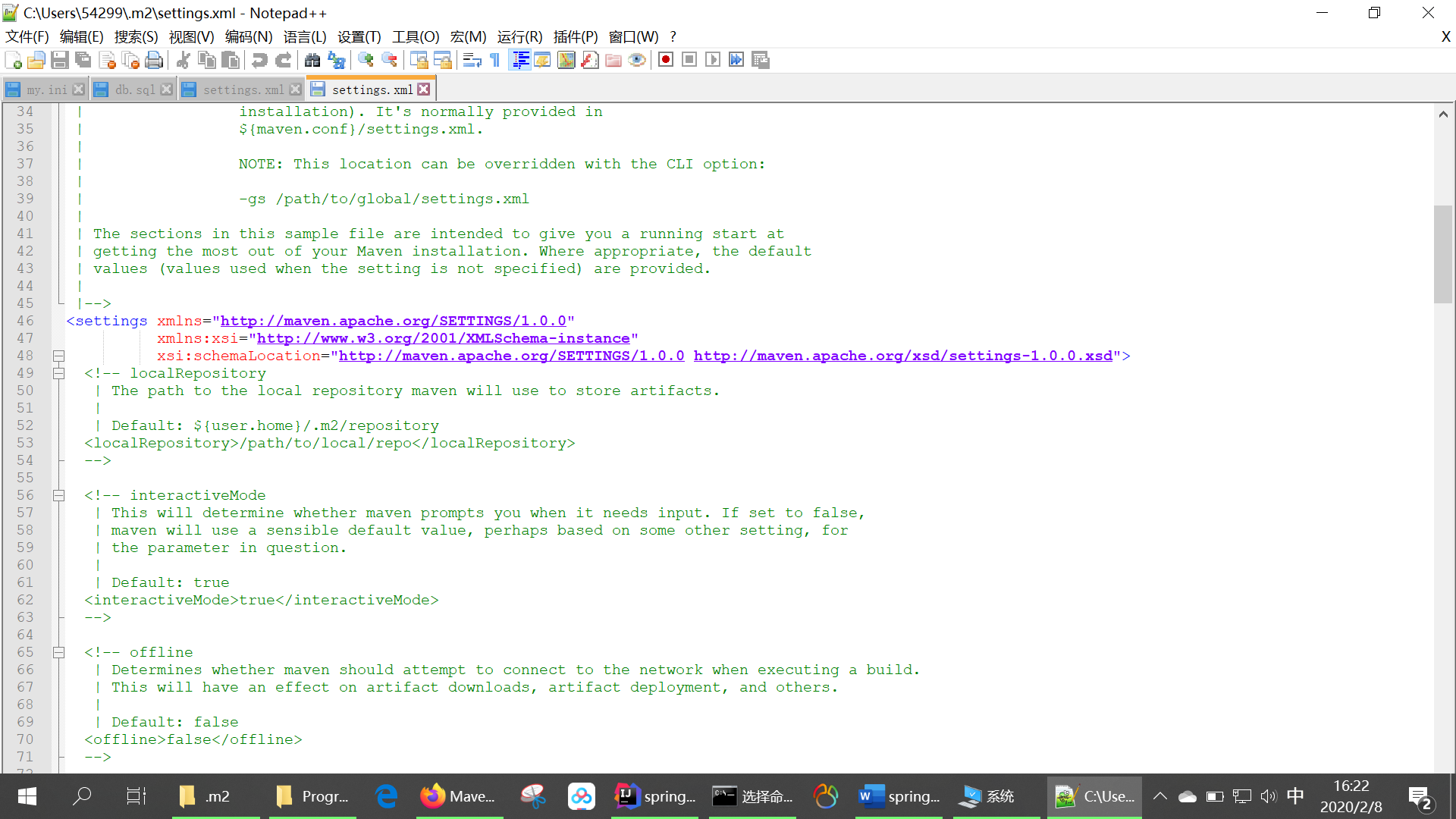


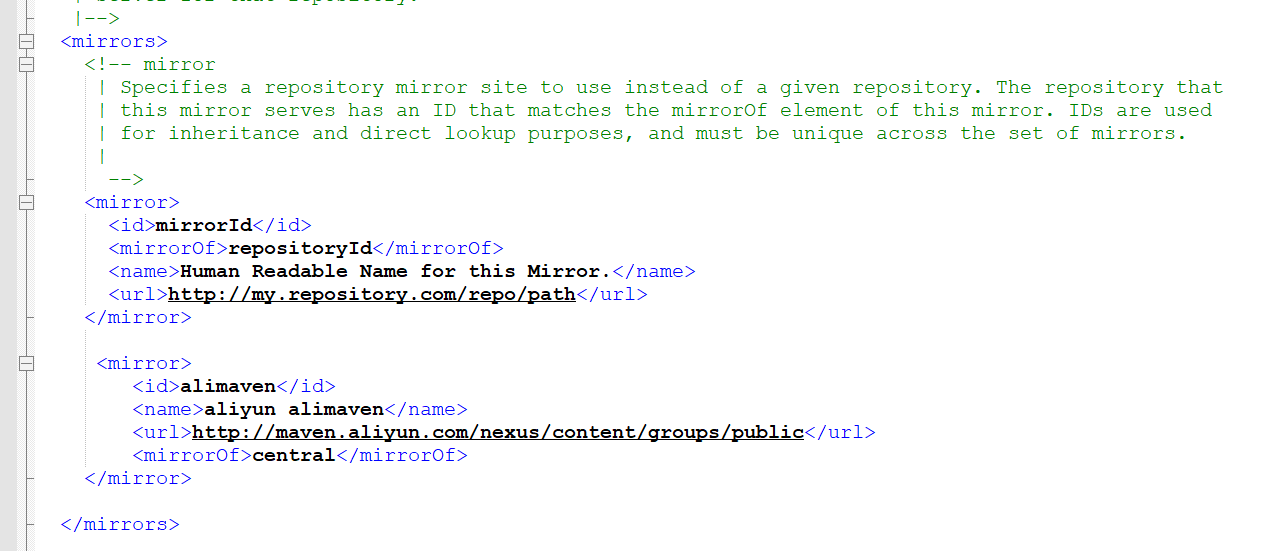
Idea读取maven时是读取setting.xml



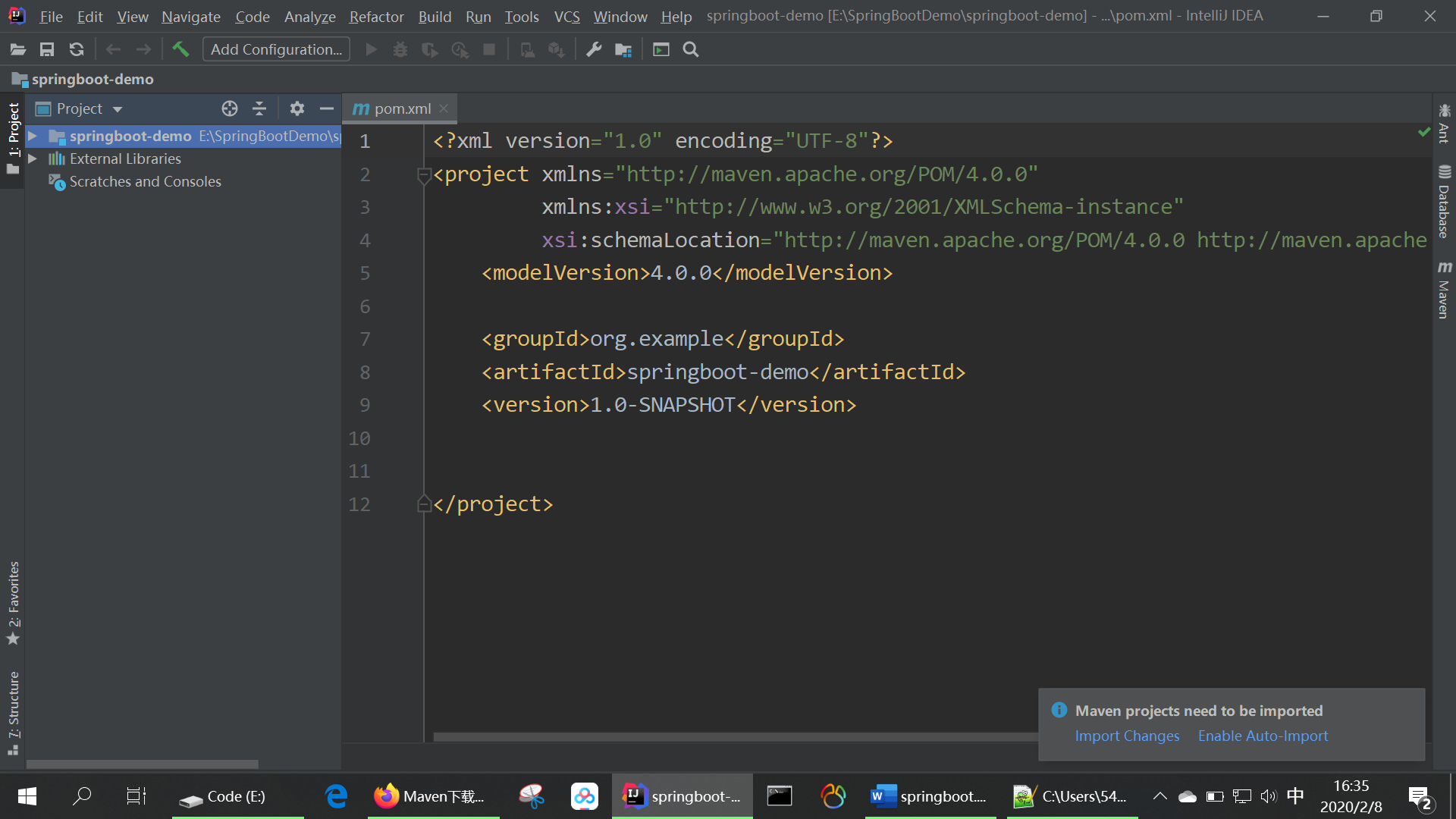
修改仓库的位置：



添加镜像，访问速度快一点

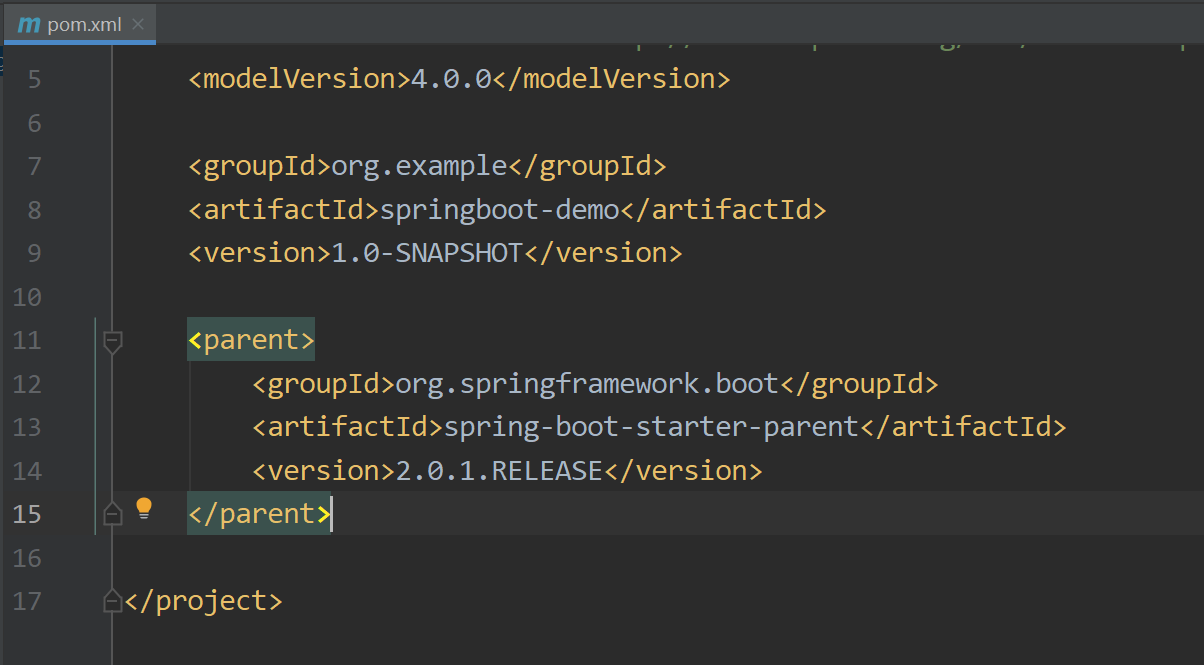


新创建一个maven工程



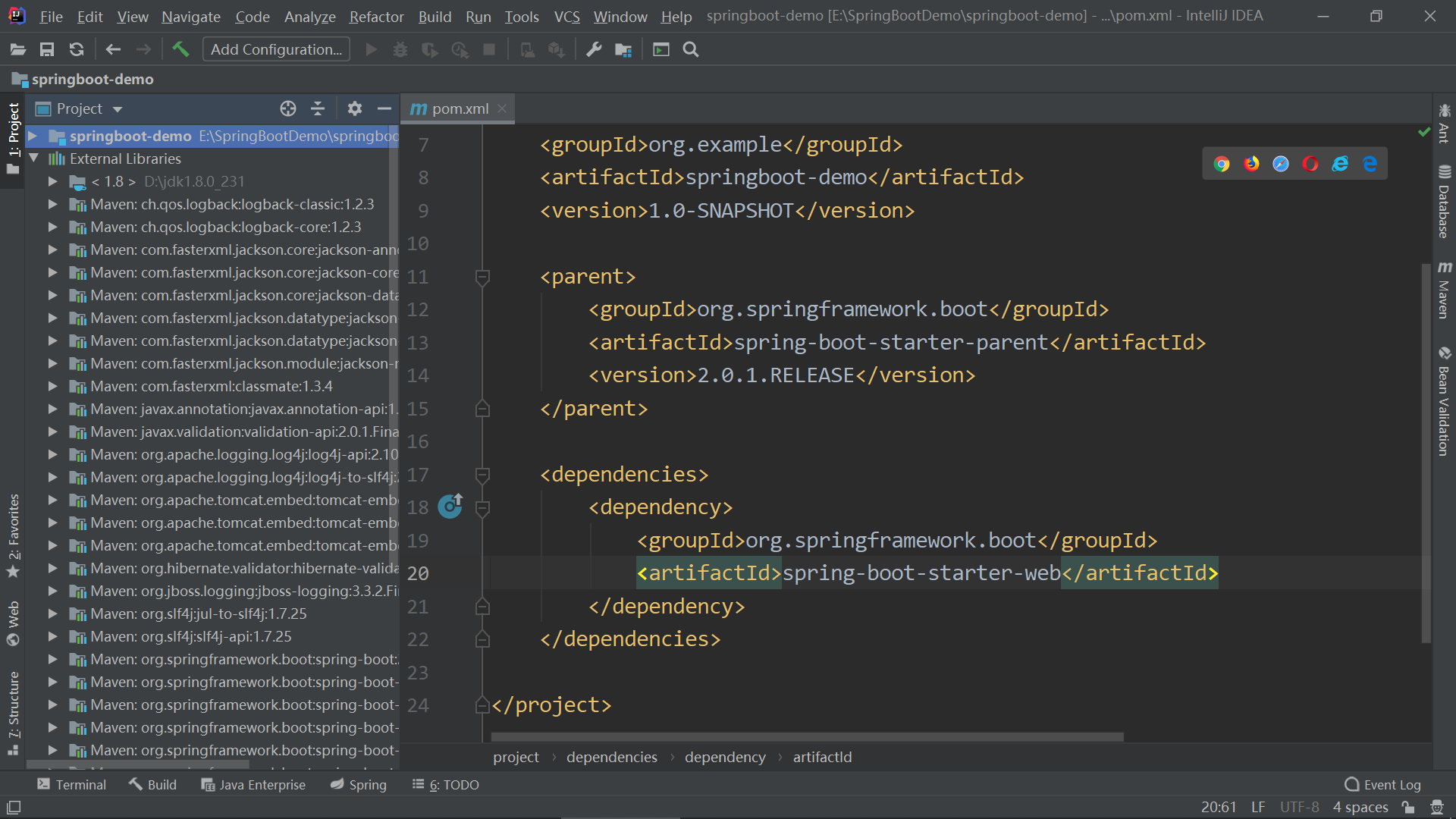
1.2添加依赖

1.2.1添加父类工程坐标



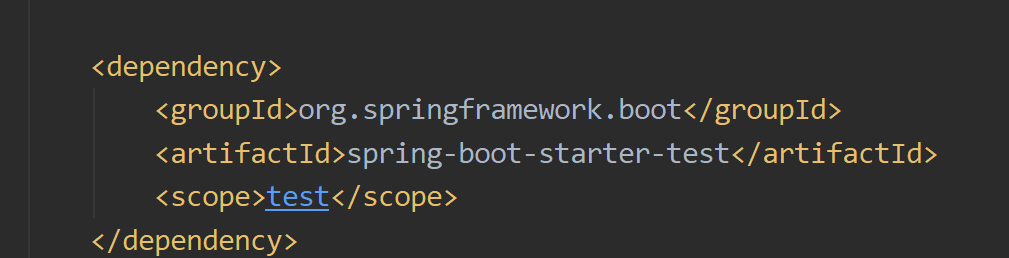
1.2.2 添加web启动器

为了让SpringBoot帮我们完成各种自动配置，我们引入SpringBoot提供的启动配置依赖，我们称之启动器，因为我们是Web项目，这里我们引入web启动器：



（自动引入架包）

1.2.3添加单元测试



2.2.4管理jdk版本

默认情况下，maven工程的jdk版本是1.5，而我们开发使用的是1.8，所以这里需要修改jdk版本，只需要简繁的添加以下属性即可：



2.2.5完整的pom

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>

<artifactId>springboot-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<parent>

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

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

<version>2.0.1.RELEASE</version>

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

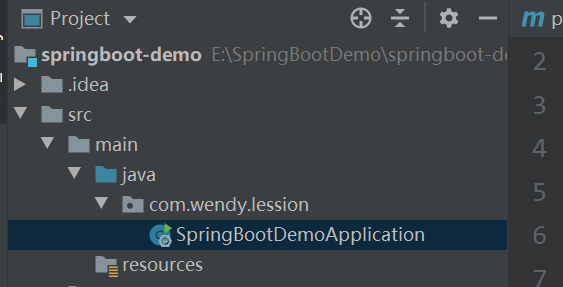
</dependencies>

</project>

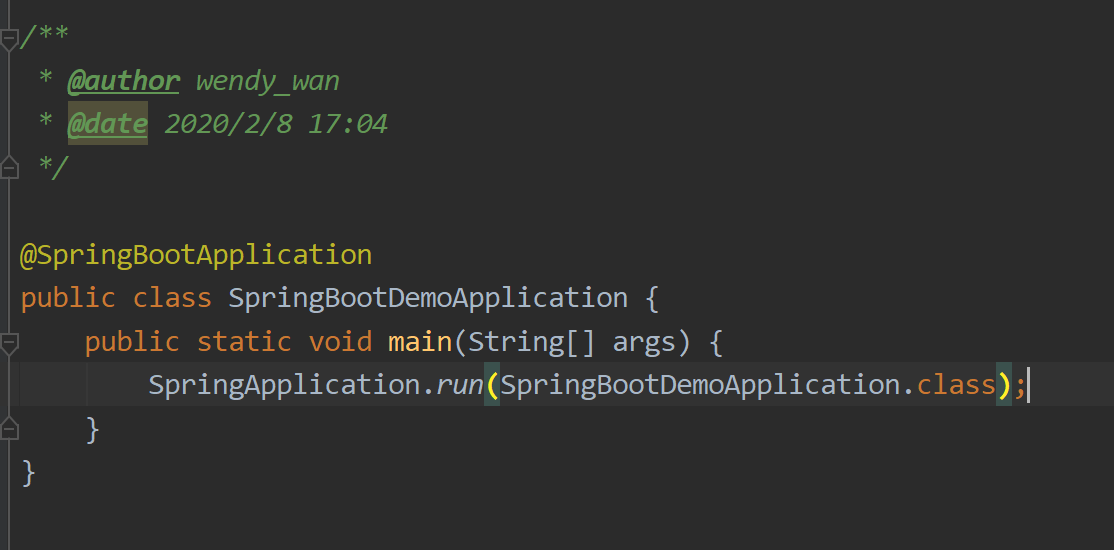
2.3启动类

Spring Boot项目通过main函数即可启动，我们需要创建一个启动类：

（启动类必须放在包目录下）

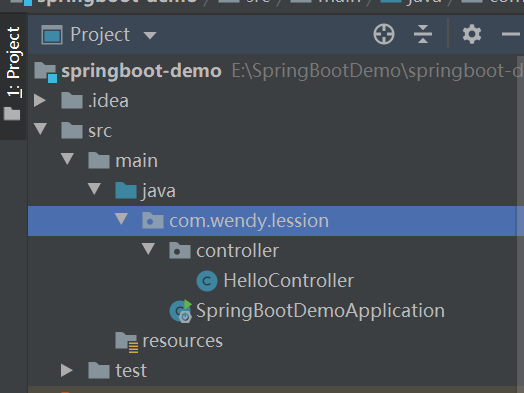


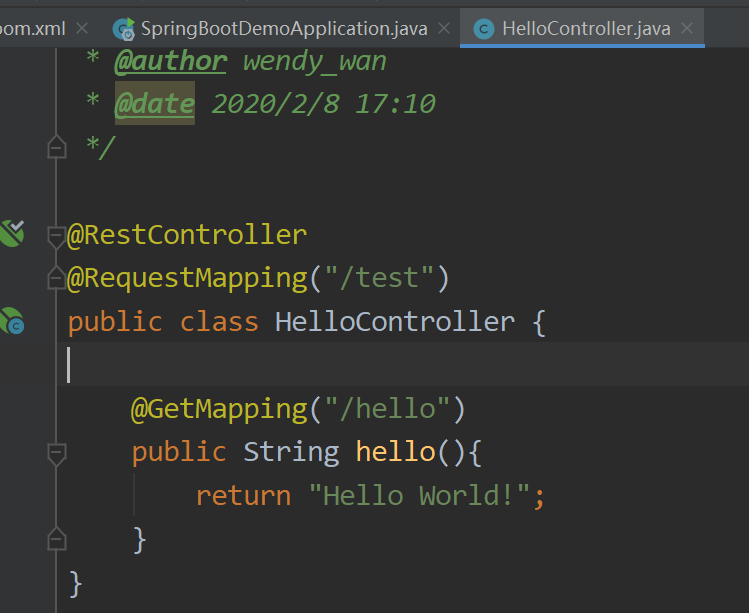
然后填写main函数：



2.4编写测试类controller

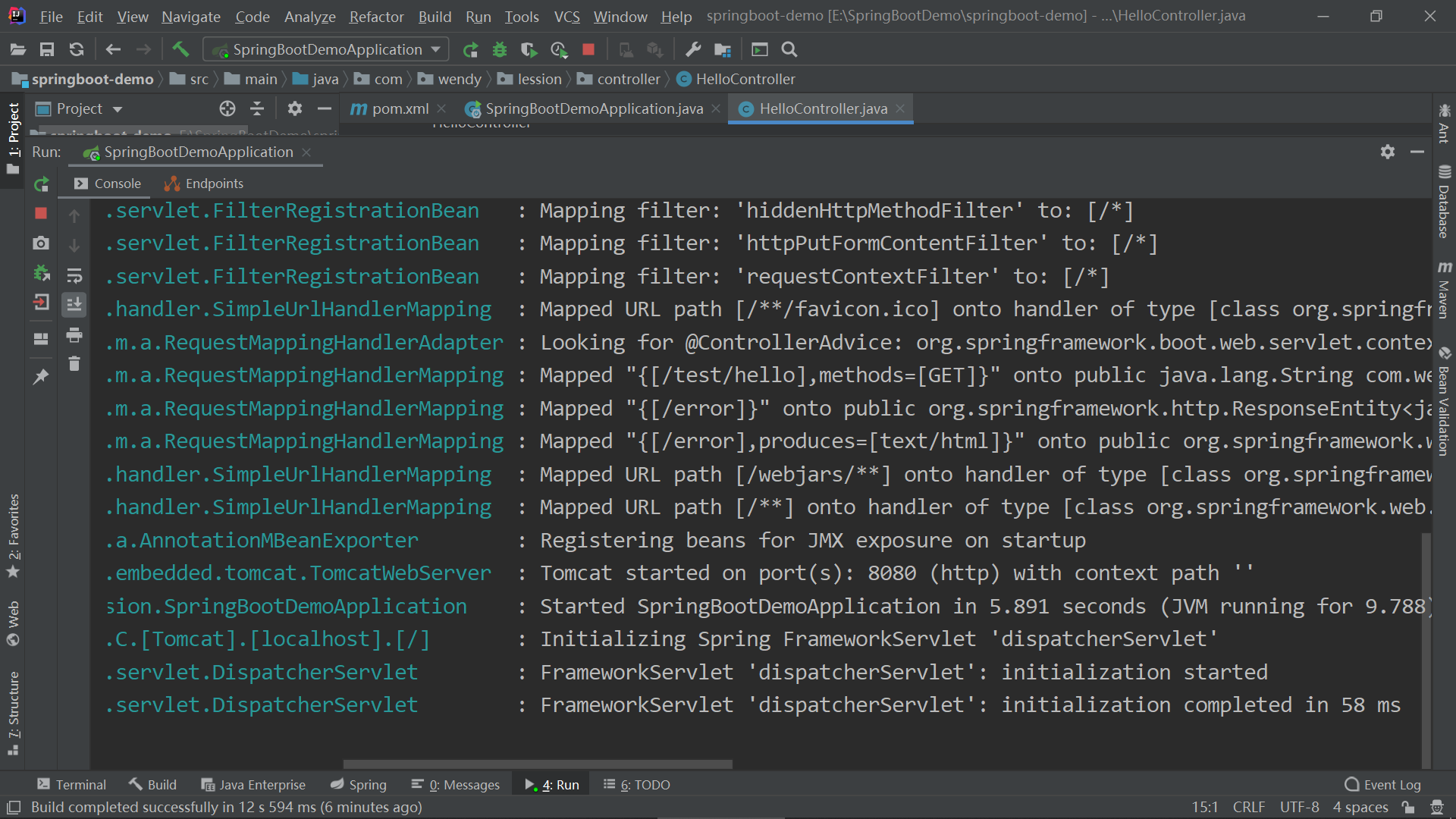
1、创建HelloController.java



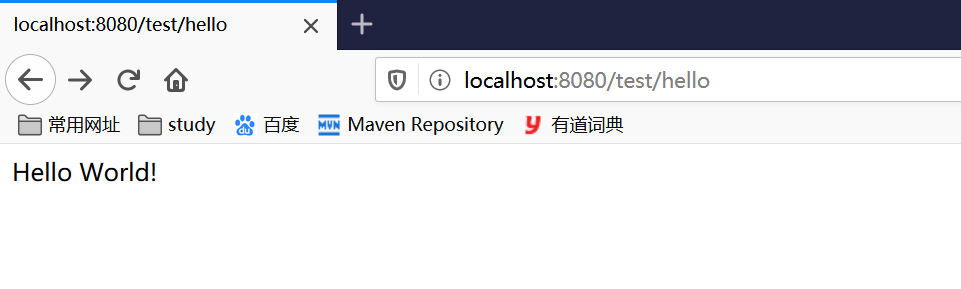


2.5启动测试

运行main函数，查看控制台：



浏览器访问：http://localhost:8080/test/hello



@RestController注解相当于@Controller+@ResponseBody合在一起的作用，如果Web层的类上使用了@RestController注解，就代表这个类中所有的方法都会以JSON的形式返回结果，也相当于JSON的一种快捷使用方式。

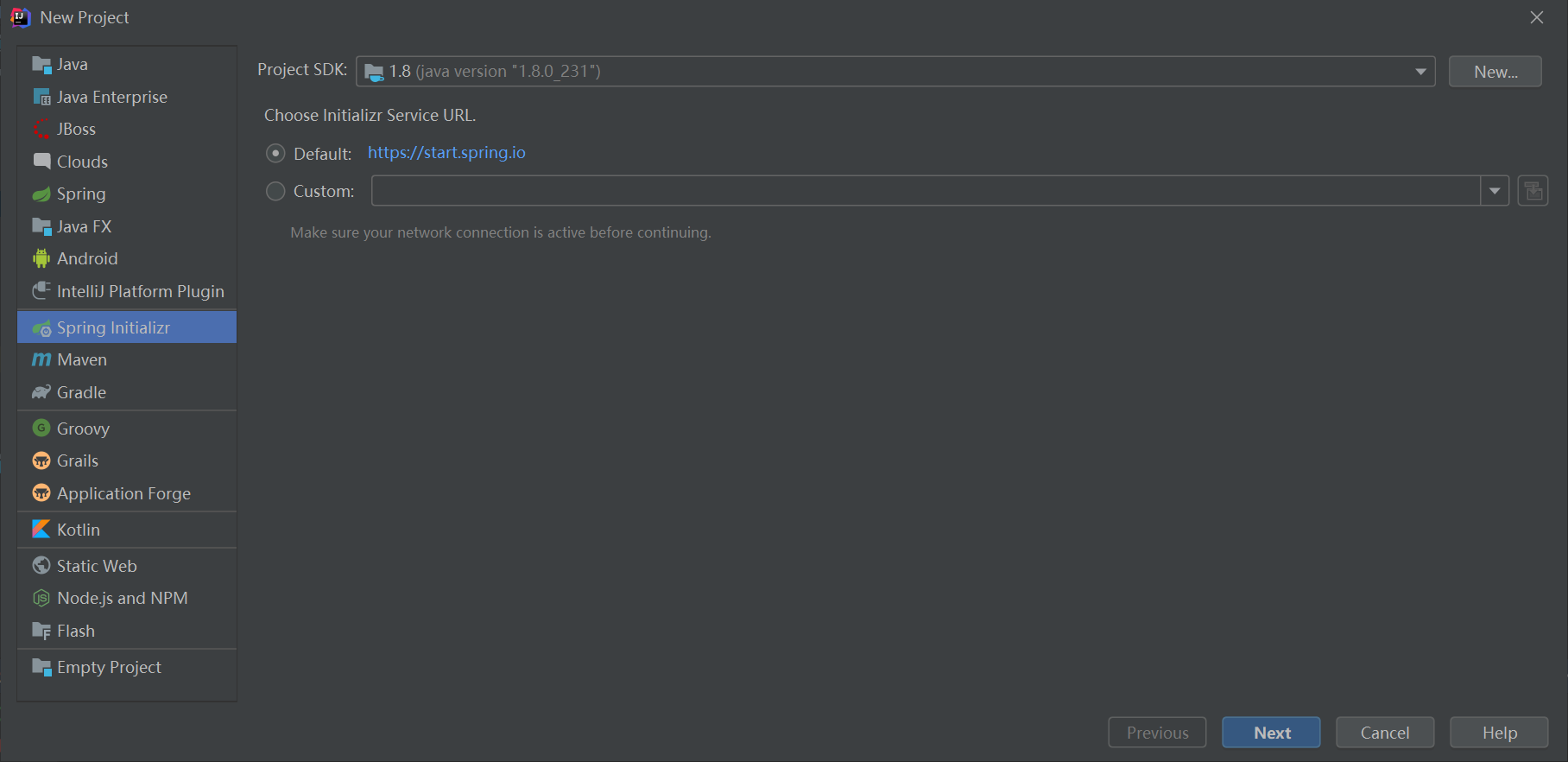
@GetMapping("/hello")是一个组合注解，等于@RequestMapping(name = "/hello",method = RequestMethod.*GET*)的缩写，以/hello的方式请求。

如果@PostMapping则为method = RequestMethod.*POST* 是指只可以使用Post的方式去请求，如果使用Get的方式请求的话，则会报405不允许访问的错误。@PutMapping，@DeleteMapping则method会以此类推。

3.Spring工具创建Spring Boot2.x web应用

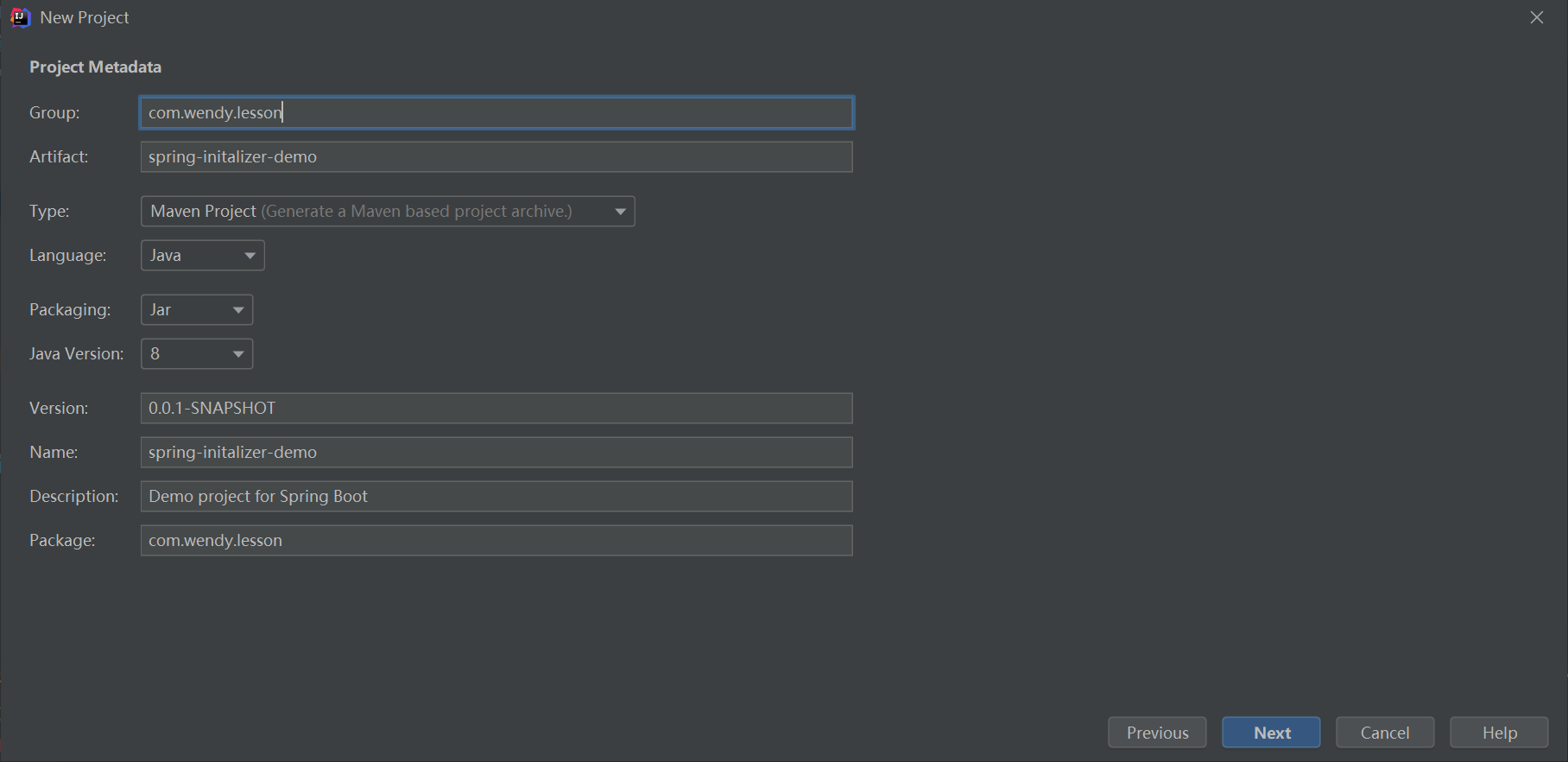
1、创建一个新的工程

2、使用Spring initial来构建



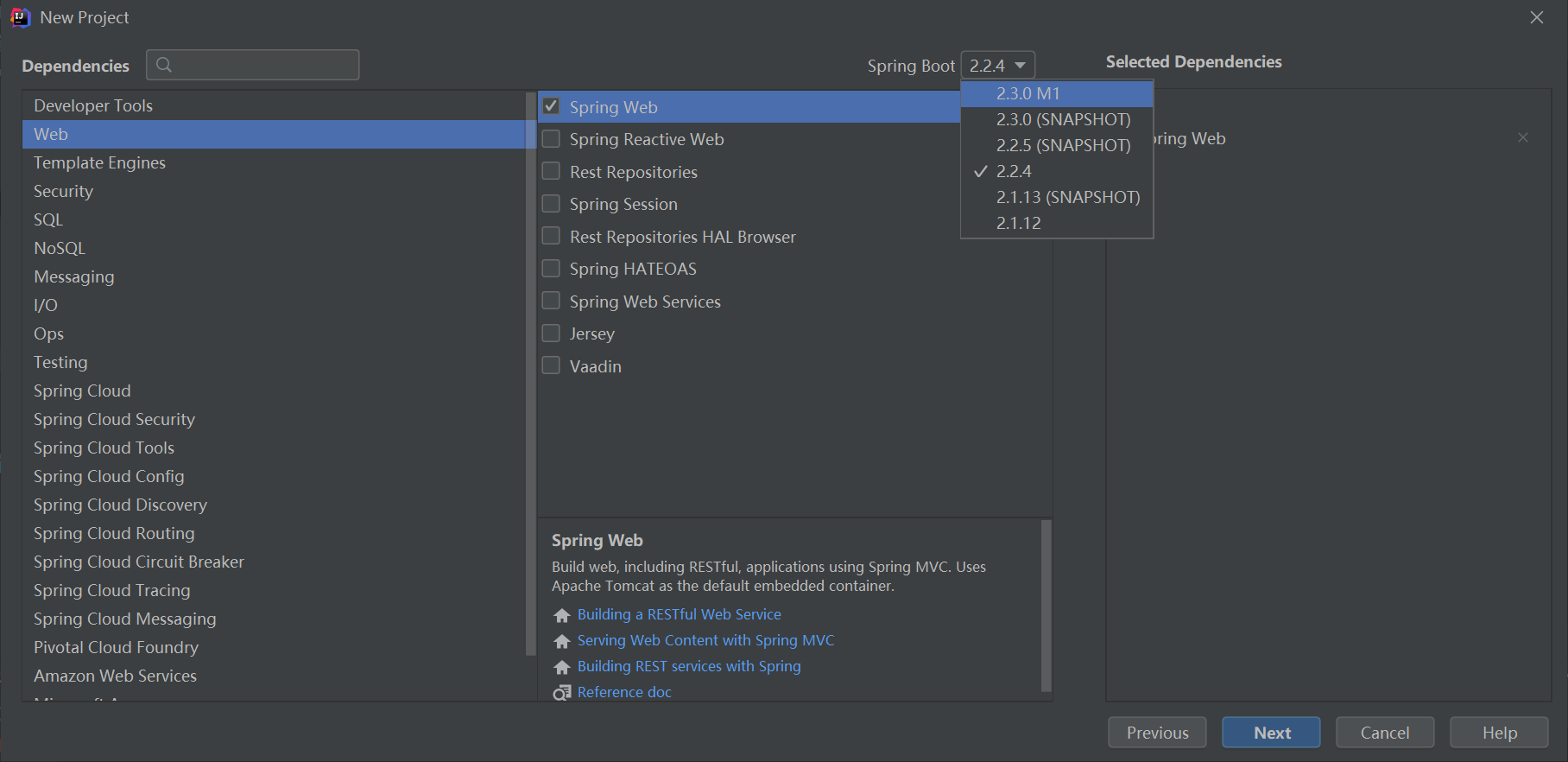
点击next

3、填写坐标



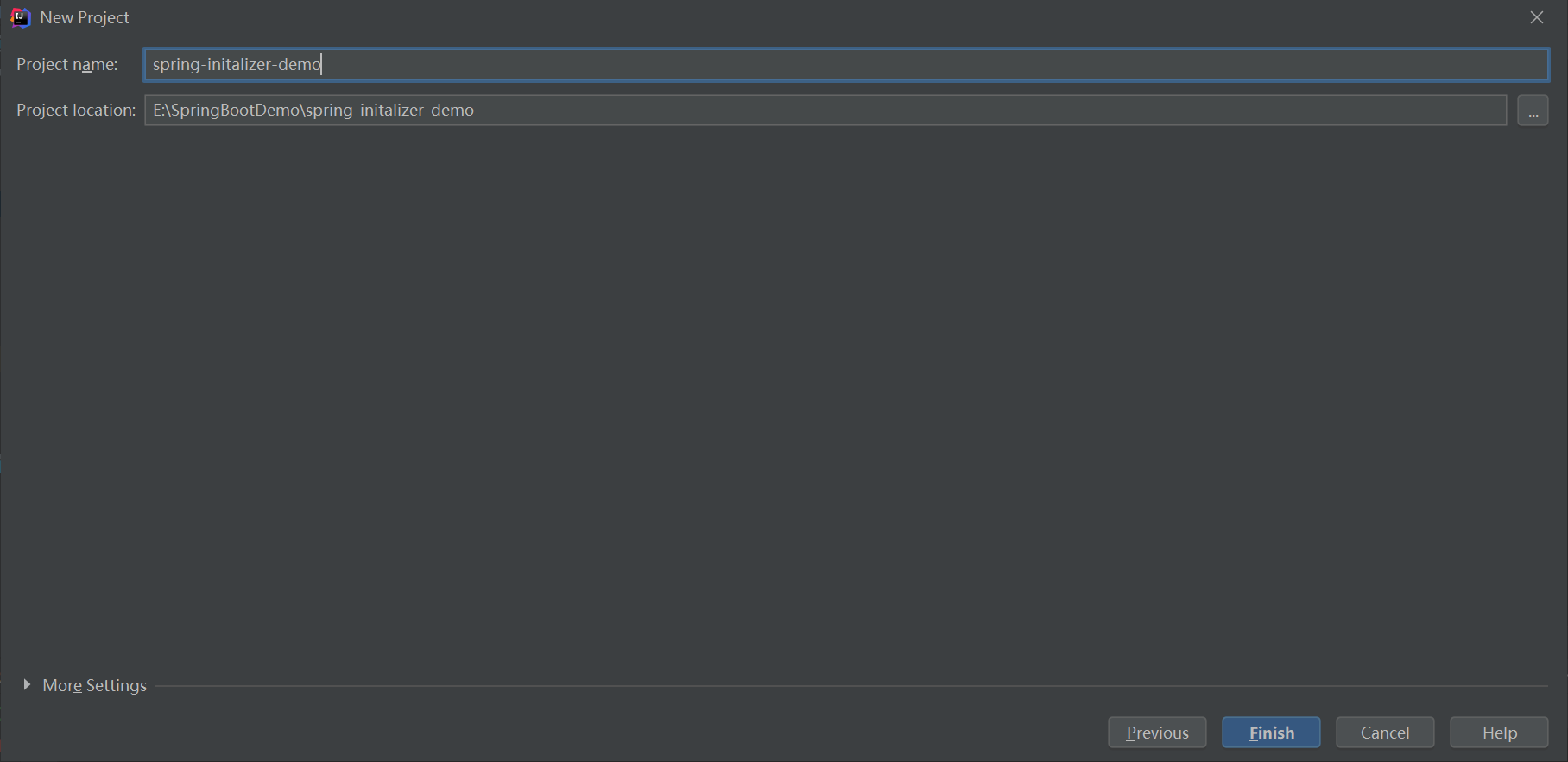
(填写信息，Artfact项目名)

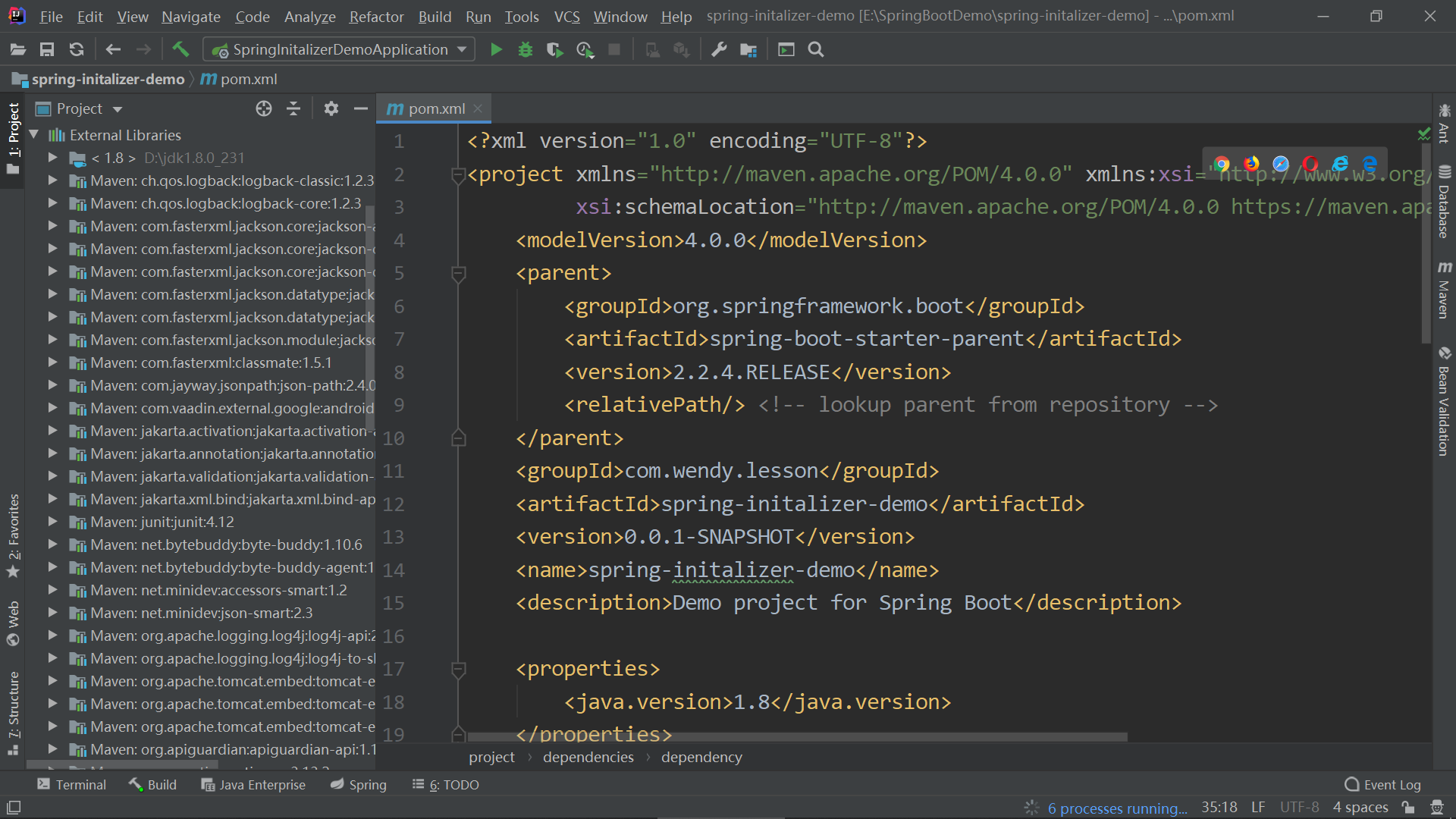
4、选择web容器



（选择web，可以自动导入架包）

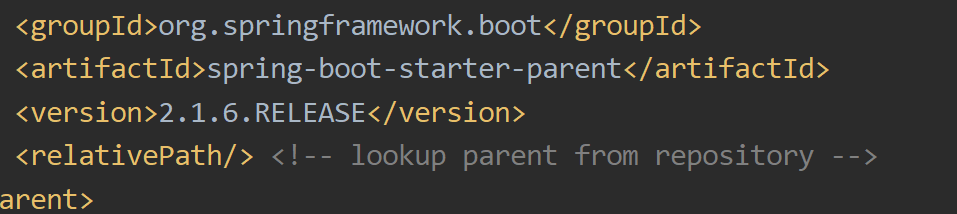
5、目录结构



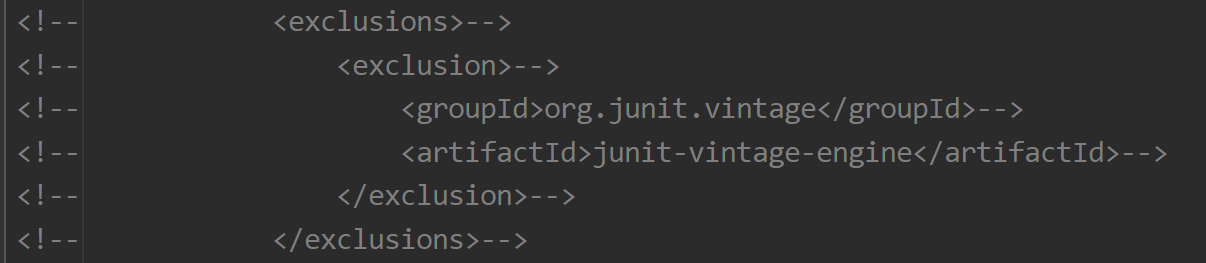


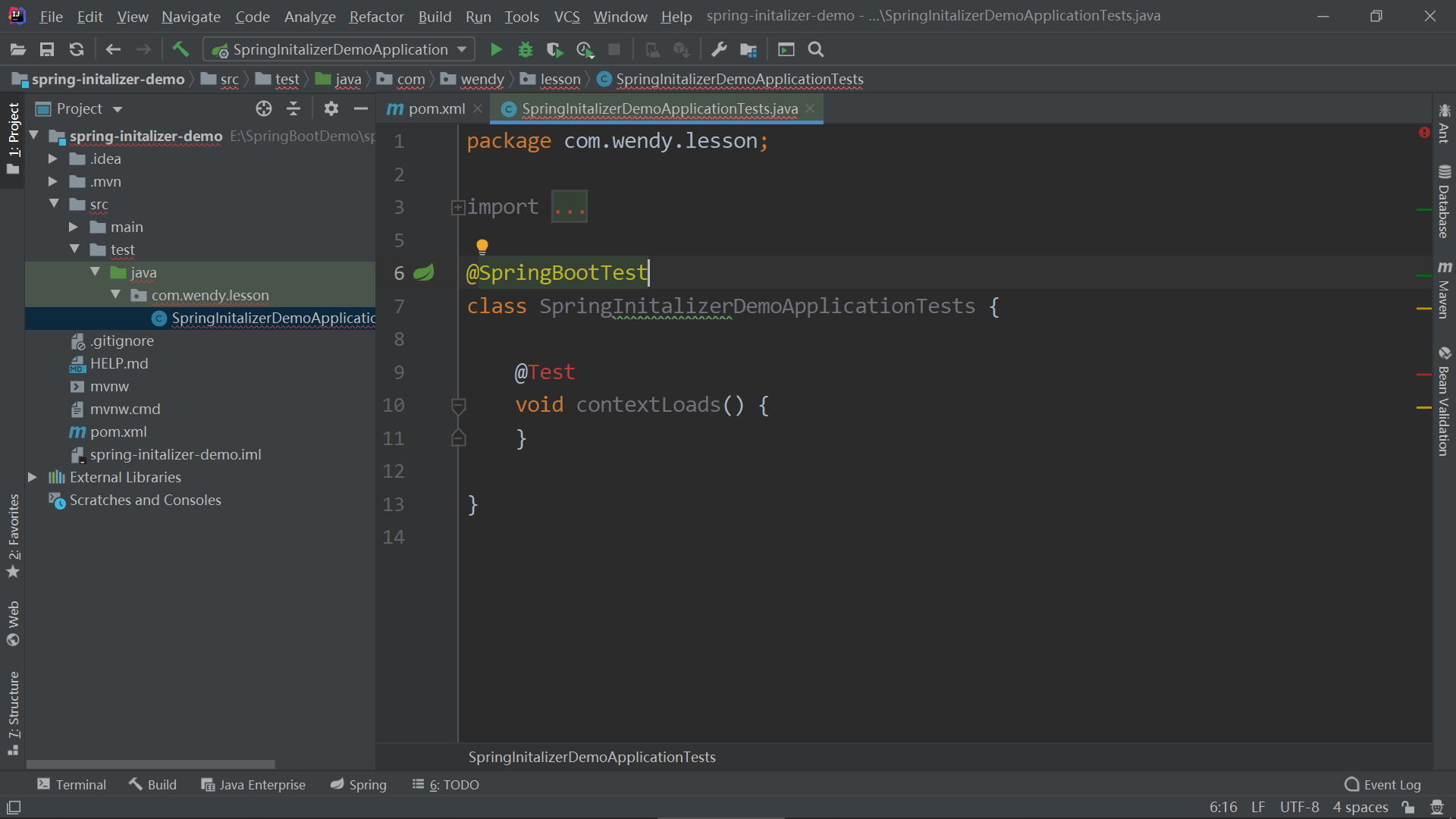
架包已经自动导入

版本号改成2.1.6

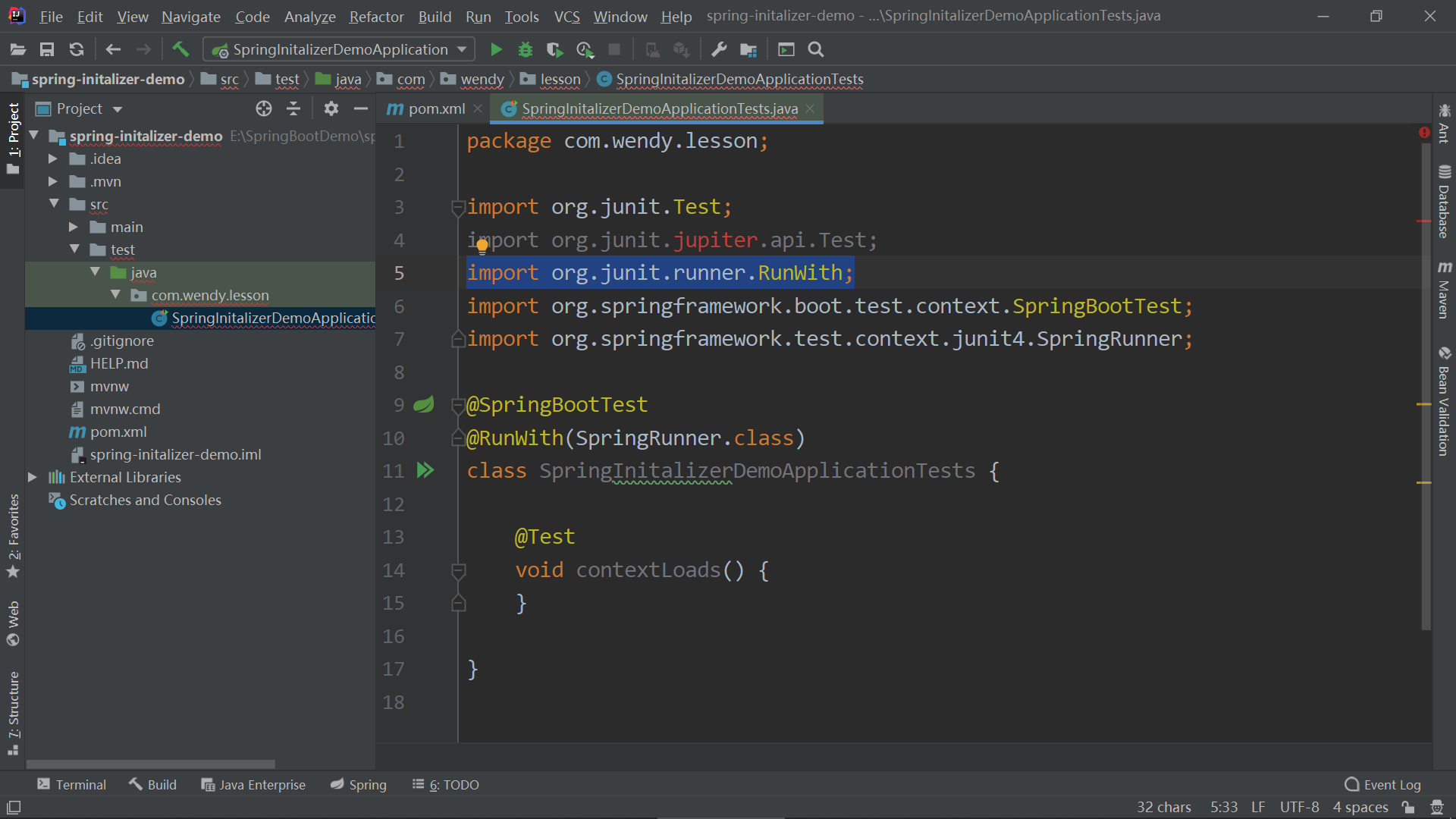


删除自带的单元测试

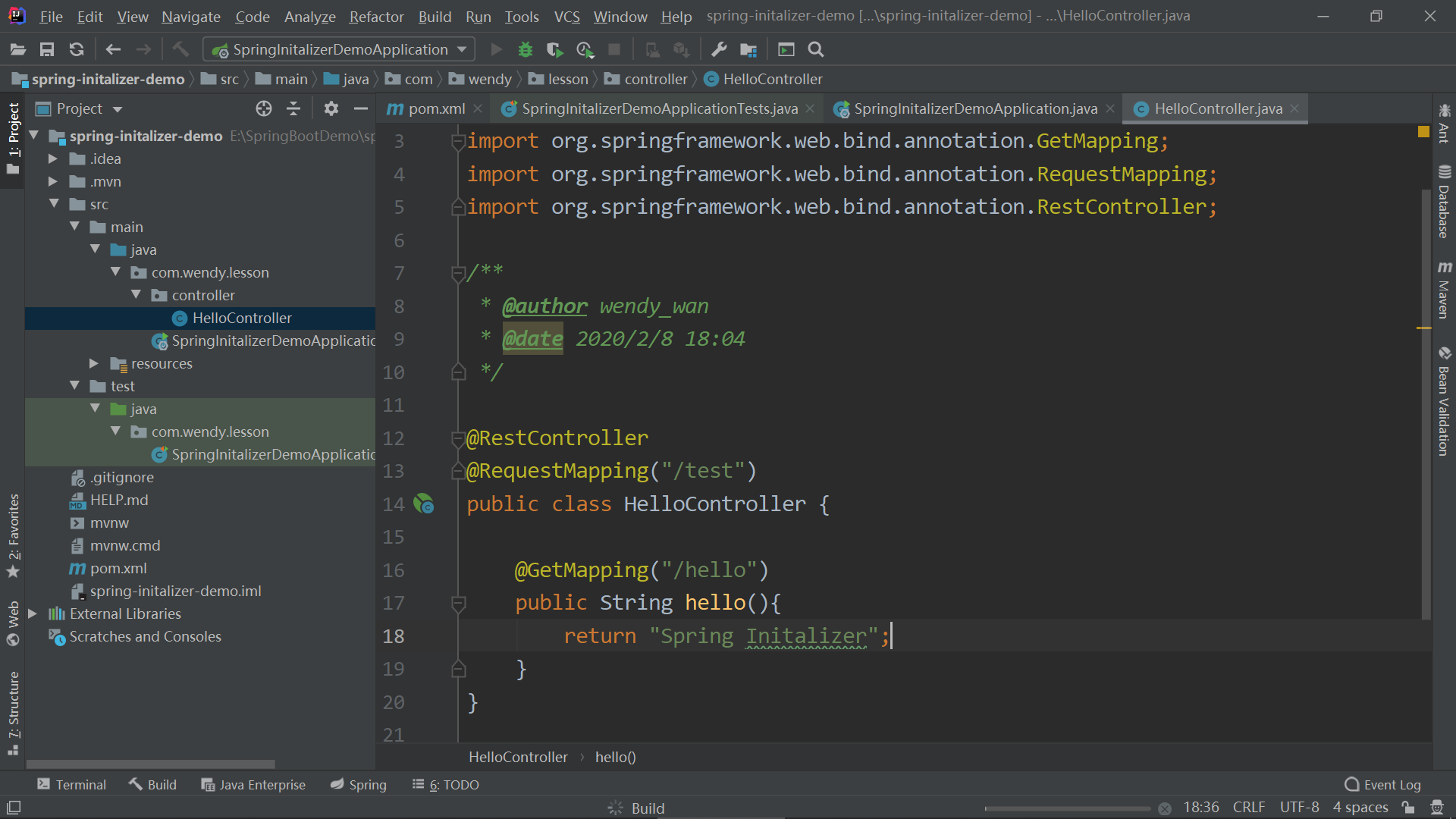


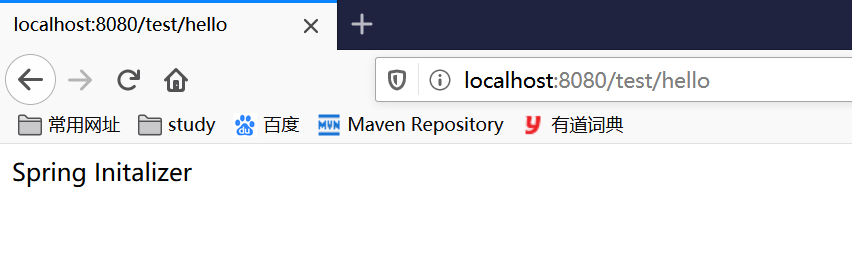


修改单元测试代码：



Web工程已经创建好了





pom文件为基本的依赖管理文件

resource资源文件

static静态资源

templates模板资源

application.properties配置文件

SpringinitalizerDemoApplication程序入口

完整版的pom

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.1.6.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.wendy.lesson</groupId>  
 <artifactId>spring-initalizer-demo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>spring-initalizer-demo</name>  
 <description>Demo project for Spring Boot</description>  
  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
<!-- <exclusions>-->  
<!-- <exclusion>-->  
<!-- <groupId>org.junit.vintage</groupId>-->  
<!-- <artifactId>junit-vintage-engine</artifactId>-->  
<!-- </exclusion>-->  
<!-- </exclusions>-->  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**4、maven构建多模块父子项目**

4.1Dependencies&DependencyMangement的区别

dependencies:即使在子项目中不写该依赖项，那么子项目仍然会从父项目中继承依赖项（全部继承）

dependencyMangement：只是声明依赖，并不实现引入，因此子项目需要显示的声明需要用的依赖，如果不在子项目中声明依赖，是不会从父项目中继承下来的，只有在子项目中写了该依赖项，并且没有指定具体版本，才会从父项目中继承该项，并且version和scope都读取自父pom，另外如果子项目中指定了版本号，那么会使用子项目中指定的jar版本。

4.2工程目录结构

wendy-service: 服务层

wendy-web ：控制层

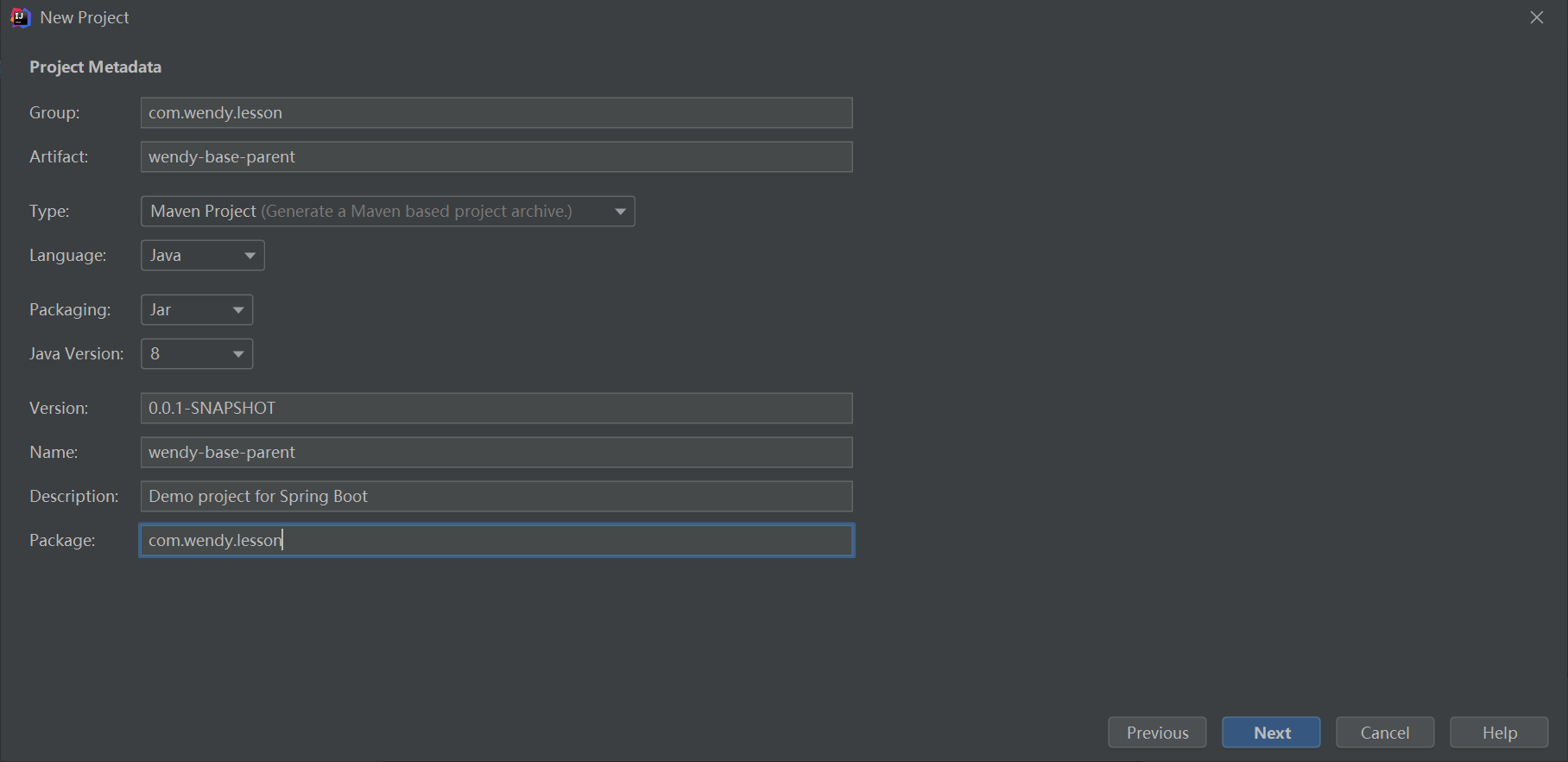
4.3搭建步骤

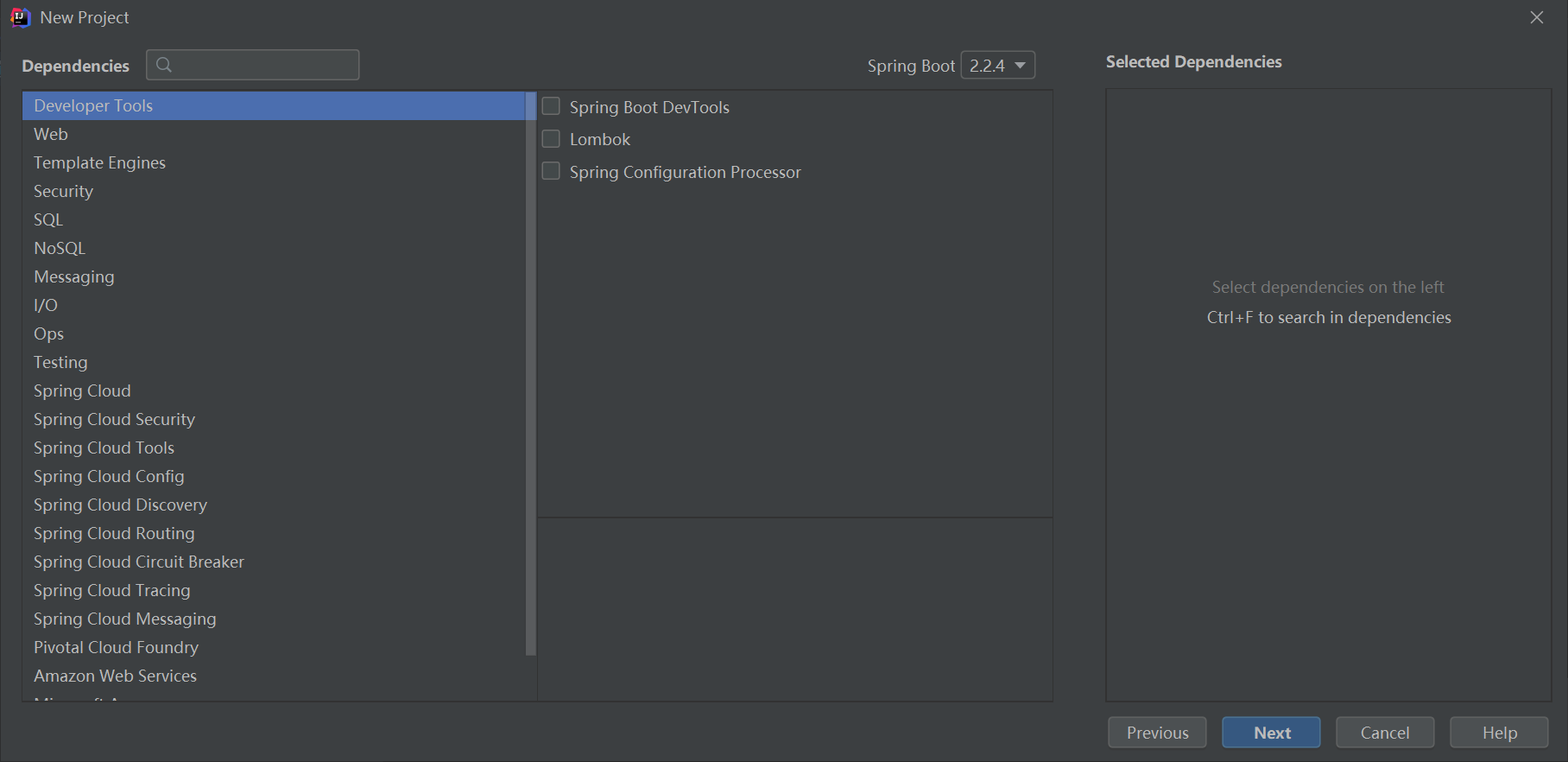
由于我们是多模块的。所以我们抛出一个wendy-base-parent来管理子项目的公共的依赖，在我们项目顶层的POM文件中，我们会看到DependencyMangement元素，通过它元素来管理jar包的版本，让子项目中引用一个依赖而不用显示的列出版本号，Maven会沿着父子层次向上走，直到找到一个拥有DependencyMangement元素的项目，然后它会使用在这个DependencyMangement元素中的指定版本号。

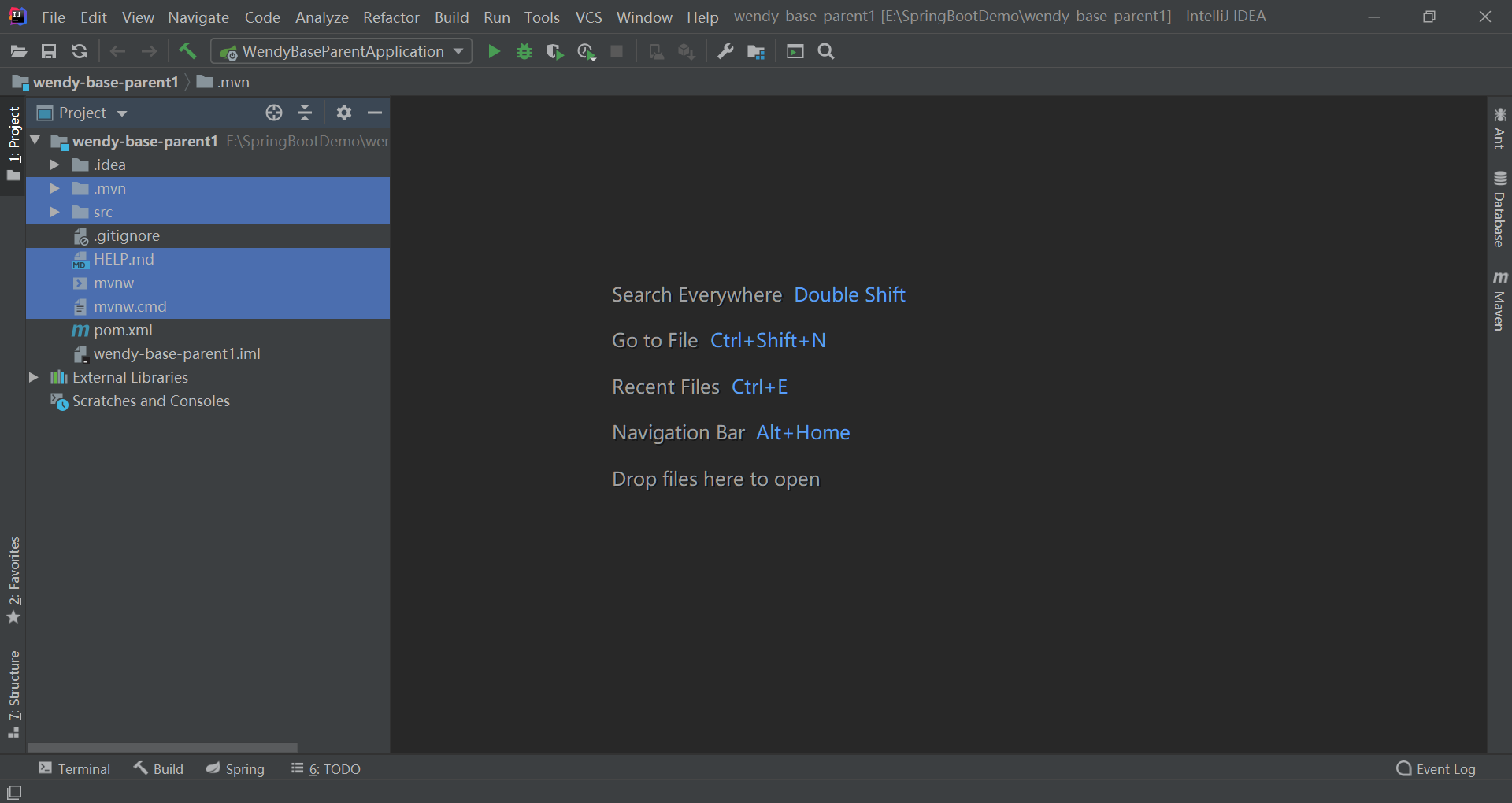
4.3.1父类工程

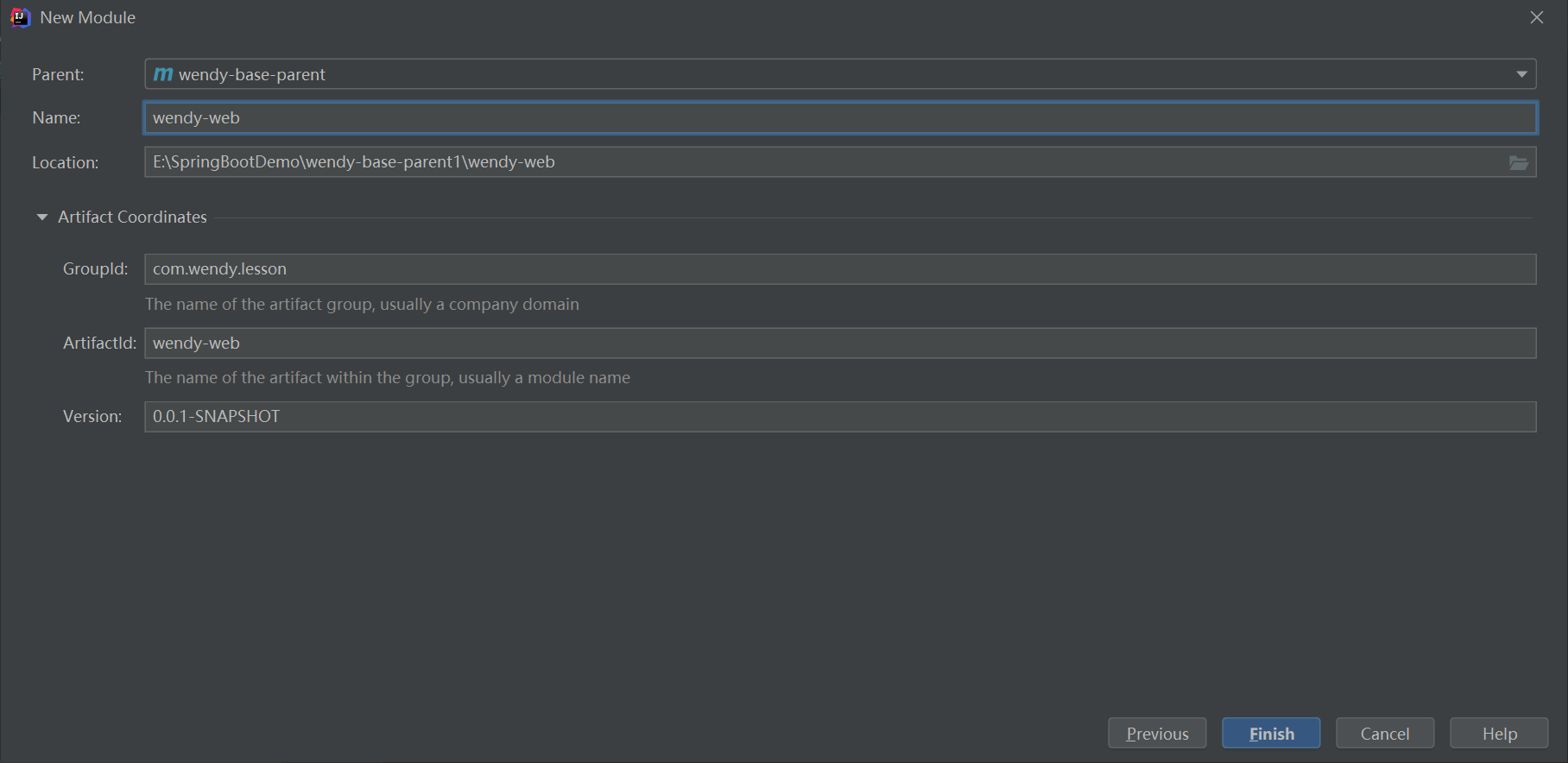
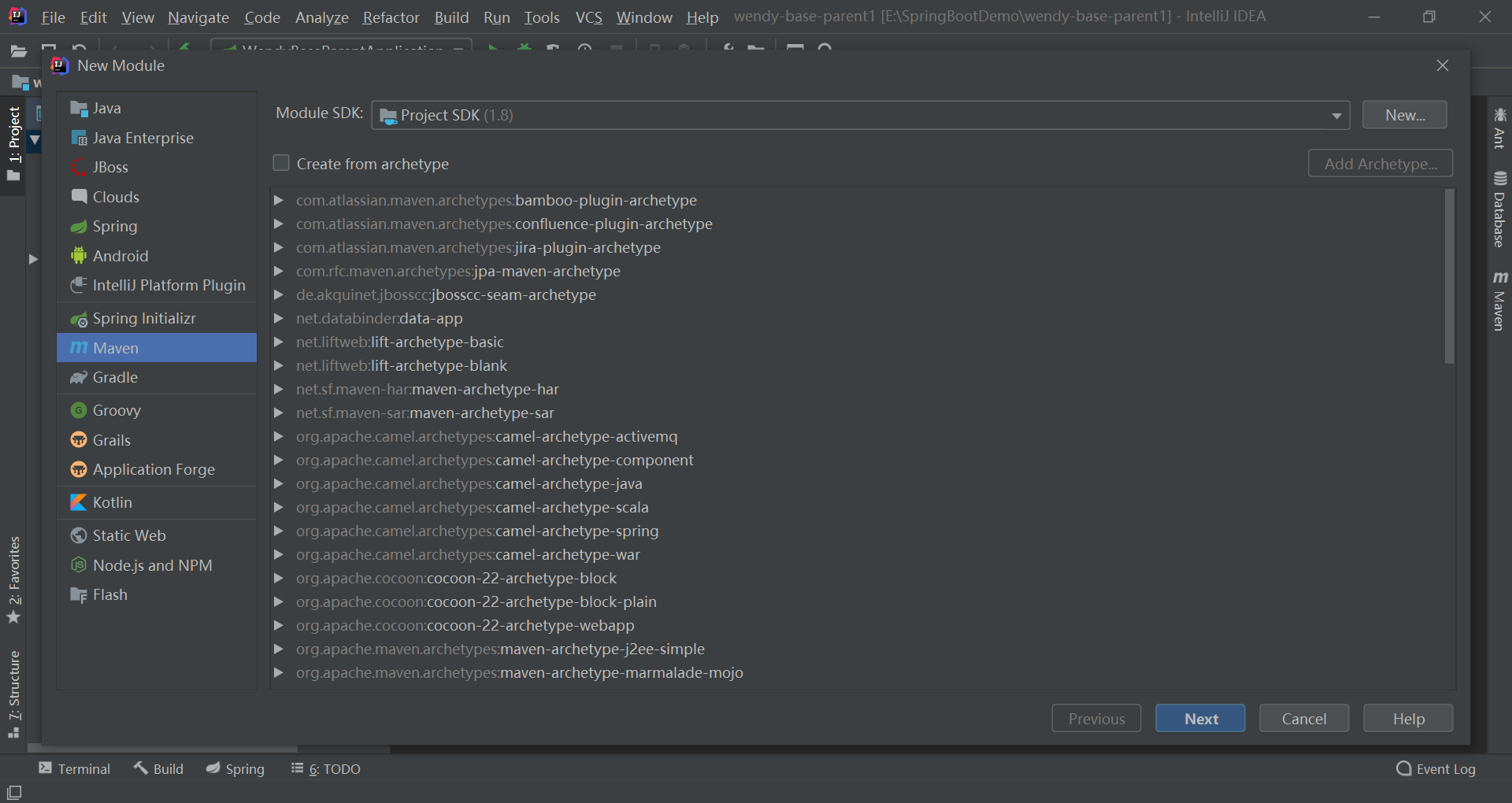
1、IDEA工具栏选择创建一个项目

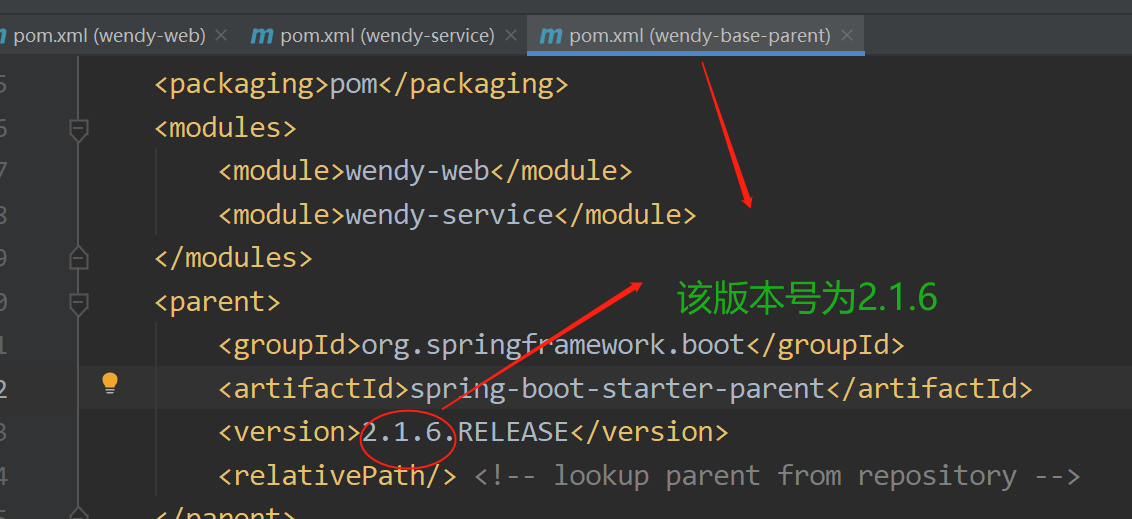
2、选择Spring initializr默认选择Dafault，点击next





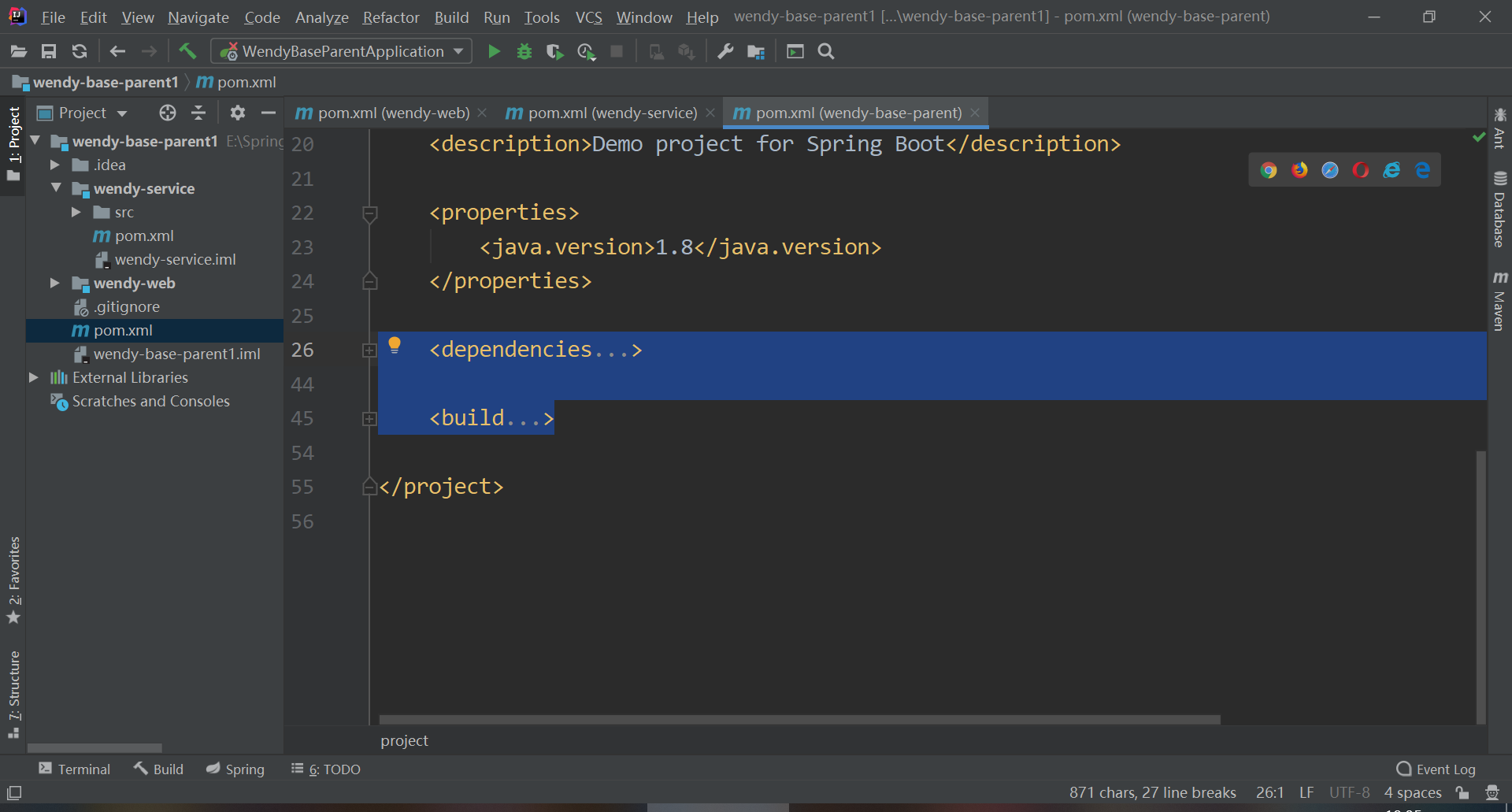


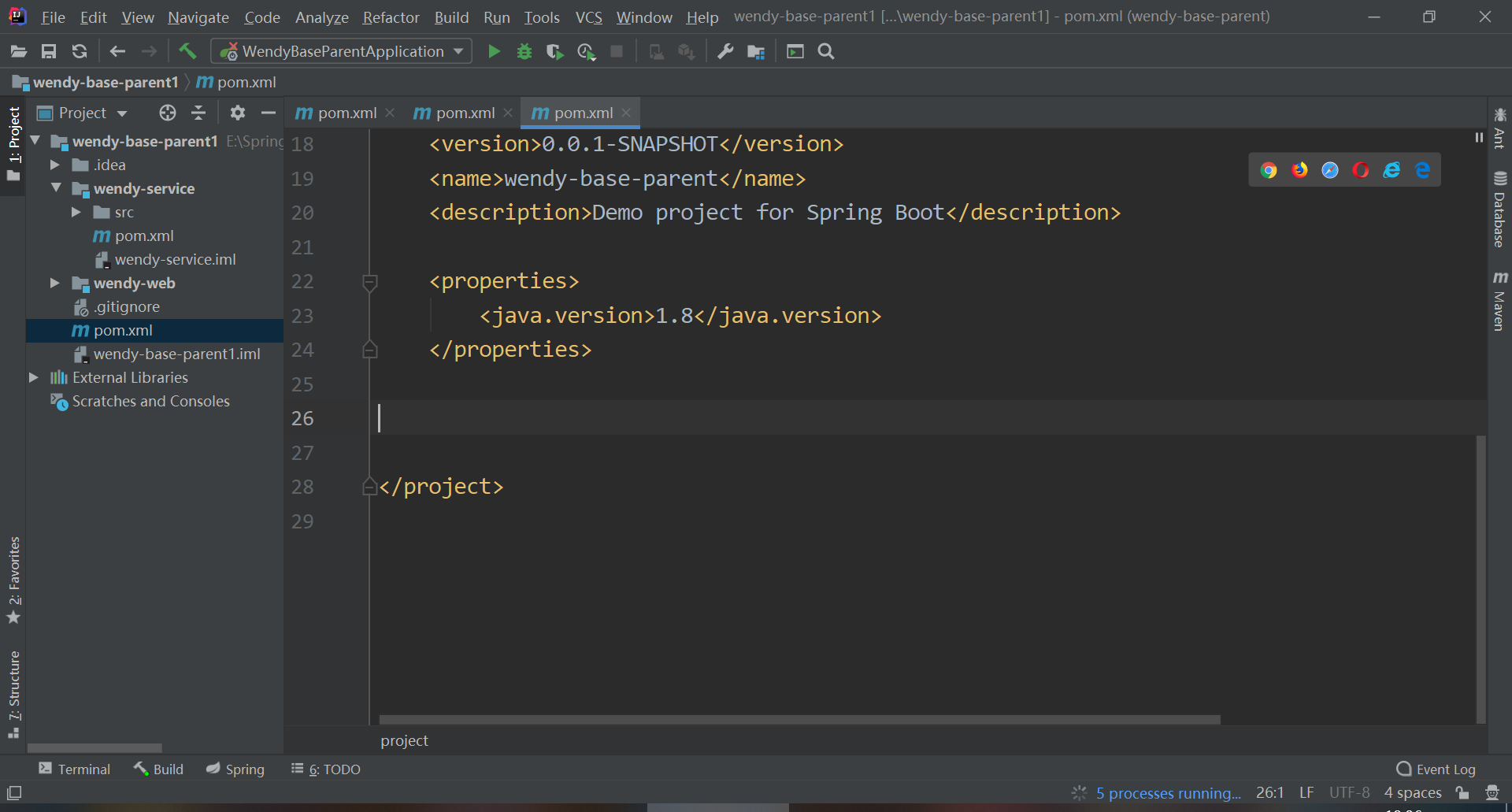




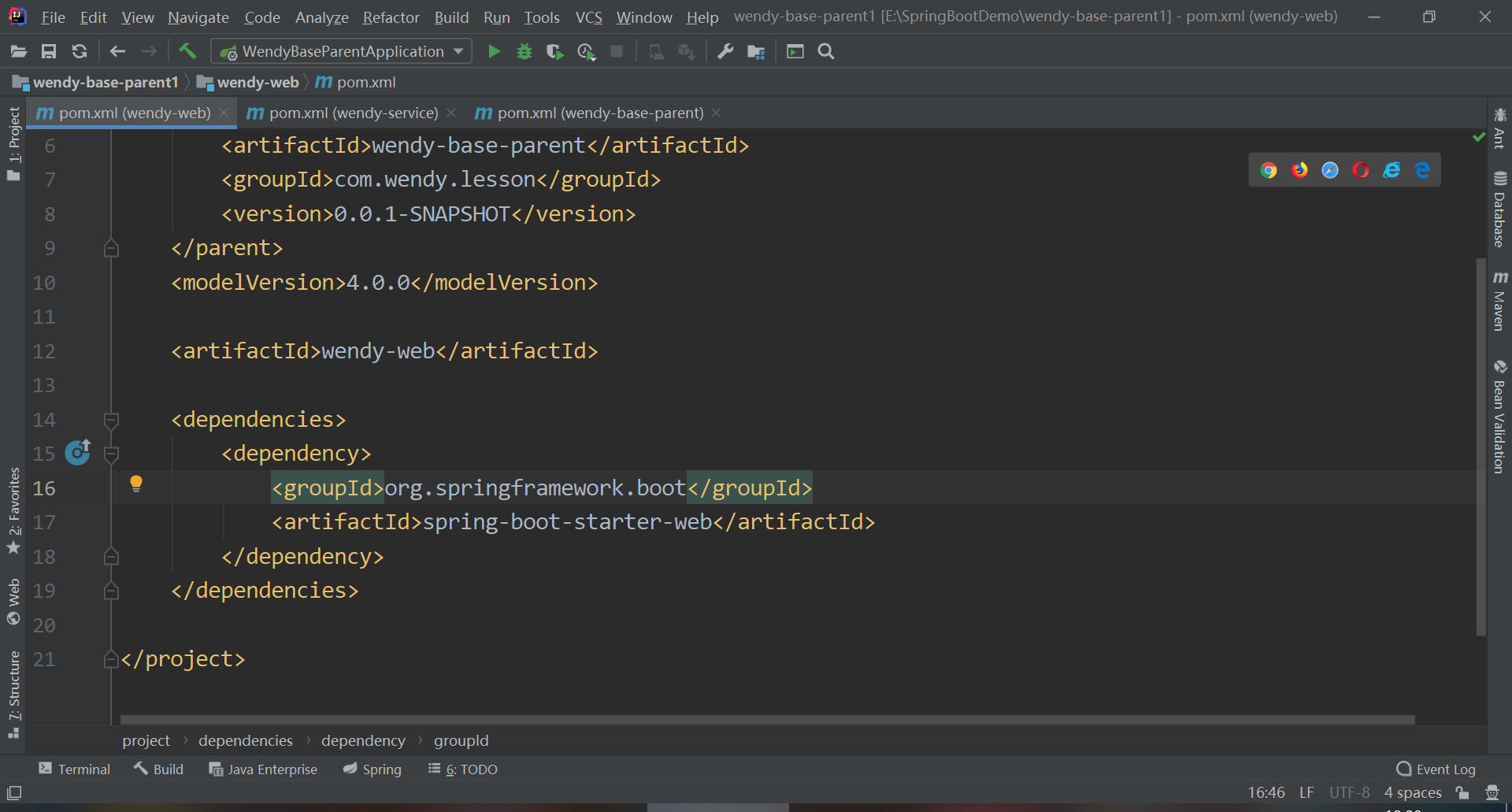
我们使用的mangementdependency，所以可以删除dependency

Build是中的插件是为了打包，但是项目中不需要，所以可以删除



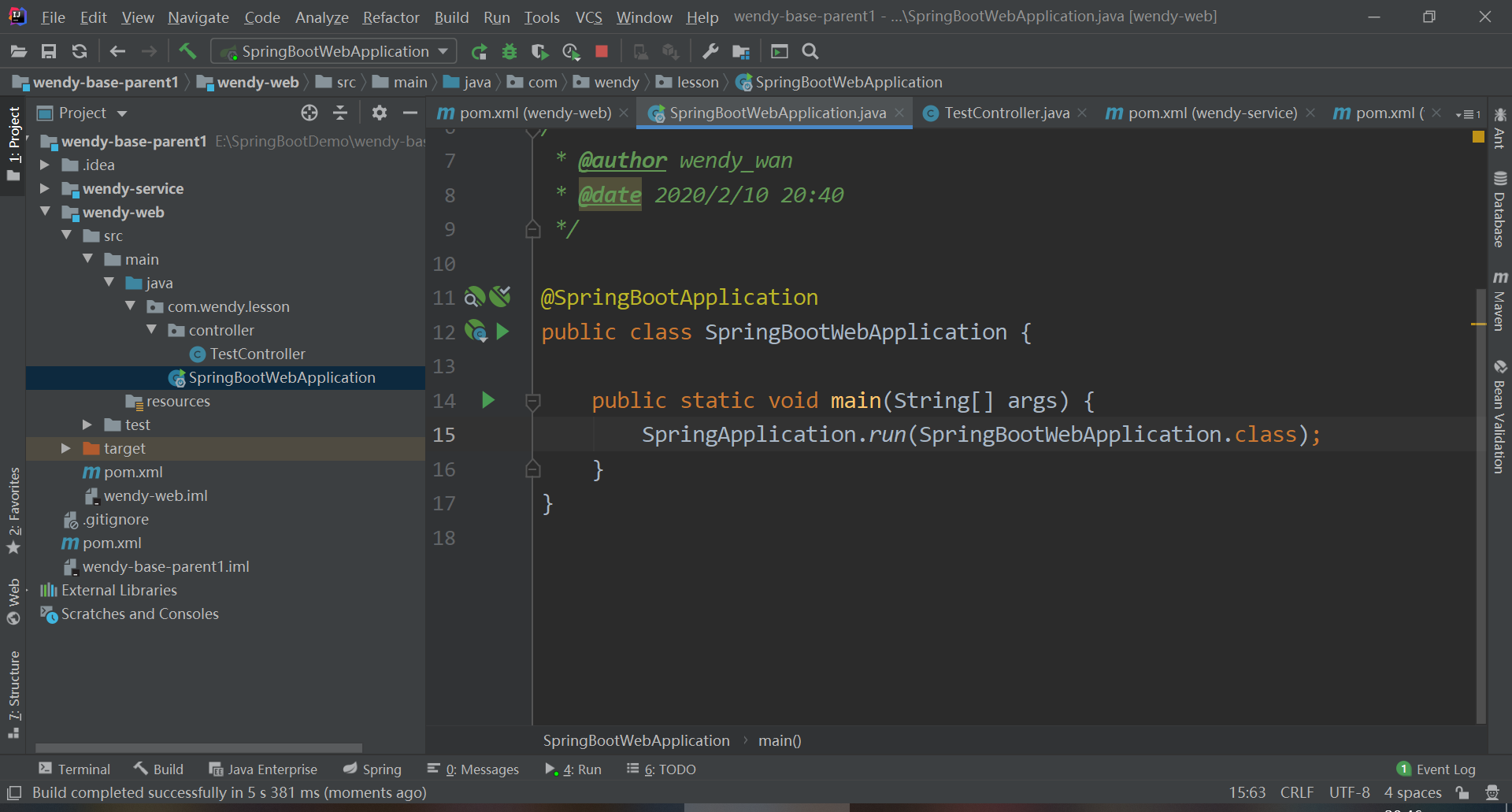


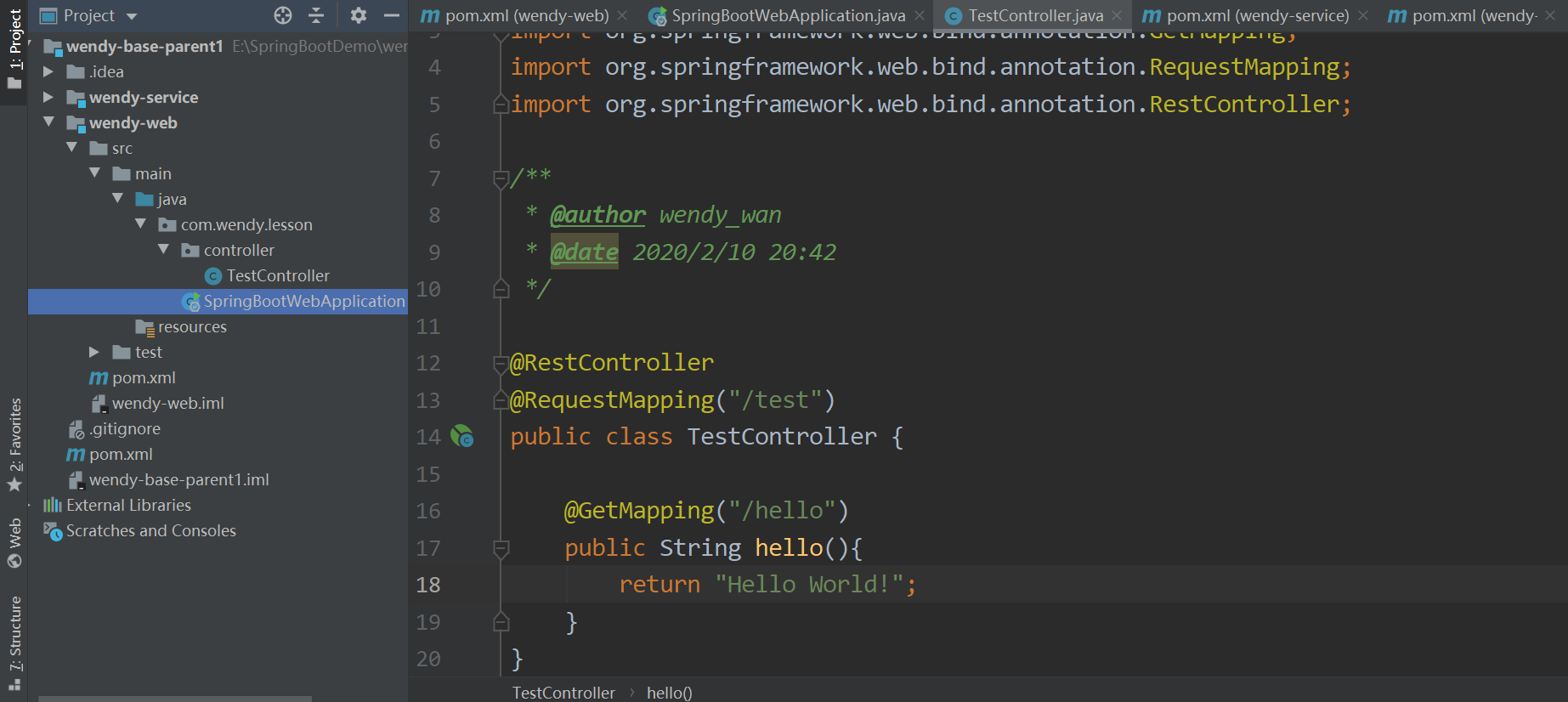
运行启动项目是在web层，所以在pom中配置一个web启动器



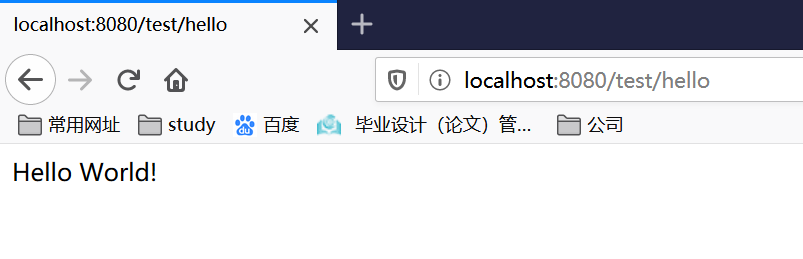
再创建一个启动类

（创建一个包，创建一个启动类。再创建一个控制层包以及类）



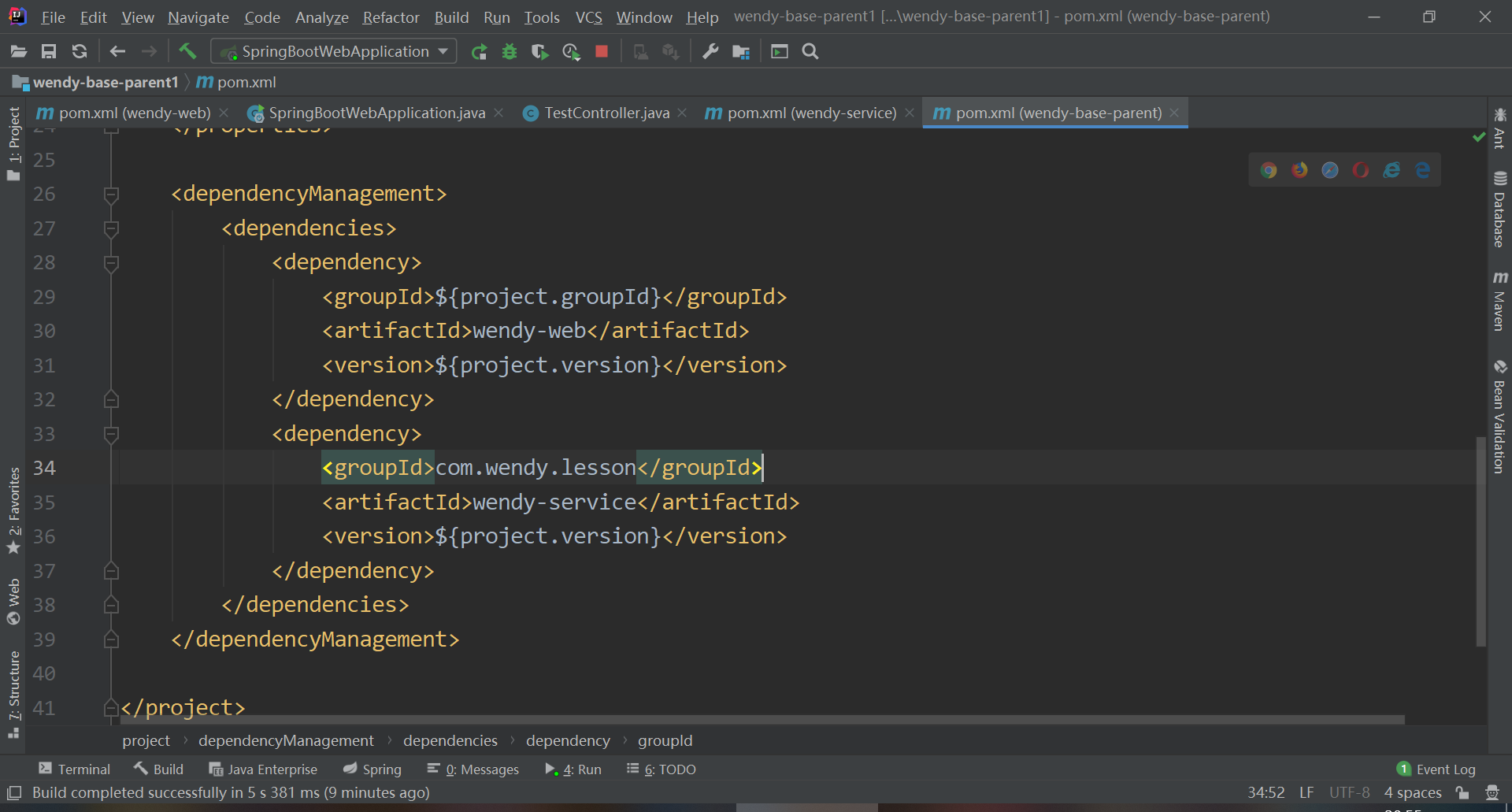


运行主函数：

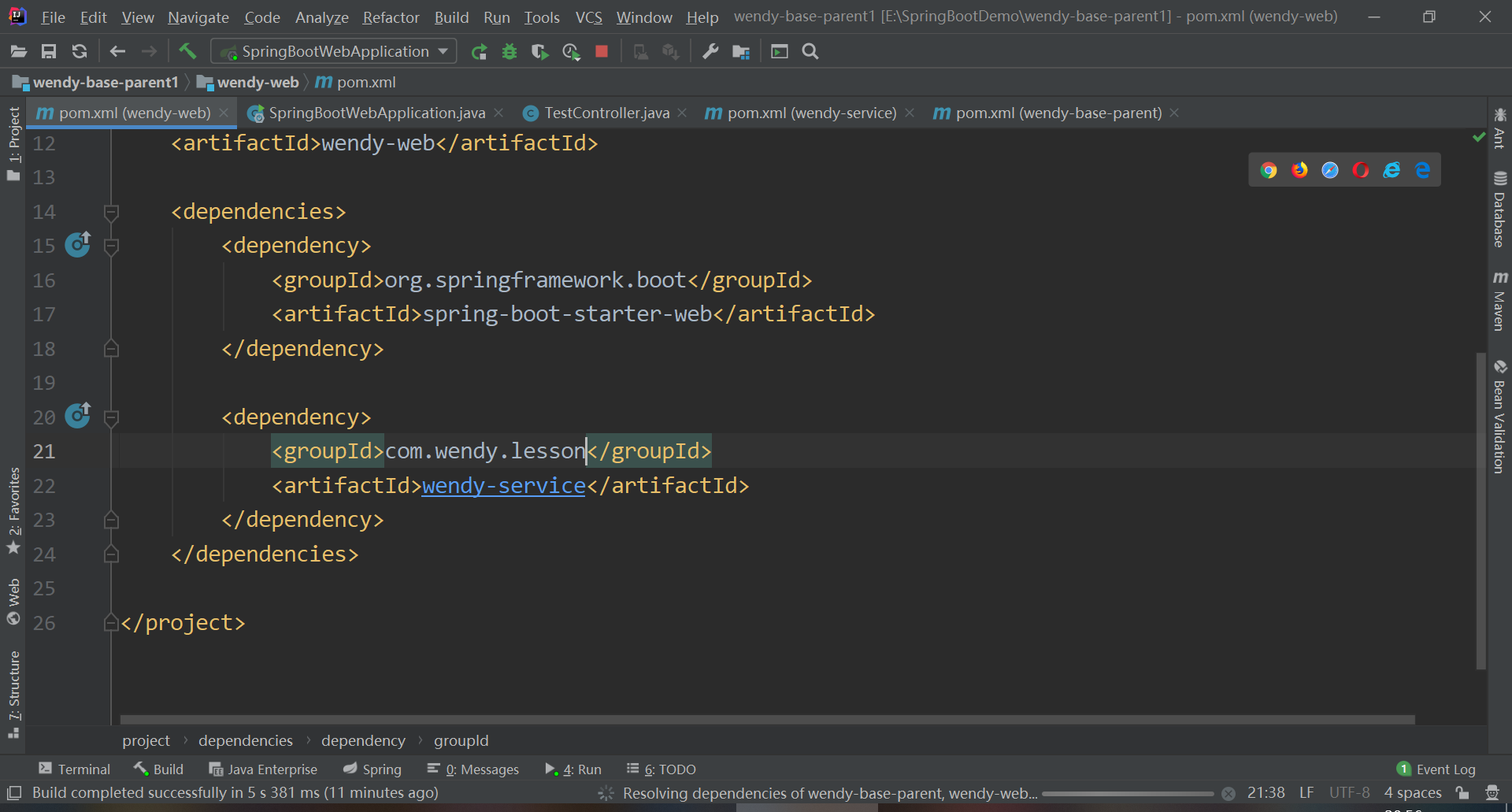


Web层调用service层。所以在pom要声明管理这些子模块。

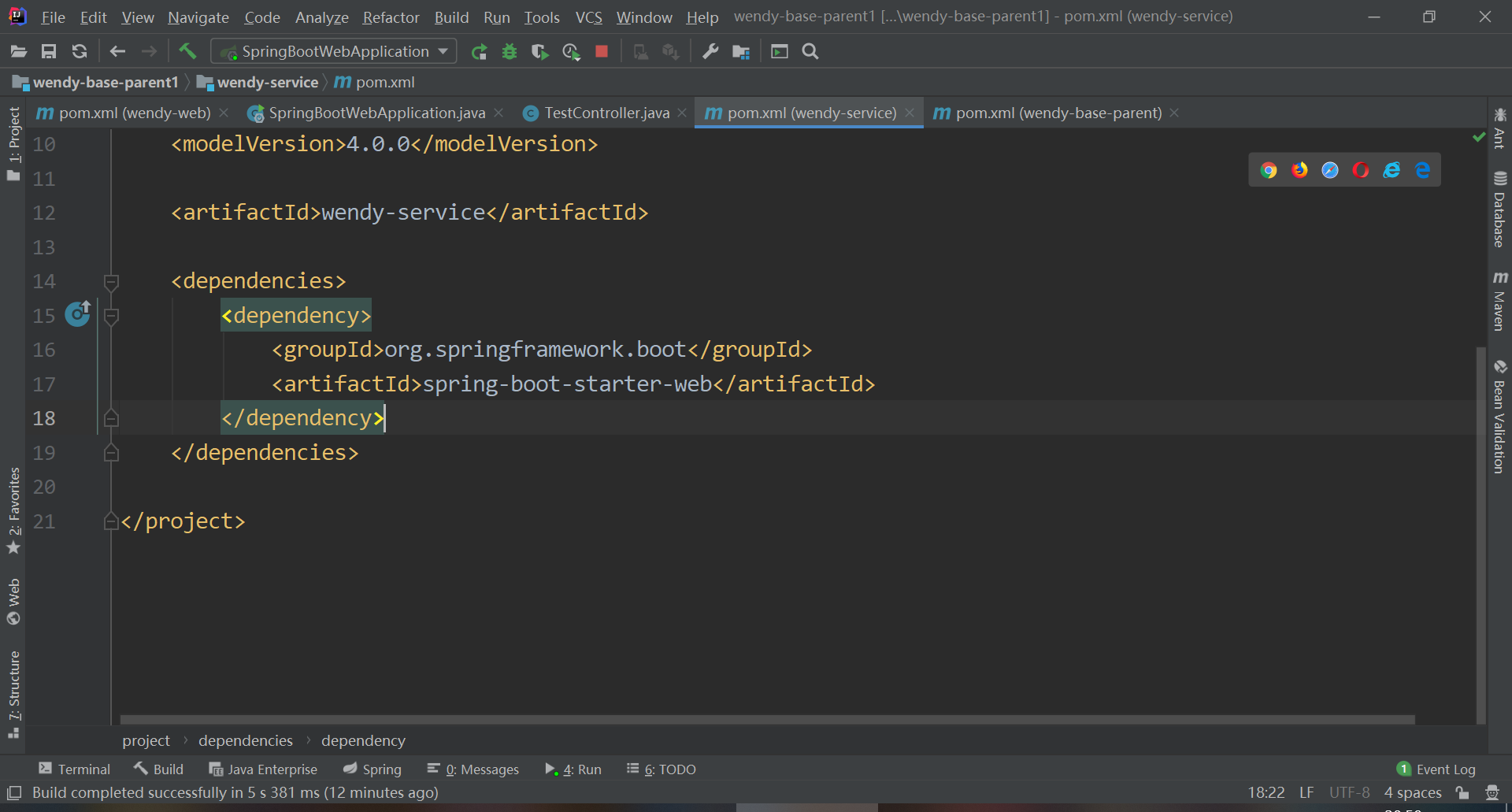
父级配置：



在web中的pom加入service的依赖



在service中的pom文件中添加配置：



前端访问web层，进入TestController方法，然后这个方法就会调用service层中的业务逻辑，业务逻辑访问完再返回给接口层，