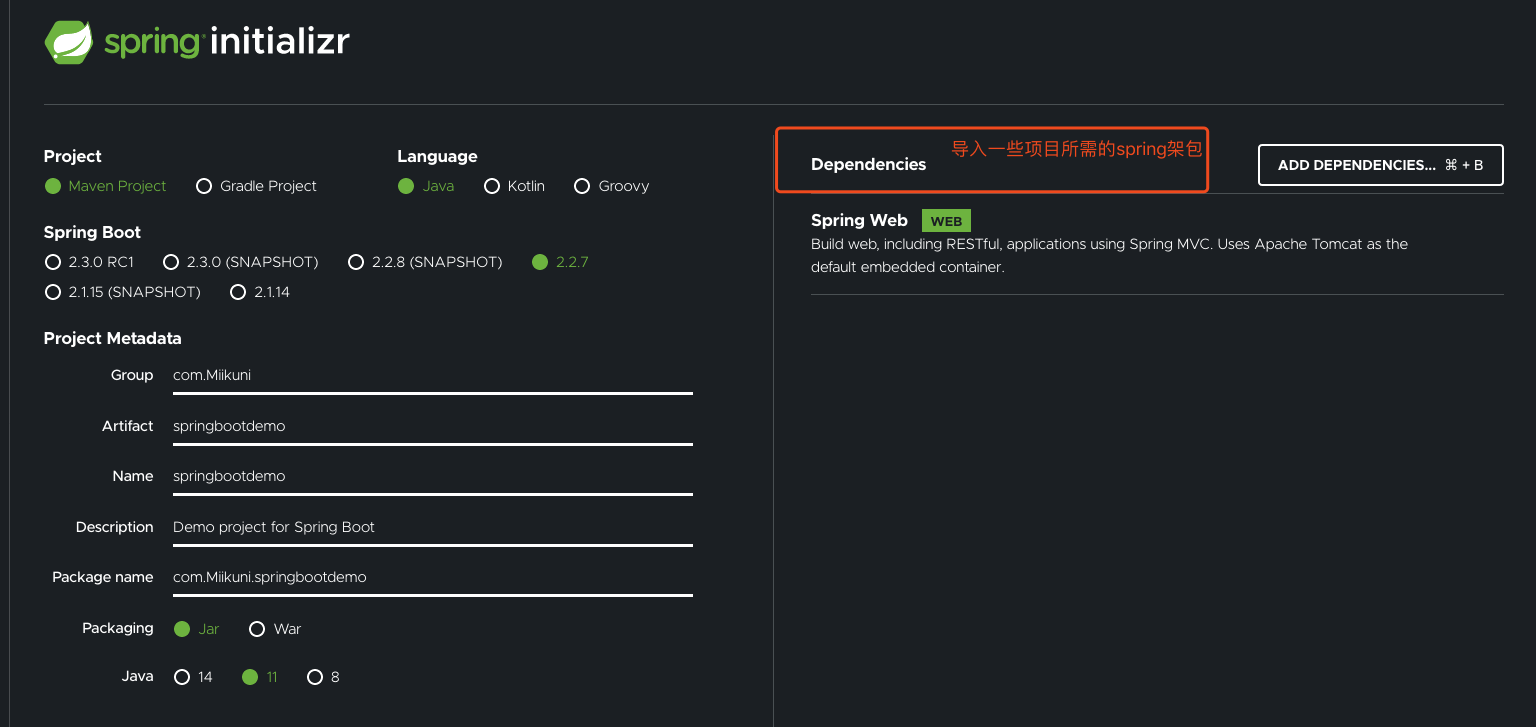
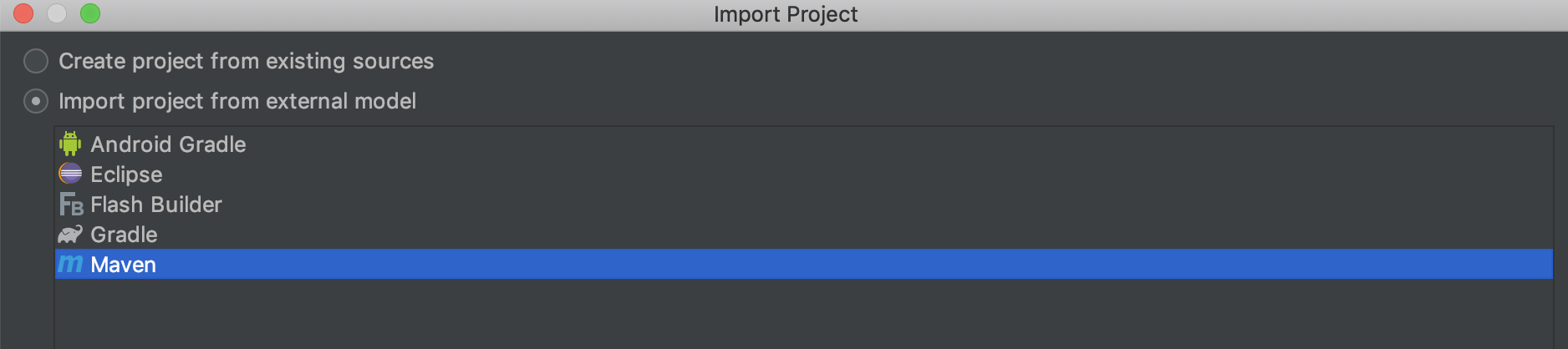
springboot

学习springboot的笔记

1.通过官网的quick start

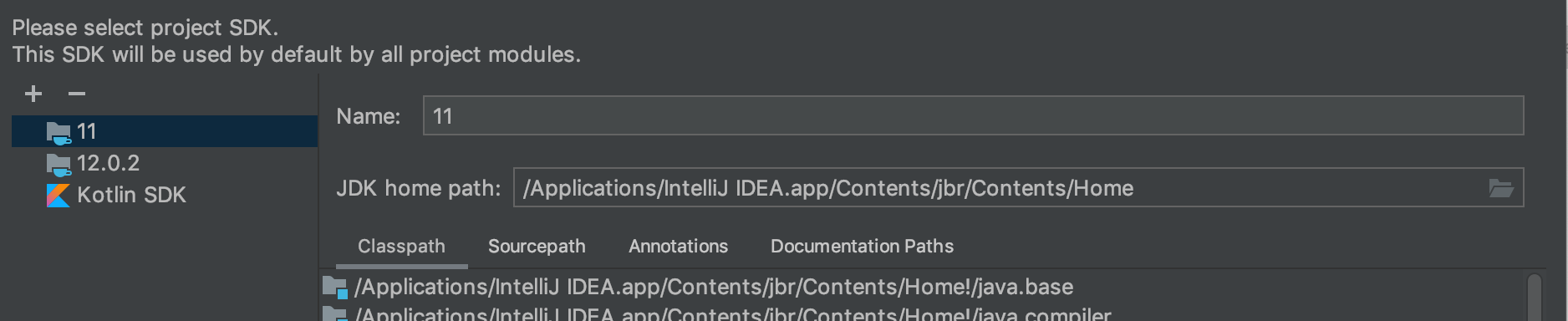
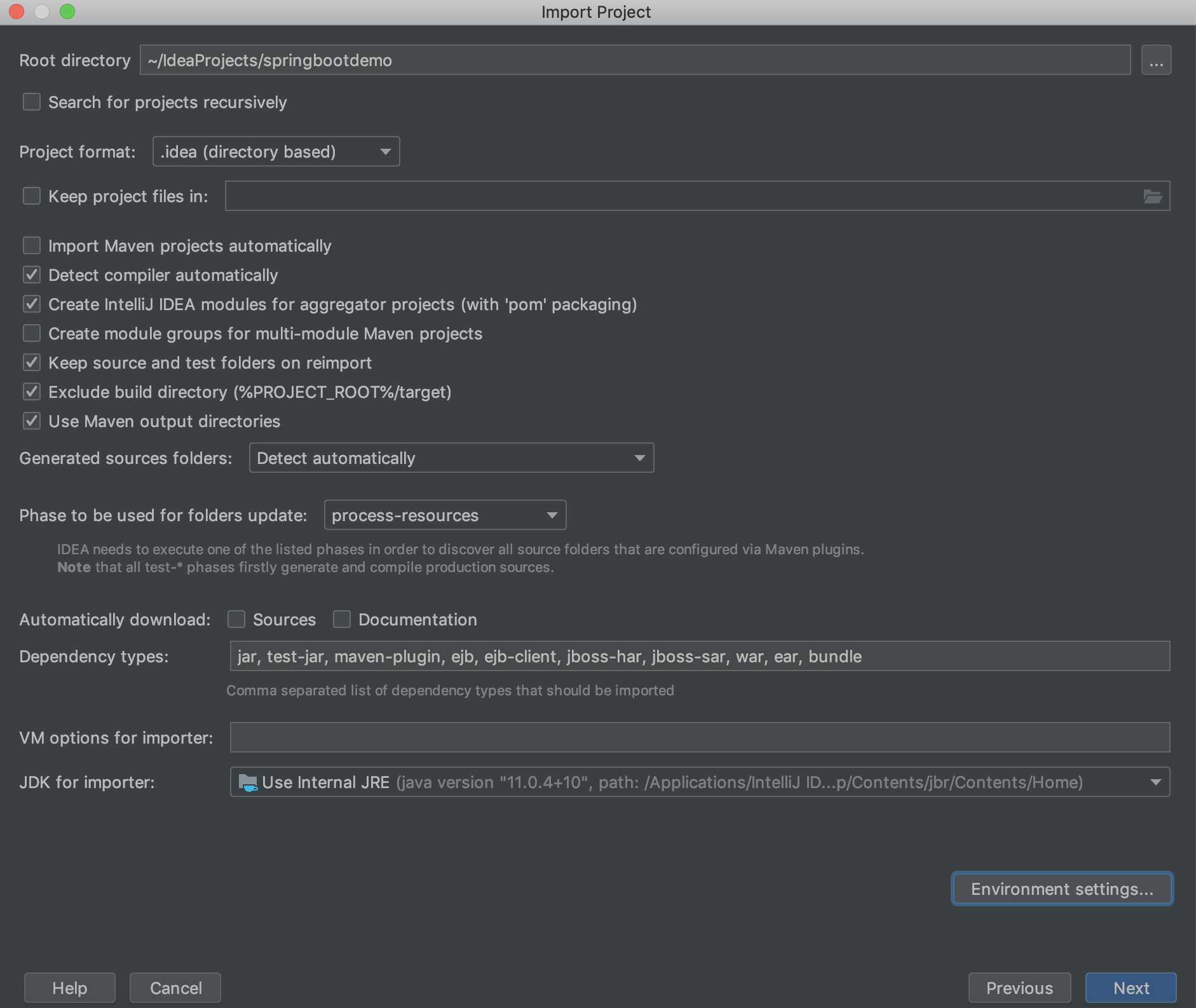
2.然后create生成并下载文件

3.在idea中导入刚下载的项目，会提示要导入什么模版，上面创建项目的时候使用的是maven，则选择导入maven

此页面可以直接下一步，也可以设计maven的相关版本（点击environment setting）

下面页面选择jdk版本，然后下一步

二.启动类

Spring boot的启动类的作用是启动spring boot项目，基于Main方法来运行的。

注意：启动类在启动时会做注解扫描（@Controller，@Service，@Respositroy….），扫描位置为同包或者子包下的注解，所以启动类的位置要放于包的根目录下。

1.启动类与启动器的区别：

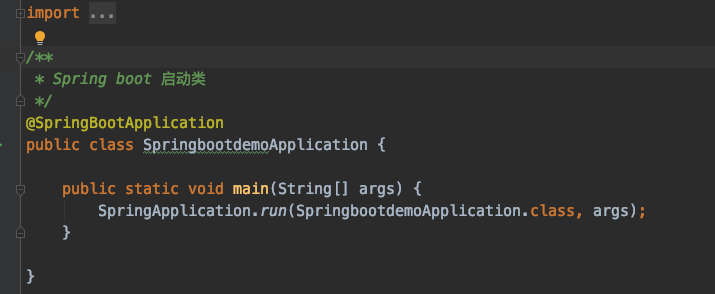
* 启动类表示项目的启动入口
* 启动器表示jar包的坐标

2.启动类

三.启动器（pom文件中设置）

Spring boot将所有的功能场景都抽取出来，做成了一个个的start（启动器），只需要在项目里面导入这些starter相关场景的所有依赖都会导入进来，要用什么功能就导入什么场景，在jar包管理上非常方便。

Spring boot提供了44个启动器

Spring-boot-starter：spring boot的核心启动器，包含自动配置，日志和yaml。

spring-boot-starter-actuator：帮助监控和管理应用。

spring-boot-starter-web：支持全栈式web开发，包括Tomcat和spring-webmvc。

四.配置文件

提供了一个名称为application的全局配置文件，支持properties格式和Yaml格式。

1.properties格式

修改Tomcat监听端口server.port=8888

2.YAML格式：扩展名可以是yaml或者yml。

基本格式的要求

* 大小写敏感
* 使用缩进代表层级关系
* 相同的部分只出现一次

server:

port: 8888

3.配置文件存放位置

* 当前项目根目录中
* 当前项目根目录下的一个/config子目录中
* 项目的resources即classpath根路径中
* 项目的resources即classpath根路径下的/config目录中

4.配置文件加载顺序

相同位置的加载顺序：

* 如果同一个目录下有application.yml和application.properties，默认先读取properties的。
* 如果同一个配置属性，在多个配置文件都有配置，默认使用第一个读取到的，后面读取不会覆盖前面读取到的

不同位置的加载顺序：

* 当前项目根目录下的一个/config子目录中（最高）

config/application.properties

* 当前项目根目录中（其次）

application.properties

* 项目中resources的/config目录中（一般）

resources/config/application.properties

* 项目中resources根路径中（最后）

resources/application.properties

5.配置文件中的占位符

语法：${}

占位符作用：

* “${}”中可以获取框架提供的方法中的值如：random.int等。

生成随机数

${random.value} - 类似uuid的随机数，没有-连接

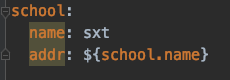
${random.int} - 随机取整型范围内的一个值

${random.long} - 随机取长整型范围内的一个值

${random.long(100,200)} - 随机生成长整型100-200范围内的一个值

${random.uuid} - 生成一个uuid，有-连接

${random.int(10)} - 随机生成一个10以内的数

* 占位符可以获取配置文件中的 键的值 赋予另一个键作为值

6.bootstrap配置文件

spring boot中有两种上下文对象，一种是bootstrap，另外一种是application。bootstrap是应用程序的父上下文，bootstrap加载优先于application。bootstrap主要用于从额外的资源来加载配置信息，还可以在本地外部配置文件中解密属性。这两个上下文共用一个环境，它是任何spring应用程序的外部属性的来源。bootstrap里面的属性会优先加载，它们默认也不能被本地相同配置覆盖。

bootstrap配置文件特征

* bootstrap由父ApplicationContext加载，比application优先加载
* bootstrap里面的属性不能被覆盖

bootstrap与application的应用场景

application配置文件主要用于Springboot项目的自动化配置。

bootstrap配置文件有以下应用场景

* 使用spring cloud config 配置中心时，需要在bootstrap配置文件中添加连接到配置中心的配置属性 来加载外部配置中心的配置信息。
* 一些固定的不能被覆盖的属性
* 一些加密/解密的场景

五.spring boot 的核心注解

1.@SpringBootApplication 是springboot启动类。此注解等同于@Cofiguration+@EnableAutoConfiguration+@ComponentScan的组合

2.@SpringBootConfiguration 是@Configuration注解的一个派生，和@Configuration一样，标志这个类是一个配置类，@SpringBootConfiguration是springboot的注解，@Configuration是spring的注解

3.@Configuration 通过对bean对象的操作替代spring中xml文件

4.@EnableAutoConfiguration springboot自动配置：尝试根据你添加的jar依赖自动配置你的Spring应用。是@AutoConfigurationPackage和@Import注解的组合

5.@AutoConfigurationPackage 是自动注入主类下所在包 下所有的加了注解的类（@Controller，@Service等），以及配置类（@Configuration）

6.@Import({AutoConfigurationImportSelector.class}) 直接导入普通的类，导入实现了ImportSelector接口的类，导入实现了ImportBeanDefinitionRegistrar接口的类

7.ComponentScan 组件扫描，可自动发现和装配一些Bean

六.Springboot在Controller中常用注解

1.@RestController 相当于@Controller+@ResponseBody注解：此注解的Controller中方法无法返回页面，所以没办法跳转并传输数据到另一个页面，所以InternalResourceViewResolver也不起作用，返回的内容就是Return里的内容。

2.@GetMapping 是@RequestMapping(method=RequestMethod.GET)的缩写

3.@PostMapping 是@RequestMapping(method=RequestMethod.POST)的缩写

4.@PutMapping 是@RequestMapping(method=RequestMethod.PUT)的缩写

5.@DeleteMapping 是@RequestMapping(method=RequestMethod.DELETE)的缩写

七.Springboot整合Web层技术

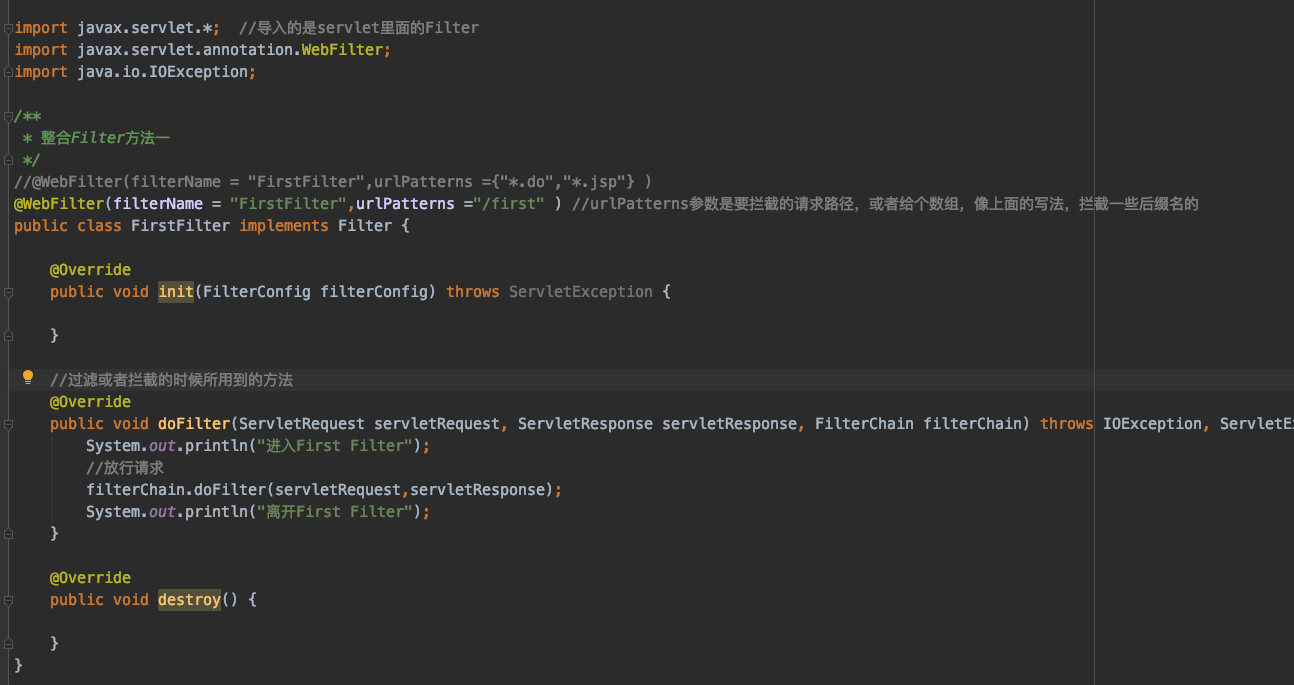
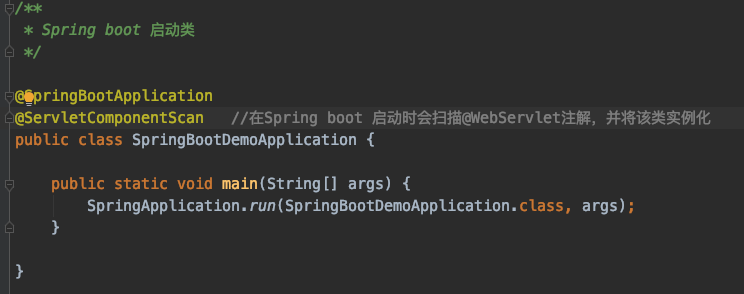
1.整合Servlet方法一：

* 通过注解扫描完成Servlet组件的注册
* 创建Servlet
* 修改启动类

2.整合Servlet方式二

* 通过方法完成Servlet组件的注册
* 创建Servlet
* 创建Servlet配置类

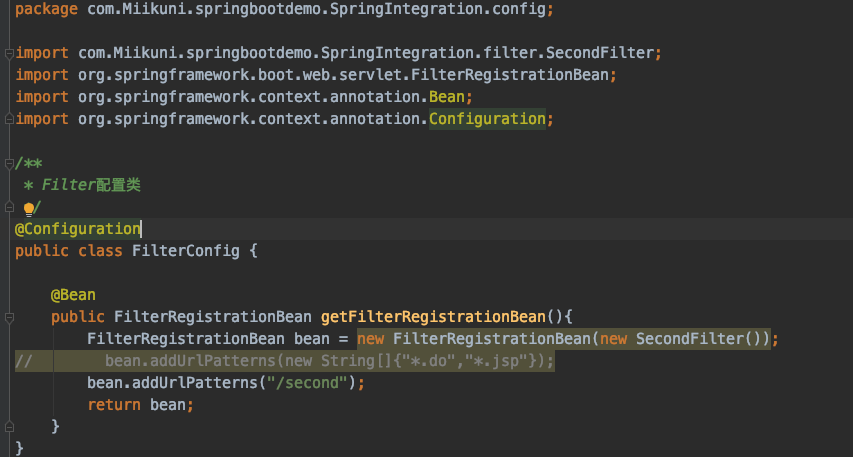
3.整合Filter方式一

* 通过注解扫描完成Filter组件注册
* 创建Filter
* 修改启动类

filter是Servlet的一个子技术，所以@ServletComponentScan

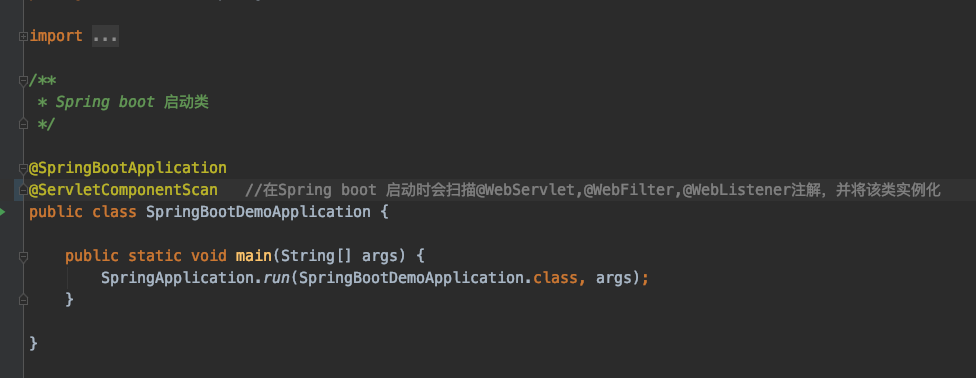
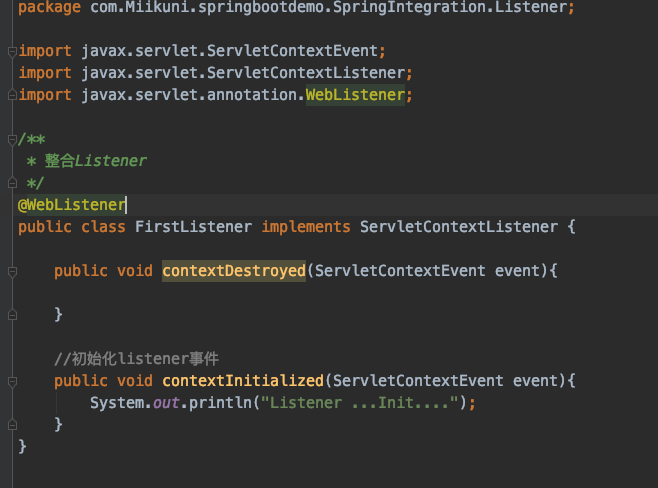
也会扫描到

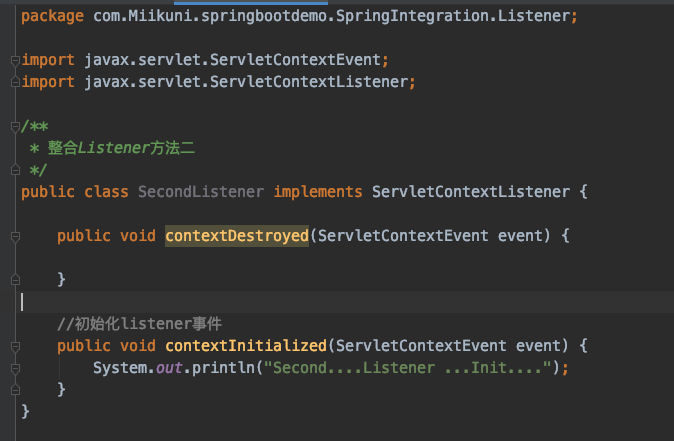
4.整合Filter方式二

* 通过方法完成Filter组件注册
* 创建Filter
* 创建Filter配置类

5.整合Listener方法一

* 通过注解扫描完成Listener组件注册
* 编写Listener
* 修改启动类

6.整合Listener方式二

* 通过方法完成Listener组件注册
* 编写Listener
* 创建Listener配置类

八.Springboot访问静态资源

在springboot项目中没有之前常规web开发的WebContext（Webapp），它只有src目录。在src/main/resources下面static和templates。Spring boot默认在static目录中存放静态页面，templates中放动态页面。

1.static目录：存放静态资源的目录名称必须是static。（可以存放网页，图片，视频等）

2.templates目录：在Spring boot中不推荐使用jsp作为视图层技术，而是默认使用Thymeleaf来做动态页面，templates目录就是存放Thymeleaf页面的。此目录下的资源是不允许直接通过外部url的方法来直接访问的（就是地址栏直接用url访问）

2个目录下有同名的文件则以templatesde目录优先

3.静态资源存放其他位置

* springboot访问静态资源的位置：

这里的classpath是指resources目录，对于资源查找顺序就是以下的顺序

classpath:/META - INF/resources/

classpath:/resources/

classpath:/static/

classpath:/public/

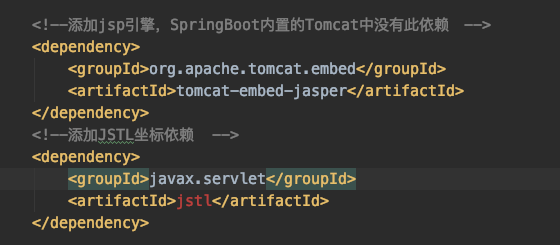
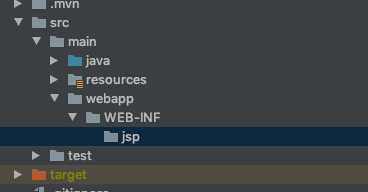
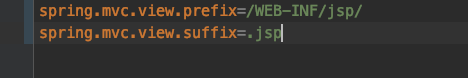
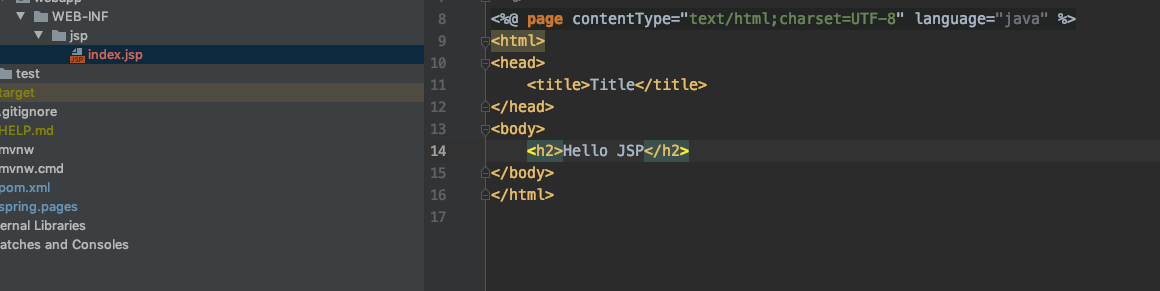
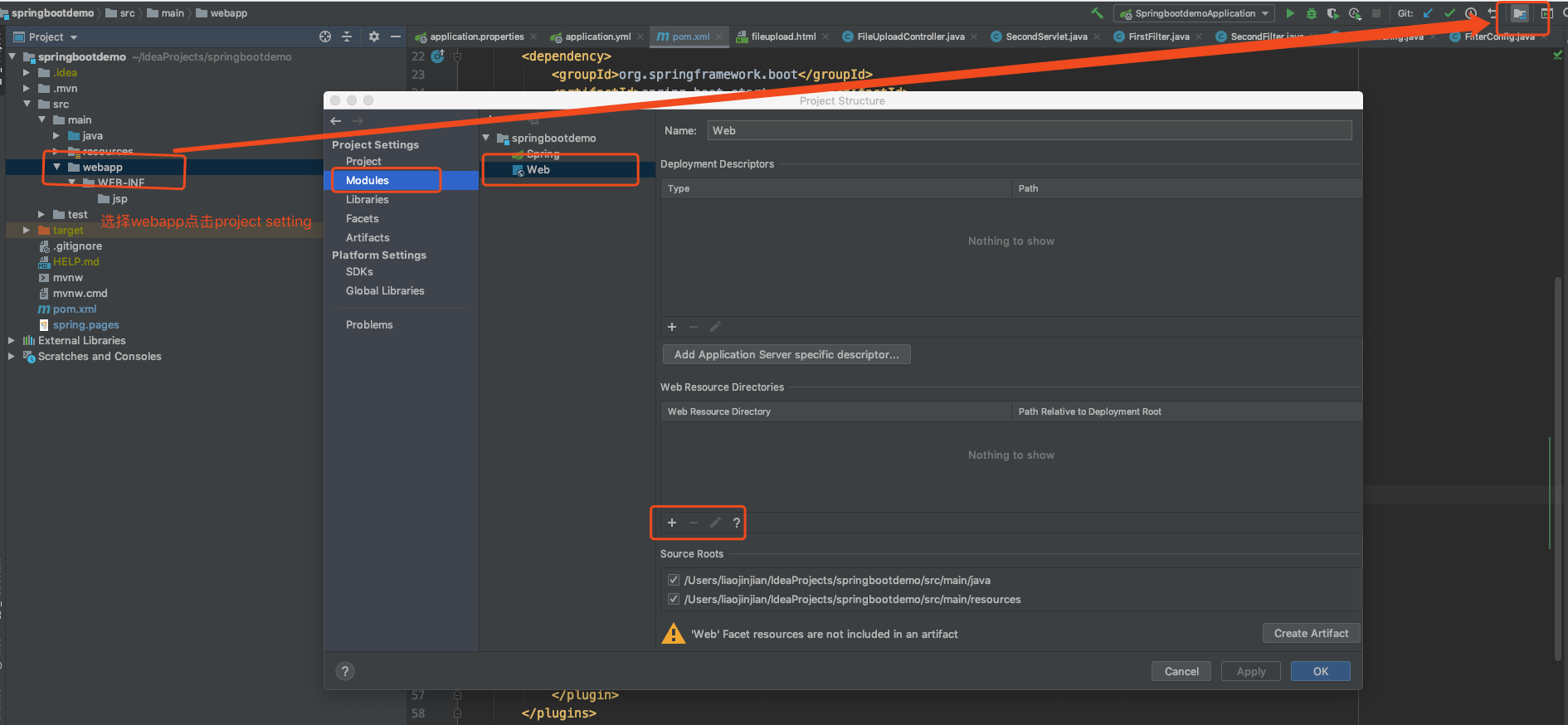
* 自定义静态资源位置

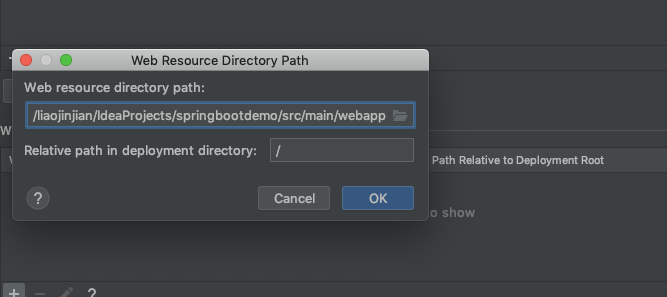
九.Springboot 文件上传

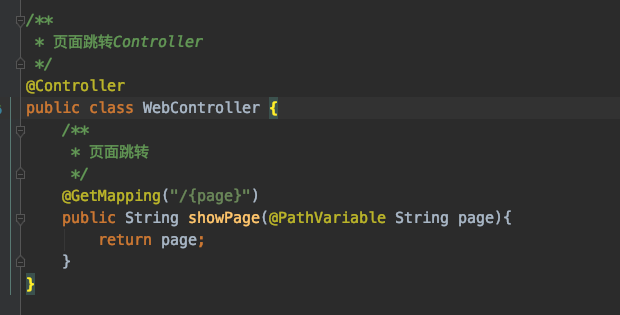
1. 编写上传页面
2. 编写controller
3. 修改上传文件大小

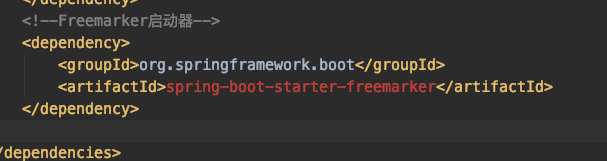
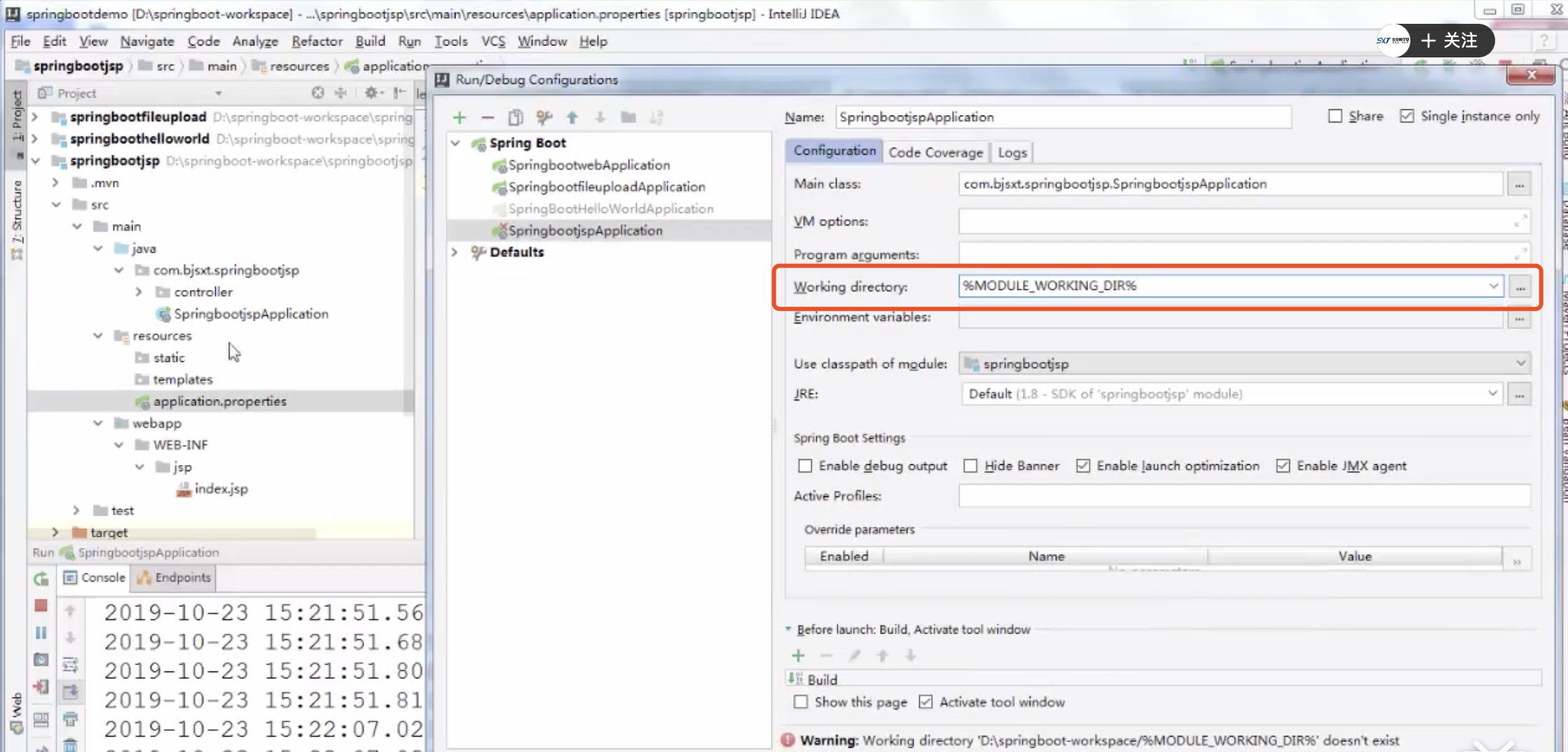
十.Springboot整合视图层技术

1. Springboot整合JSP技术

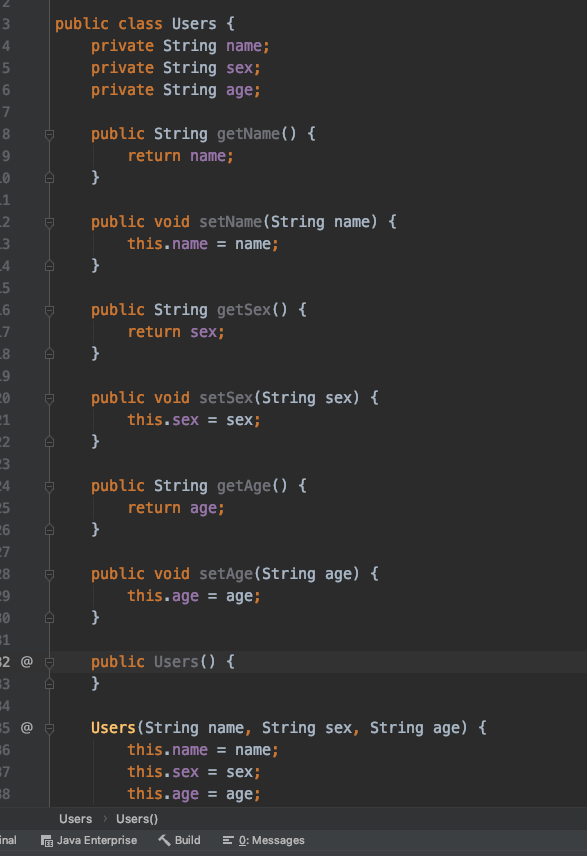
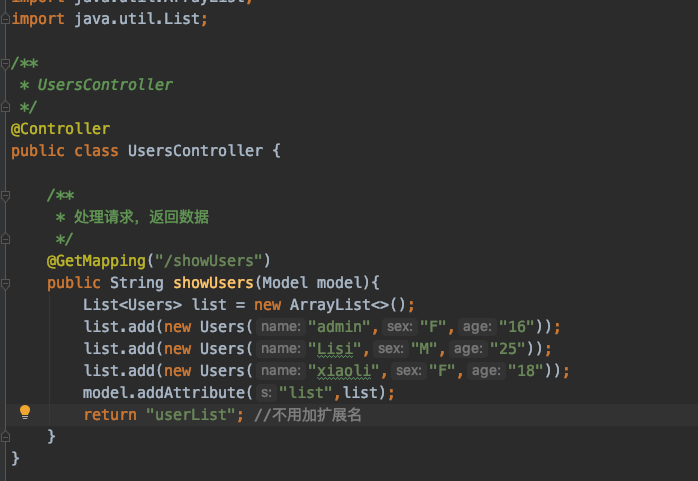
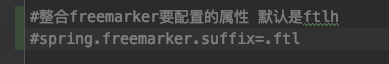
* 添加JSP引擎和JSTL标签库
* 创建webapp目录
* 标记为web目录



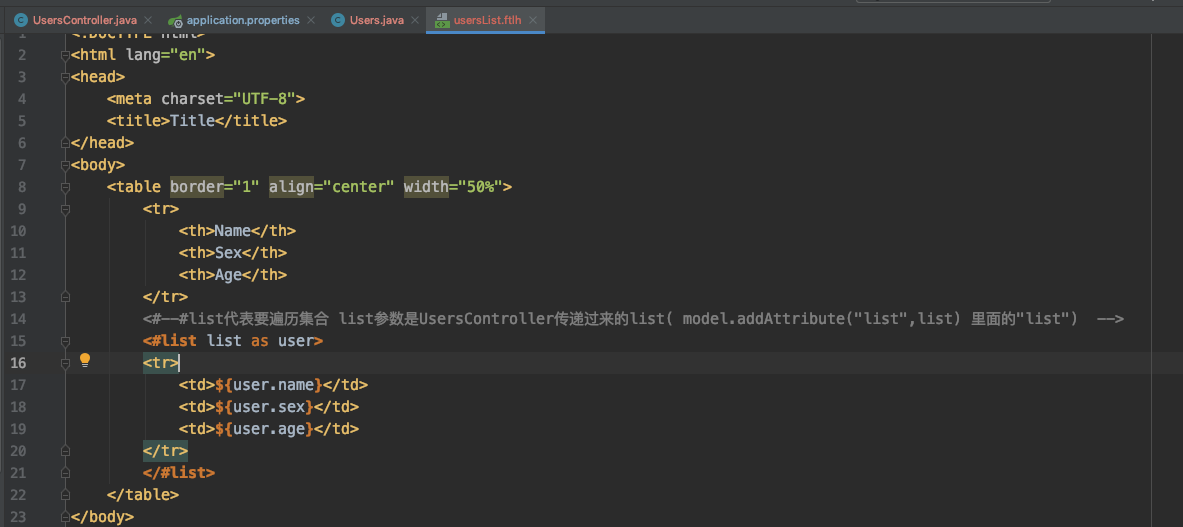
* 创建JSP
* 修改配置文件，配置视图解析器
* 创建controller

如果在idea中项目结构为聚合工程，那么在运行jsp是需要指定路径，如果是独立项目则不需要

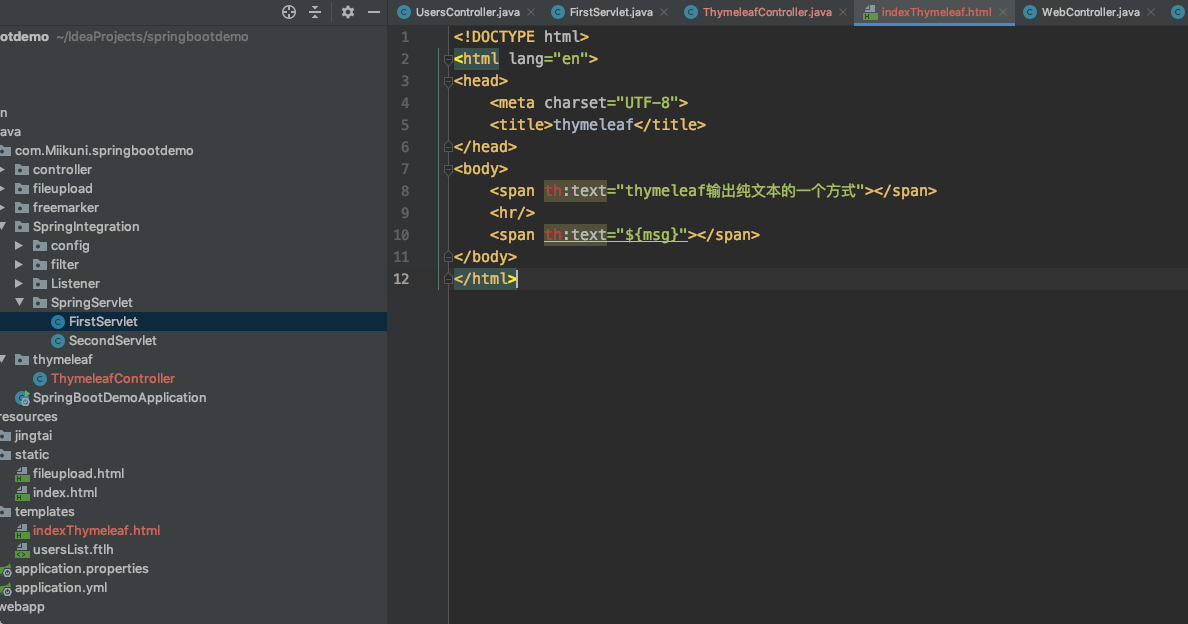
2. Springboot整合Freemarker

* 修改pom文件，添加Freemarker启动器
* 创建Users实体
* 创建Controller
* 创建视图
* 修改配置文件后缀

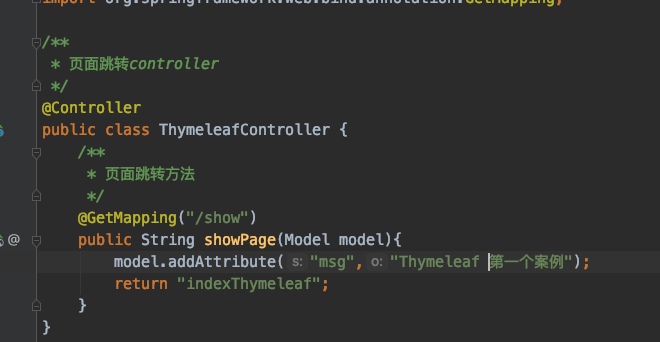
3. Springboot整合Thymeleaf

3.1 Thymeleaf的视图类型是html，可以作为静态原型，Thymeleaf可以处理HTML，XML，JavaScript，CSS甚至纯文本。Thymeleaf是原生的，不依赖于标签库，它能够在接受原始HTML的地方进行编辑和渲染，因为它没有于Servlet规范耦合，所以Thymeleaf模版能进入jsp所无法涉足的领域。

3.2 Thymeleaf基本使用

* 修改POM文件，添加Thymeleaf启动器依赖
* 创建Controller
* 创建视图

3.3Thymeleaf语法

命名空间：xmlns:th=“http://www.thymeleaf.org”

3.3.1字符串与变量输出操作

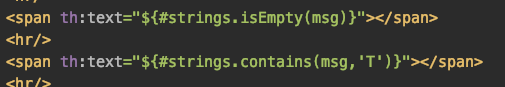
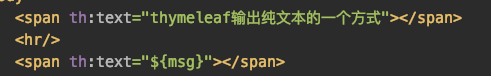
* th:text 在页面中输出值 th:text=“${变量名}”可以输出后端变量
* th:value 可以将一个值放入到input标签的value中

3.3.2 字符串操作

Thymeleaf提供了一些内置对象，内置对象可以直接在模版中使用。这些对象是以#引用的。

* 使用内置对象的语法

大部分内置对象的名称都是以s结尾的，如：strings，numbers，dates

string的方法和java中的string方法基本相同

${#strings.isEmpty(key)} 判断字符串是否为空，如果为空就返回true

${#strings.contains(msg,’T’)} 判断字符串是否包含指定的子串，如果包含返回true

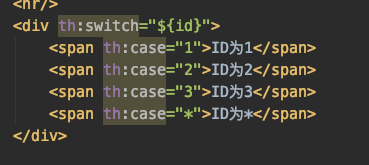
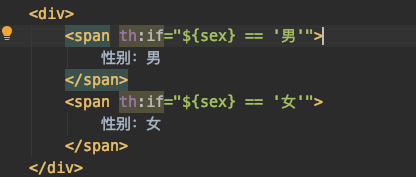
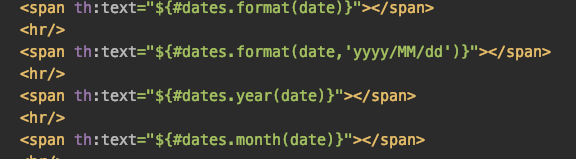
${#strings.startWith(mag,’a’)} 判断当前字符是否以子串开头，是则返回true

3.3.3 日期格式化处理

和java的date类方法相似

${#dates.format(key)} 格式化日期，默认以浏览器语言为格式化标准

${#dates.format(key,’yyyy/MM/dd’)} 按照自定义的格式做日期转换

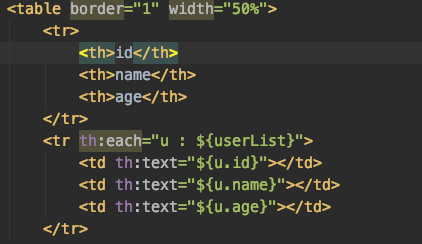
${#dates.year(key)} ${#dates.month(key)} ${#dates.day(key)}

3.3.4 条件判断

Thymeleaf

* th:if 条件判断
* th:switch/th:case 与java的switch语句等效，如果有多个匹配结果只选择第一个显示。
* th:case=“\*”表示java中switch的default，即没有case的值为true时显示th:case=“\*”的内容

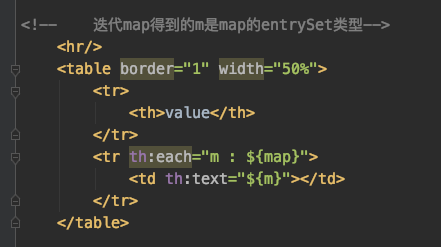
3.3.5 迭代遍历

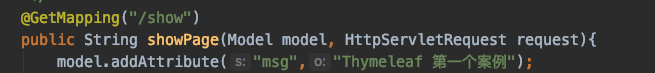
* th:each 迭代list，set集合
* th:each状态变量

1. index：当前迭代器的索引从0开始
2. count：当前迭代对象的计数从1开始
3. size：被迭代对象的长度
4. odd/even：布尔值，当前循环是否是偶数/奇数从0开始
5. first：布尔值，当前循环是否是第一条，如果是返回true否则返回false
6. last：布尔值，当前循环是否是最后一条，是返回true

th:each="u,var : ${userList}"

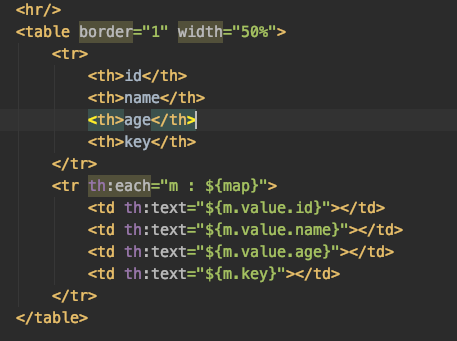
var 为迭代器

3.3.6 th:each 迭代map

3.3.7 操作域对象

* HttpServletRequest

request.setAttribute(“req”,”HttpServletRequest”);

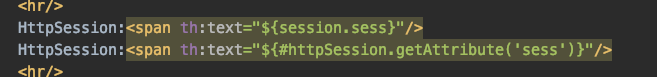
<span th:text="${#request.getAttribute('req')}"/>

<span th:text="${#httpServletRequest.getAttribute('req')}"/>

* HttpSession

request.getSession().setAttribute("sess","HttpSession");

<span th:text="${session.sess}"/>

<span th:text="${#httpSession.getAttribute('sess')}"/>

* ServletContext

request.getSession().getServletContext().setAttribute(“app","application");

<span th:text="${application.app}"/>

<span th:text=“${#servletContext.getAttribute('app')}"/>

3.3.8 URL表达式

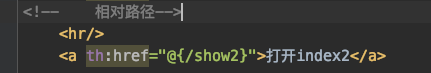
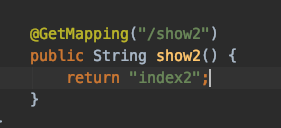
* 在thymeleaf中URL的表达式的语法格式为@{}
* 绝对路径

<a th:href=“@{http://www.baidu.com}">打开百度</a>

* 相对路径

1. 相对于当前项目的根

<a th:href=“@{/show2}">相对路径</a>

打开项目里面的请求路径

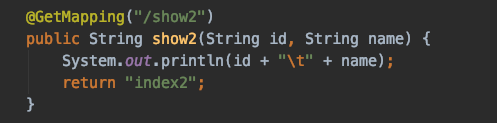
1. 相对于服务器路径的根

<a th:href=“@{~/project2/resourcename}”>相对于服务器的根</a>

如果tomcat中部署了多个项目，则可以用这种方式跨项目访问，project2为项目名，resourcename为路径名称

* 在URL中传递参数
* 普通格式中URL中传递参数

<a th:href="@{/show2?id=1&name=admin}">普通URL传递参数方式一</a>

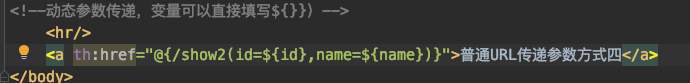
通过?来添加传递参数，&来连接多个参数

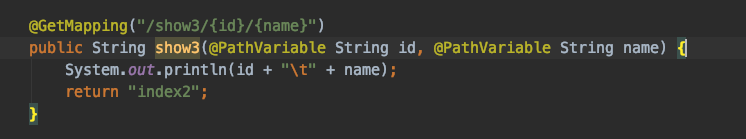
<a th:href=“@{/show2(id=2,name=khan)}">普通URL传递参数方式二</a>

在（）在直接填写参数，参数用，隔开

<a th:href=“@{'/show2?id='+${id}+'&name='+${name}}">普通URL传递参数方式三</a>

传递参数的方法：

<a th:href="@{/show2(id=${id},name=${name})}">普通URL传递参数方式四</a>

* 在restful格式的URL中传递参数

<a th:href="@{/show3/{id}(id=200)}">restful传递参数方式一</a>

<a th:href=“@{/show3/{id}/{name}(id=200,name=admin)}">restful传递参数方式二</a>

<a th:href=“@{/show4/{id}(id=200,name=admin)}">restful传递参数方式三</a>

id是使用restful传值，name是通过key value传值

<a th:href=“@{/show4/{id}(id=${id},name=${name})}">restful传递参数方式四</a>

3.3.9 Thymeleaf的一些配置

十一.Springboot整合持久层技术

1.整合JDBC

1.1 创建表

CREATE TABLE `test`.`users` (

`userid` int(0) NOT NULL AUTO\_INCREMENT,

`username` varchar(255) NULL,

`usersex` varchar(255) NULL,

PRIMARY KEY (`userid`)

);

id是自增的

1.2

