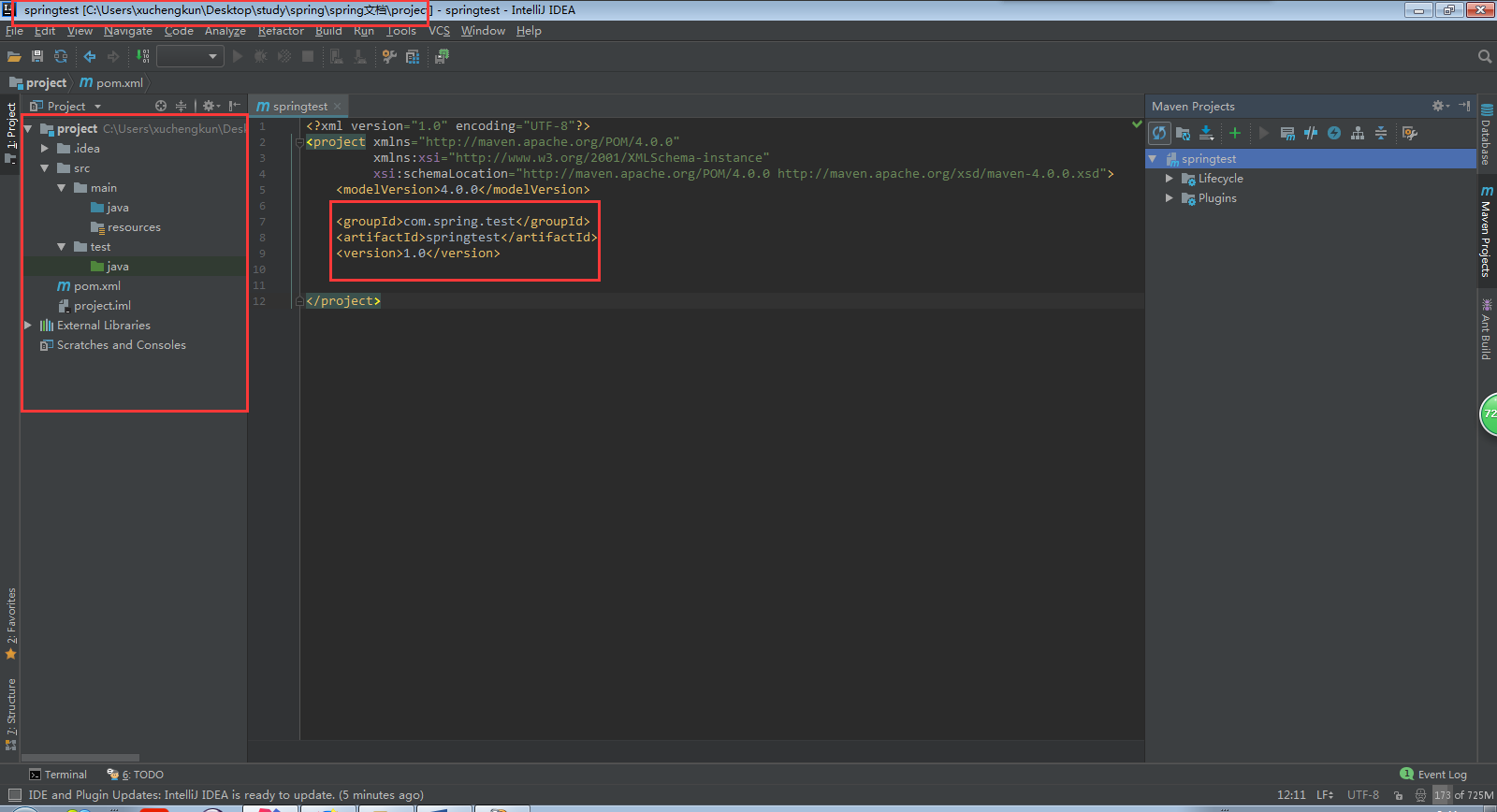
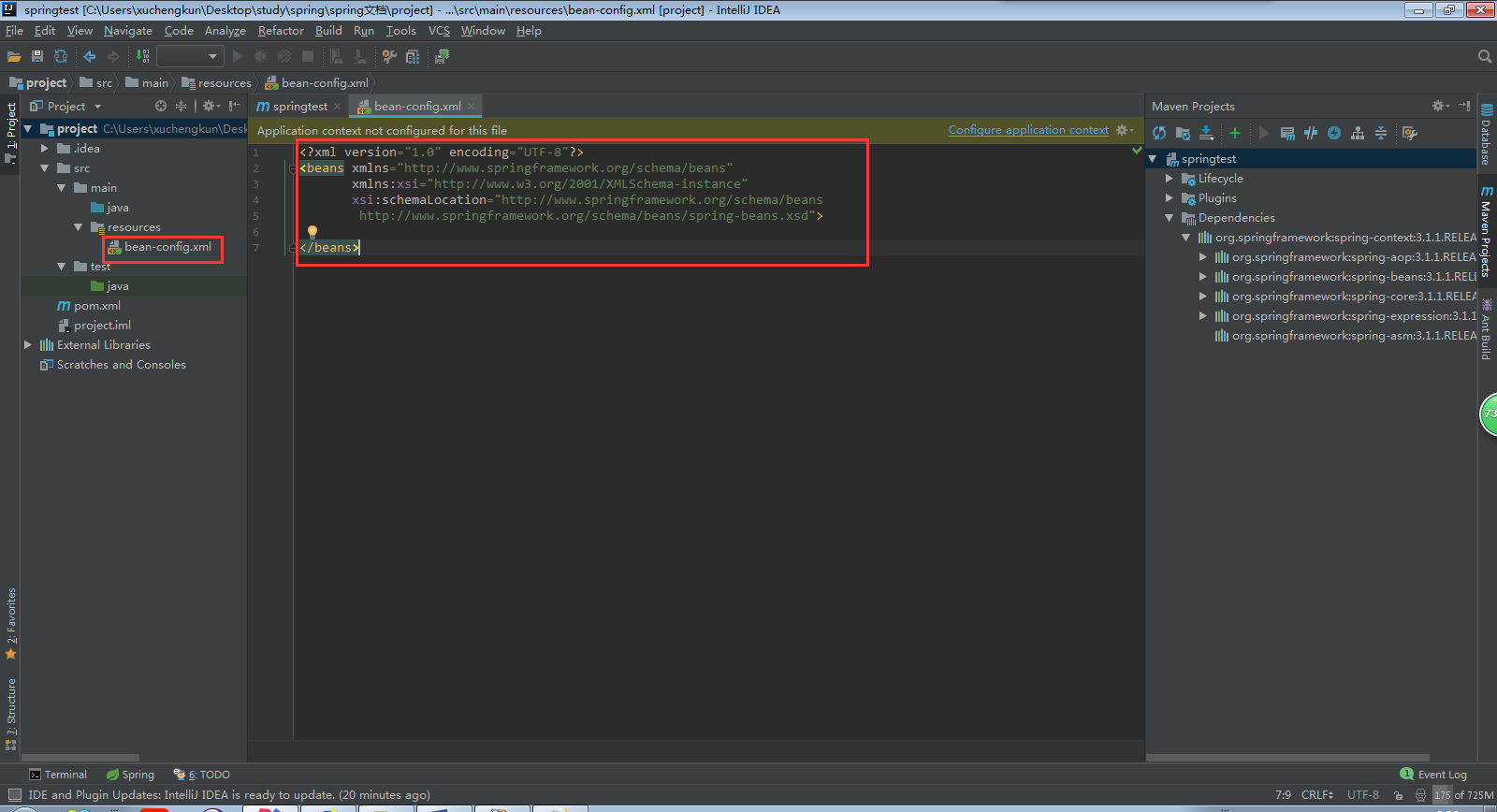
## 1.新建Maven项目

在IDEA上新建maven项目。



## 2.添加applicationContext.xml配置文件

在resources目录下，配置applicationContext.xml文件，会看见标签头的beans有很多的红色。下面的图片是之前的，名字改成applicationContext.xml。



对应的配置：

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

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

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

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

http://www.springframework.org/schema/beans/spring-beans.xsd"></beans>

在pom.xml文件中引入

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.1.1.RELEASE</version>

</dependency>

## 3.增加单元测试

为了增加单元测试，在pom.xml文件中引入junit包：

<dependency>

<groupId>junit</groupId>

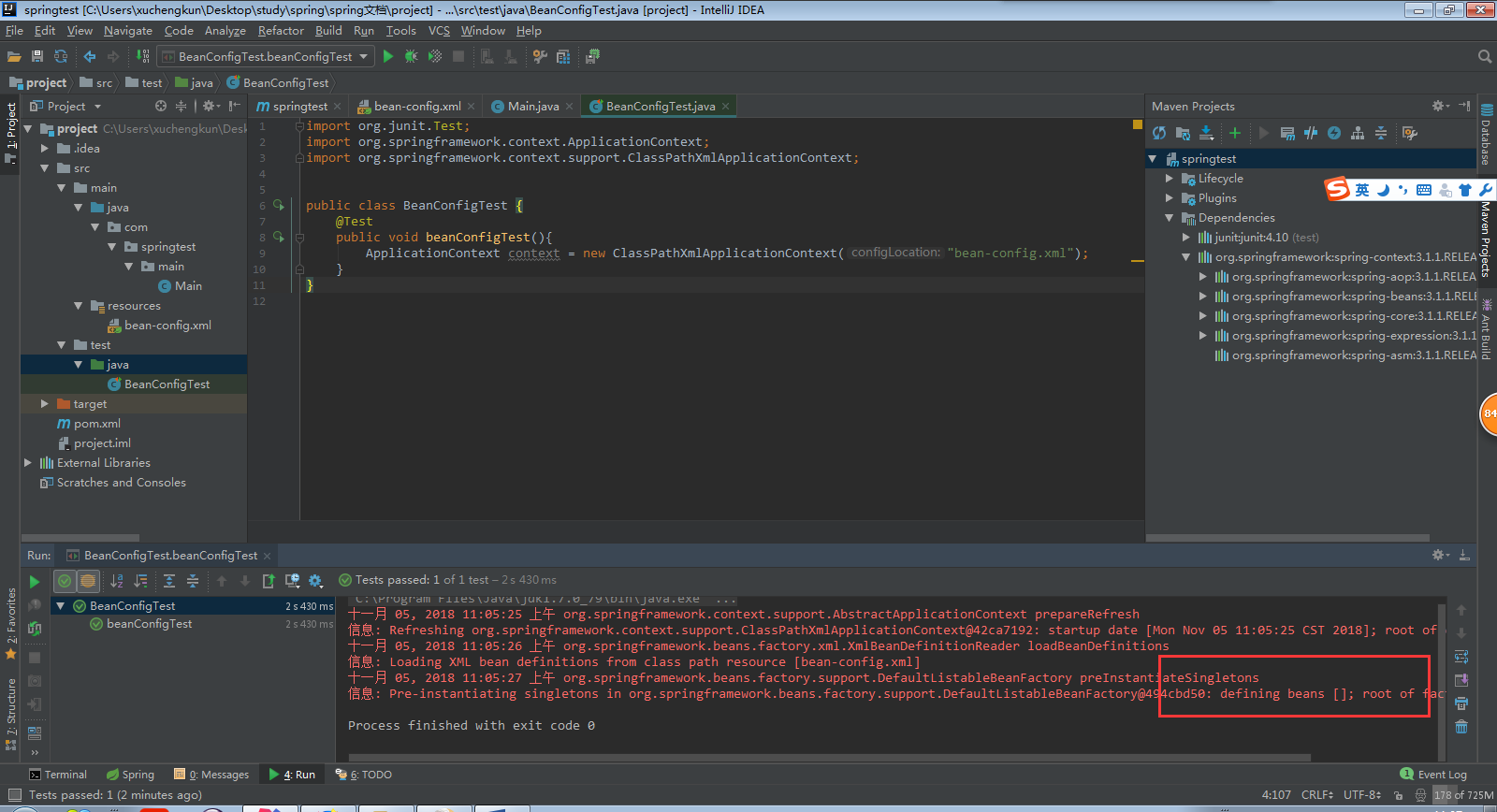
<artifactId>junit</artifactId>

<version>4.10</version>

<scope>test</scope>

</dependency>

测试配置文件的加载情况：



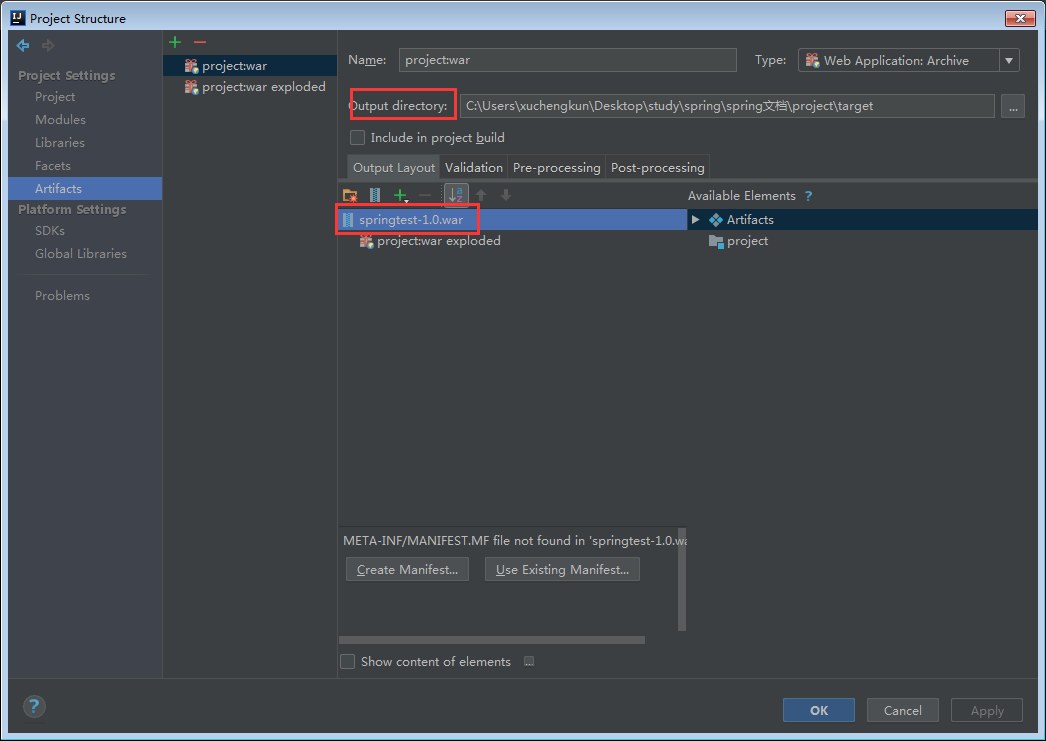
可以看出已经成功加载。

## 4.改成web项目

因为之前使用的是maven项目，现在要改成maven web项目。

### 4.1.修改maven的打包方式

在pom.xml中上部添加<packaging>war</packaging>，刷新，可以在project structure中看见多出来war包(之前是没有的)



其中输出目录是用来将打包的war包放在哪里

### 4.2.创建web目录和web.xml

在src/main下创建webapp/WEB-INF这两个目录

在WEB-INF下创建web.xml文件

配置，并指定欢迎页面

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://java.sun.com/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
 http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 version="3.5">  
 <welcome-file-list>  
 <welcome-file>index.jsp</welcome-file>  
 </welcome-file-list>  
</web-app>

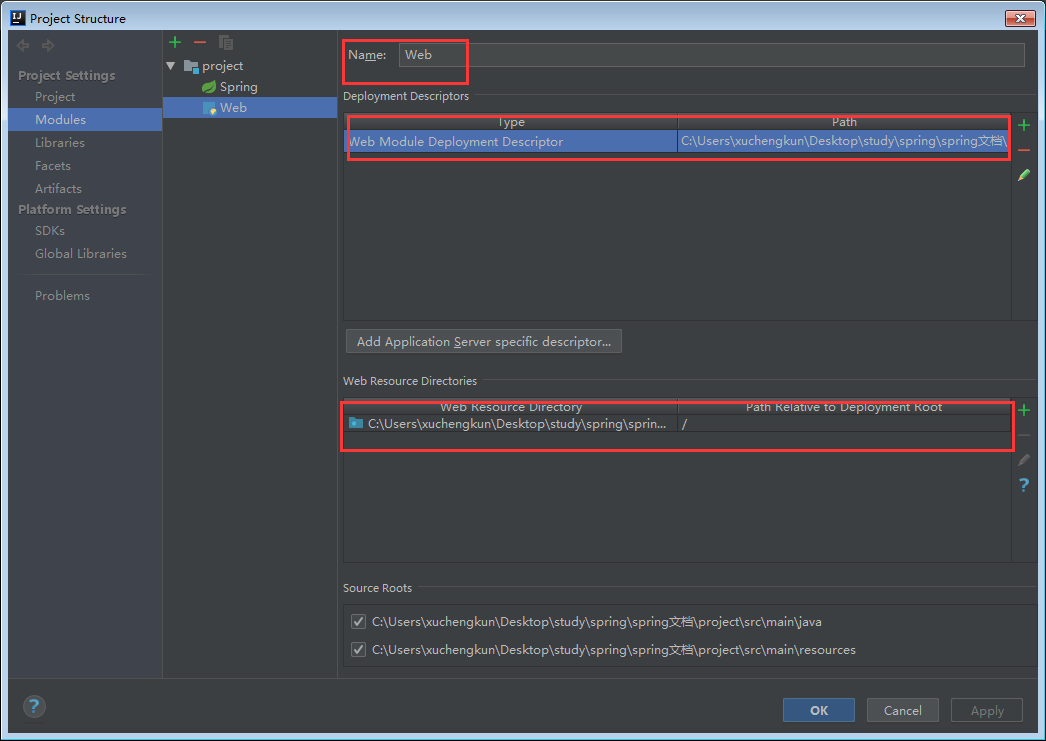
在webapp下创建index.jsp页面

<%--  
 Created by IntelliJ IDEA.  
 User: xuchengkun  
 Date: 2018/11/5  
 Time: 11:47  
 To change this template use File | Settings | File Templates.  
--%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
 <head>  
 <title>欢迎</title>  
 </head>  
 <body>  
 <h1>欢迎</h1>  
 </body>  
</html>

### 4.3.添加web模块

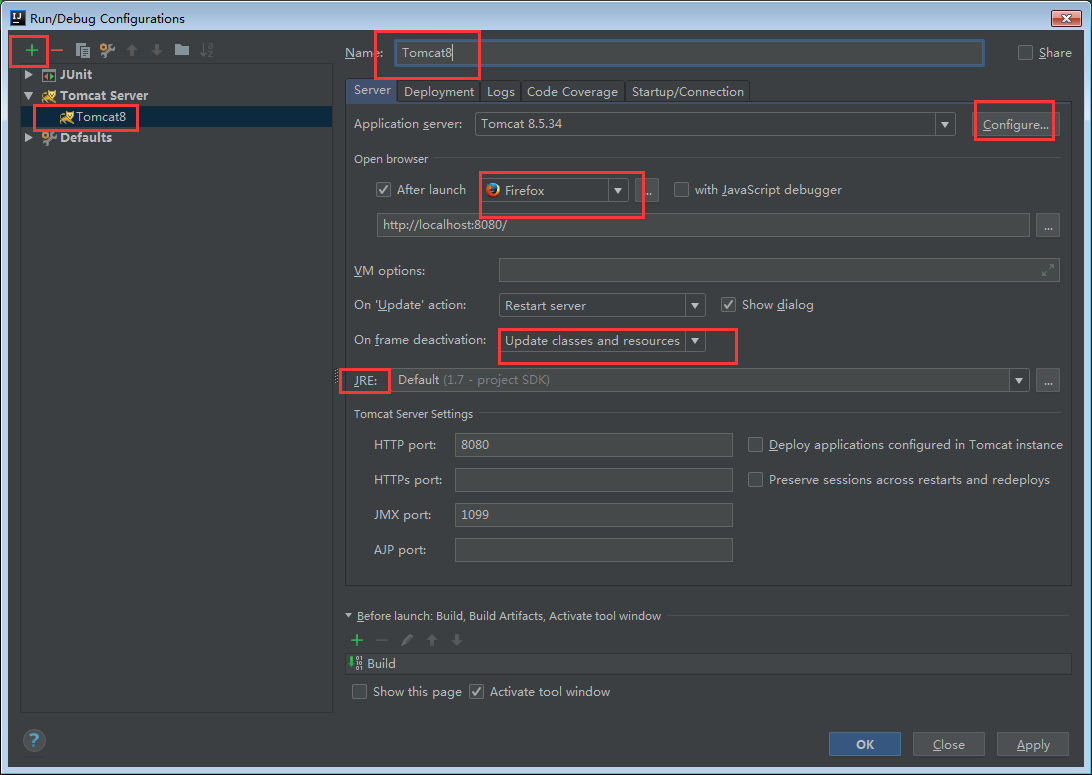
打开file->project structrue->Facts，点击+号，添加web。   
修改Deployment的Path到web.xml上。

修改Web Resources目录到webapp上



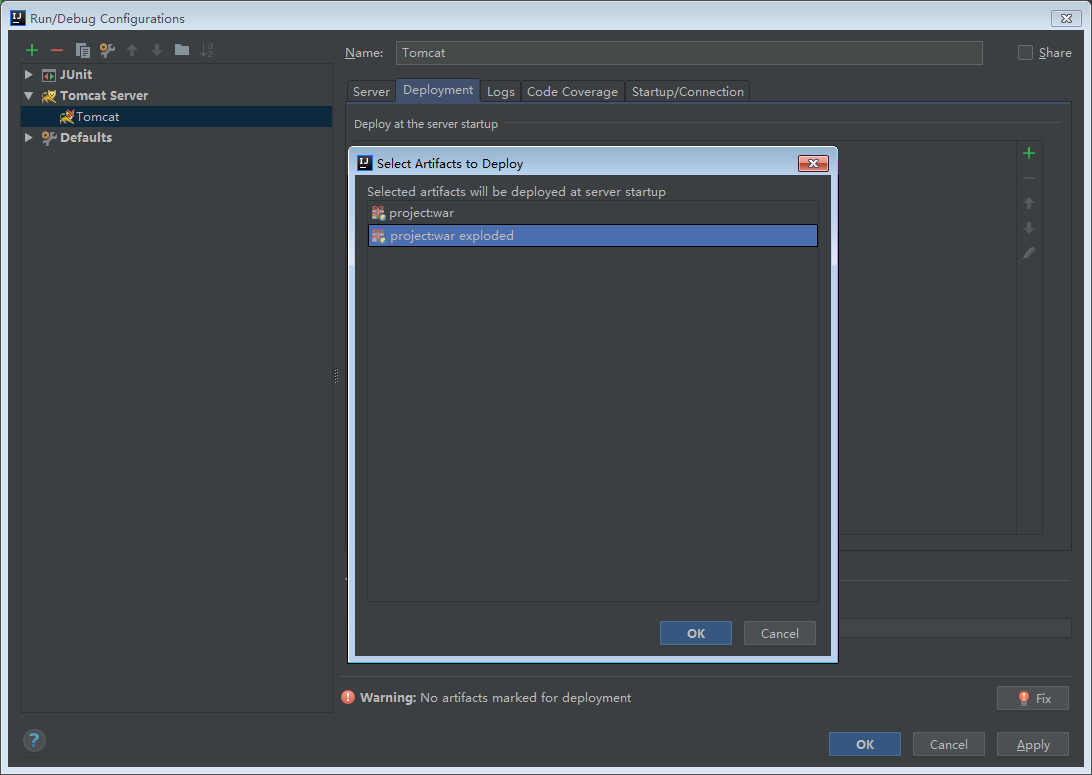
### 4.4 添加Tomcat

打开Run->Edit Configurations，点击+号，选择Tomcat服务器。

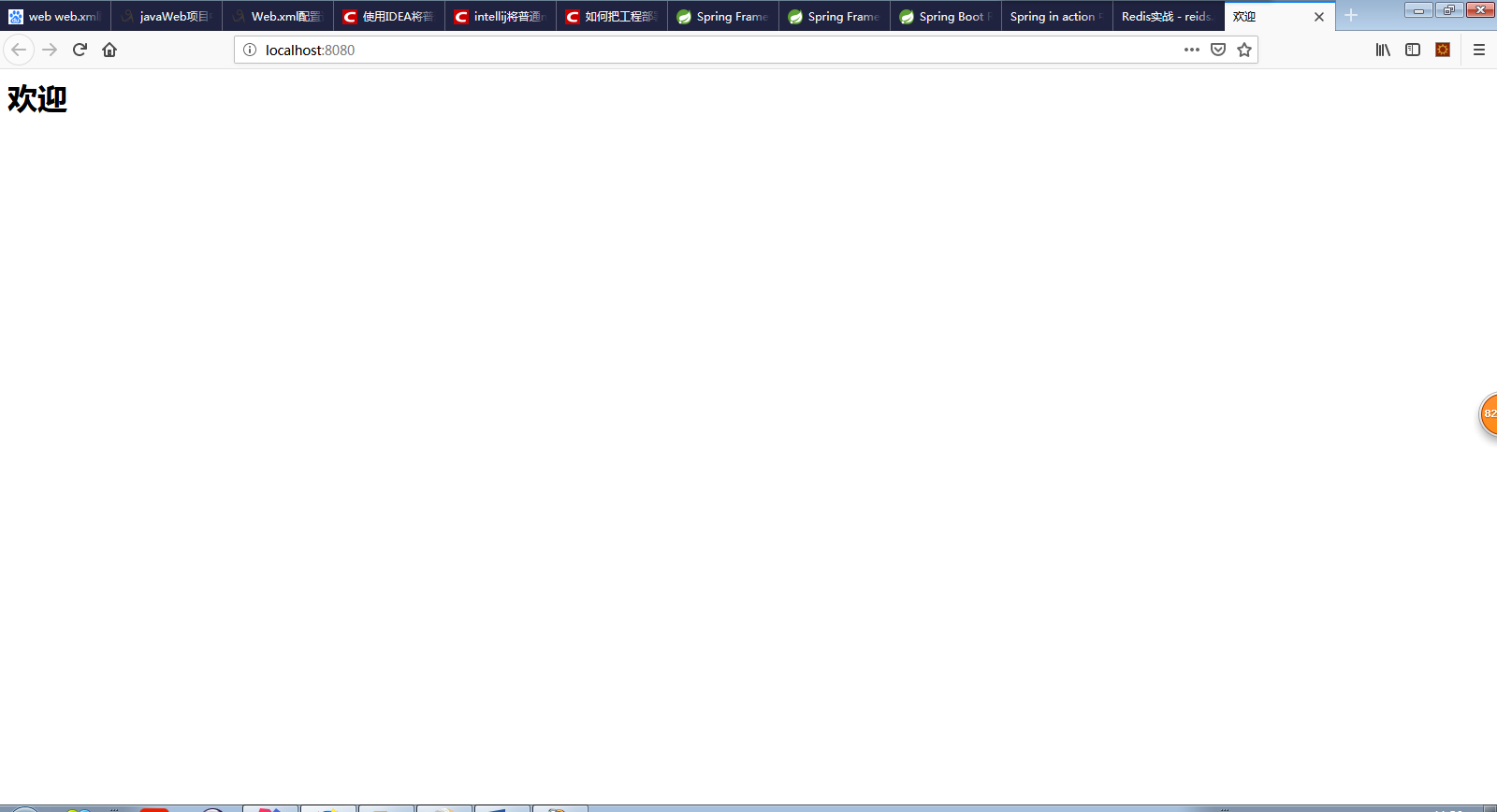


注意要修改服务器名字，启动的浏览器，增加热部署update classes and resources，最后注意jre版本。

### 4.5 部署war包



### 4.6 启动Tomcat



## 5.新建层次目录

将springtest重命名为family

在src/main/java/com/family目录下新建：

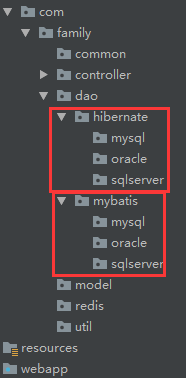
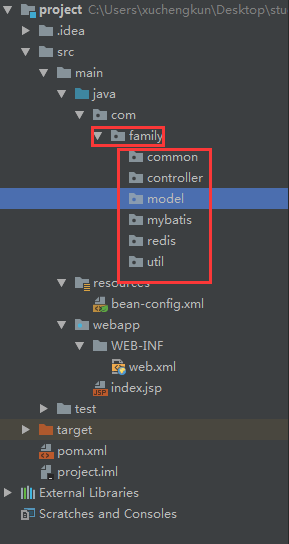
common中就是放入一些公共的包，类，接口等等

model放入实体类

controller放入控制器

dao 和数据库交互 后面可能会集成mybatis hibernate，每种分为mysql，oracle，sqlserver

注意：下面左边图片也是旧的，这里多了dao文件目录



## 6.实现请求的映射

我现在要实现一个家庭物品存储位置分布的页面显示

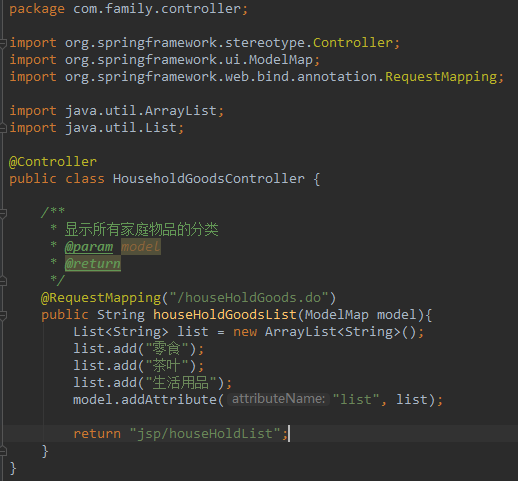
### 6.1 创建一个控制器

在controller目录下创建HouseholdGoodsController.java，

为了表示它是一个控制器，在class上面加上@Controller

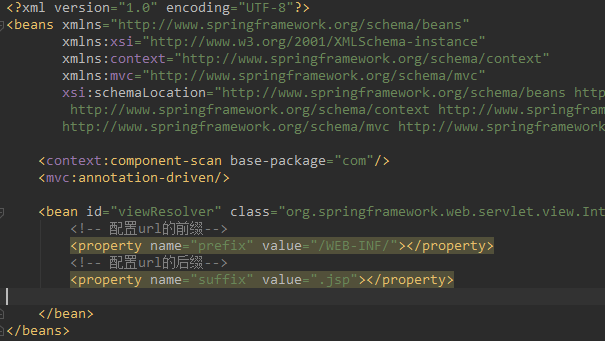
创建一个方法加上@RequestMapping发现没有这个注解，在pom.xml中引入一个新的依赖

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>3.1.1.RELEASE</version>  
</dependency>



创建一个映射方法，通过ModelMap将数据返回，并跳转到houseHoldList.jsp页面

为了启动这个控制器，形成bean放入spring容器中，在bean-config.xml文件中配置



其中component-scan属性指的是要扫描的基础包。在加上一个视图解析器用来拼接返回的URL。

### 6.2 创建一个jsp页面

在WEB-INF下新建jsp目录专门用来存放jsp页面的。同时创建一个houseHoldList.jsp页面。修改index.jsp，增加一个链接用于跳转。

<a href="houseHoldGoodsList.do">物品分类</a>

但是启动Tomcat会发现根本请求不到控制器。使用查看元素，发现请求地址并没有涉及到项目，只是单纯的在localhost后面多了一个上面的链接。

所以这里为了获取web项目的根路径，添加了，：

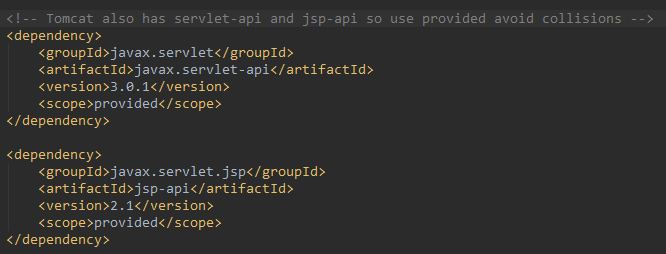
**<%** String path = request.getContextPath();  
 String basePath = request.getScheme()+"://"  
 +request.getServerName()+":" + request.getServerPort() + path+"/";  
**%>**

在<head>标签中添加：

<base href=" **<%=**basePath**%>**">

这样所有的链接都会在新签名自动加上basePath。

然后发现getContextPath是红色的，这是因为没有引入包jsp-api和servlet-api的原因，无法解析。

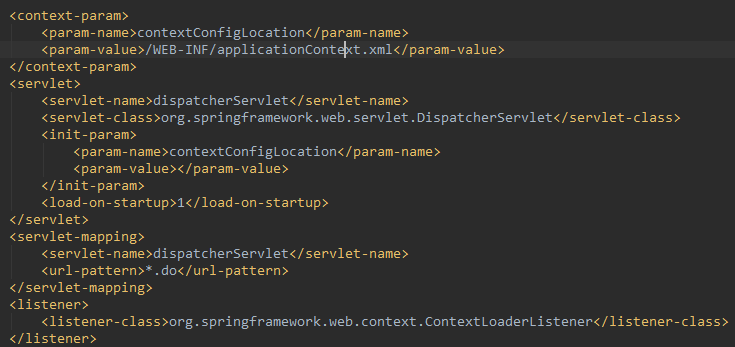


使用provided的原因：tomcat中也有servlet-api包，这样就会发生了冲突。

可以通过添加<scope>provided</scope>解决，因为provided表明该包只在编译和测试的时候用，所以，当启动tomcat的时候，就不会冲突了。

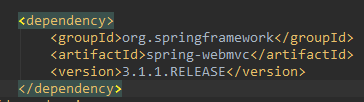
### 6.3 创建DispatcherServlet处理请求

在web.xml文件中配置：



其中这个Dispatcher会拦截所有以.do结尾的请求。然后会加载bean-config.xml配置文件。

可以看见红色的地方，会出现没有这个类的情况，这里引入依赖spring-webmvc：



### 6.4 渲染前端页面

为了能够遍历spring控制器传回来的list，这里使用jstl，所以这里需要引入标签库

为了能够不在每个页面都重复写taglib的定义，在webapp下新建common/taglib.jsp用来充当公共的标签库。

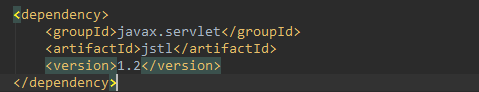
在houseHoldList.jsp页面中



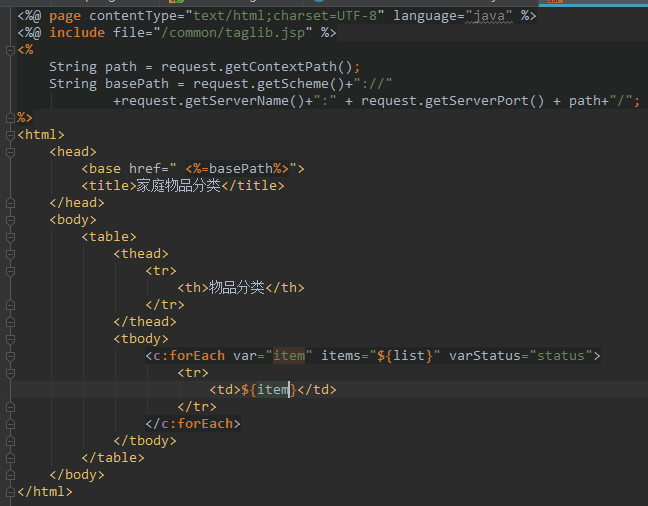
在taglib.jsp中加上



红色，没有引入jstl，在maven中加上：



使用c:forEach遍历到表格中



## 7. 整理请求流程

Spring解析web.xml文件，通过contextConfigLocation加载spring配置文件，解析配置文件，扫描组件，注册映射，启用注解，实例化视图解析器。

index.jsp请求发起houseHoldGoods.do请求，会通过获取项目根目录来拼接请求/houseHoldGoods.do，通过DispatcherServlet分派器，来分派请求，对应到映射之中，通过ModelMap回带会分派器，拼接返回的URL，前缀+str+后缀，进行渲染。

## 8. 最后项目结构

