SpringBoot入门

目录

[课程介绍 1](#_Toc523470574)

[目录 1](#_Toc523470575)

[1. 第一个SpringBoot程序 1](#_Toc523470576)

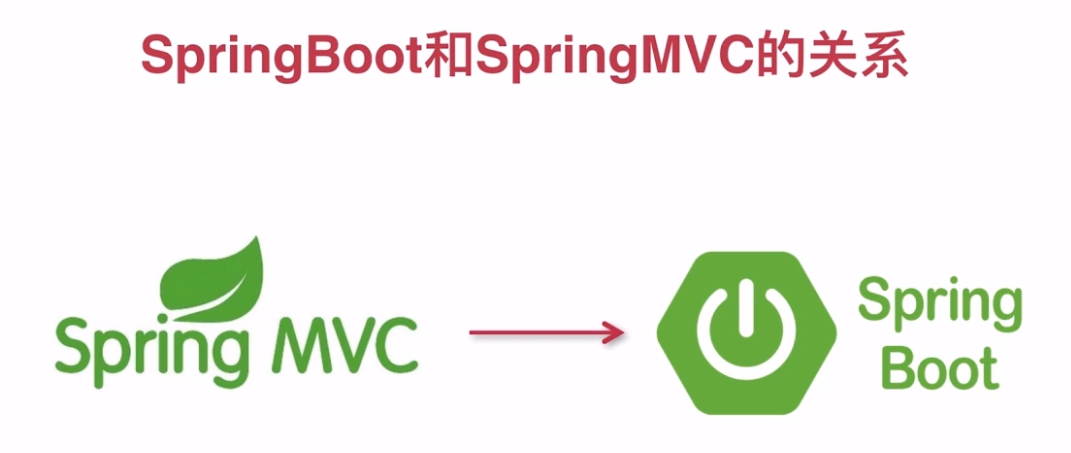
[2. 自定义属性配置 11](#_Toc523470577)

[3. Controller的使用 17](#_Toc523470578)

[4. Spring-data-jpa 21](#_Toc523470579)

[5. 事务管理 21](#_Toc523470580)

看视频学习你值得拥有



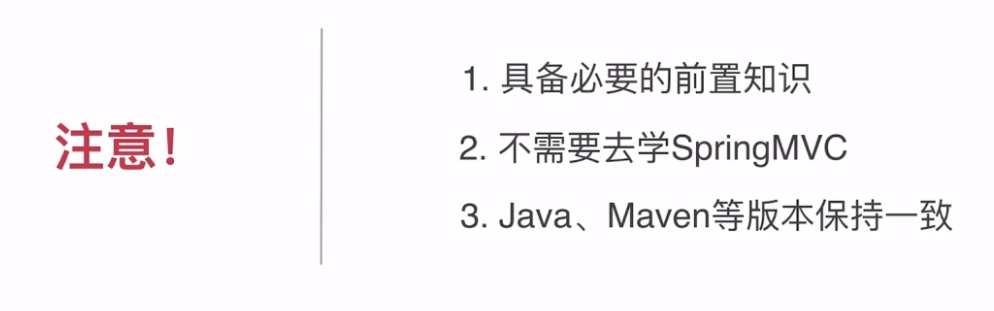
Springboot是spring MVC的升级版两者没有必然的联系



# 课程介绍

## 目录

### 第一个SpringBoot程序



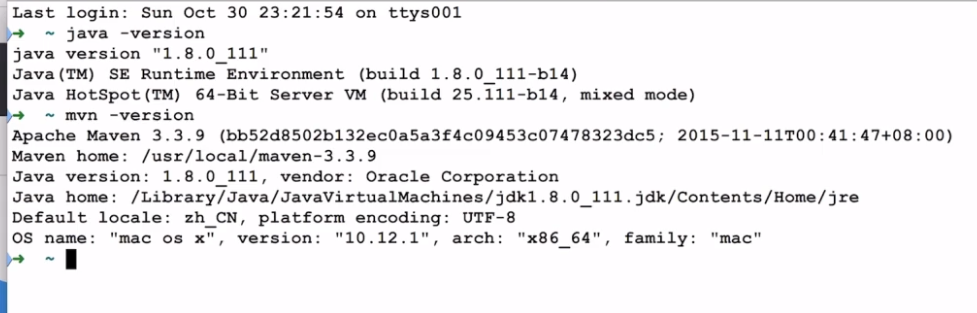


图1.1开发环境图

视频作者廖老师的开发jdk和maven如上图1.1所示

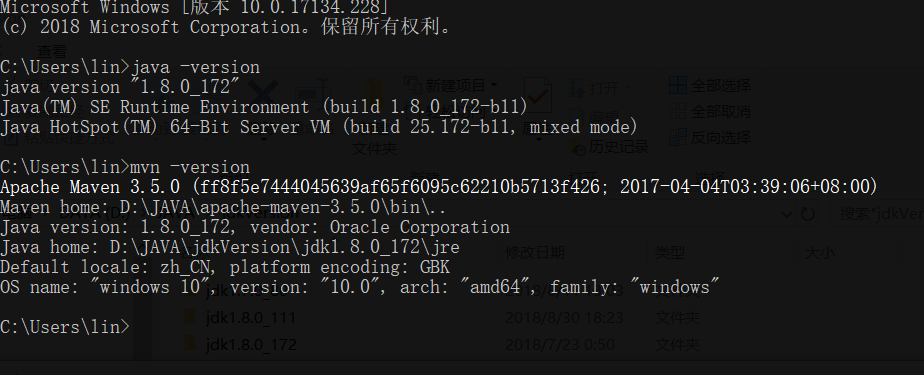


图1.2韦少的开发环境图

我看一下,我的版本比他高会不会向下兼容,希望不要出现不兼容的情况,不要刨坑埋自己,如果是版本的问题,再回来改一下环境.

好了,现在开始用IDEA创建一个springboot的项目

File->new->project

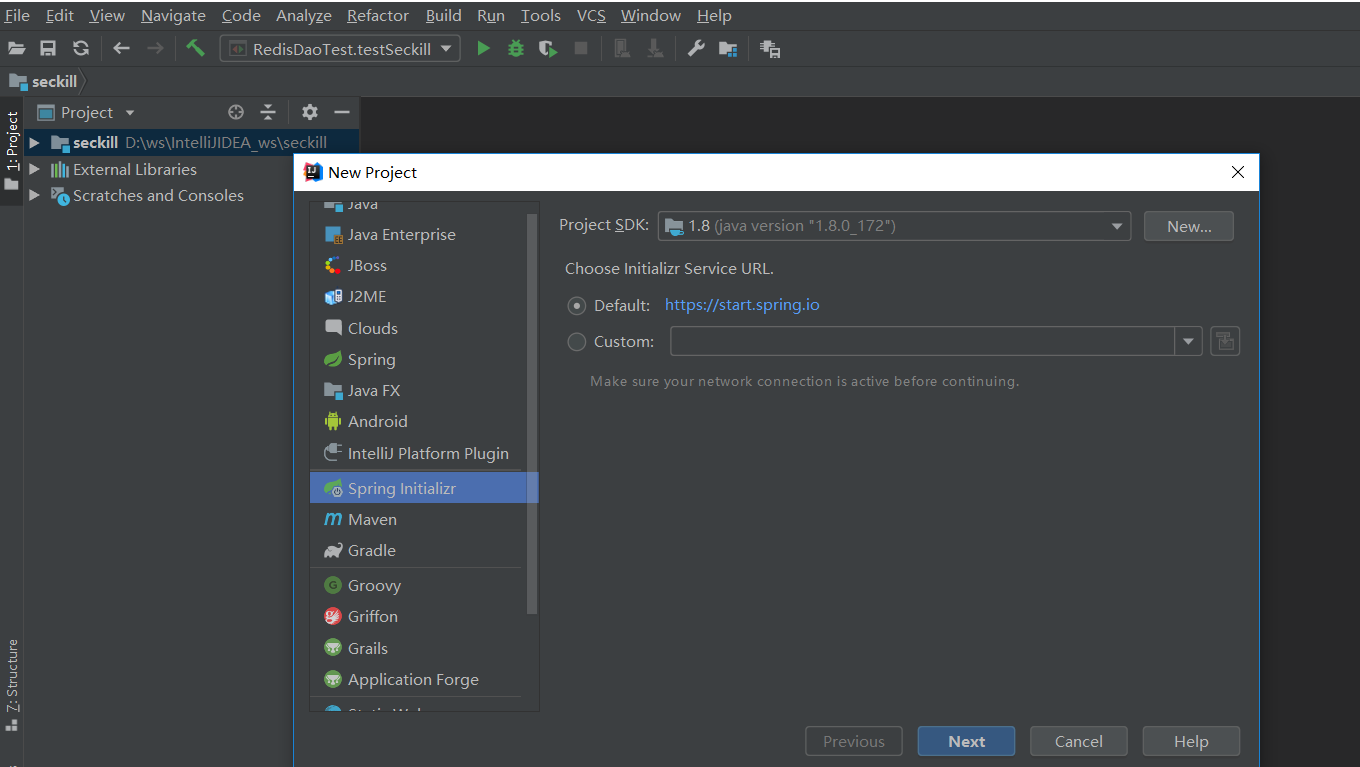


图1.3

Next->

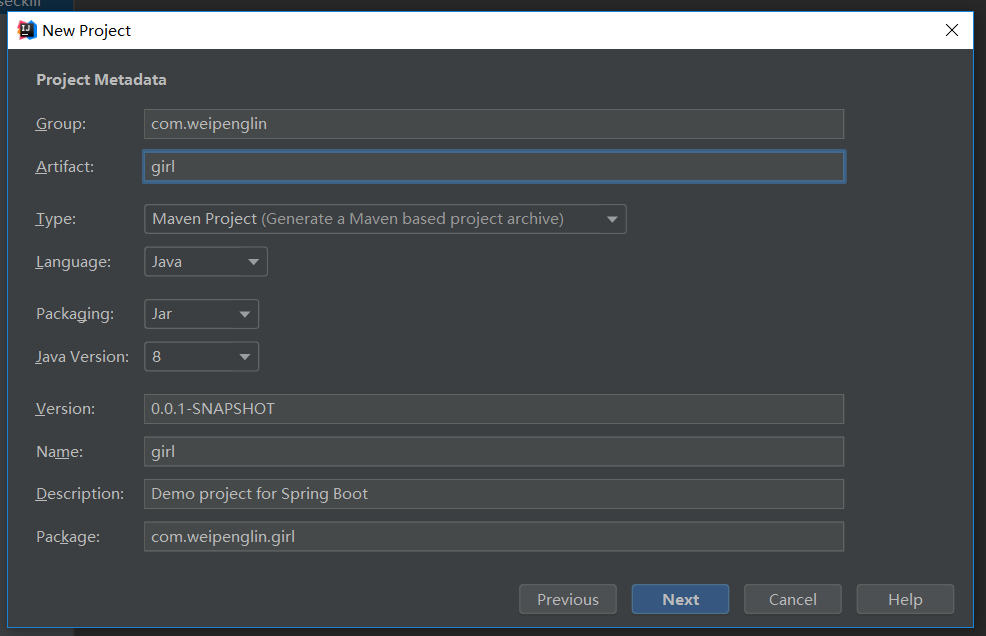


图1.4

Group和Airfact自己起,这是以后再maven中的jar包的路径

Next->

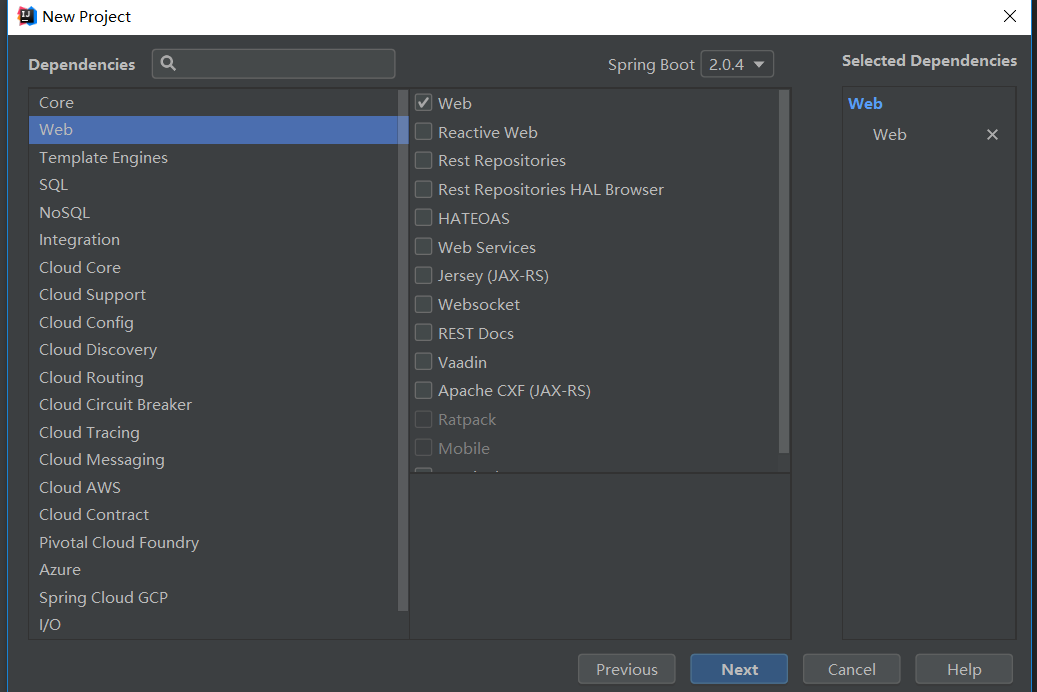


图1.5

只用选择web一项就可以,这些都是spring的一些组件

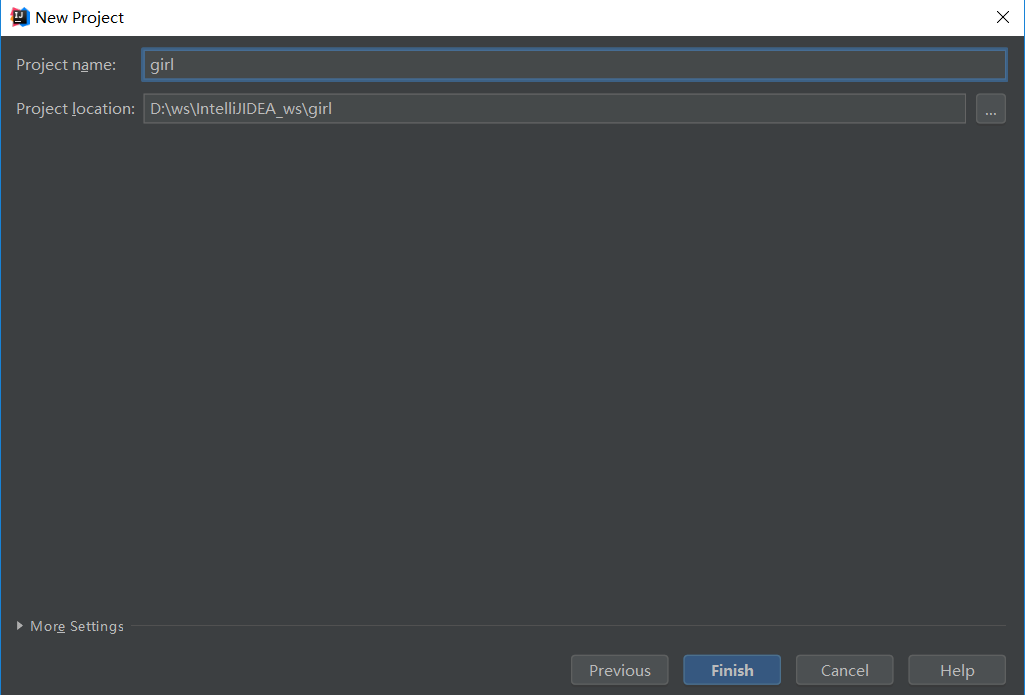


图1.6

然后finish，注意文件的保存路径不能有中文。

在cmd中查看vim .m2/settings.xml文件。

Window版的下载地址和方法：

<http://www.cnblogs.com/xiekeli/archive/2012/08/13/2637176.html>

视频作者settings.xml如下图1.7



图1.7

我的环境也是阿里云的，如图1.8

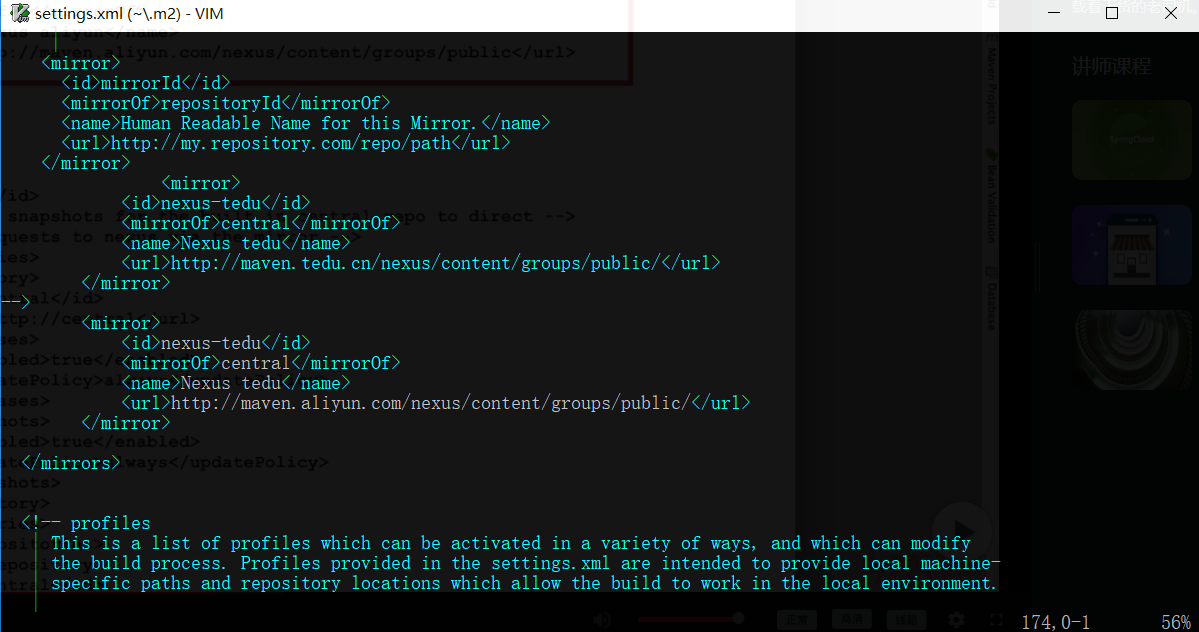


图1.8

删除不需要的三个文件，如图1.9

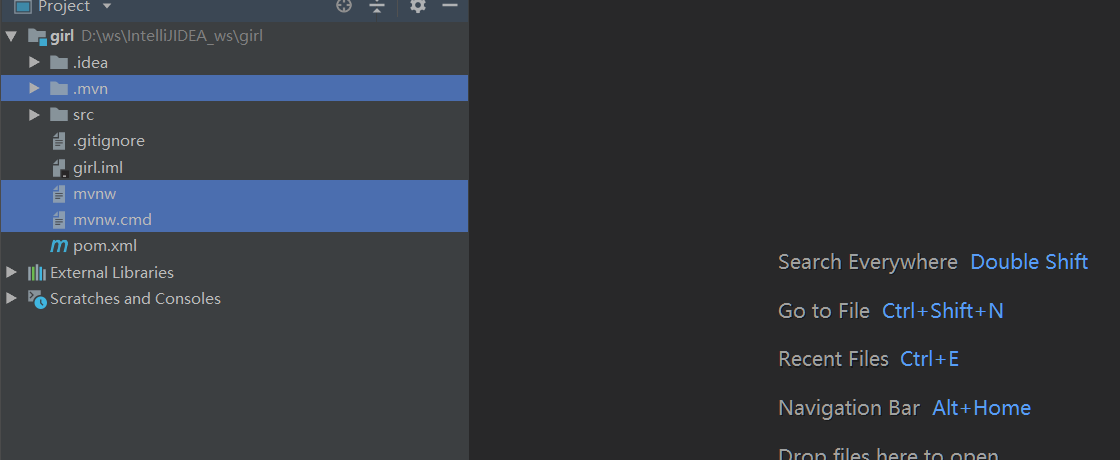


图1.9

pom.xml->会有一些信息，熟悉maven管理项目的同学对此并不陌生。

下面分别是springweb项目必须要引用额jar、单元测试用、使用maven构建项目的一个插件

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

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

<artifactId>spring-boot-maven-plugin</artifactId>

src->中只有一个GirlApplication，并且只有一个main方法，这个方法是用来启动spring的。方法中的传参一般是（类名.class，args）,这个类想要启动这个项目，必要要有@SpringBootApplication

@SpringBootApplication

public class GirlApplication {

public static void main(String[] args) {

SpringApplication.run(GirlApplication.class, args);

}

}

resource->application.properties:是springboot的配置文件

test->目录是test的类

启动项目方式1（也是最常用的）：在main方法的类中右击启动项目

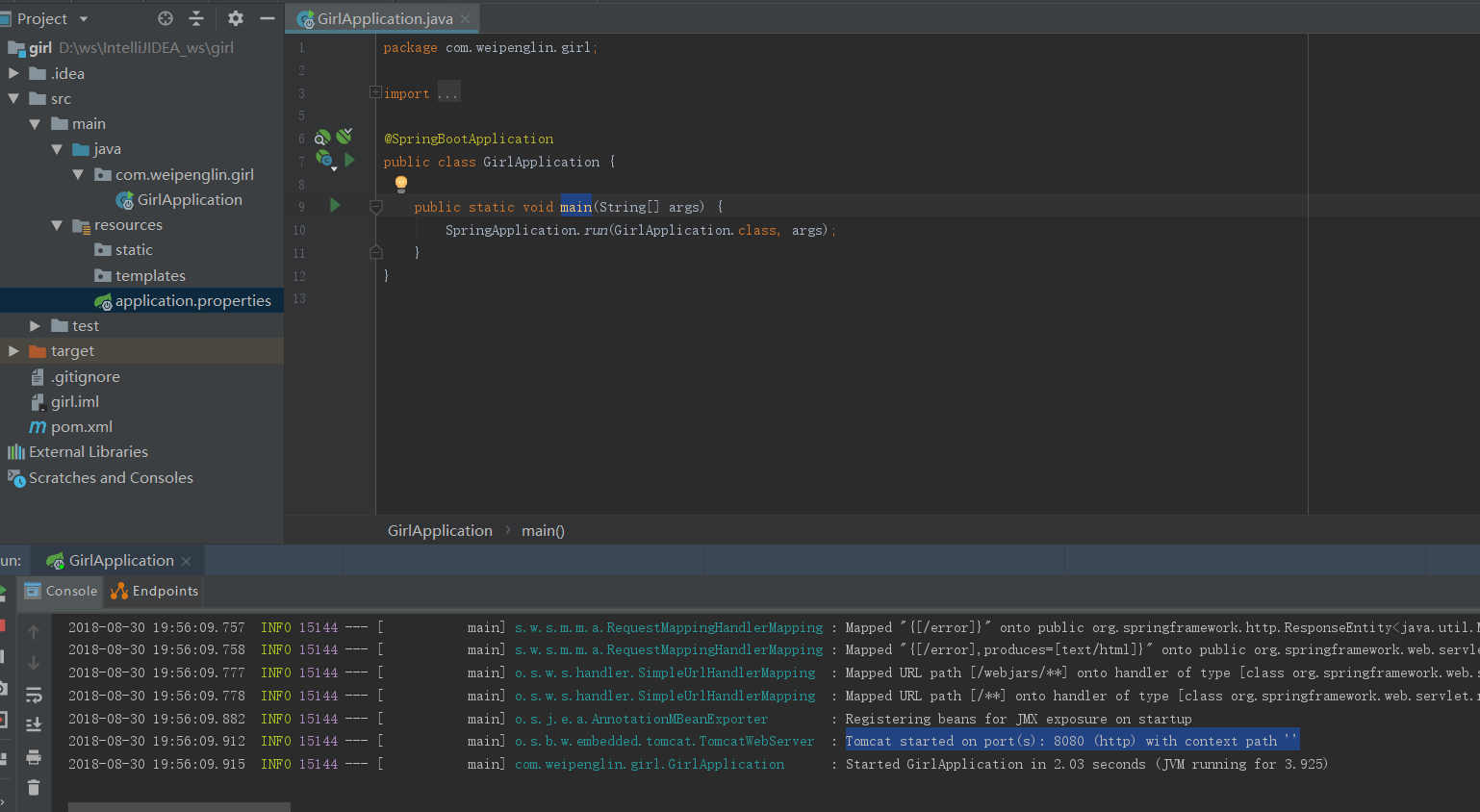


图1.10

然后新建一个helloControoler类

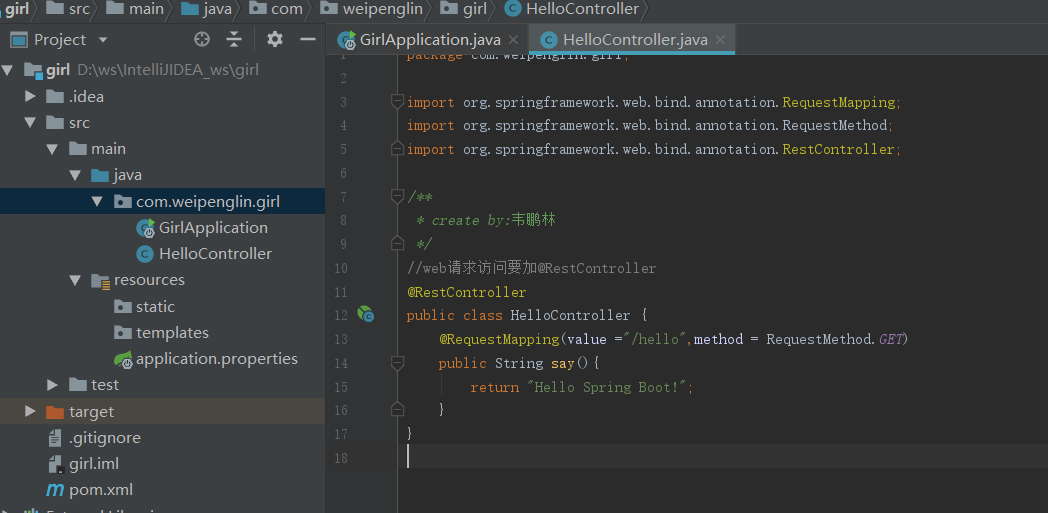


图1.11

IDEA给类和方法添加自动添加注释的方法在下面这个文章：

<https://www.jianshu.com/p/09139b425cc3>

重新启动项目，在浏览器输入<http://localhost:8080/hello>，显示如下

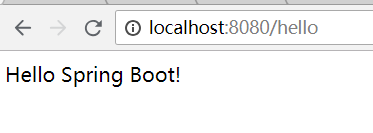


图1.12

我看到视频作者在cmd中风骚的使用ll等linux的命令，百度一下。微软新出的powershell可以向linux环境下敲命令，在cmd中敲powershell就可以进入。但是有的命令还是敲不了，比如ll

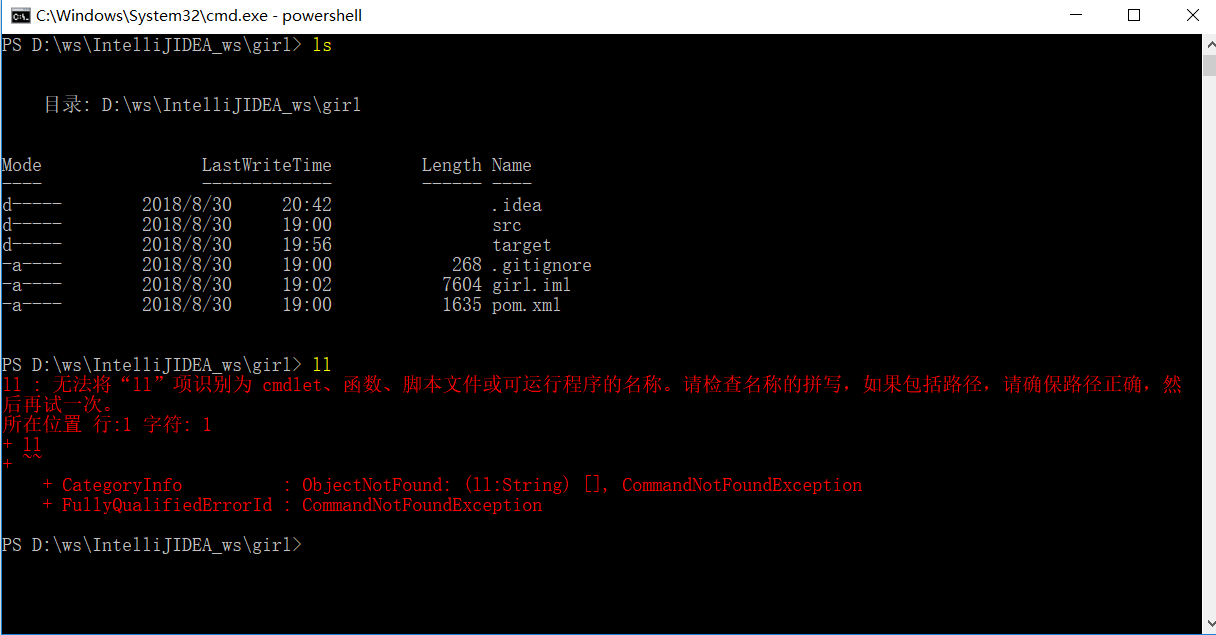
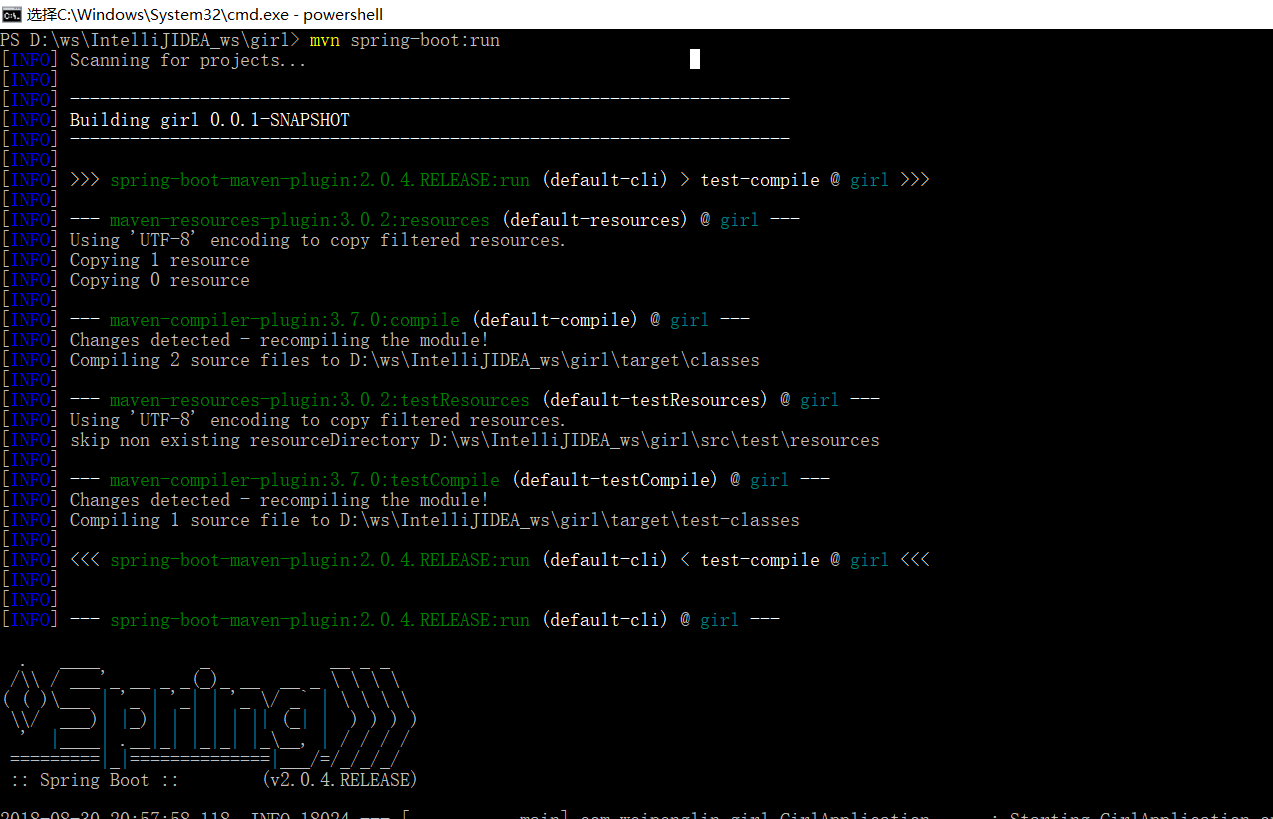


图1.13

启动项目方式2：

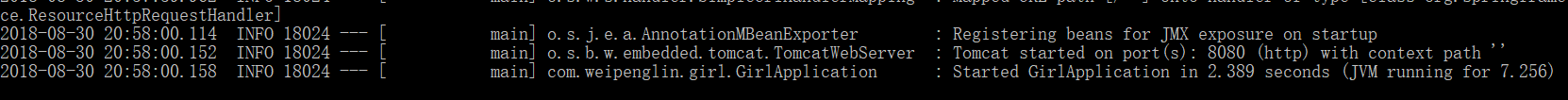
视频作者展示了在项目目录下的cmd命令行方式启动项目



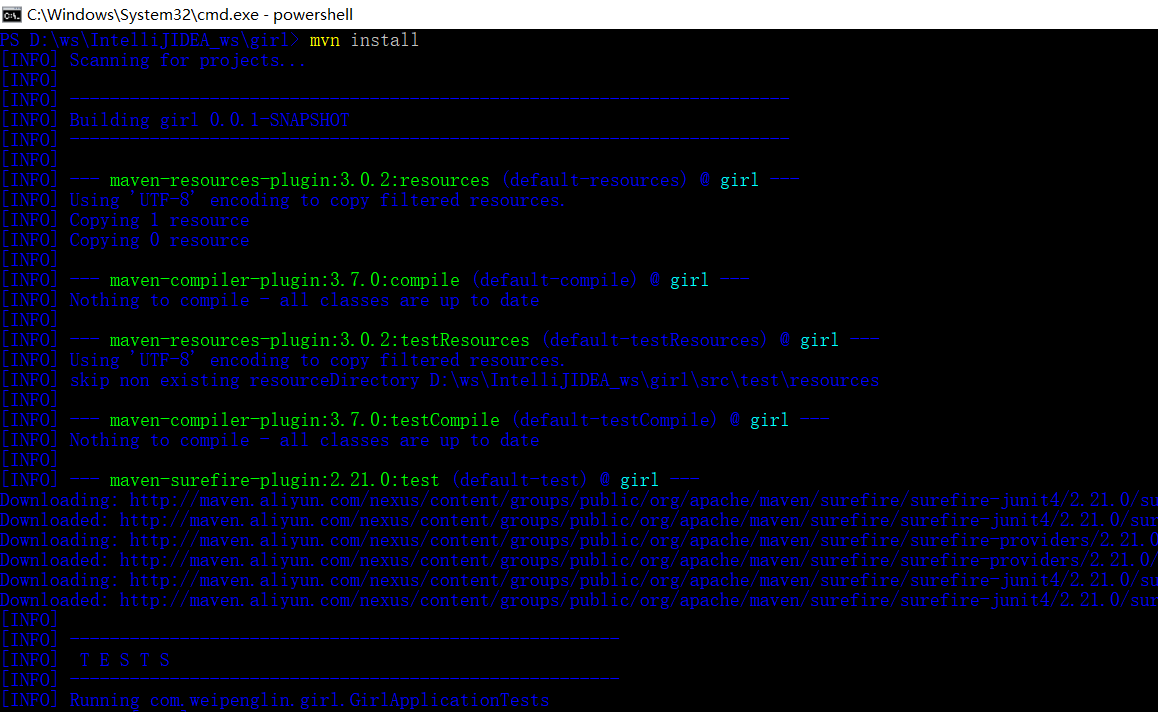
.

.

.

图1.14

启动项目方式3：



.

.

.

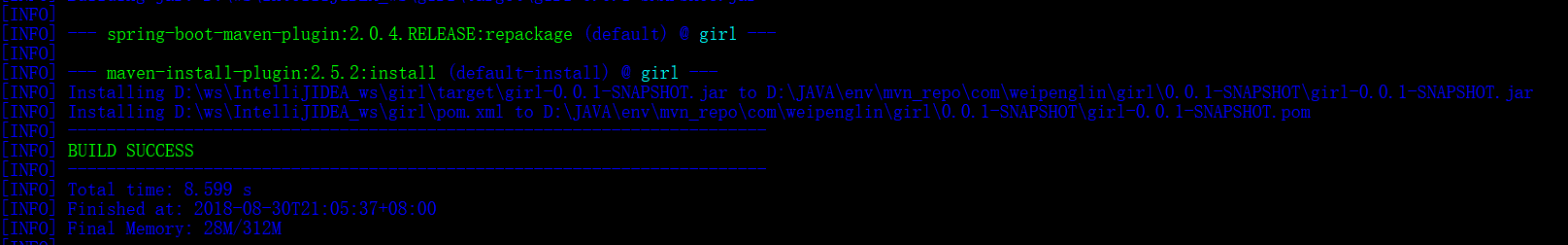


图1.15

cd 到target目录下，然后java -jar .\girl-0.0.1-SNAPSHOT.jar来启动maven项目

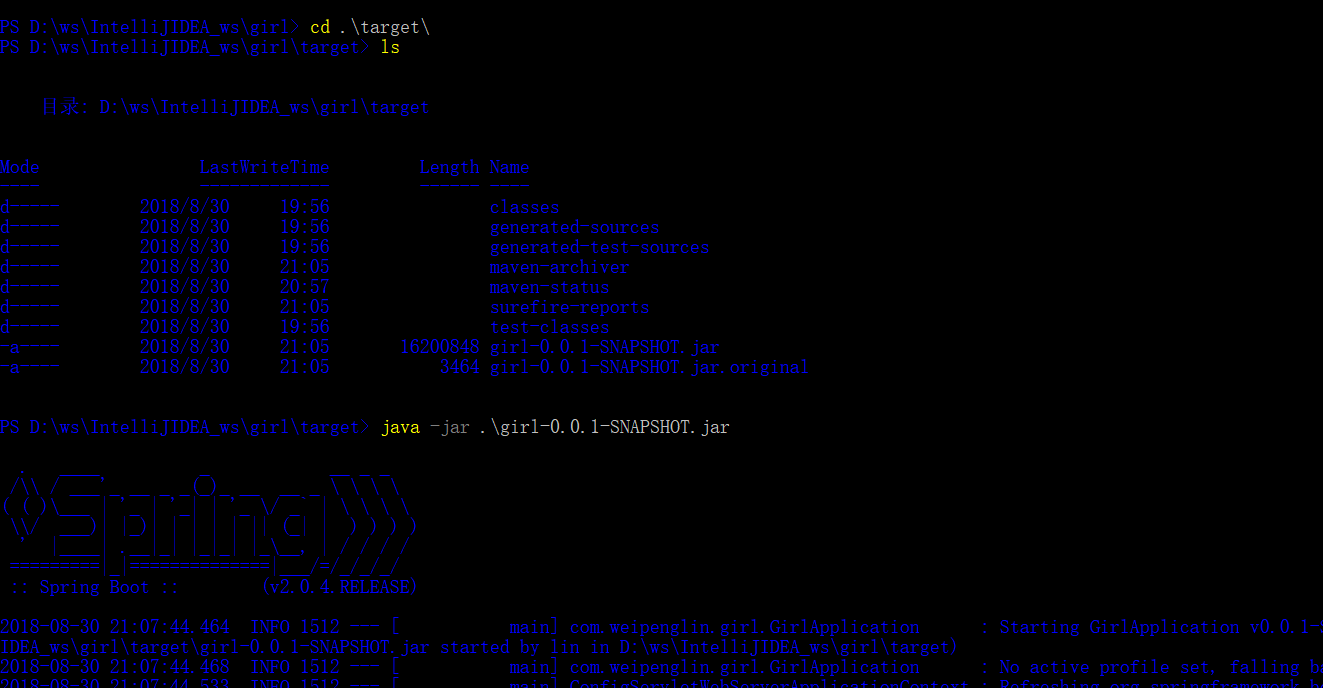


图1.16

### 自定义属性配置

删掉resources下的static、templates文件夹.配置项目文件application.properties,然后启动浏览器,输入地址查看一下

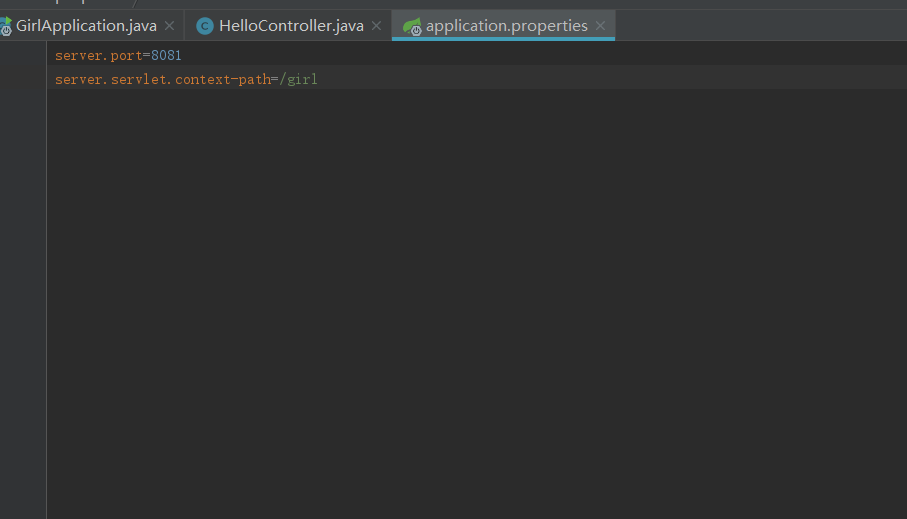
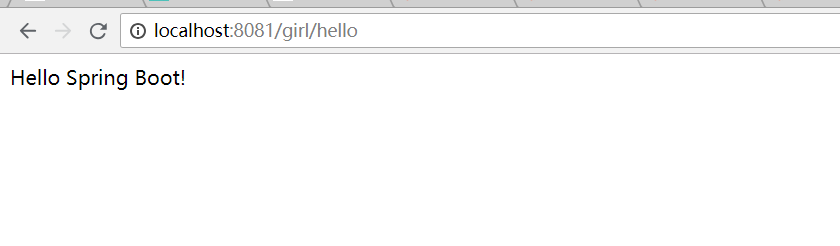


图2.1

图2.2

视频作者有推荐使用application.yml文件的配置方式,本文作者在实际操作当中发现,文件有点小”bug”(是我刚才没弄清楚)

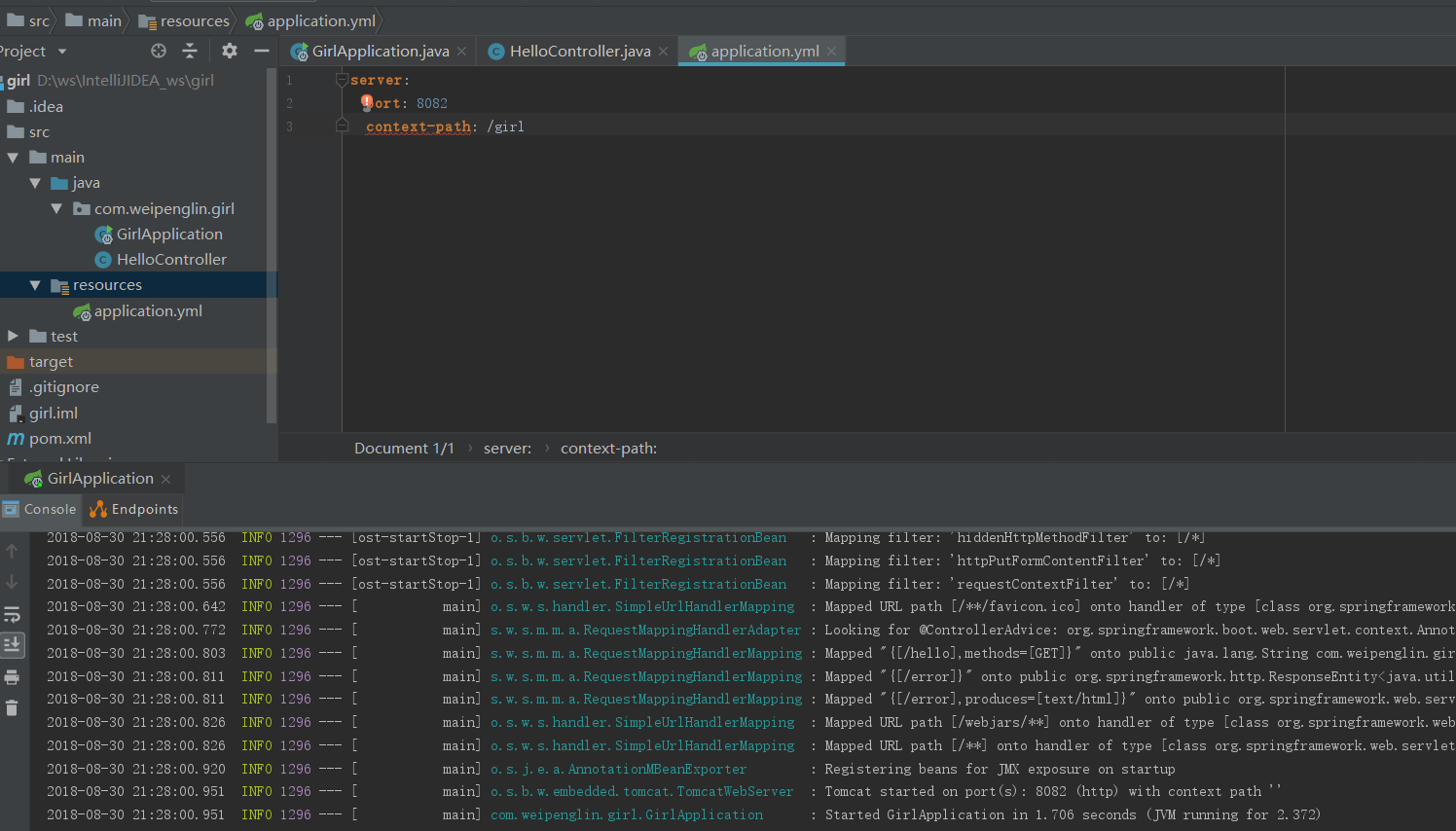


图2.3

application.yml中的context-path: /girl有告警

但是却能启动成功,试着浏览器访问一下,提示不成功

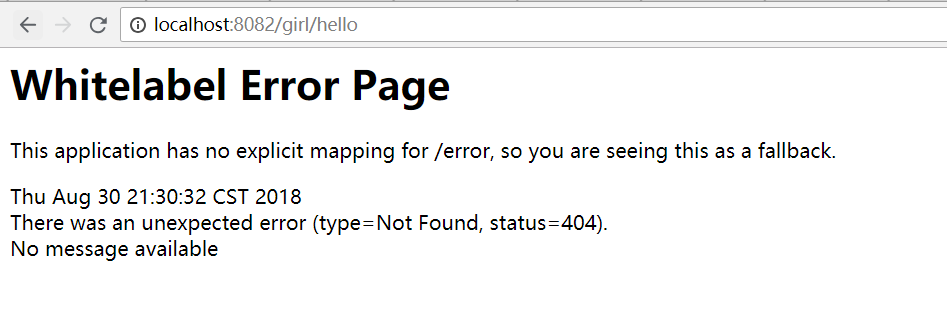


图2.4

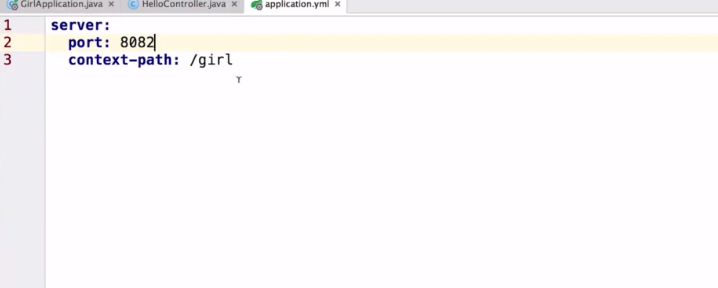
****

图2.5

图2.5的配置,现在已经不能用了,根据提示配置就好

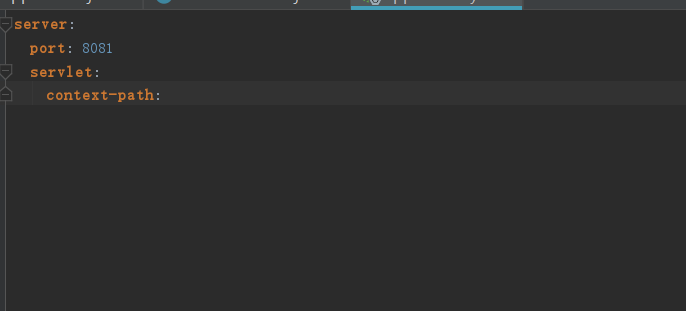


图2.6

访问页面也能正常访问.心得:根据提示敲代码能避免一些失误.

然后我们改下配置文件

server:

port: 8080

cupSize: B

controller这样改:

@RestController

public class HelloController {

/\*\*

\* 使用注解把配置文件中的内容注入到cupSize这个属性变量中来

\*/

@Value("${cupSize}")

private String cupSize;

@RequestMapping(value ="/hello",method = RequestMethod.GET)

public String say(){

return cupSize;

}

}

启动项目访问

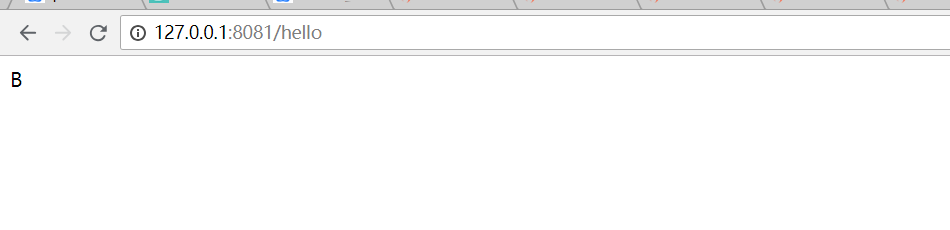


图2.7

然后可以输入年龄等其他信息,和cupSize的方式一样

如何在配置文件里面在使用配置呢?

server:

port: 8080

cupSize: B

age: 18

content: "capSize: ${cupSize}, age: ${age}"

注意:注意单词的拼写cupSize和capSize和age: 后面一定要有空格.

\*/

@Value("${cupSize}")

private String cupSize;

@Value("${age}")

private Integer age;

@Value("${content}")

private String content;

@RequestMapping(value ="/hello",method = RequestMethod.GET)

public String say(){

return content;

}

问题:当然一个女生可以有很多类,不能每次都改配置文件,每次都在controller里面加autowired.

答案:可以新建一个properties的类,里面定义属性,在类上加注解, @ConfigurationProperties(prefix ="girl")对应配置文件中的:

girl:

cupSize: B

age: 18

@ConfigurationProperties注解把配置文件中配置注入到类中属性变量

@Autowired

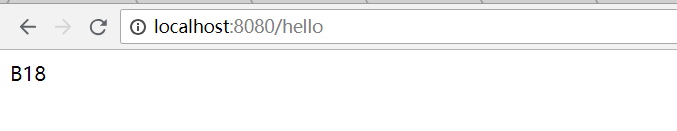
private GirlProperties girlProperties;

@RequestMapping(value ="/hello",method = RequestMethod.GET)

public String say(){

return girlProperties.getCupSize() + girlProperties.getAge();

}

然后在controller中通过注入到girlProperties,通过调用GirlProperties类的方法打印配置文件中的属性的值. 图2.8

接下来比如cupSize 在家喜欢B,在外面喜欢F.不能随便去修改配置,视频作者说”优秀的开发工程师不是去修改配置的”.

从这个例子可以映射到,我们的开发环境和生产环境之间的不同.

视频作者讲到:我们复制application.yml,该名称application-dev.xml(开发环境),application-prod.xml(生产环境),然后,在本地application.yml配置

spring:

profiles:

active: dev

然后就提到的开启webserver的多种启动方式

首先:在项目文件目录的cmd窗口mvn install

然后:输入

java -jar .\girl-0.0.1-SNAPSHOT.jar --spring.profiles.active=prod



图2.9

启动配置的8081端口的prod的服务

然后在application.yml中配置并启动服务,然后在浏览器分别访问两个服务

spring:

profiles:

active: dev



图2.10

总结:

@Value

@Component

@ConfigureProperties

最简单的是用@Value来实现配置的注入,随着配置的增多,我们就要为配置分组,所以就要使用@Component、@ConfigureProperties

多环境配置，就是dev和prod

### Controller的使用

图3.1

需求：现在希望访问127.0.0.1:8080/hello和127.0.0.1：8080/hi都能访问我们的方法怎么做？

答案：url写成一个集合，在RquestMapping中这样写

图3.2

vlue=”/hello” 改成图2-12，然后启动测试验证如图图3.3

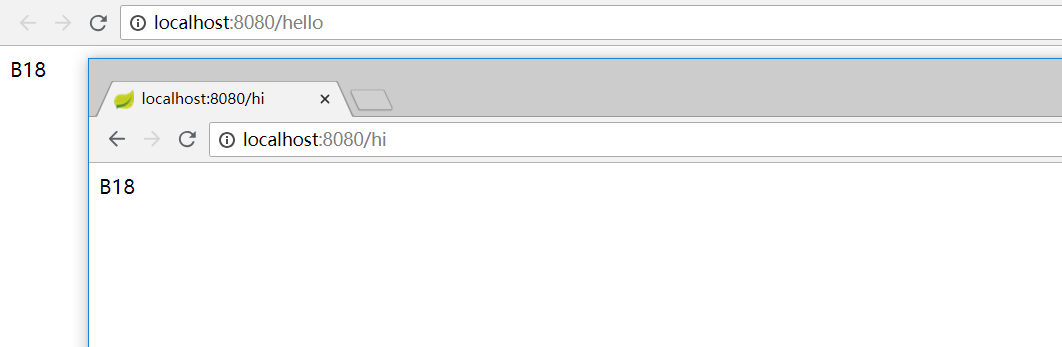


图3.3

测试验证ok

仔细观察会发现在console中有这样一段输出,mapper中hello和hi的映射

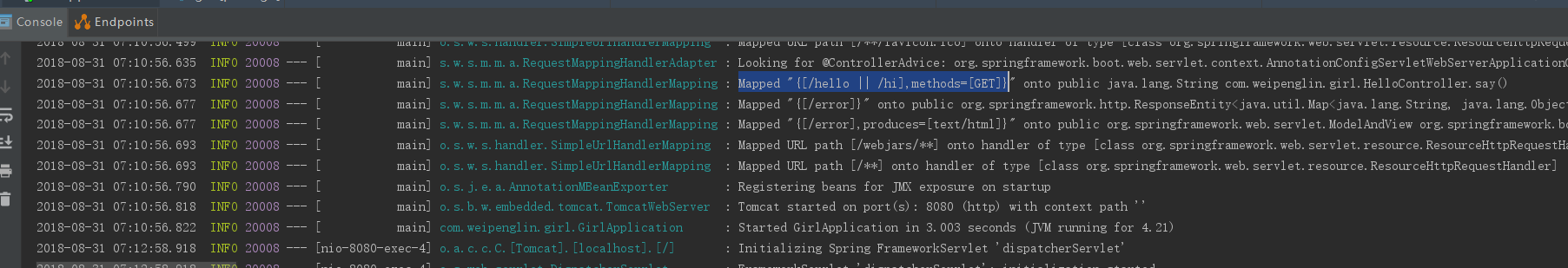


图3.4

当然映射可以加在类上，意思就是给类中的全部方法不要按照这个路径写，由于我这样的代码写了很多，不需要再测试。不熟悉的小伙伴可以自己测试玩。

上述我们方法请求方式用的是RequestMethod.*GET ，*现在我试着用一下啊Post请求试一下。 RequestMethod.*POST*

视频作者用Pstman来发送post请求由于我没有安装这个软件，现在我先来安装一下，具体的教程如下：

<https://blog.csdn.net/fxbin123/article/details/80428216>

我的环境有google浏览器，下载安装即可。重启webserver服务，测试如图3.5

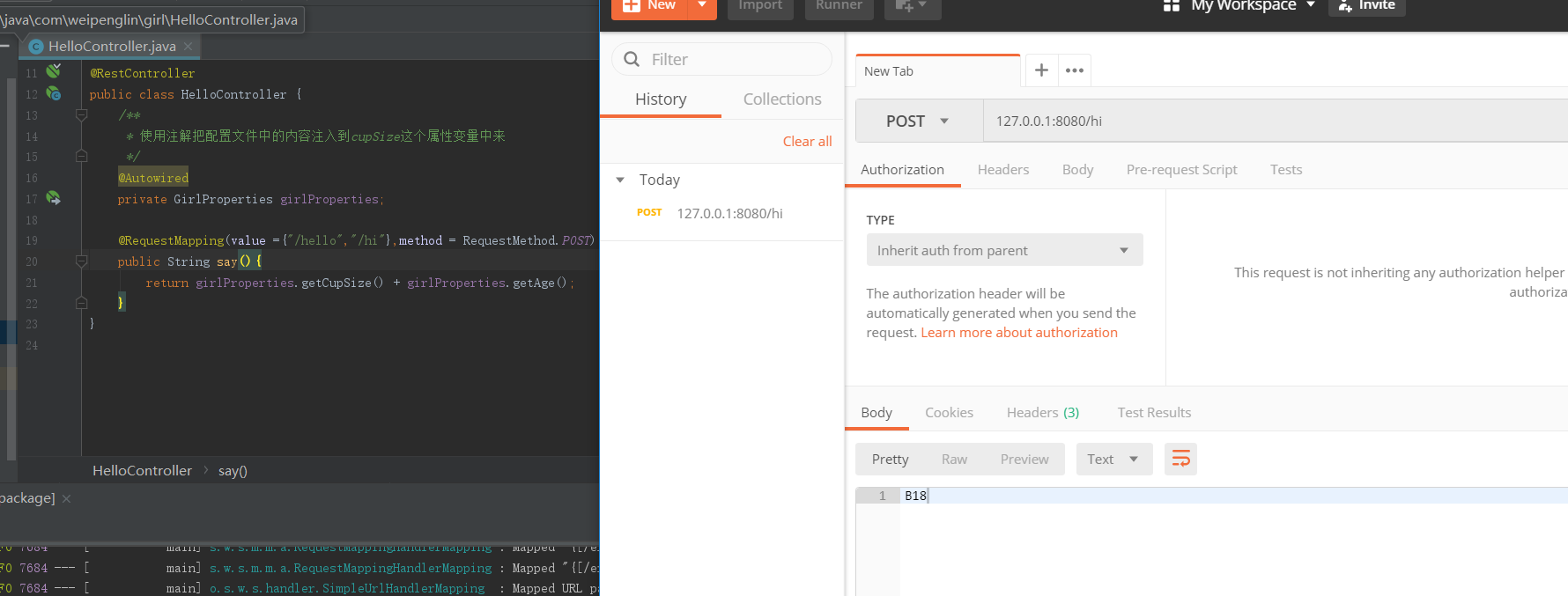


图3.5

思考：如果我们不加method = RequestMethod.POST 测试一下post和get请求

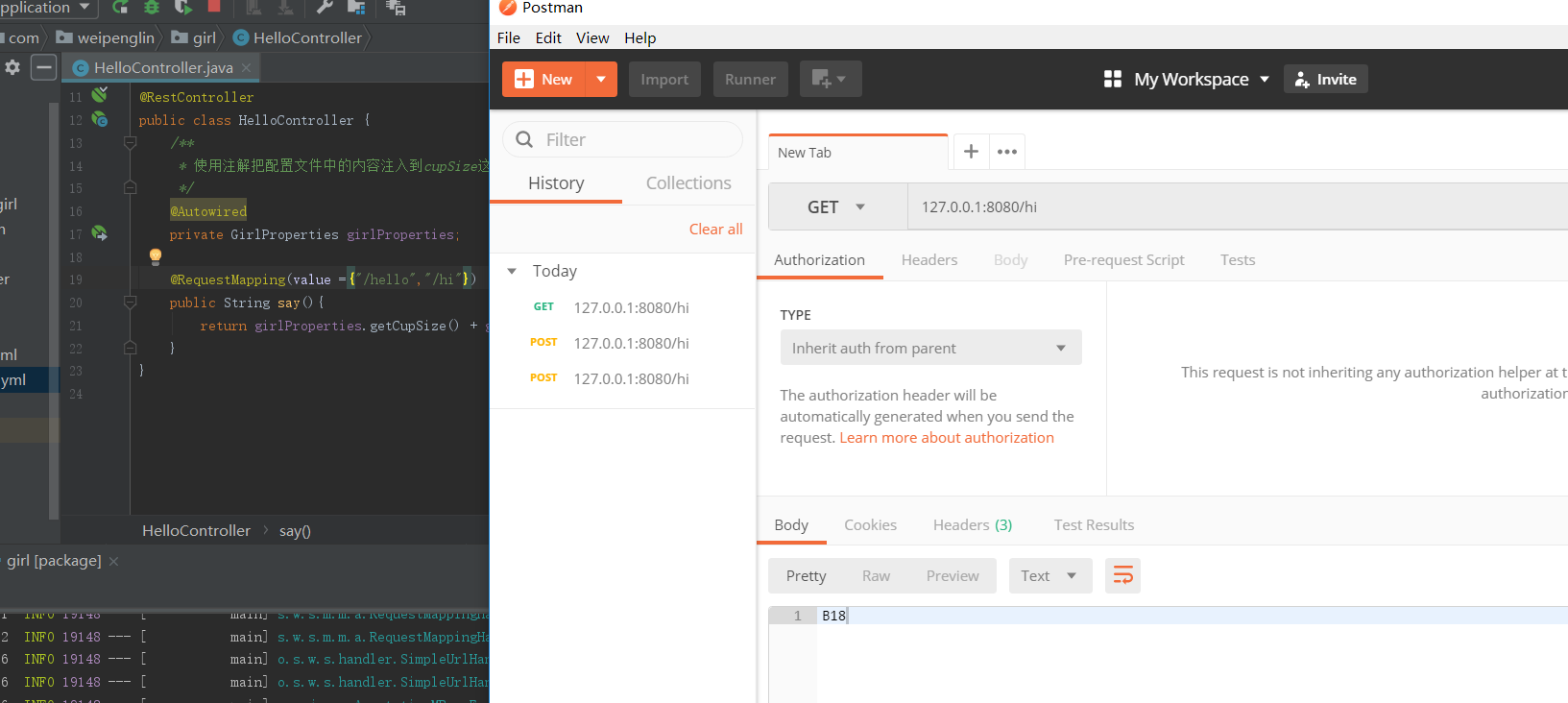


图3.6

测试结果是不加method = RequestMethod.POST这样的method限制都可以访问，因为get和post适用于不同的业务场景，所以这个地方不能省

图3.7

测试一下，如图3.8所示

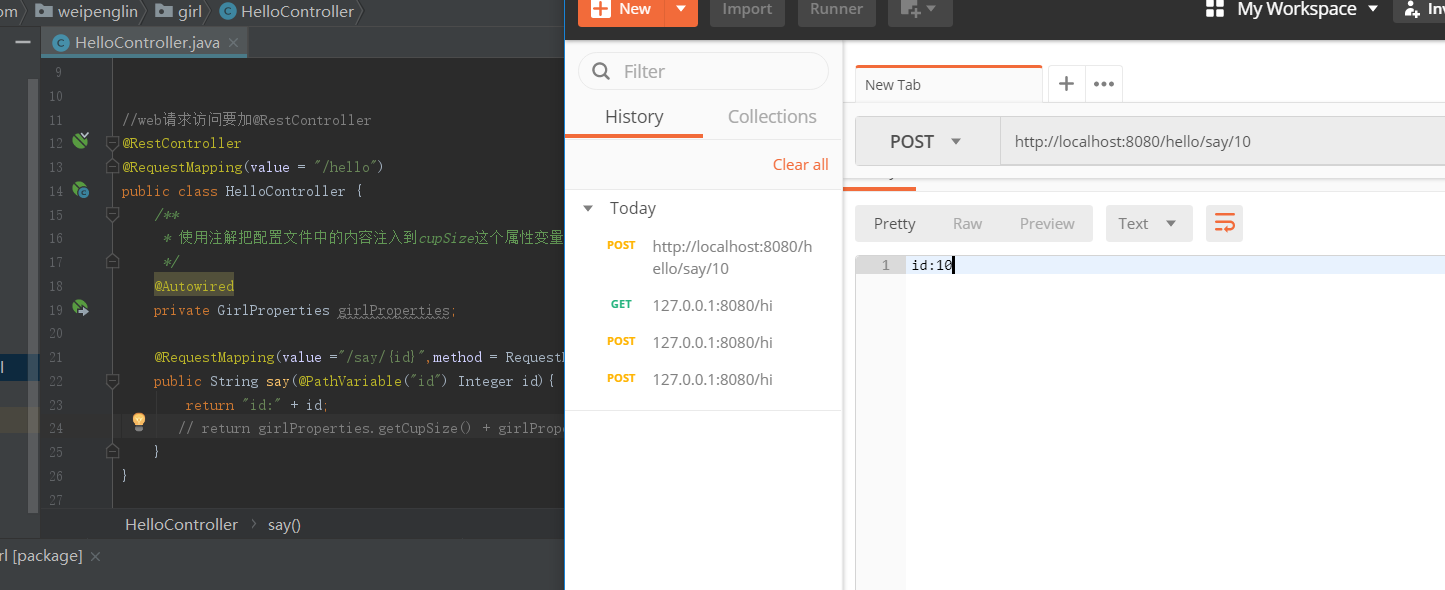


图3.8

也可以写成这样：

@RequestMapping(value ="/{id}/say",method = RequestMethod.POST)

测试以实ok的，这样url比较清爽

传统的url请求路径这样：

<http://localhost:8080/hello/say?id=100>



图3.9

测试结果和Requestmapping、方法中的用@RequestParam来标注。

注意：@RequestParam中的标注的id要和url(http://localhost:8080/hello/say?id=100)中的id保持一致，不用和方法中形参保持一致，这里为了区分，把形参改名myid。

不管是POST还是GET都用@RequestParam来获取属性里面的值

新需求：我们不在url输入数字，希望在程序中给id默认值



图3.10

方法参数注解写成下面所示：

public String say(@RequestParam(value = "id",required = false,defaultValue = "0") Integer myId)

新需求：@RequestMapping(value ="/say",method = RequestMethod.POST)

这个注解太长了，有没有简化方法，有

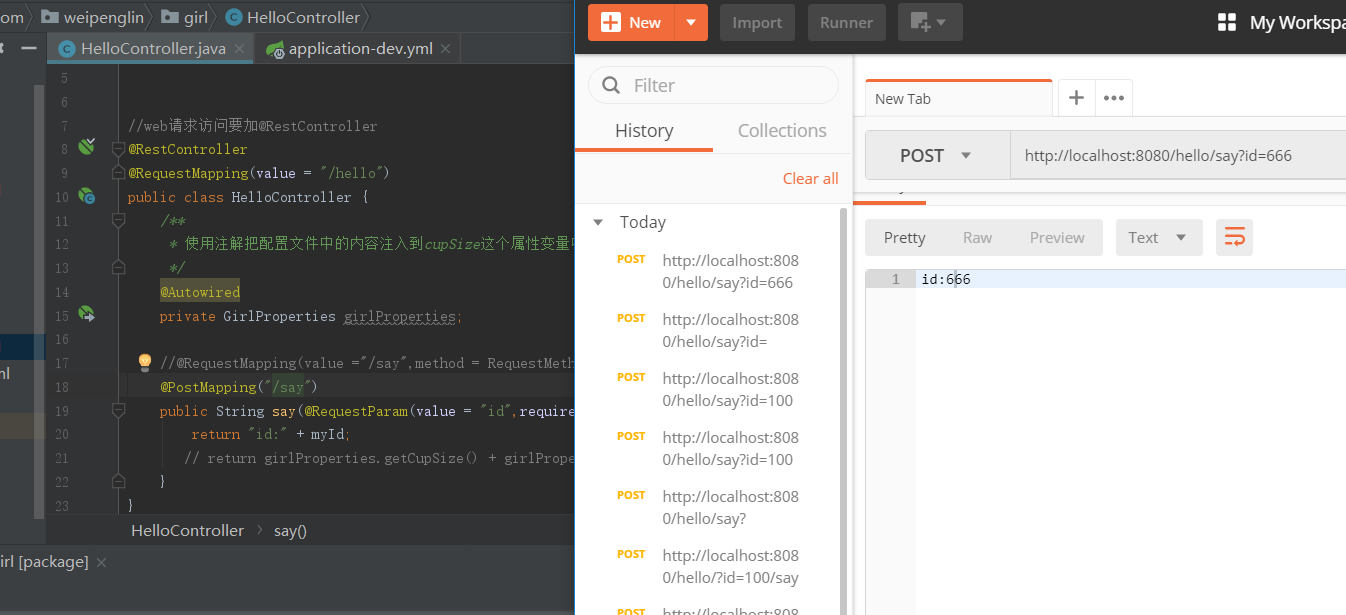


图3.11

测试图图3.11所示。其实这个和之前@RestController有点相似的地方，是联合@ [RequestMapping和RequestMethod.POST组成@PostMapping](mailto:RequestMapping和RequestMethod.POST组成@PostMapping)

### Spring-data-jpa

什么是jpa？

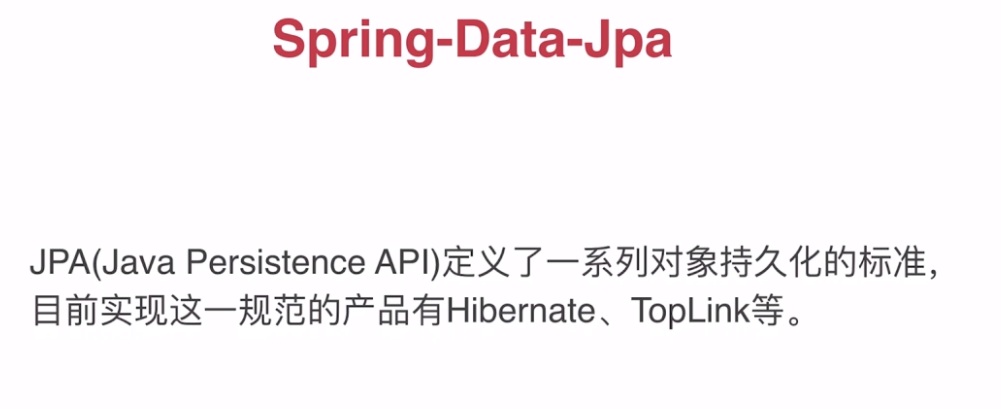


图4.1

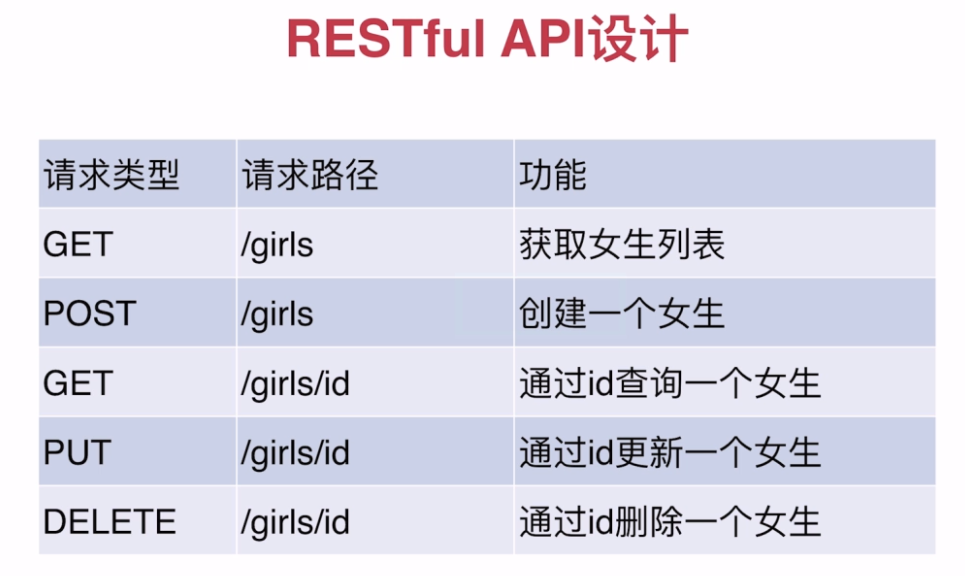


图4.2

pom添加两个依赖

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

<version>1.5.6.RELEASE</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.44</version>

</dependency>

在yml中对jpa和musql做一些相应的配置。

爬了几个坑

1. spring-boot-starter-data-jpa写了两次。错误报unique
2. 删了spring-boot-starter-data-jpa之后要右击pom文件Reimport
3. java.lang.IllegalStateException: The following classes could not be excluded because they are not auto-configuration classes:

这个坑不好爬，

学习一下sprinboot的自动装配

现在我把mysql的面试登陆去掉。然后在

17：12 2018/8/31这个坑我怕出来，从上午11左右到现在

坑1(最深的坑)：导错包了

https://blog.csdn.net/stupid\_java\_learner/article/details/78346820

Girl实体类中的@Id注解import错包了。

错误：

import org.springframework.data.annotation.Id;

正确：

import javax.persistence.\*;

坑2：

Girl实体类没加@Table(name="Girl")

坑3：

Girl没有加 @Configuration

一句话概括就是 @Configuration 中所有带 @Bean 注解的方法都会被动态代理，因此调用该方法返回的都是同一个实例。

https://blog.csdn.net/isea533/article/details/78072133

坑4：

@EnableAutoConfiguration这个注解在我环境下一致报错，想删掉。

其实这个注解会默认加载一下东西。这个报警不用管，最后启动项目没问题

很多Spring Boot开发者总是使用 @Configuration ， @EnableAutoConfiguration 和 @ComponentScan 注解他们的main类。由于这些注解被如此频繁地一块使用（特别是你遵循以上最佳实践时），Spring Boot提供一个方便的 @SpringBootApplication 选择。  
该 @SpringBootApplication 注解等价于以默认属性使用 @Configuration ， @EnableAutoConfiguration 和 @ComponentScan

有几篇写的很好的文档推荐：

<https://www.cnblogs.com/woms/p/5754200.html>（条理清晰）

<https://blog.csdn.net/a67474506/article/details/52013634>(理论性很强)

<https://blog.csdn.net/qq_19674905/article/details/79367921>（形象生动）

坑5：

@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class, HibernateJpaAutoConfiguration.class})

前期是加的Girl.class(我寻思Gir不就是数据源吗？加他就对了)，这坑也怕了很长时间。最后还是要加上datasource和hibernate

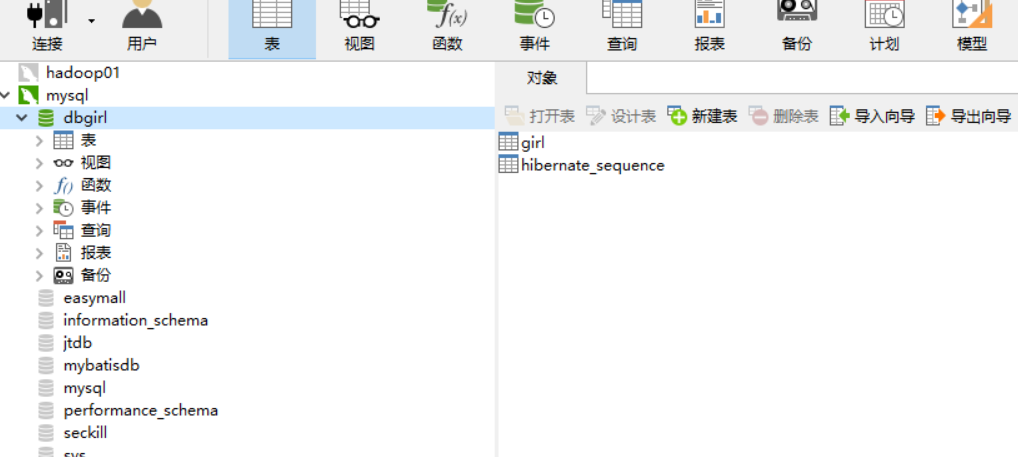


图4.3

为了这一刻，我肝了6个小时。2333



图4.4

新建Controller来测试一下这些API

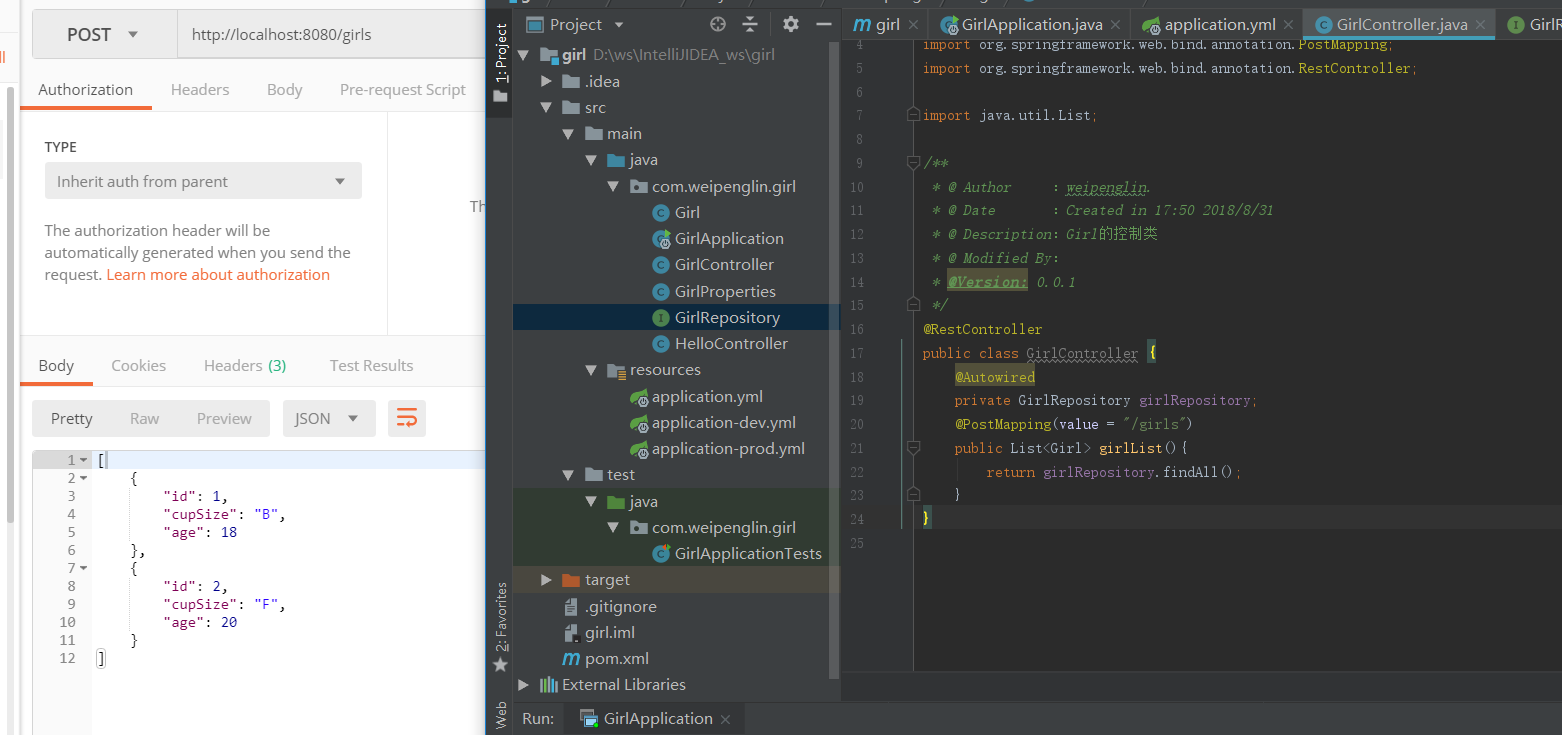


图4.5

报错：

com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException: Duplicate entry '1' for key 'PRIMARY'

取消数据表中自增属性（这是我自己加的，真的是自己给自己埋坑）

<https://blog.csdn.net/zr_1877/article/details/79006060>

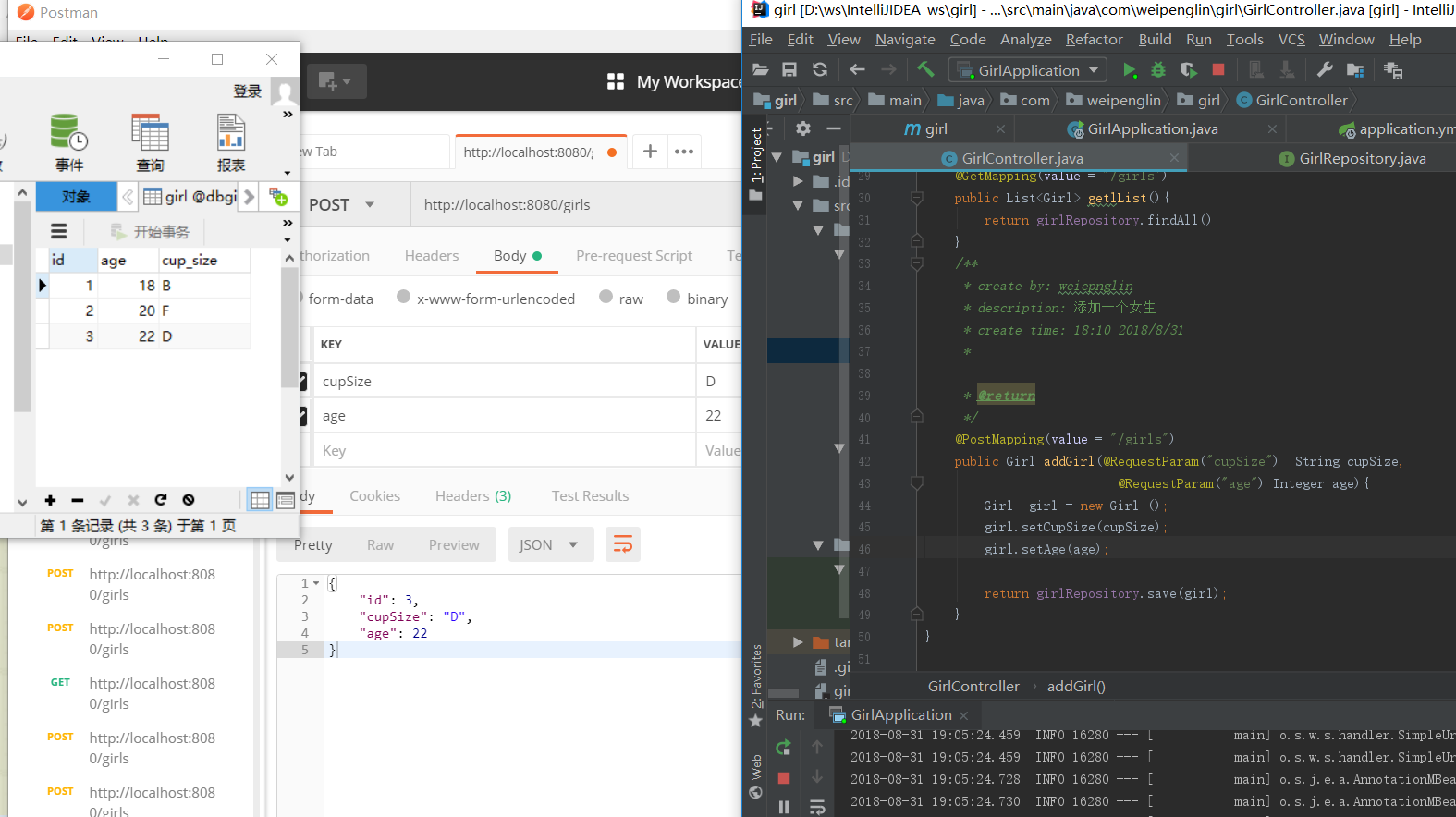


图4.6



图4.7

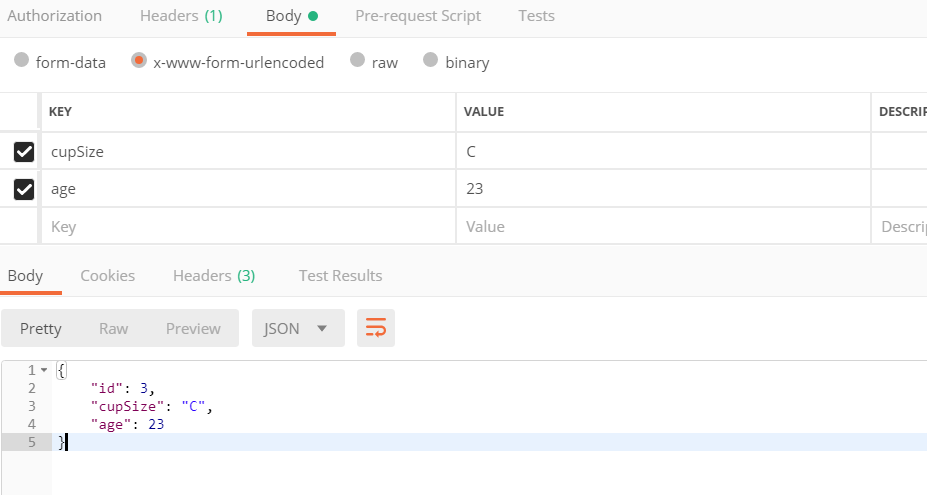


图4.8

注意：put提交，不能使用form-data

小小得瑟一下，删除操作我加了一个try-catch，比视频者好一些，在console能看见。下面是我写的crud

/\*\*

\* create by: weiepnglin

\* description: 添加一个女生

\* create time: 18:10 2018/8/31

\*

\* @return

\*/

@PostMapping(value = "/girls")

public Girl addGirl(@RequestParam("cupSize") String cupSize,

@RequestParam("age") Integer age){

Girl girl = new Girl ();

girl.setCupSize(cupSize);

girl.setAge(age);

return girlRepository.save(girl);

}

/\*\*

\* c查询一个女生

\* @param id

\* @return

\*/

@GetMapping(value = "/girls/{id}")

public Optional<Girl> queryOneGirl(@PathVariable("id") Integer id){

Optional<Girl> girl = girlRepository.findById(id);

return girl;

}

/\*\*

\* 更新一个女生

\* @param id

\* @param cupSize

\* @param age

\*/

@PutMapping(value = "/girls/{id}")

public Girl updateOneGirl(@PathVariable("id") Integer id,

@RequestParam("cupSize") String cupSize,

@RequestParam("age") Integer age){

Girl girl = new Girl ();

girl.setId(id);

girl.setCupSize(cupSize);

girl.setAge(age);

return girlRepository.save(girl);

}

//删除一个女生

@DeleteMapping(value = "/girls/{id}")

public void deleteOneGirl(@PathVariable("id") Integer id){

try {

girlRepository.deleteById(id);

System.out.println("删除成功!");

} catch (Exception e) {

System.out.println("删除失败!");

e.printStackTrace();

}

}

还有个根据年龄来找女生，需要在GirlRepository声明一个接口

public interface GirlRepository extends JpaRepository<Girl ,Integer> {

//通过年龄来查询 findByAgez这个名字固定,不能乱改

public List<Girl> findByAge(Integer age);

}

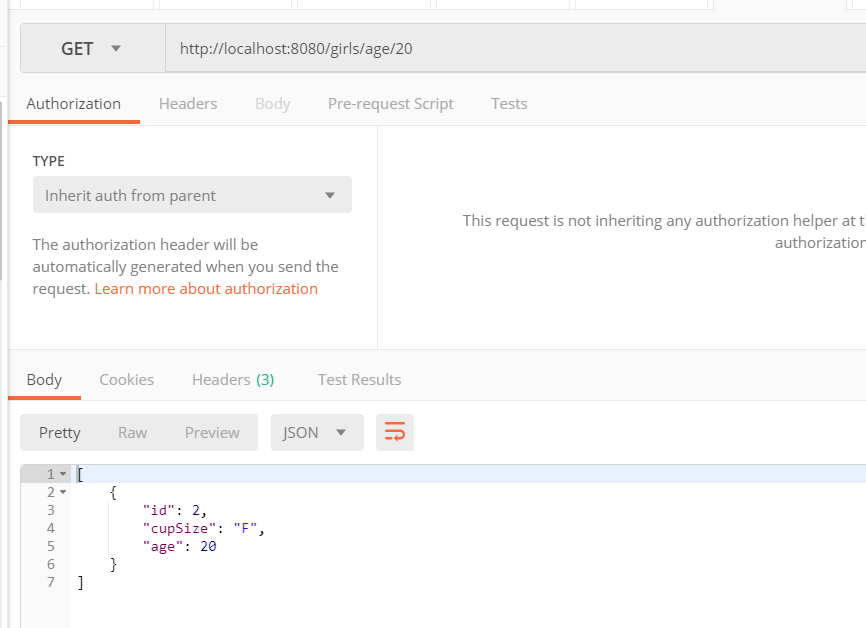


图4.9

### 事务管理

实现两个女生的插入

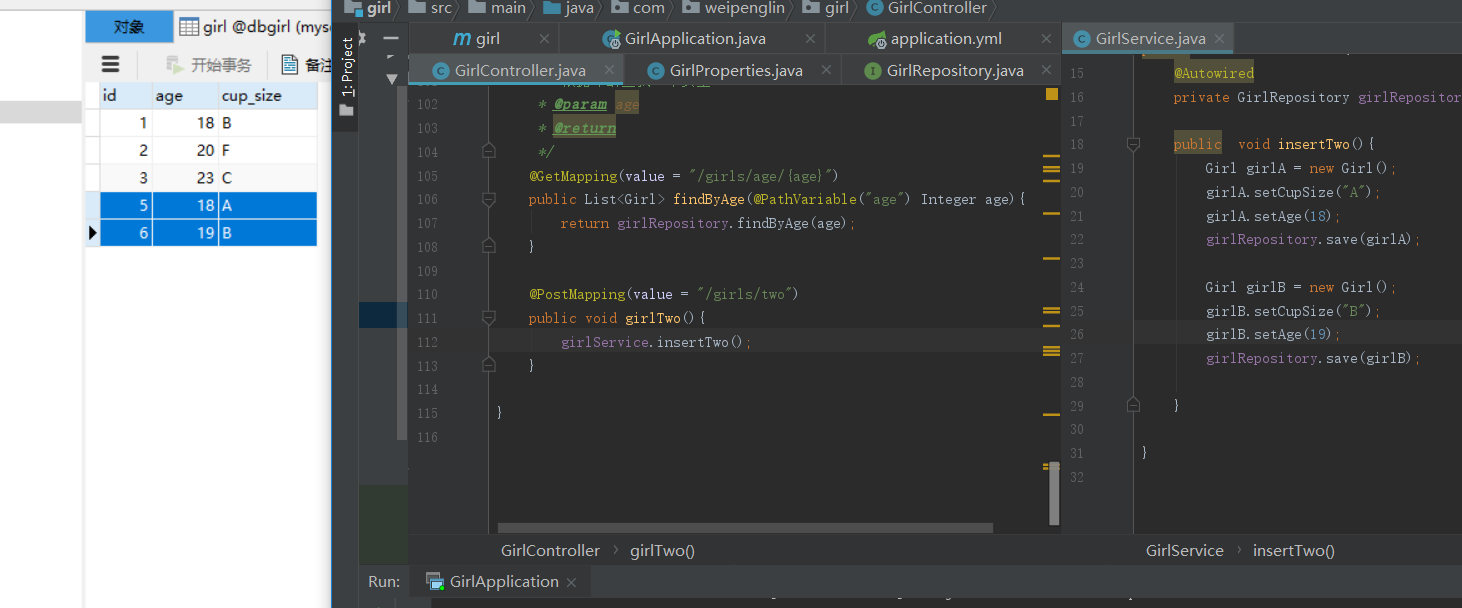


图5.1

修改Girl表中的cupSize字段长度，再给GirlService的insertTwo方法加上@Transactional

然后测试一下

如果出现事务不起作用，八成是数据库的引擎不是Innodb

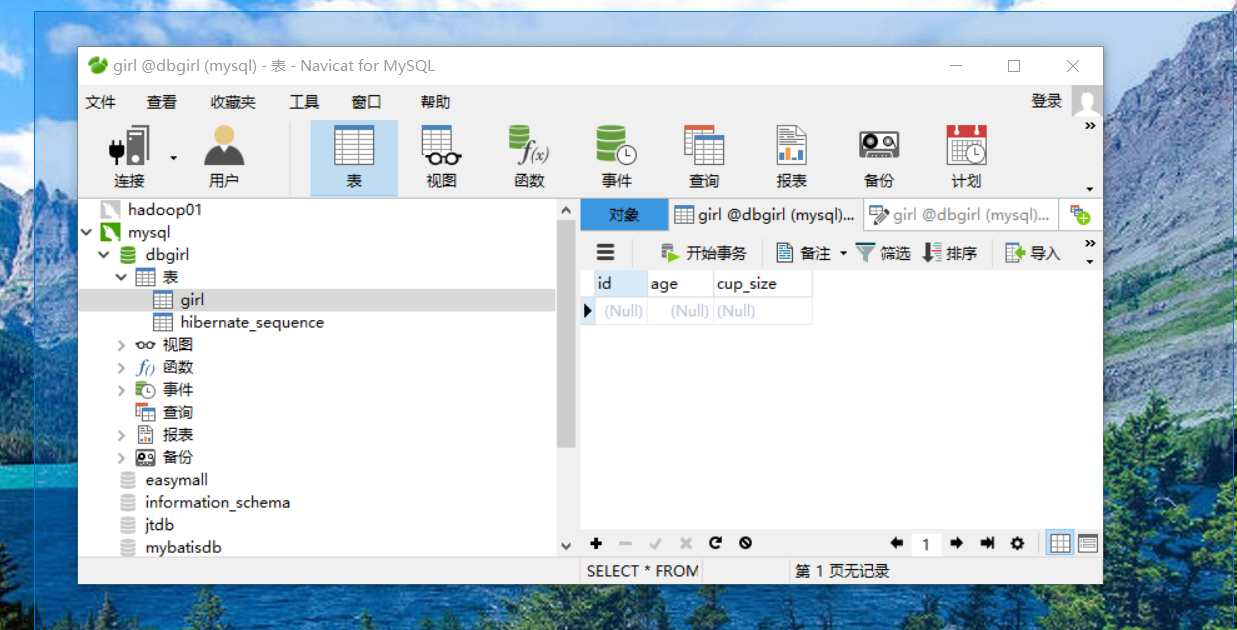
改过来就好了

<https://blog.csdn.net/BestLovelyMin/article/details/74942932>

<https://blog.csdn.net/leejianjun/article/details/72467983>

修改表的引擎

alter table hibernate\_sequence engine=innodb;



实测没问题