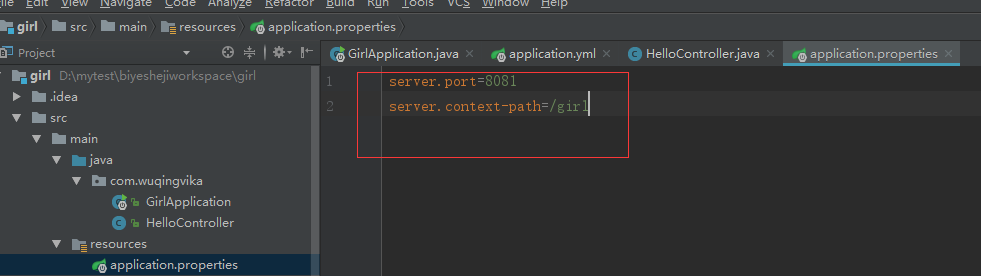
结果：



然后 浏览器访问http://localhost:8080/hello就行了

### 1.2 项目属性配置

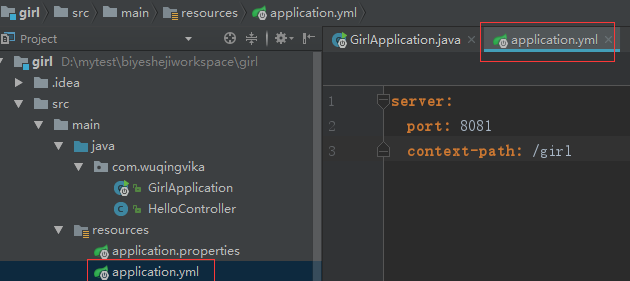
#### 1.2.1 配置端口



然后重新启动， 浏览器访问http://localhost:8081/girl/hello就行了

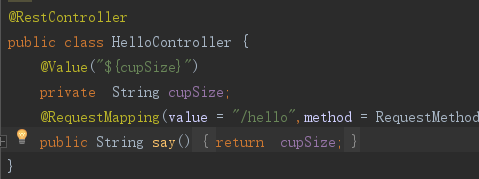
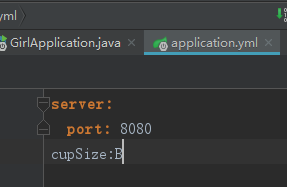
#### 1.2.2 简化版配置文件(比较常用也推荐使用)

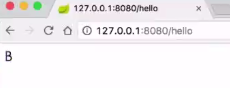
创建application.yml,配置如下 相同前缀可用Tab错开，注意冒号之后留有一个空格。然后删掉application.properties文件。



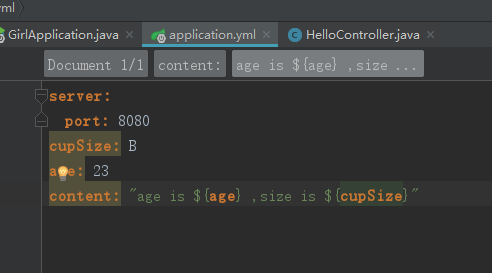
#### 1.2.3 配置自定义属性

（1）cupSize为自定义属性。



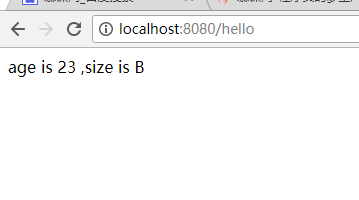


#### 1.2.4 自定义多个属性

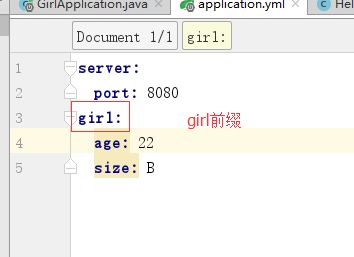


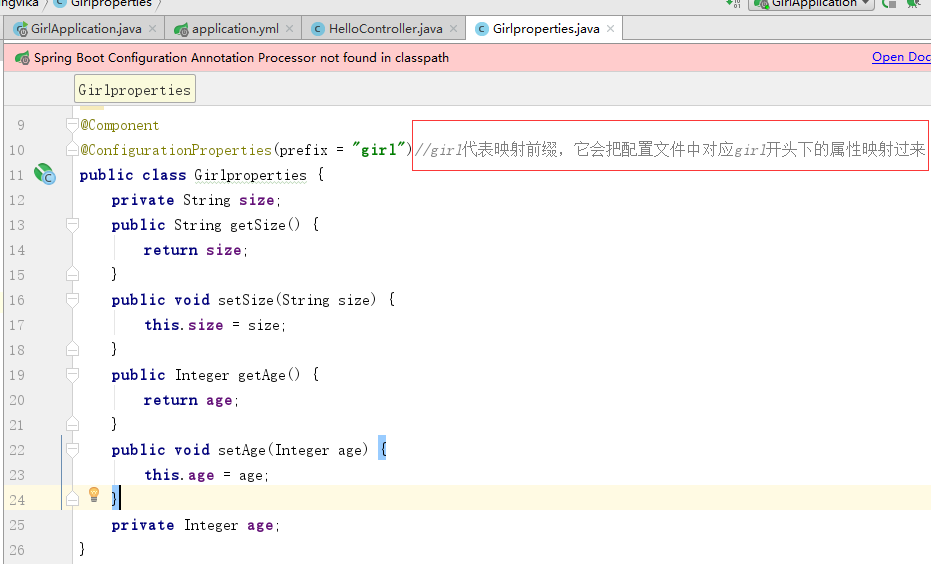


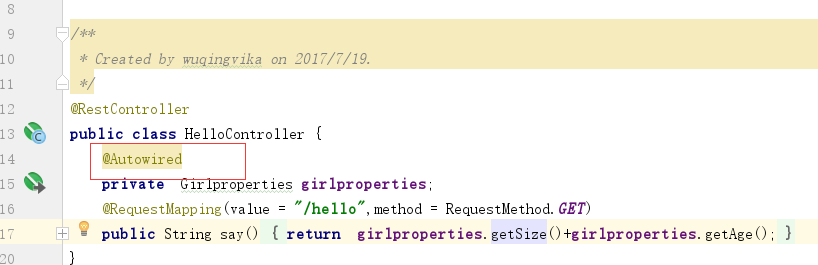
访问结果：



#### 1.2.5 封装类

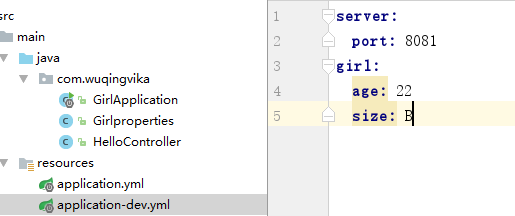


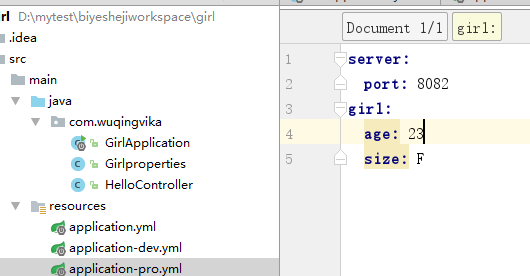




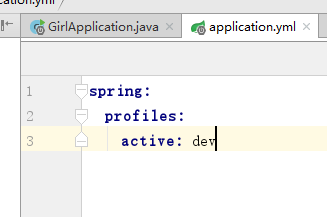
#### 1.2.6 切换配置文件

1.如果说开发环境是尺寸为B，端口为8081，生产环境为F端口为8082。

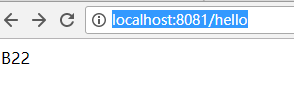




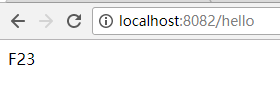
那么怎么在两个环境中切换呢，只需要将application.yml定义激活为对应的值就好。



如上已经配置为开发环境，则在浏览器中访问对应端口结果如下



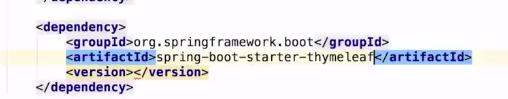
相反如果为生产环境，则结果为：



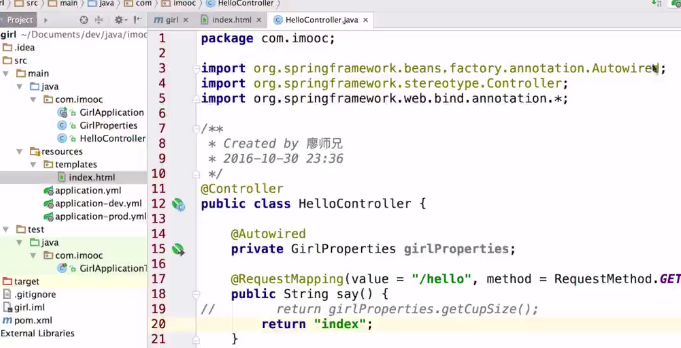
### 1.3 Controller的使用

#### 1.3.1 Controller

1 添加组件



在resource下添加一个templates包。在这个目录下，新建一个index.html.



RestController等同于controller+ResponseBody



#### 1.3.2 URL映射

请求可以为hello或hi



也可以在类上面写，访问时只需要拼接就行(/hello/say)



#### 1.3.3 设置请求参数



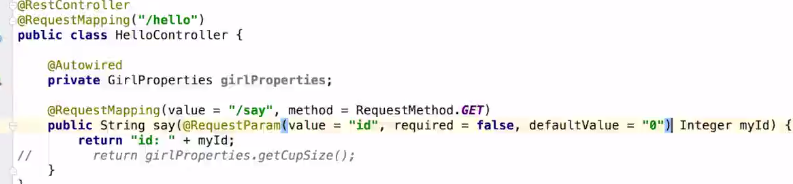
(1)pathvariable



(2)RequestParam 这时只要引号内Id取名一样就行



扩展（是否为必传：false，默认值为0）

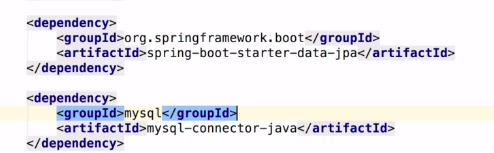


简化RequestMapping,有GetMapping(推荐),还有PostMapping等组合注解。

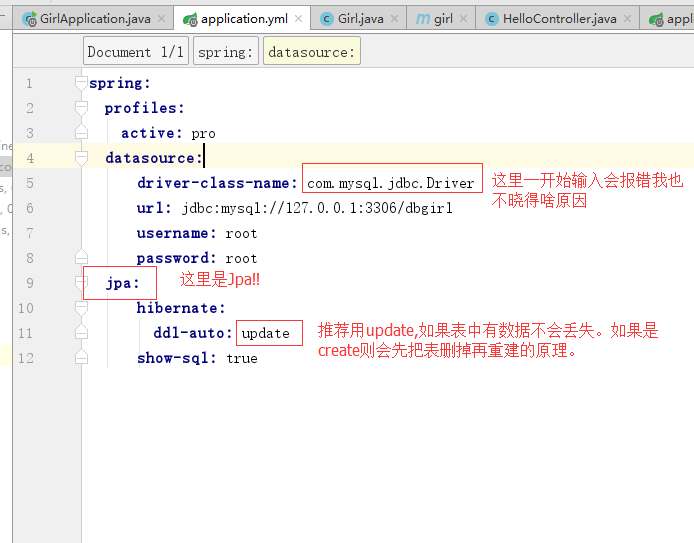


### 1.4 数据库操作

#### 1.4.1 添加组件



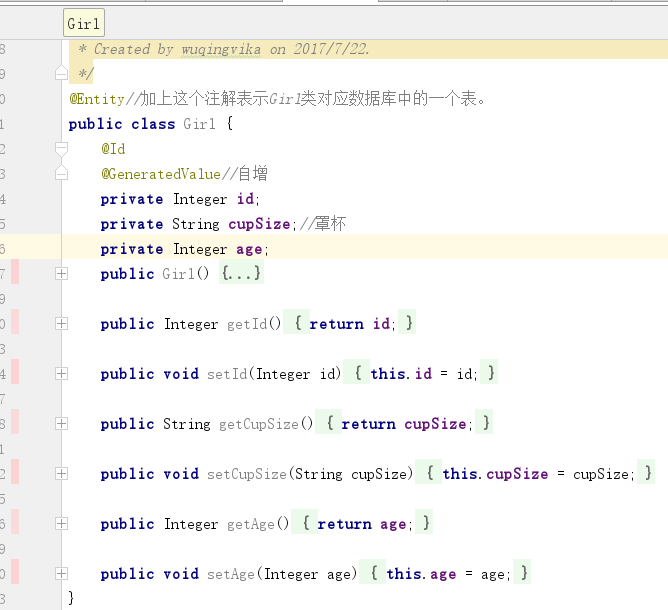
#### 1.4.2 配置文件



注意，数据库先得建立好。

#### 1.4.3 配置实体类

记得添加构造方法，以及get/setter不然报错

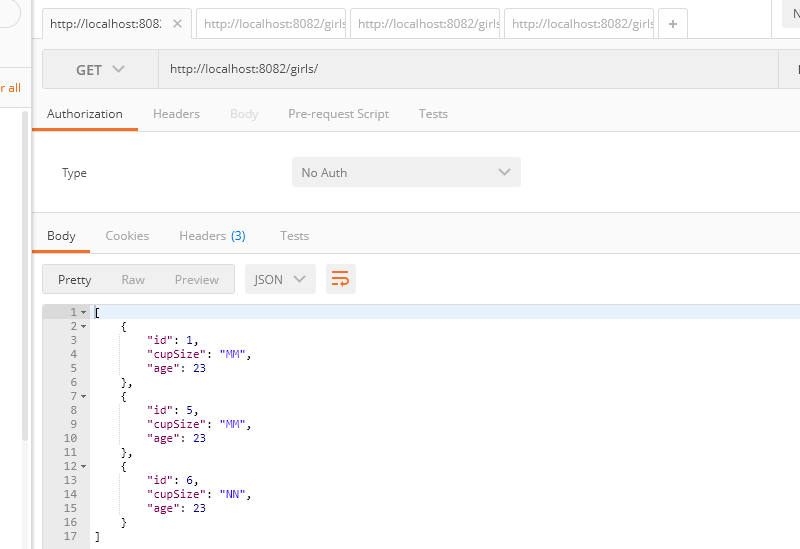




#### 1.4.4 获取女生列表

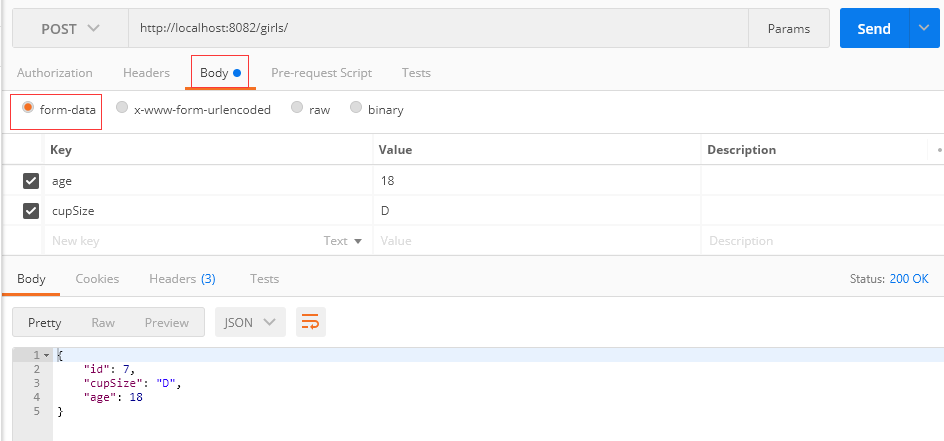


结果



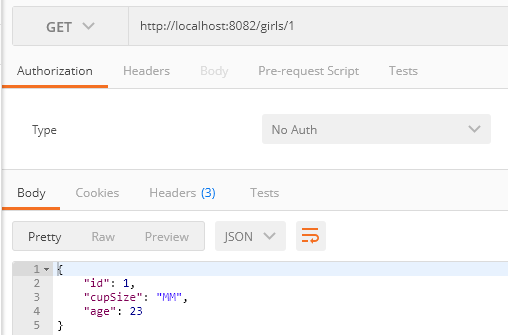
#### 1.4.5 创建一个女生

@PostMapping(value = **"/girls"**)  
**public** Girl girlAdd(@RequestParam(**"cupSize"**) String cupSize,  
 @RequestParam(**"age"**) Integer age){  
 Girl girl=**new** Girl();  
 girl.setAge(age);  
 girl.setCupSize(cupSize);  
 **return girlRepository**.save(girl);  
}



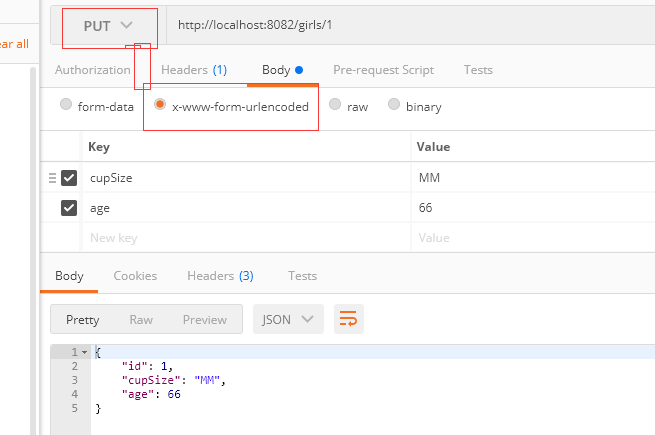
#### 1.4.6 通过id查询一个女生

@GetMapping(value = **"/girls/{id}"**)  
**public** Girl getGirlById(@PathVariable(**"id"**) Integer id){  
 **return girlRepository**.findOne(id);  
}



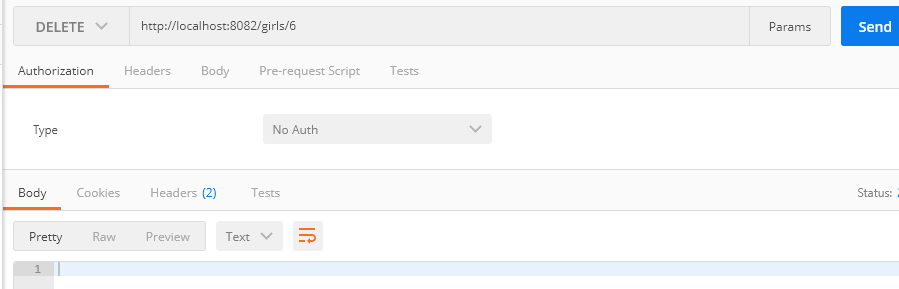
#### 1.4.7 通过id更新一个女生

@PutMapping(value = **"/girls/{id}"**)  
**public** Girl girlUpdate(@PathVariable(**"id"**) Integer id,@RequestParam(**"cupSize"**) String cupSize,  
 @RequestParam(**"age"**) Integer age){  
 Girl girl=**new** Girl();  
 girl.setId(id);  
 girl.setAge(age);  
 girl.setCupSize(cupSize);  
 **return girlRepository**.save(girl);  
}

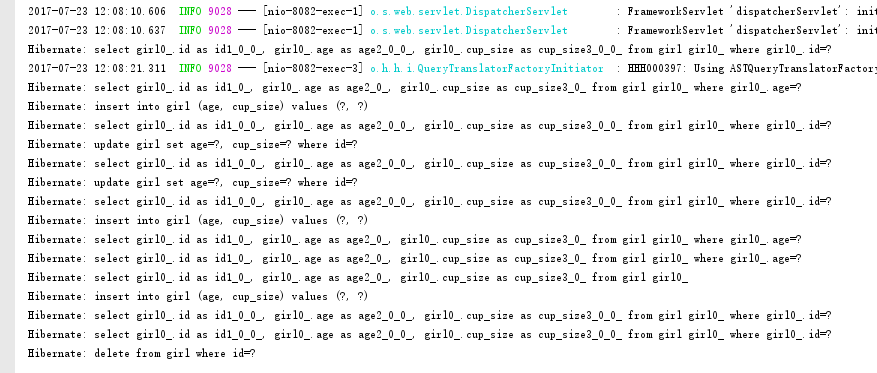


#### 1.4.8 通过id删除一个女生

@DeleteMapping(value = **"/girls/{id}"**)  
**public void** girlDel(@PathVariable(**"id"**) Integer id){  
 **girlRepository**.delete(id);  
}



控制台输入的sql



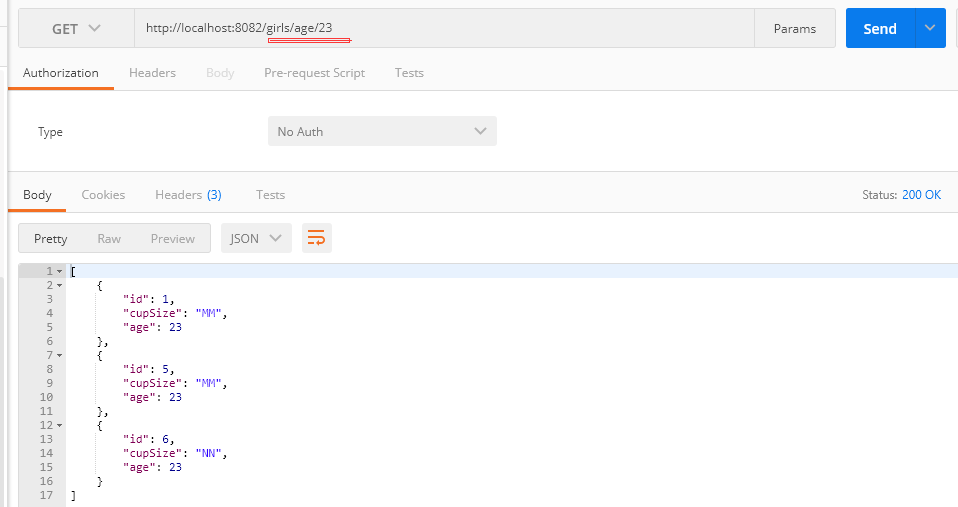
#### 1.4.9 通过age获取女生列表

@GetMapping(value = **"/girls/age/{age}"**)  
**public** List<Girl> findByAge(@PathVariable(**"age"**) Integer age){  
 **return girlRepository**.findByAge(age);  
}

注意扩展方法的命名规则，详情见图：

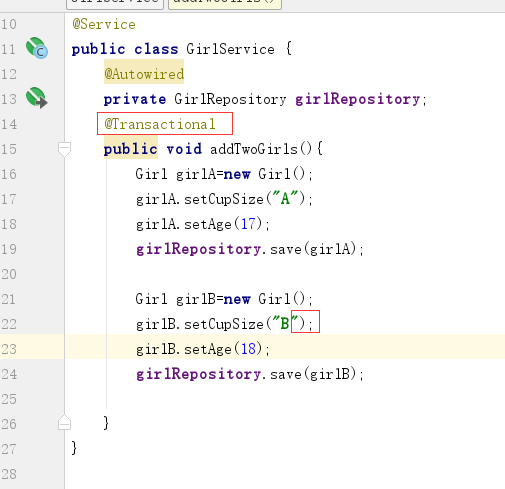


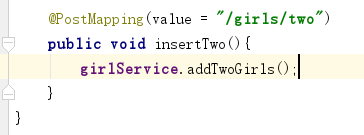
结果如下：



### 1.5 事务

添加两个女生，要么一起添加成功，要么一起添加失败。





然后慢慢测试吧。

## 2、PostMan安装使用

1.下载postman，Postman-REST-Client.zip（<http://download.csdn.net/download/u010246789/9528471#comment>）

2.解压后找到文件Postman-REST-Client\_v0.8.4.14.crx

3.将Postman-REST-Client\_v0.8.4.14.crx后缀名改为rar

4.将Postman-REST-Client\_v0.8.4.14.rar解压缩

5.将解压缩后文件夹中\_metadata改为metadata

6.打开chrome更多工具-->扩展程序，点击开发者模式，显示加载已解压的扩展程序

7.点击加载已解压的扩展程序，选中第4步解压缩的文件夹即可

8.点击安装完成后的详细信息，可以创建快捷方式，而后双击即可使用。

具体见 <https://wenku.baidu.com/view/7339bcd9580216fc710afd99.html>

心累搞了好久呢

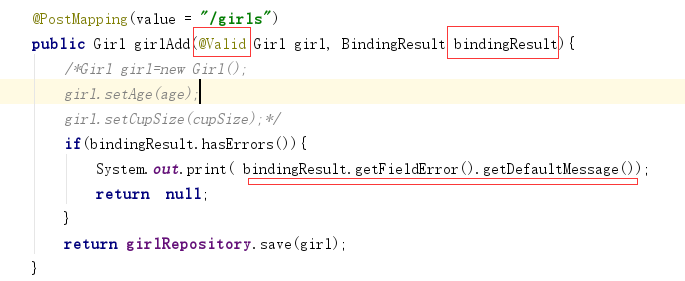
后来直接在google应用里添加应用就ok的。

## 3、SpringBoot Web篇

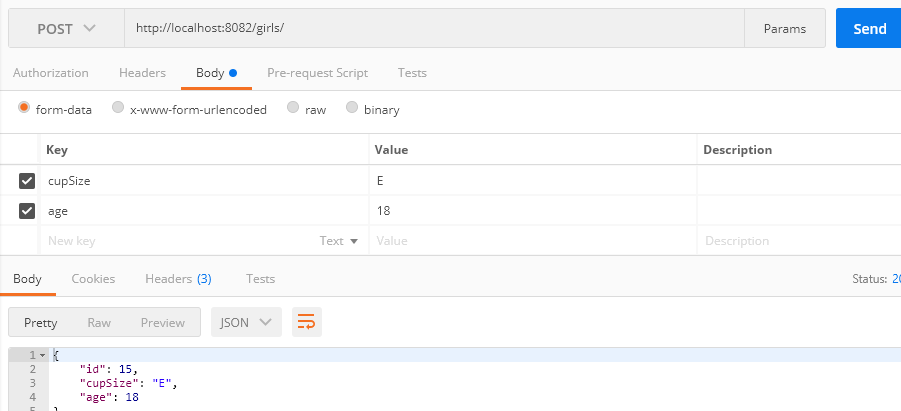
### 3.1 表单验证

验证年龄小于18不能添加

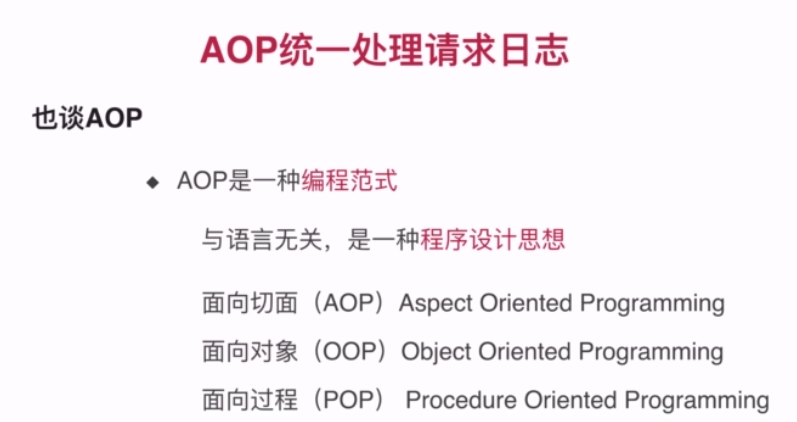




结果（成功）

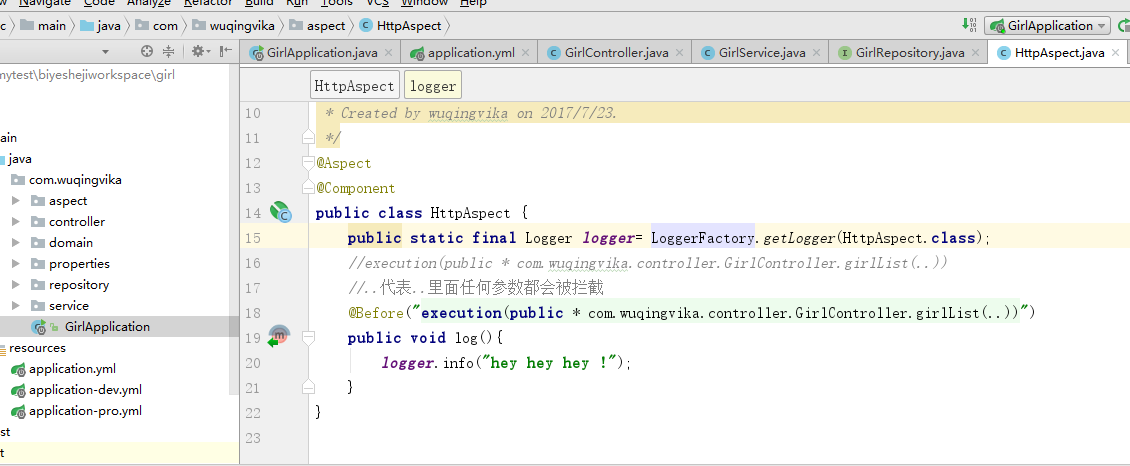


### 3.2 使用Aop处理请求

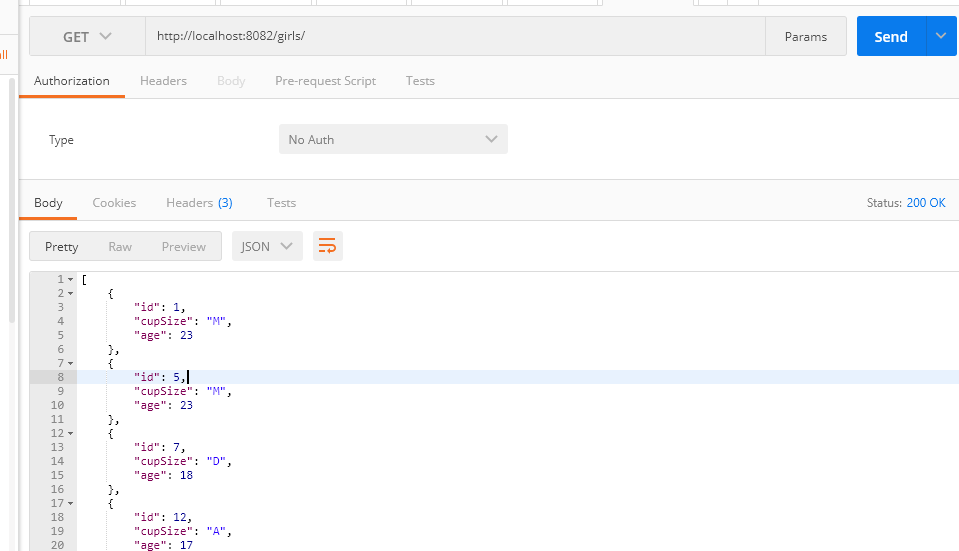


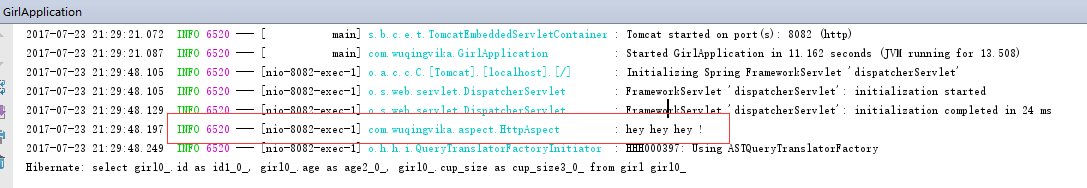


#### 3.2.1 拦截一个方法



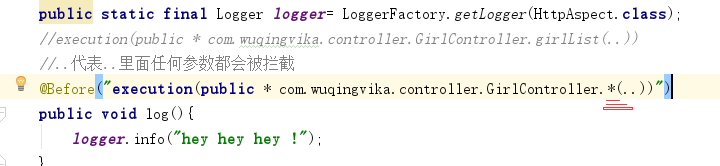


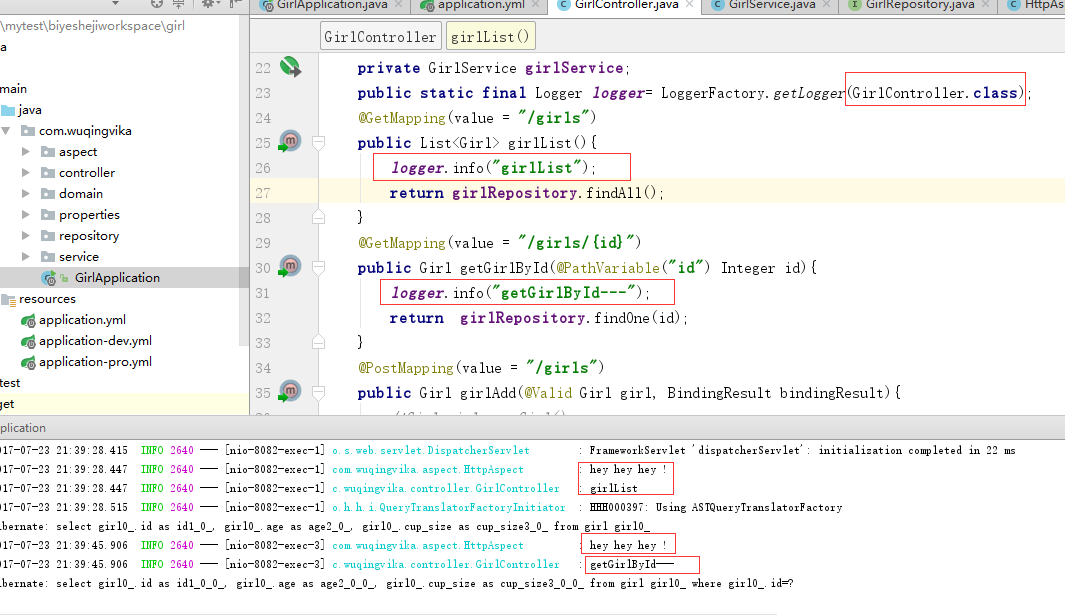




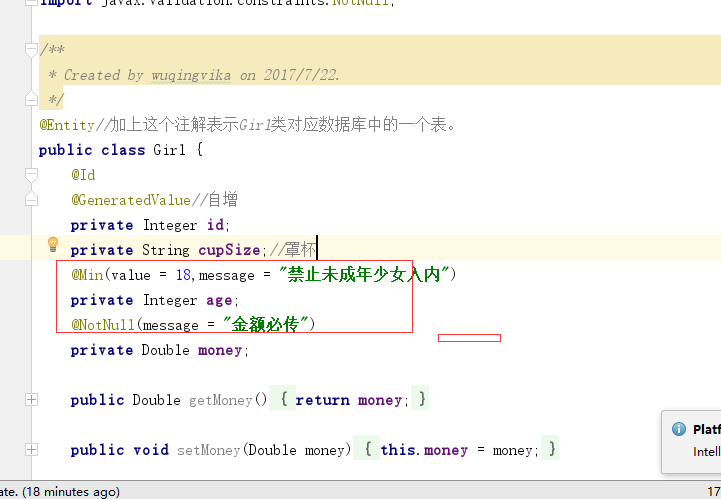
#### 3.2.2 拦截所有方法

拦截GirlController中所有方法。@Before(**"execution(public \* com.wuqingvika.controller.GirlController.\*(..))"**)

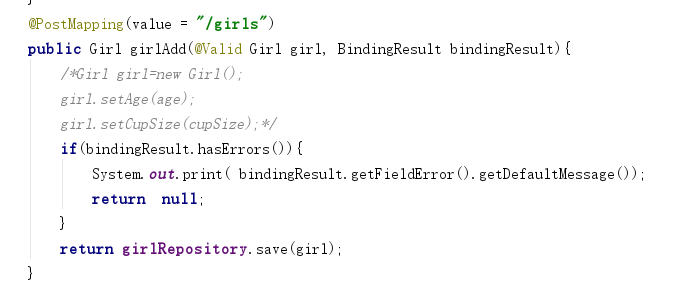




#### 3.2.3表单验证



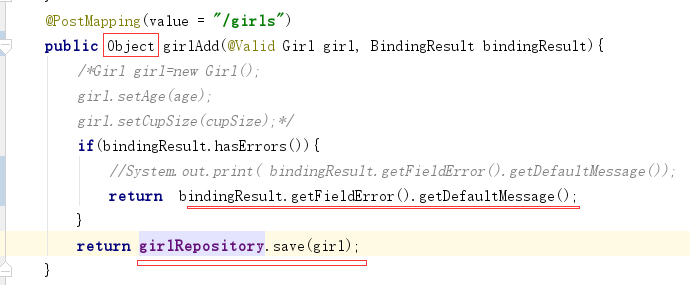
**import** javax.validation.Valid;



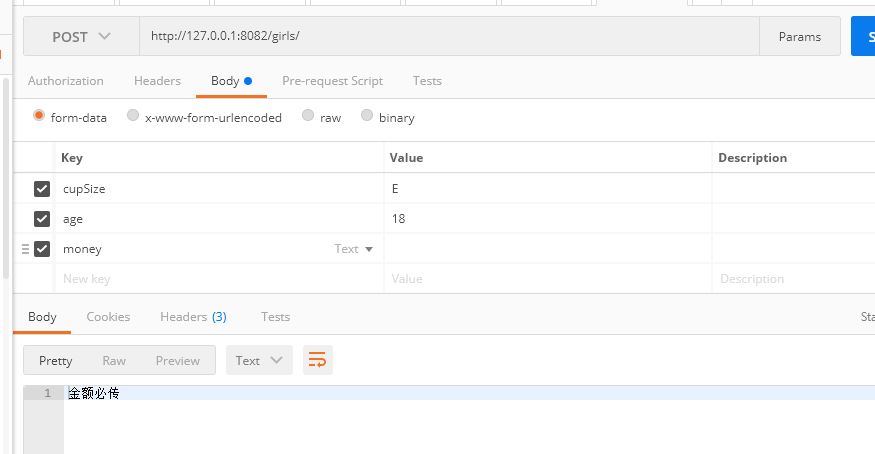
**package** com.wuqingvika.aspect;  
  
**import** org.aspectj.lang.JoinPoint;  
**import** org.aspectj.lang.annotation.\*;  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.stereotype.Component;  
**import** org.springframework.web.context.request.RequestContextHolder;  
**import** org.springframework.web.context.request.ServletRequestAttributes;  
  
**import** javax.servlet.http.HttpServletRequest;  
  
*/\*\*  
 \* Created by wuqingvika on 2017/7/23.  
 \*/*@Aspect  
@Component  
**public class** HttpAspect {  
 **private final static** Logger ***logger***= LoggerFactory.*getLogger*(HttpAspect.**class**);  
 *//execution(public \* com.wuqingvika.controller.GirlController.girlList(..))  
 //..代表..里面任何参数都会被拦截* @Pointcut(**"execution(public \* com.wuqingvika.controller.GirlController.\*(..))"**)  
 **public void** log(){  
  
 }  
 @Before(**"log()"**)  
 **public void** doBefore(JoinPoint joinPoint){  
 ServletRequestAttributes attributes=(ServletRequestAttributes) RequestContextHolder.*getRequestAttributes*();  
 HttpServletRequest request=attributes.getRequest();  
 *//url* ***logger***.info(**"url={}"**,request.getRequestURL());  
 *//method* ***logger***.info(**"method={}"**,request.getMethod());  
 *//ip* ***logger***.info(**"ip={}"**,request.getRemoteAddr());  
 *//类方法* ***logger***.info(**"Class.Method={}"**,joinPoint.getSignature().getDeclaringTypeName()+**"."**+joinPoint.getSignature().getName());  
 *//参数* ***logger***.info(**"args={}"**,joinPoint.getArgs());  
 ***logger***.info(**"hey hey hey !"**);  
 }  
  
 @After(**"log()"**)  
 **public void** doAfter(){  
 ***logger***.info(**"yo yo yo"**);  
 }  
  
 @AfterReturning(returning = **"object"**,pointcut = **"log()"**)  
 **public void** afterReturning(Object object){  
 *//如果有参数返回* ***logger***.info(**"response={}"**,object.toString());  
 }  
  
}

### 3.3 统一异常处理

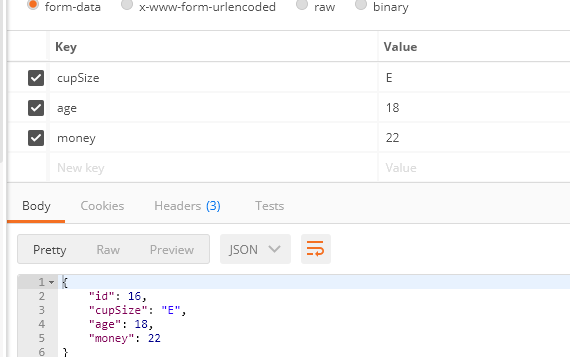
将异常抛到页面。



失败，抛出错误 信息。

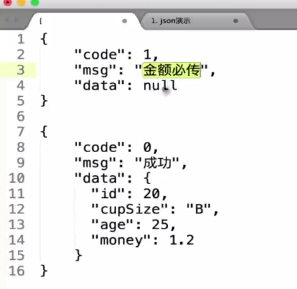


成功，返回对象。

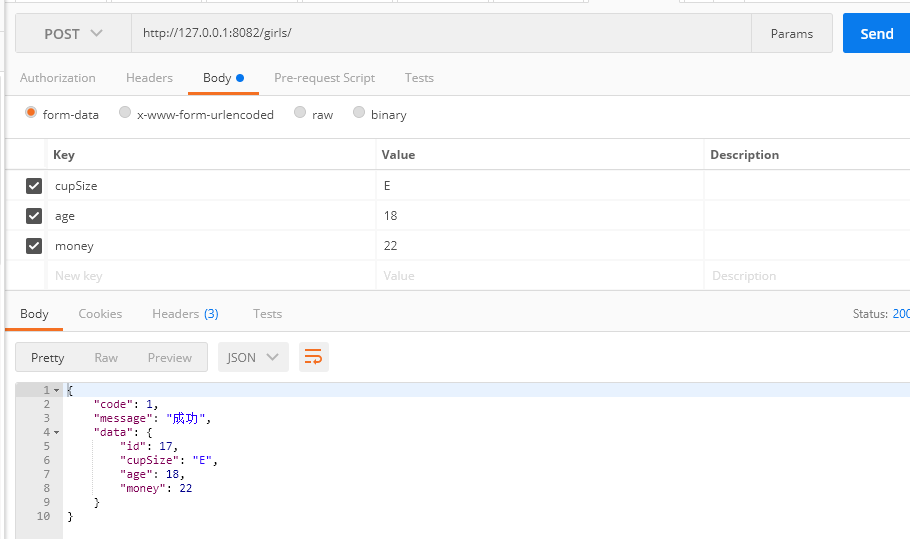


#### 3.3.1 定义Http请求返回最外层对象

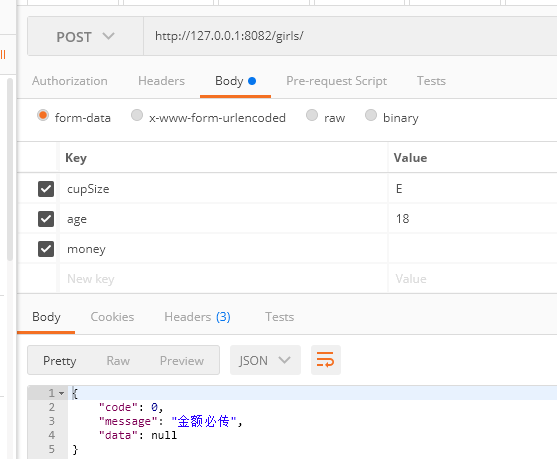
实现效果。



成功返回结果

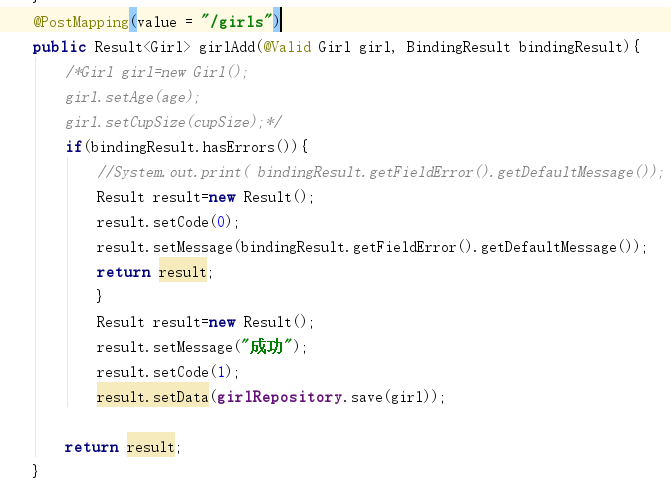


失败结果

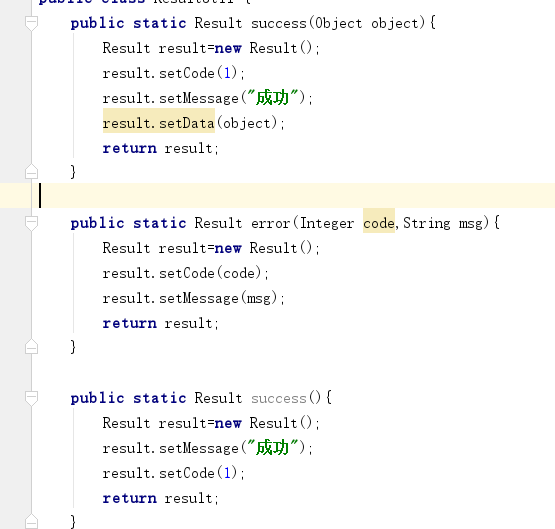


具体实现方法，定义http请求返回最外层对象Result.java

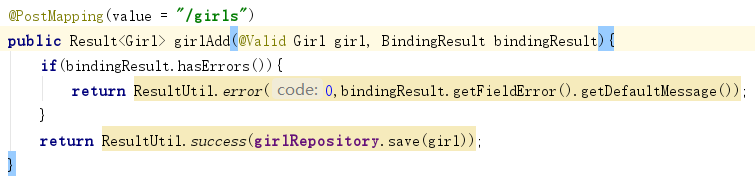
**package** com.wuqingvika.domain;  
**public class** Result<T> {  
 **private** Integer **code**;*//错误码* **private** String **message**;*//提示信息* **private** T **data**;*//具体的内容***注：加上get/set方法。**



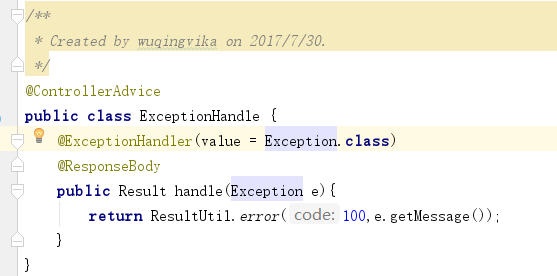
#### 3.3.2 定义工具类ResultUtil



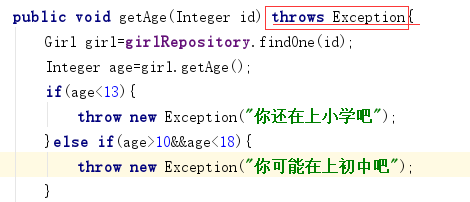
简化后的代码：



#### 3.3.3 捕获异常ExceptionHandle



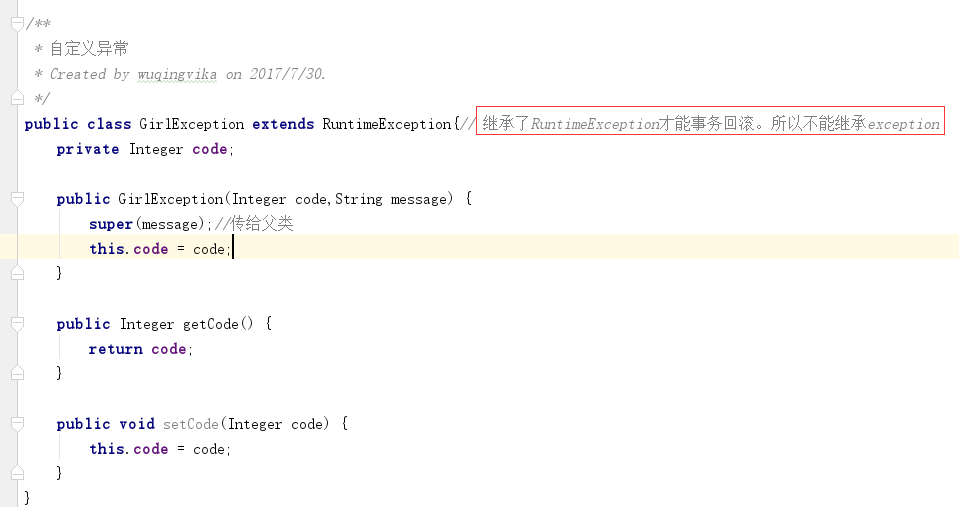
但是呢 这样只会是同一个异常码100。

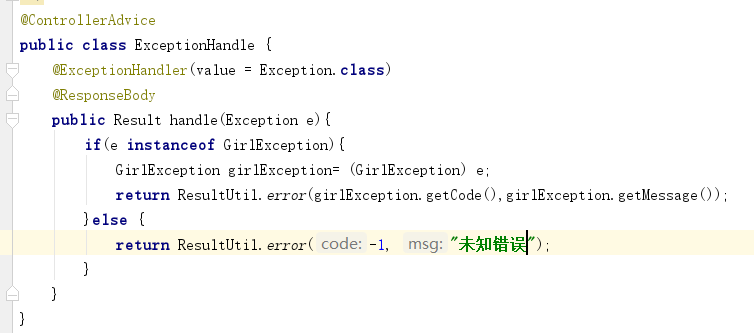


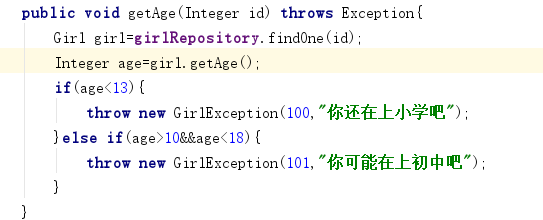
Controller里GetAge()方法同样要throws exception!!

#### 3.3.4 自定义异常GirlException

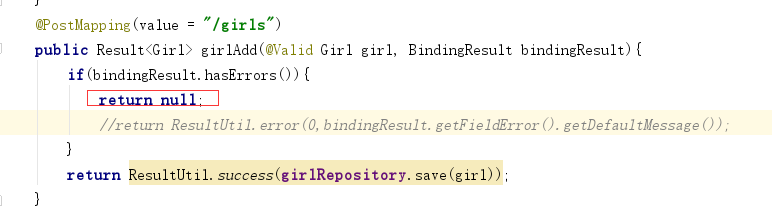
*默认Exception只接收Message一个参数，所以我们这里继承了RuntimeException，同时它也能支持事务回滚。*

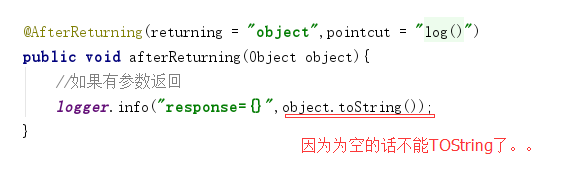


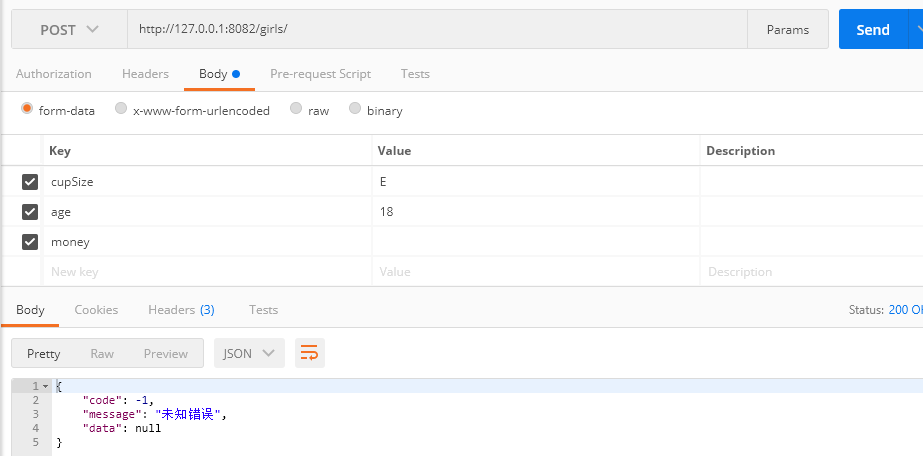




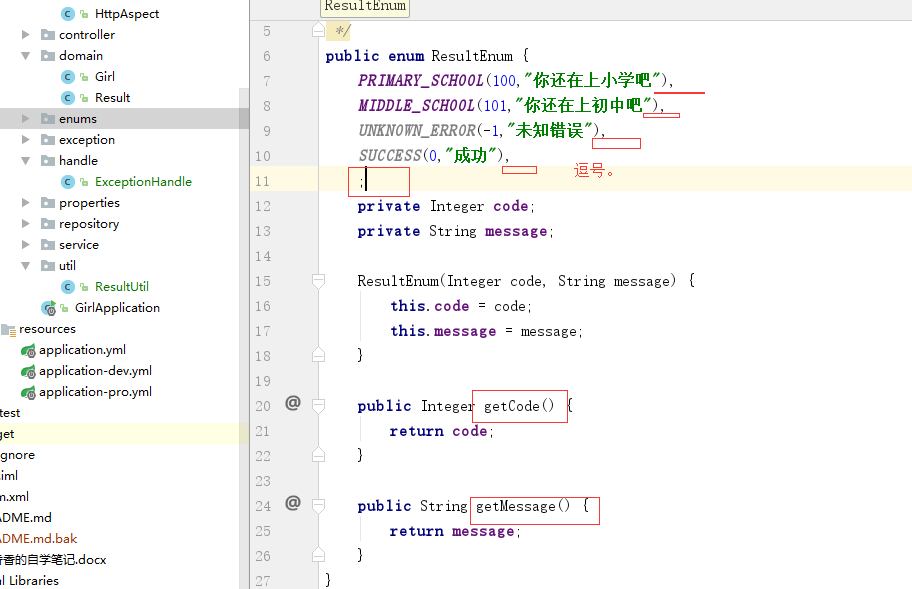
这里100，101结果可以自行测试，未知错误的情况如下：

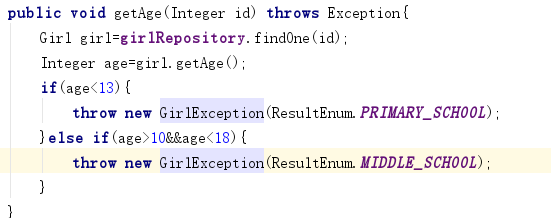




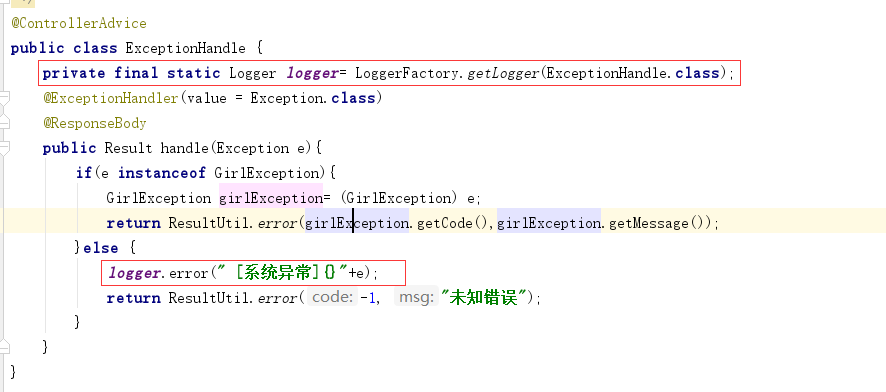


#### 3.3.5 利用枚举维护异常码

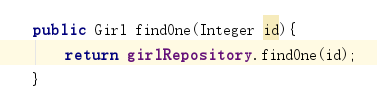






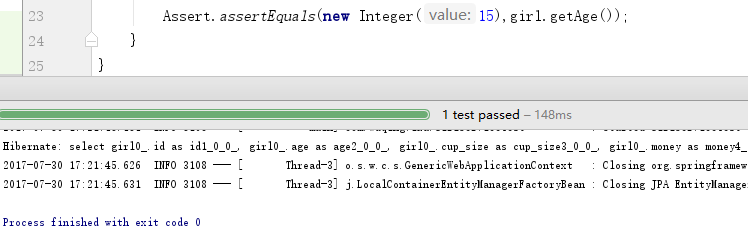


### 3.4 单元测试

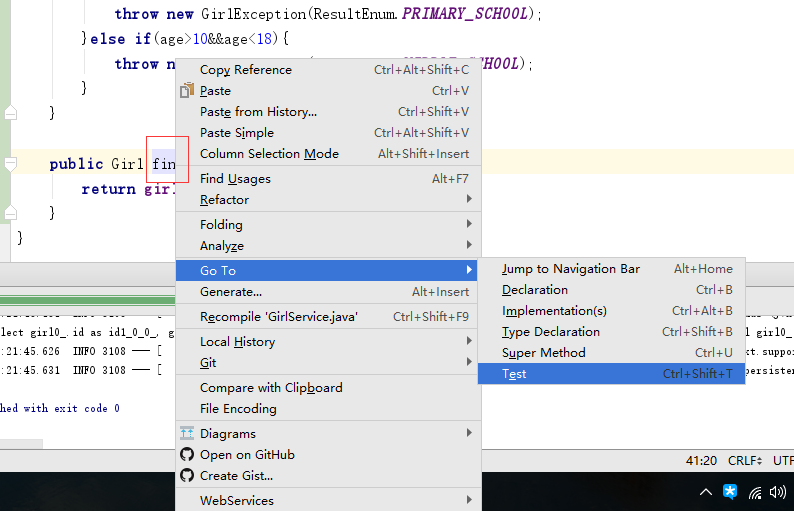




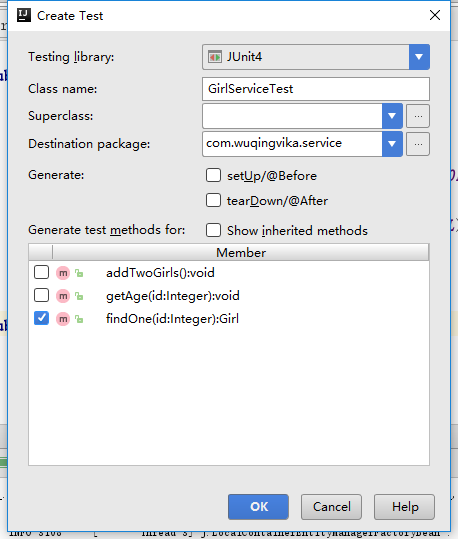
**改为正确答案后:**



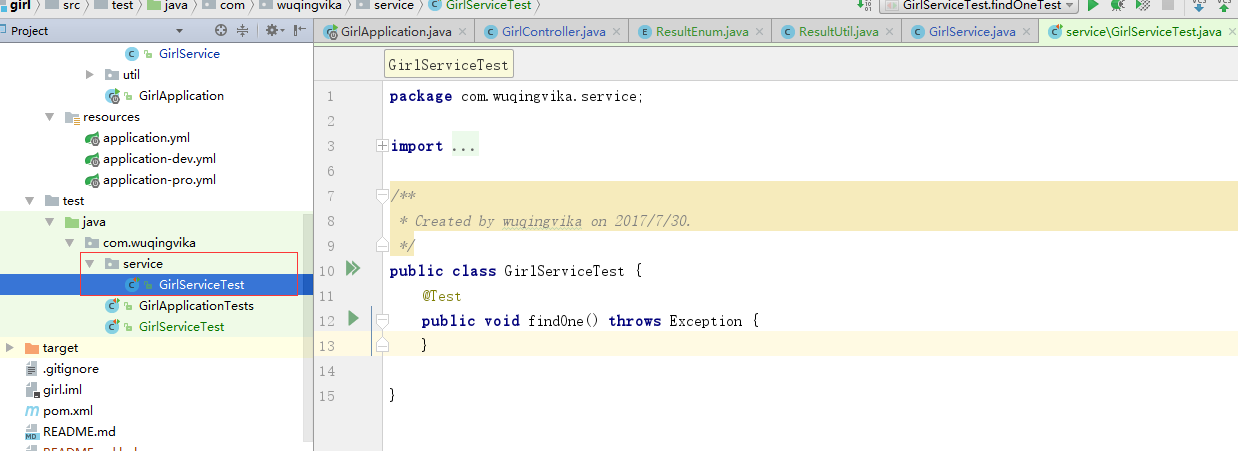
**Idea提供了快速创建测试方法的操如下：右击service层中的某一方法，Goto->Test.**



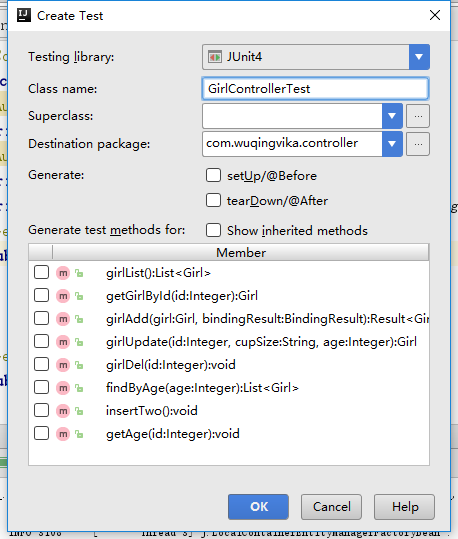
点击 Create New…



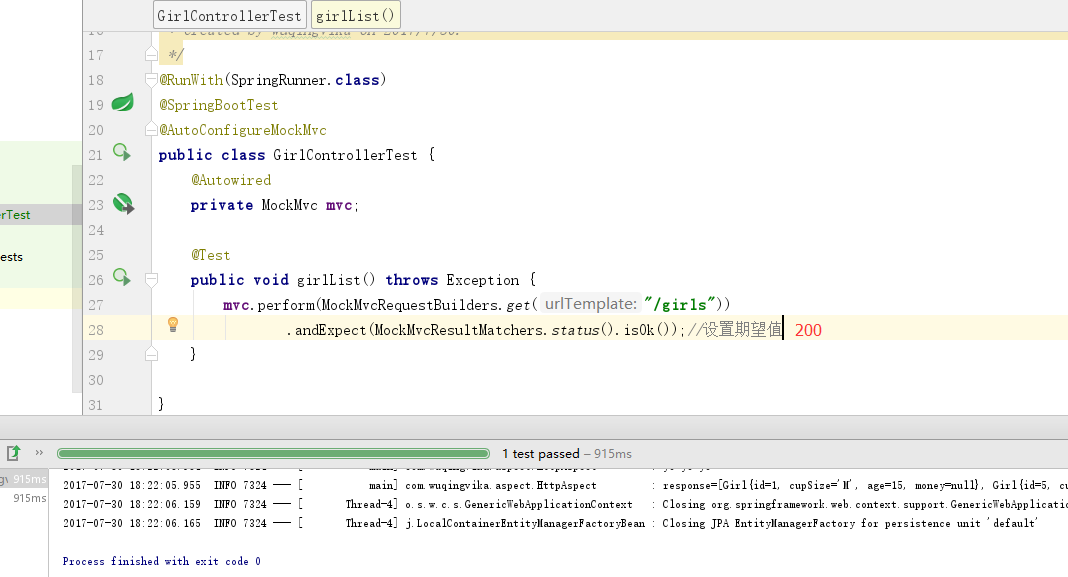
它会自动生成Service包。并在该包下创建我们的测试类。



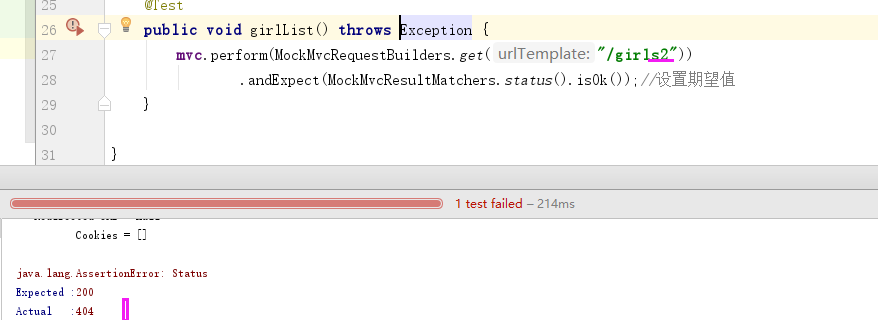
下面对GirlController进行测试：



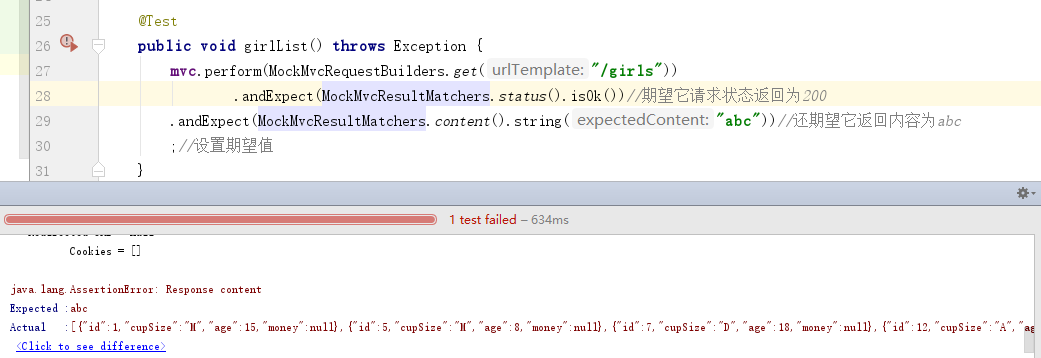
对Controller中的方法进行测试



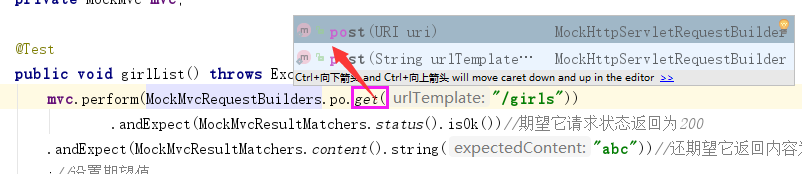
故意写个错误的Url



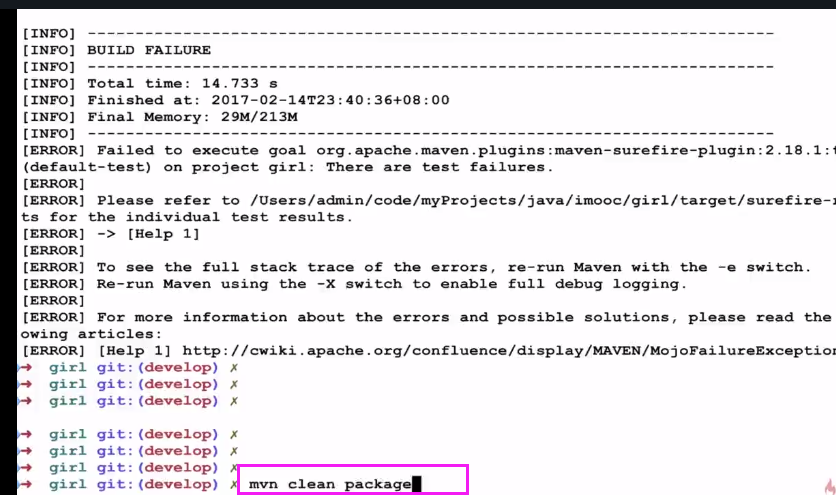
**期望它返回内容值为abc,结果如下**



**如果想换成Post提交只需这样**



最后，如果有多个测试方法，不想一个个测试，可以通过Maven..clean package，它会把所有的测试方法都走一遍的。



**如果想跳过测试，只需在后面加上一句:**

