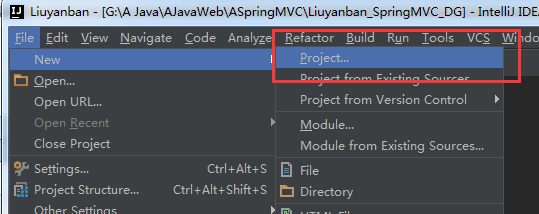
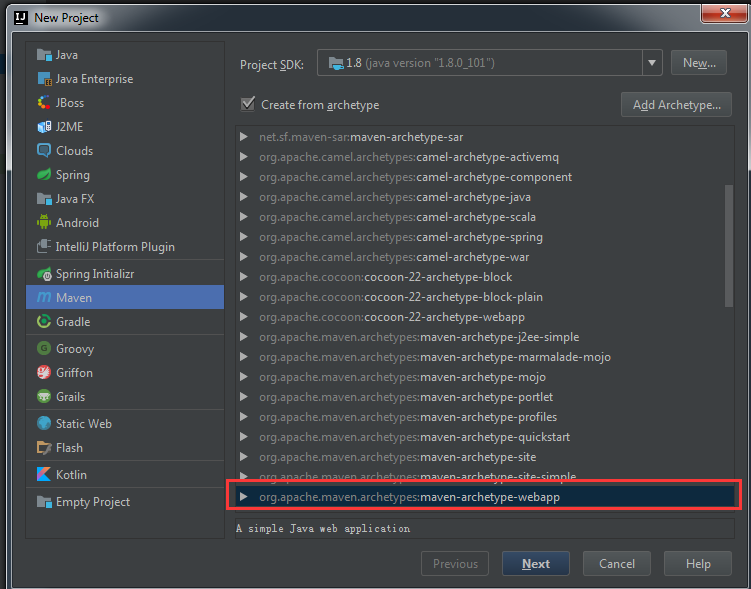
**一、首先需要有一个IntelliJ IDEA 可以去官网下载，虽然是收费的，然而并不算贵。**

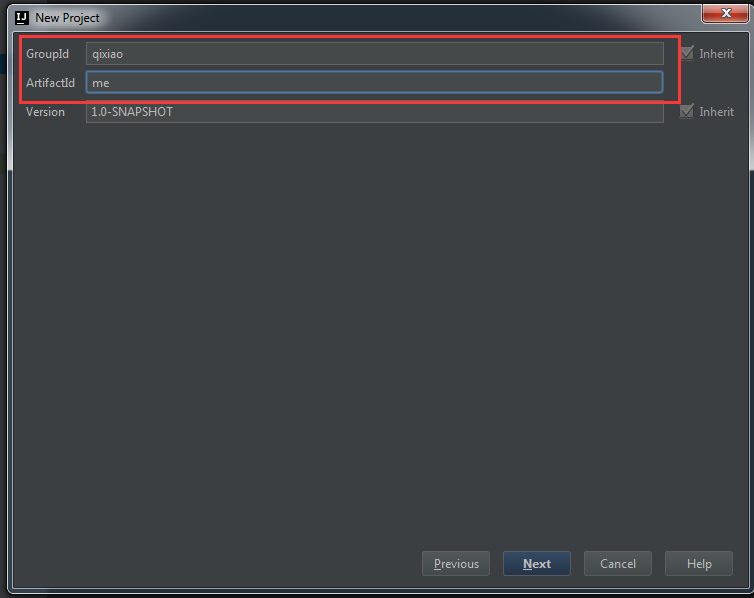
**二、运行编译器新建项目**



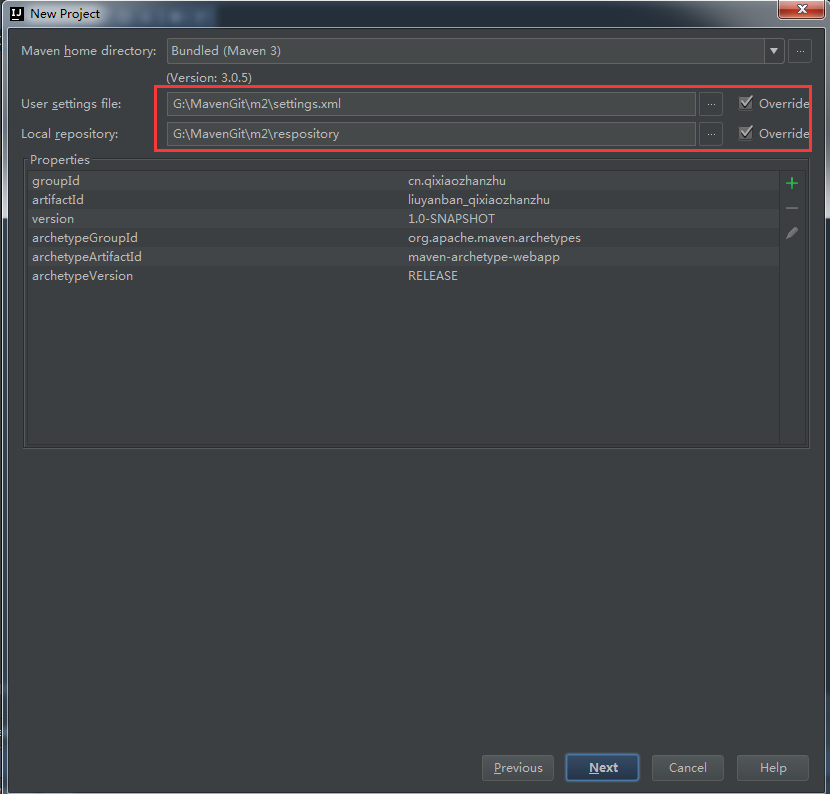
**在弹出的窗体中选择maven，然后勾选要建的maven模板--这里选webApp**



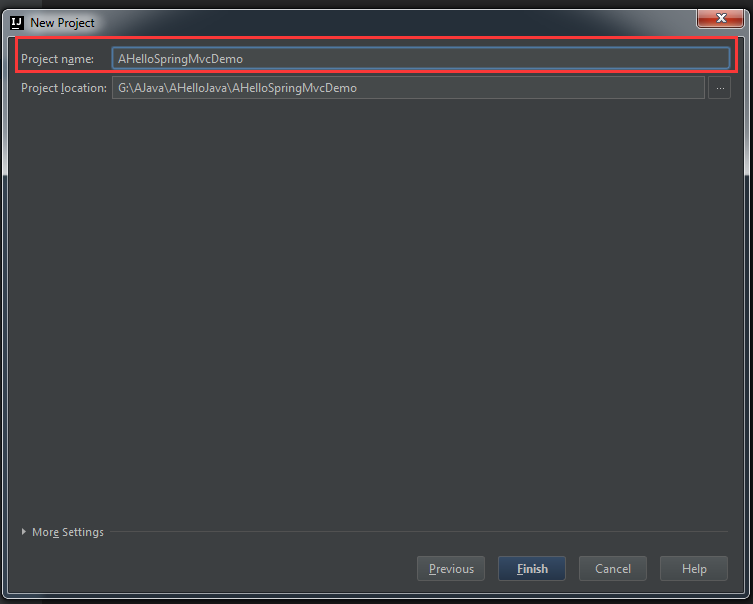
**然后填入相应的maven项目组信息（这个是比较随意的）**



**这里填写自己的maven本地仓库路径（这个不用过多解释吧，用maven的基础）；**



**这里填写项目名称以及项目生成路径，然后在下方检查填写是否正确：**

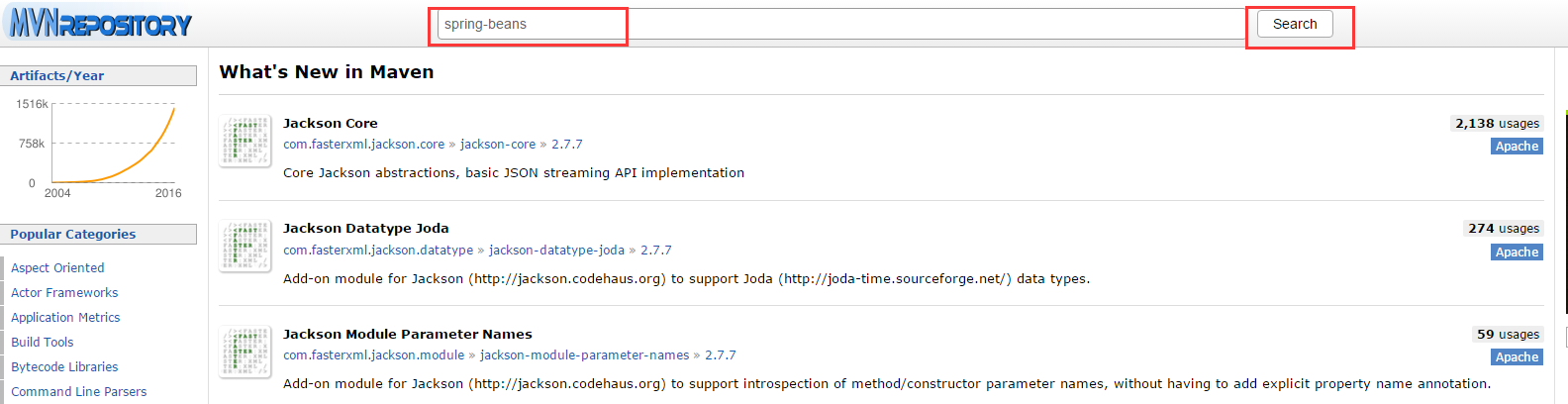


**如果正确就点击Finish->**

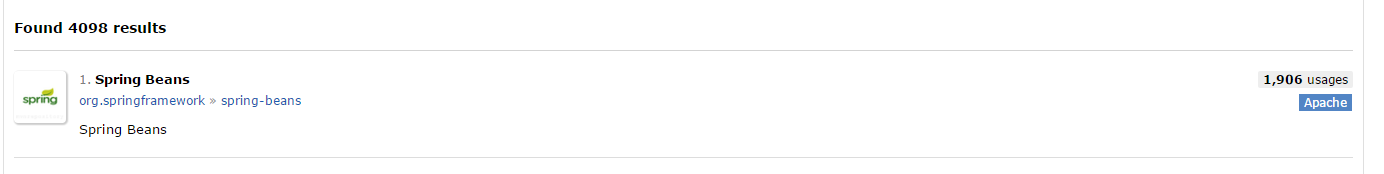
**maven会自动创建需要的一些配置信息以及目录结构，在这段时间里我们可以查找需要的jar包并在maven配置文件pom.xml里面进行配置，见下面步骤：**

**这里如果不知道要用到什么jar包就去百度SpringMVC需要的jar包，然后在maven的官方链接单独查找jar包来配置pom.xml，实例如下（这里我示范去maven官网查找xml配置的部分，具体需要的jar包去我后面展示的pom.xml里面查找）：**

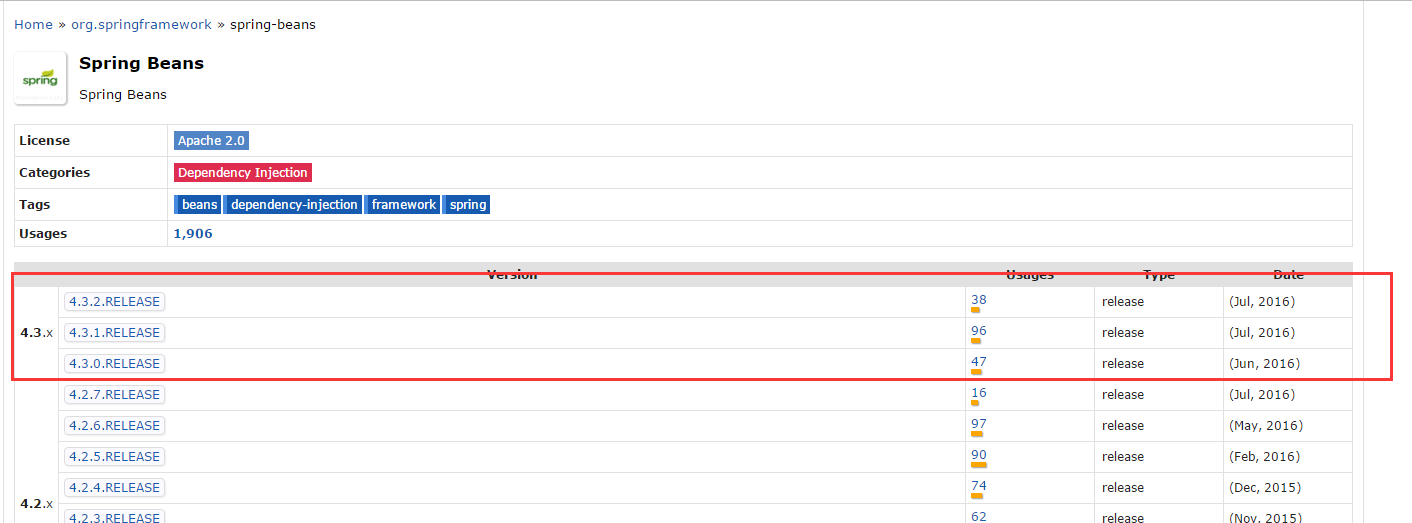
**登录**<http://mvnrepository.com/> **示例查找spring-beans**



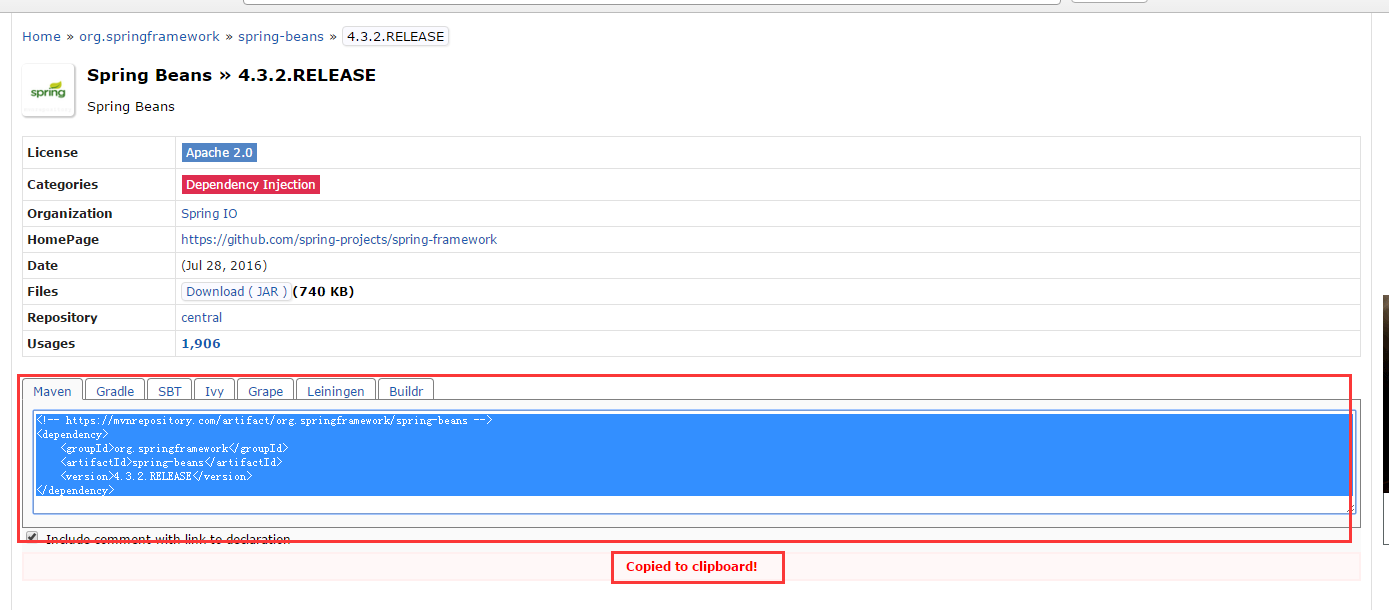
**点击查找结果**

****

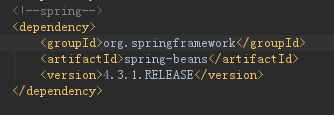
**这里可以看到最新版以及使用人数最多的版本，自己选择--配置文件里面的jar包版本最好选择同一个版本避免版本冲突；**

****

**我们选择第一个进入网页，里面可以看到maven的配置pom.xml文件写法，点击代码直接复制（自动复制）；**



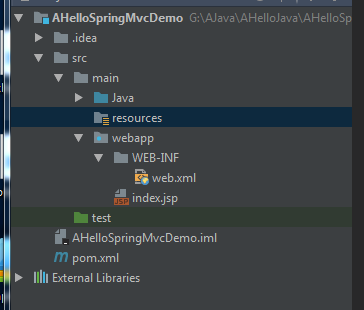
**然后将复制的代码拷贝到pom.xml文件中去即可，maven会自动下载所需要的jar包，我们不需要理会（如果有网络不顺畅情况，请自行想办法下载jar包或联系本人，然后将下载好的jar包拷贝到对应的maven本地仓库文件夹中也可以的）；**



**这里我展示我配置的maven pom.xml，maven会自动到库里面下载需要的jar包到maven仓库。**

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif pom.xml

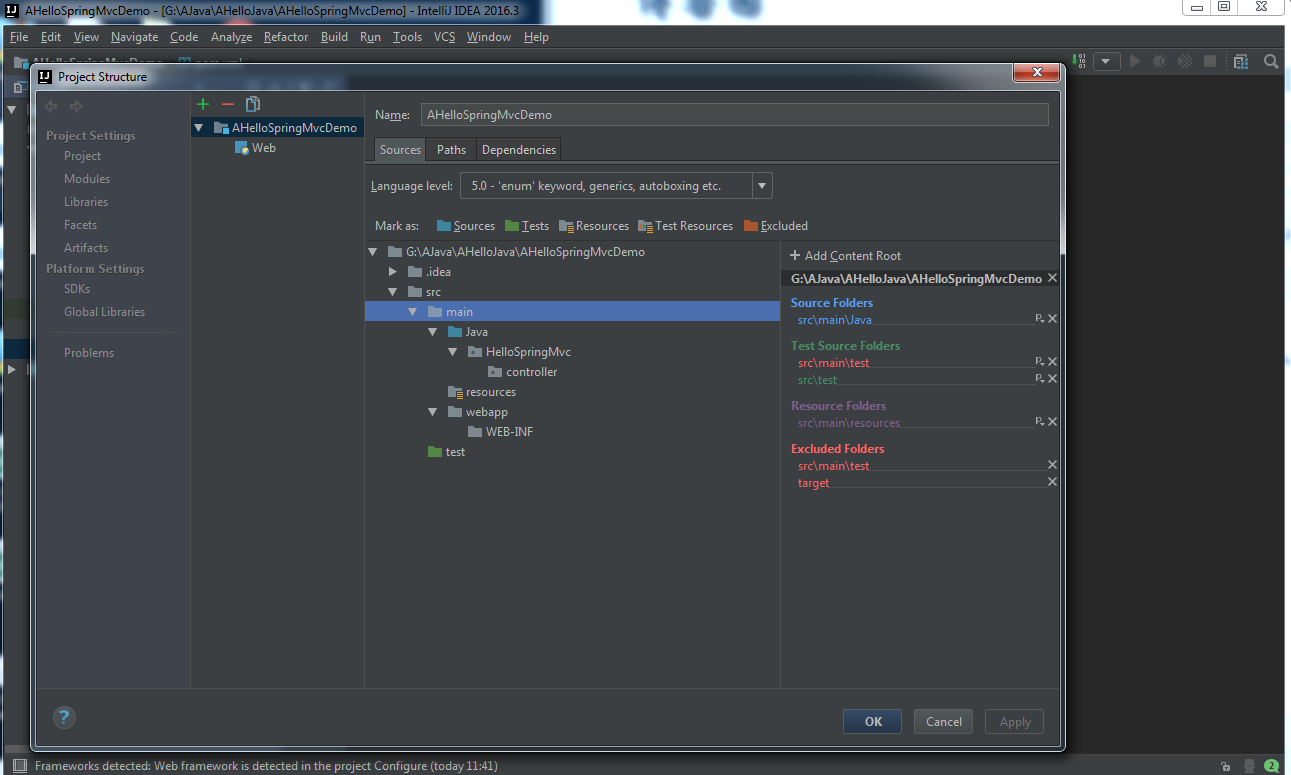
**等系统构建完成以后，我们就可以看到目录结构**



**不全的可以按照需求来补全文件结构，maven项目中的文件结构分为Sources，Tests，Resources，Test Resources，Excluded几种，我们需要详细区分各个文件夹的类型：**

http://images2015.cnblogs.com/blog/998608/201608/998608-20160829173514636-237517065.png

**配置文件夹类型方法如下：**



**可以选中文件夹点击上面的按钮来标注文件夹类型！**

**三、配置SpringMVC**

**一切就绪后接下来我们继续配置SpringMVC的具体信息：**

**首先需要配置Web.xml这个不必多说，网站项目运行第一个加载的就是web.xml，进入**

**src->main->webapp->WEB-INF->web.xml**

[复制代码](javascript:void(0);)

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

2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

3 xmlns="http://java.sun.com/xml/ns/javaee"

4 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

5 version="3.0">

6 <welcome-file-list>

7 <welcome-file>/index.jsp</welcome-file>

8 </welcome-file-list>

9 <!-- Spring MVC配置 -->

10 <servlet>

11 <servlet-name>spring</servlet-name>

12 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

13

14 <!-- 可以自定义servlet.xml配置文件的位置和名称，默认为WEB-INF目录下，名称为[<servlet-name>]-servlet.xml，如spring-servlet.xml

15 <init-param>

16 <param-name>contextConfigLocation</param-name>

17 <param-value>/WEB-INF/spring-servlet.xml</param-value>

18 </init-param>-->

19

20 <!-- load-on-startup元素标记容器是否在启动的时候就加载这个servlet(实例化并调用其init()方法) -->

21 <load-on-startup>1</load-on-startup>

22 </servlet>

23

24 <servlet-mapping>

25 <servlet-name>spring</servlet-name>

26 <url-pattern>/</url-pattern>

27 <!--<url-pattern>\*.do</url-pattern>-->

28 </servlet-mapping>

29

30 <!-- Spring配置 -->

31 <listener>

32 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

33 </listener>

34

35 <!-- 指定Spring Bean的配置文件所在目录。默认配置在WEB-INF目录下 -->

36 <context-param>

37 <param-name>contextConfigLocation</param-name>

38 <param-value>classpath:applicationContext.xml</param-value>

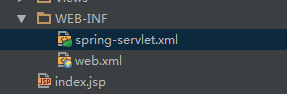
39 </context-param>

40

41 </web-app>

[复制代码](javascript:void(0);)

**然后在WEB-INF文件夹下新建 spring-servlet.xml**

****

**配置spring-servlet.xml这里采用注解的方式：**

[复制代码](javascript:void(0);)

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

2 <beans xmlns="http://www.springframework.org/schema/beans"

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

4 xmlns:context="http://www.springframework.org/schema/context"

5 xmlns:mvc="http://www.springframework.org/schema/mvc"

6 xsi:schemaLocation="http://www.springframework.org/schema/beans

7 http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

8 http://www.springframework.org/schema/context

9 http://www.springframework.org/schema/context/spring-context-3.1.xsd

10 http://www.springframework.org/schema/mvc

11 http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd">

12

13 <!-- 启动注解驱动的Spring MVC功能，注册请求url和注解POJO类方法的映射-->

14 <mvc:annotation-driven >

15

16 </mvc:annotation-driven>

17

18 <!-- 启动包扫描功能，以便注册带有@Controller、@service、@repository、@Component等注解的类成为spring的bean -->

19 <context:component-scan base-package="HelloSpringMvc.controller" />

20 <!-- 对模型视图名称的解析，在请求时模型视图名称添加前后缀 -->

21 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

22 <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>

23 <property name="prefix" value="/"/> <!-- 前缀 -->

24 <property name="suffix" value=".jsp"/> <!-- 后缀 -->

25 </bean>

26 <!-- 访问静态文件（jpg,js,css）的方法 -->

27 <mvc:resources location="/files/" mapping="/files/\*\*" />

28 <mvc:resources location="/scripts/" mapping="/scripts/\*\*" />

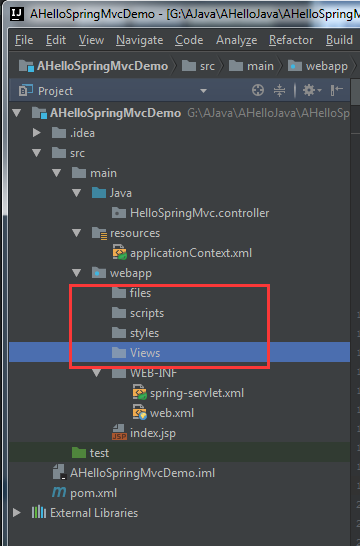
29 <mvc:resources location="/styles/" mapping="/styles/\*\*" />

30 <mvc:resources location="/Views/" mapping="/Views/\*\*" />

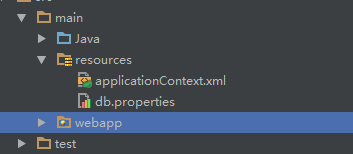
31 </beans>

[复制代码](javascript:void(0);)

**这里我们配置了静态文件路径，因此在目录结构新建几个文件夹用以放静态文件例如脚本，图片文件，视图之类的。**



**然后在Resources资源文件夹下新建applicationContext.xml**



**配置applicationContext.xml这里并没有进行配置什么信息，因为用的是mysql jdbc连接数据库，没有用到Hibernate之类的orm，这里可以将hibernate相关配置项放在里面。**

[复制代码](javascript:void(0);)

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

2 <beans

3 xmlns="http://www.springframework.org/schema/beans"

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

5 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

6 <!-- 我们可以在其中添加我们所需要配置的bean，也可以添加相应的数据库连接和事务处理等等，方便后续拓展

7 -->

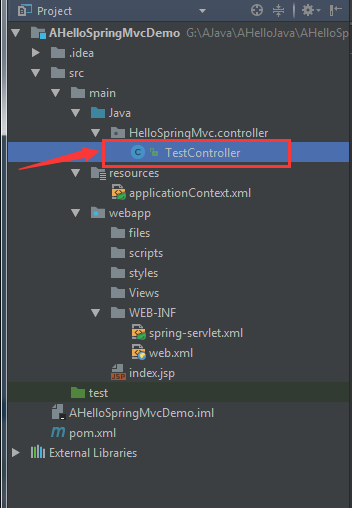
8 </beans>

[复制代码](javascript:void(0);)

**这里我们就将SpringMVC的基本XML配置文件设置完毕，接下来我们进行一个小实例进行测试配置好的SpringMVC。**

**四：测试SpringMVC配置：**

**1、在controller文件夹下新建一个类 名字叫TestController，代码如下：**

****

[复制代码](javascript:void(0);)

package HelloSpringMvc.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.ResponseBody;

/\*\*

\* Created by Administrator on 2016/8/29.

\*/

@Controller

@RequestMapping(value = "/Test/\*") //访问的url地址前缀，可以不写，写了就必须在方法url前面先加上class url 进行区分控制器

public class TestController

{

//访问地址：http://localhost:8080/Test/returnSuccess

@RequestMapping(value = "returnSuccess") //实际访问的url地址

public String returnSuccess() {

return "/Views/success"; //返回Views文件夹下的success.jsp页面

}

//访问地址：http://localhost:8080/Test/returnString

@RequestMapping(value = "returnString", produces = {"text/plain;charset=UTF-8"})

//produces用于解决返回中文乱码问题，application/json;为json解决中文乱码

@ResponseBody //用于返回字符串,不写即返回视图

public String returnString() {

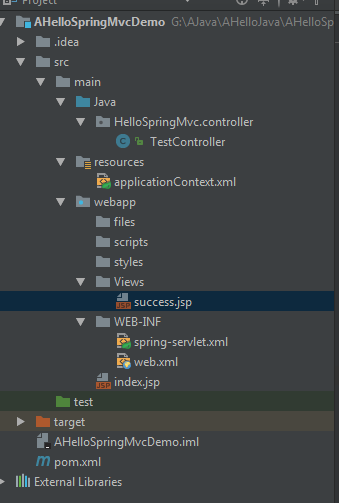
return "hello return string 这是中文，并没有乱码";

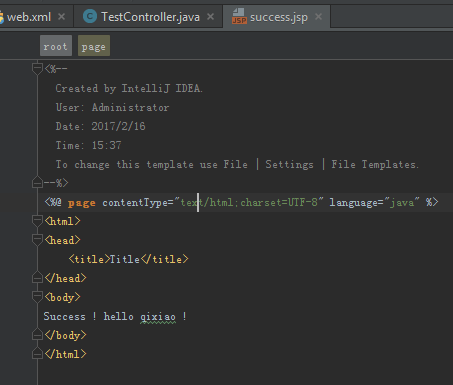
}

}

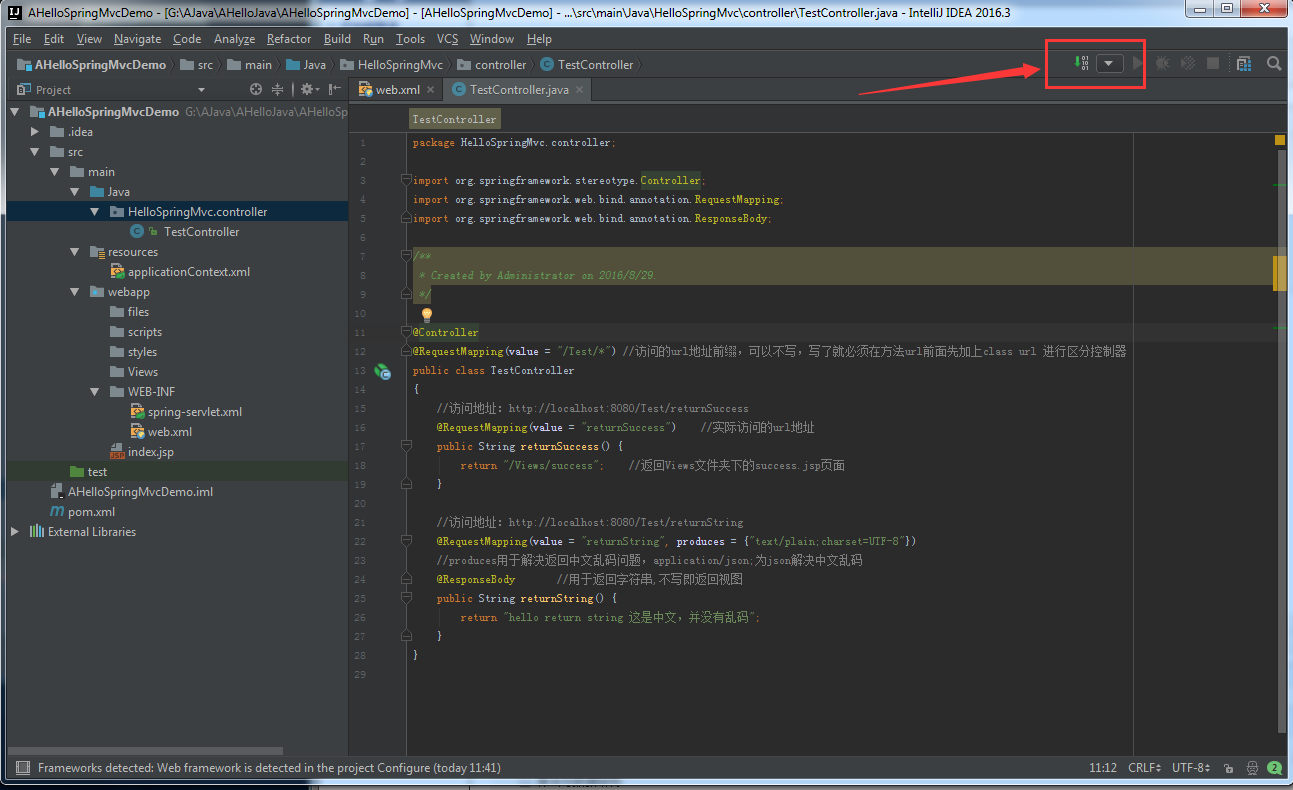
[复制代码](javascript:void(0);)

**然后我们在Views文件夹下建一个success.jsp 随便写点什么：**

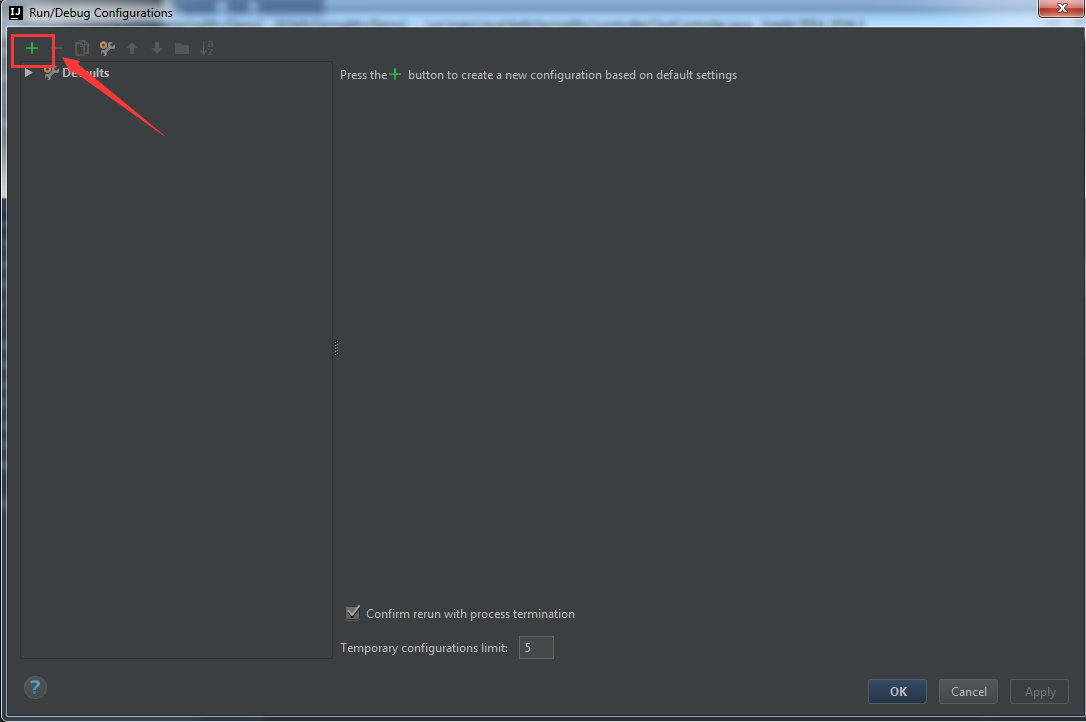
****



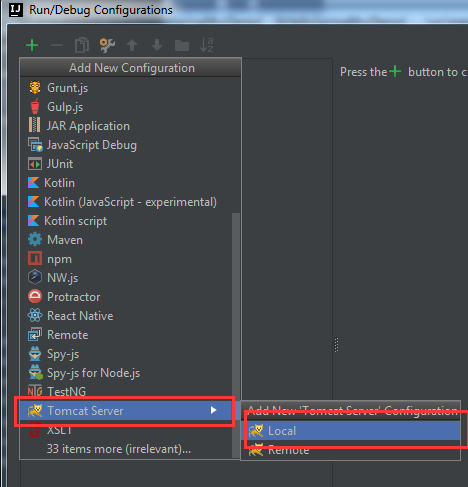
**然后试运行，IDEA需要先进行配置运行环境，配置如下->**



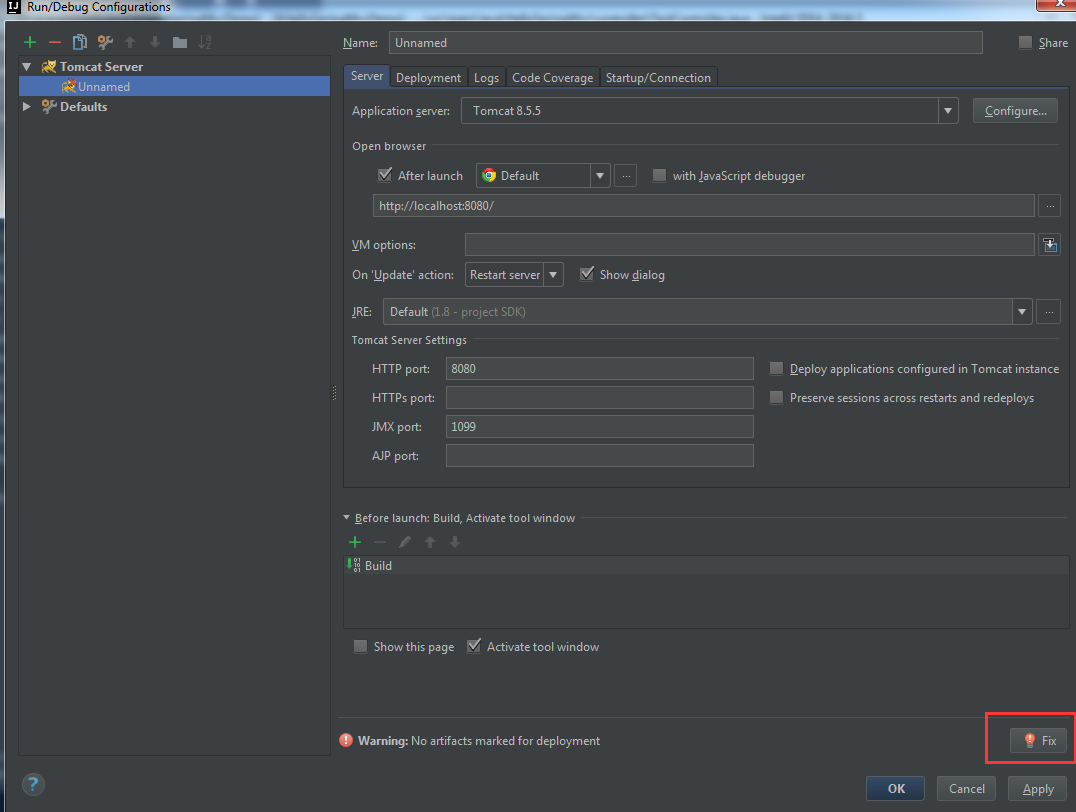
**在弹出的选项卡里面选择“+”号进行添加运行环境！**

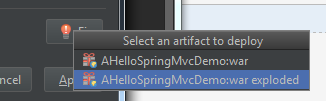


**本人这里选用TomCat启动项目:**

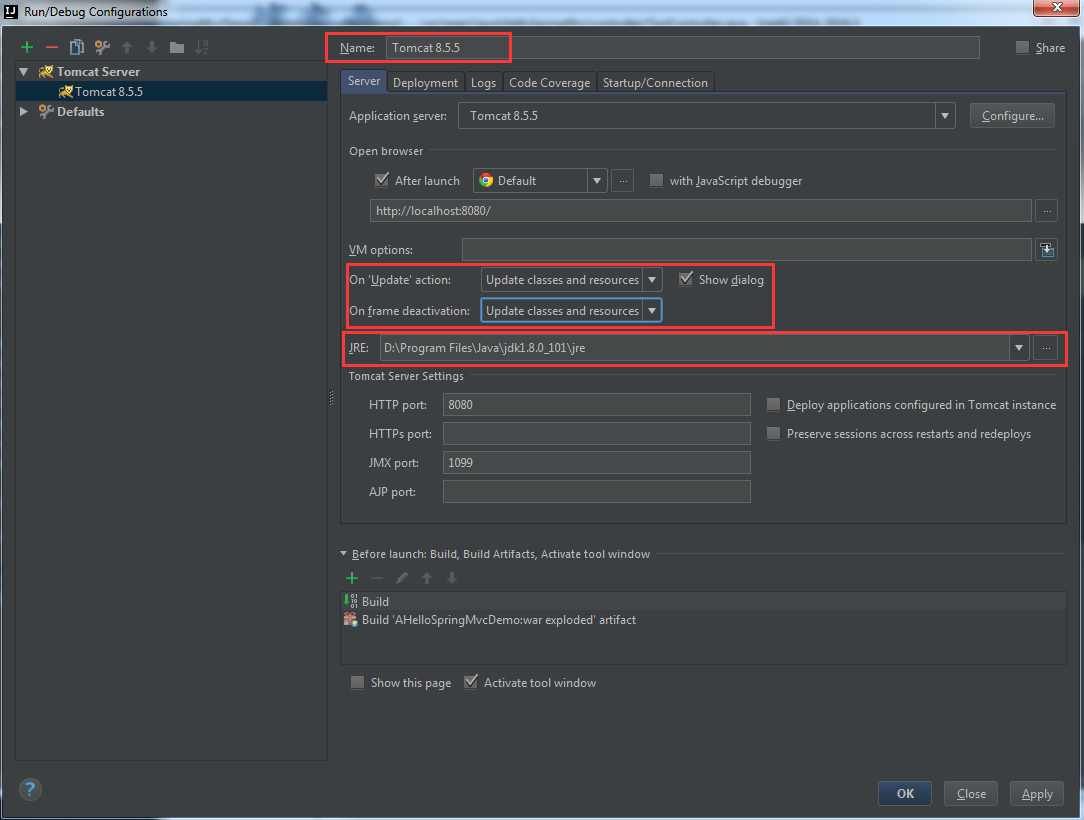


然后在弹出的选项卡选择Fix



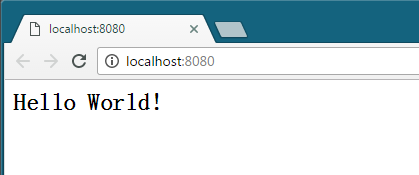


**然后应用即可，当然还有一些可选配置如下（这是我的配置）：**

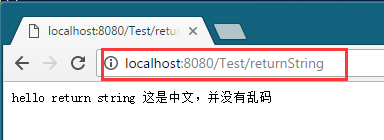


**接下来就可以运行了吧！！！**

**好,点击运行，google自动弹出了网页！**

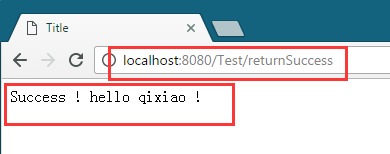
****

**运行后，在浏览器地址栏输入：http://localhost:8080/Test/returnString**



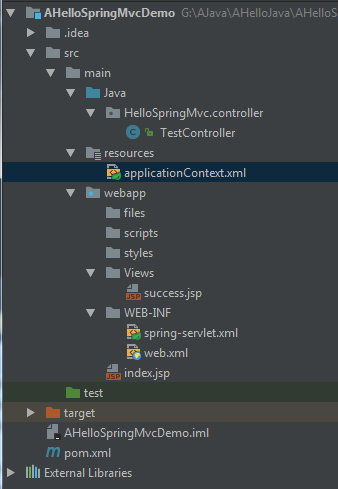
**可以看到我们运行第二个url地址可以成功返回字符串，而且并没有中文乱码！**

**下面测试第一个url地址：http://localhost:8080/Test/returnSuccess**



**可以看到，成功返回到success.jsp ! 说明我们的SpringMVC并没有配置错误。**

**再次将项目目录结构附上->**

****