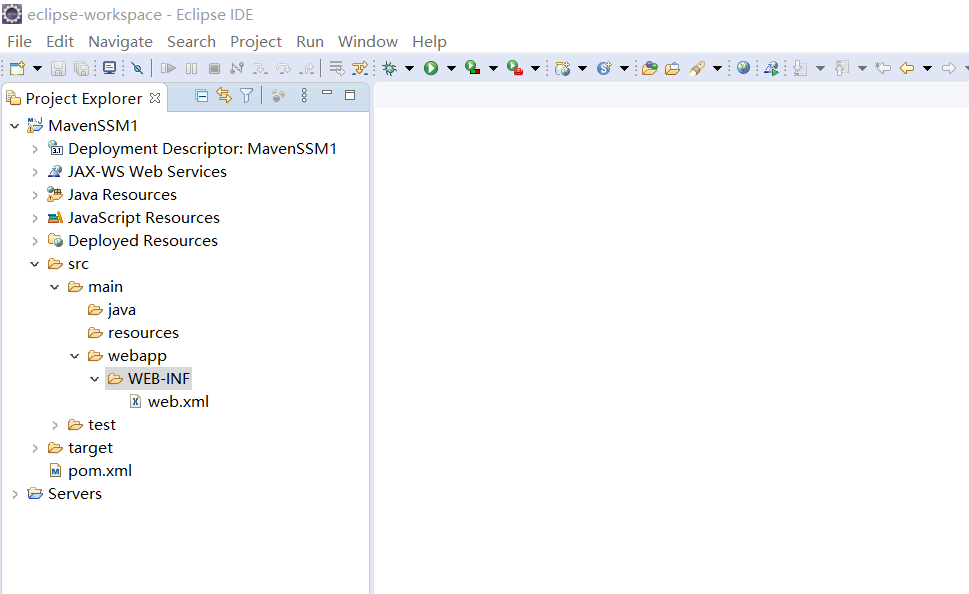
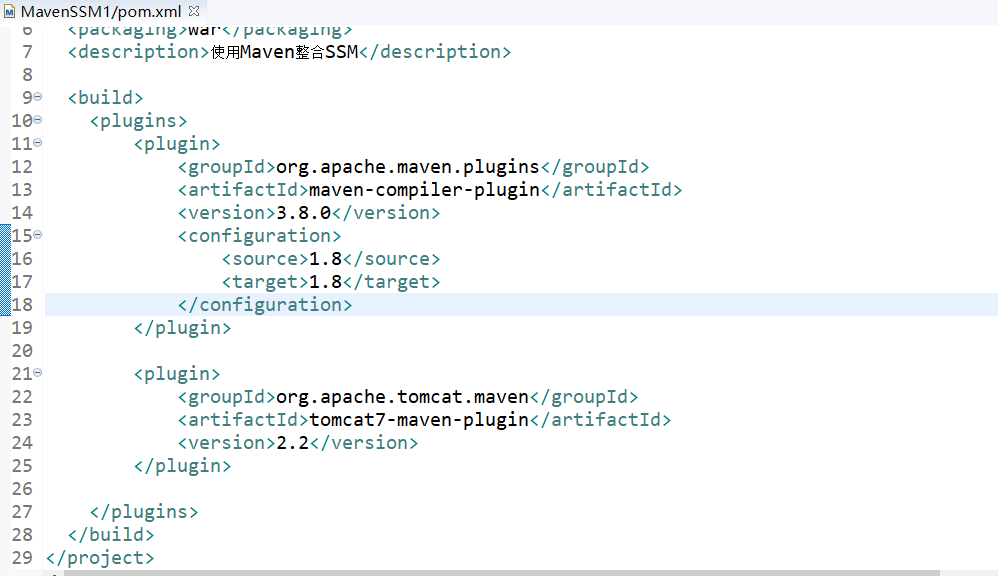
# Maven整合SSM

1. 创建Maven项目并完善web.xml文件。如下：

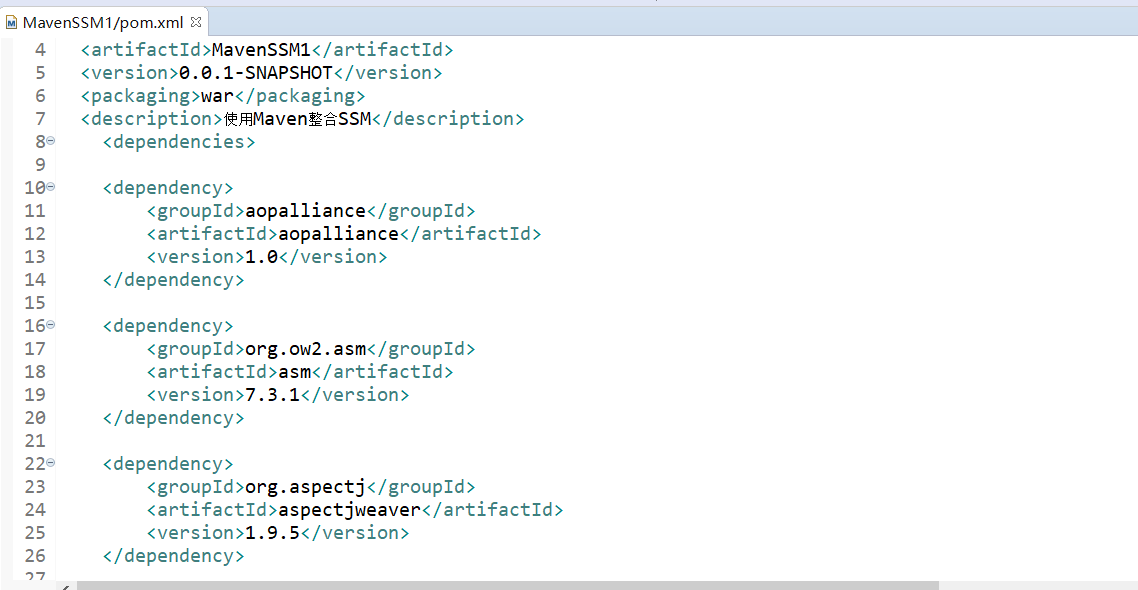


1. 在pom.xml文件中配置JDK编译插件和tomcat7插件。如下：

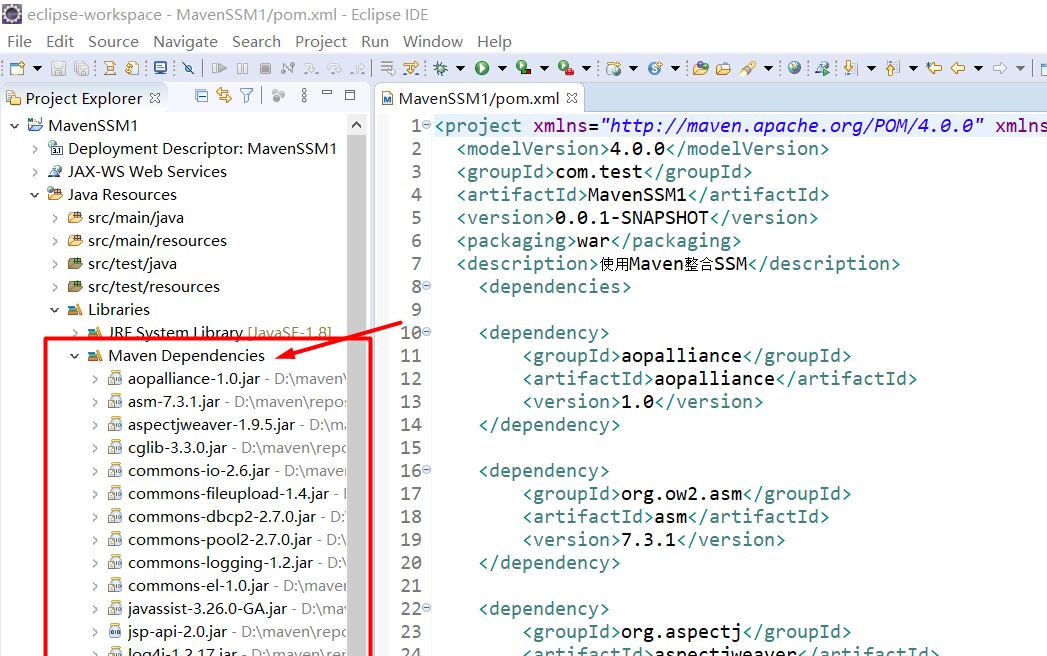


由于配置了JDK插件，所以需要更新一下Maven项目。

1. 通过依赖管理的形式导入SSM开发所需的jar包。如下：

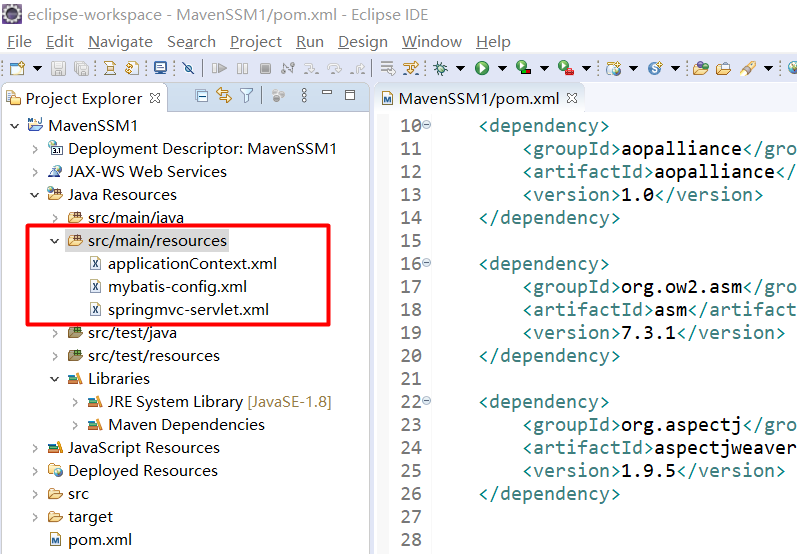


配置依赖后，在项目目录结构中会看到配置依赖导入的jar包。如下：

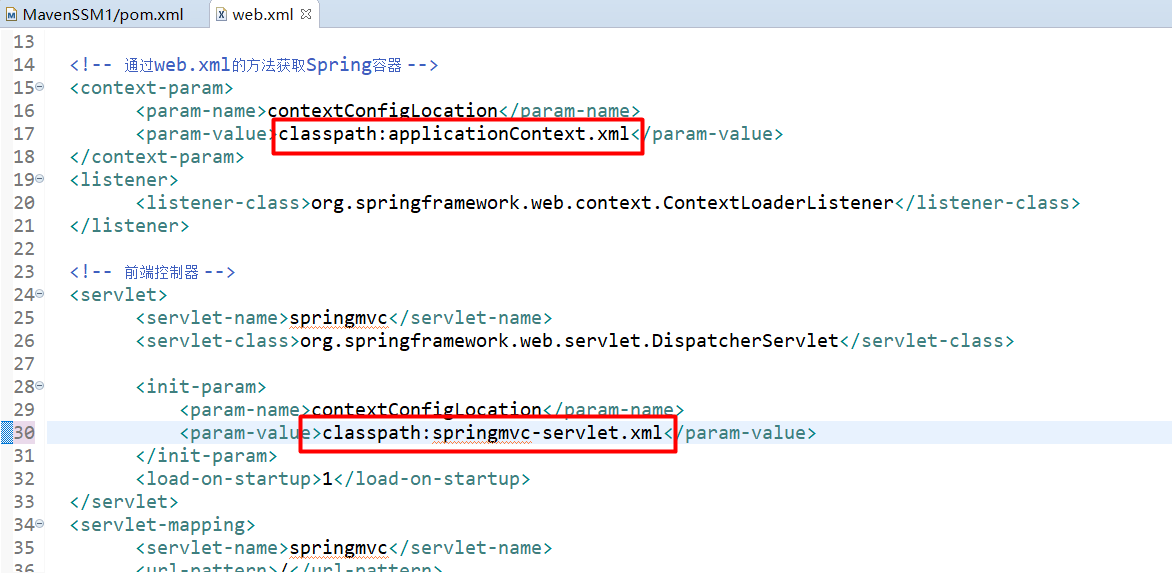


如果你的本地仓库中没有SSM所需的jar包，而你计算机中有SSM整合所需的jar包，这时候，你需要根据这些jar包的名字去中央仓库中寻找依赖模板，然后粘贴到pom.xml文件中。这时它会自动根据你配置的依赖去本地仓库中查找，但由于本地仓库没有，所以它会去中央仓库中下载到本地仓库中。之后再次进行Maven整合SSM时就不必去中央仓库查找依赖模板了。

1. 在src/main/resource目录下创建SSM框架的配置文件。如下：

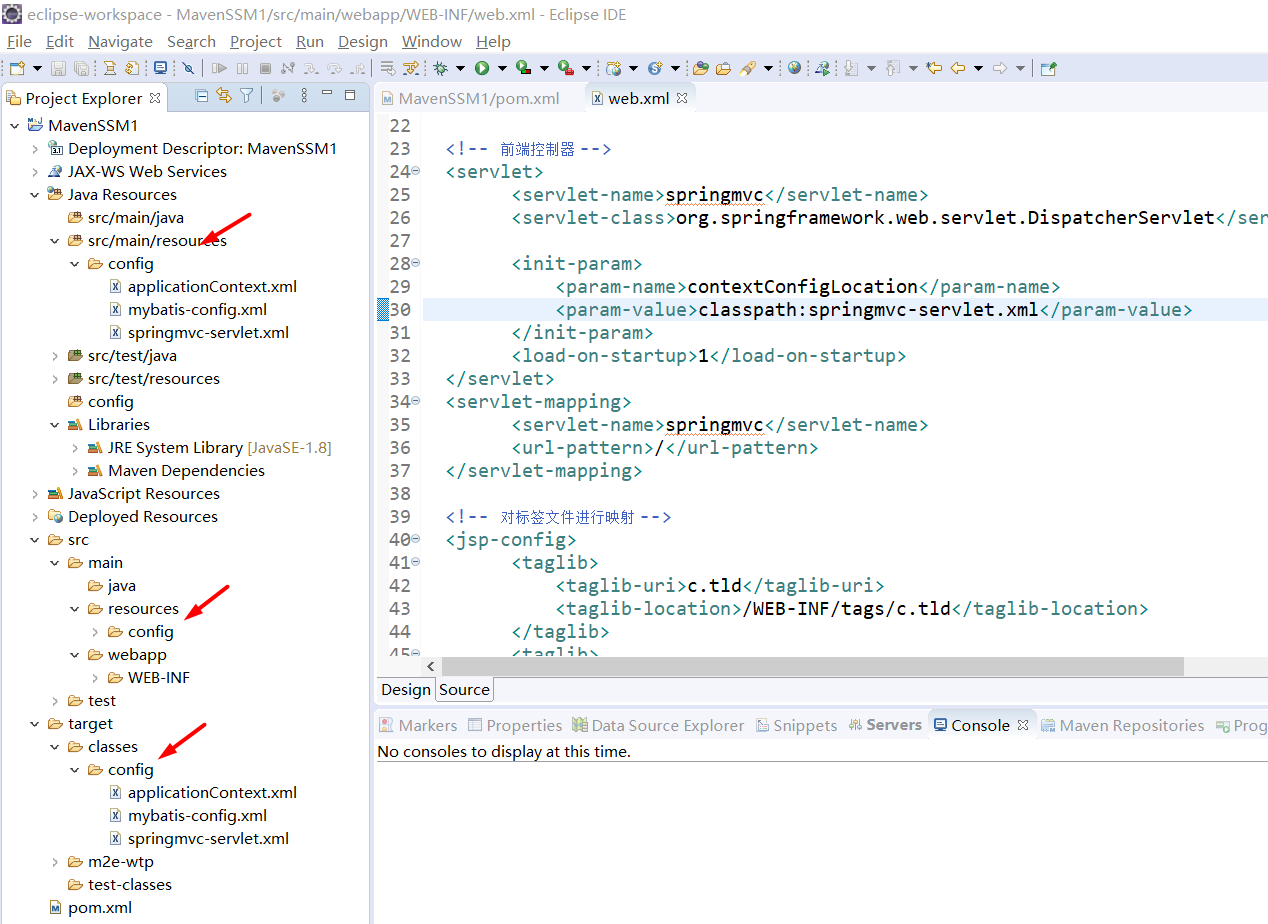


这里需要注意的是，src/main/java和src/main/resource/映射目录为classes目录，所以你在src/main/resource目录下创建配置文件就好比将配置文件保存在classes目录下。如果你是直接在src/main/resources根目录下创建SSM配置文件的话，那么你在web.xml定位配置文件时如下：

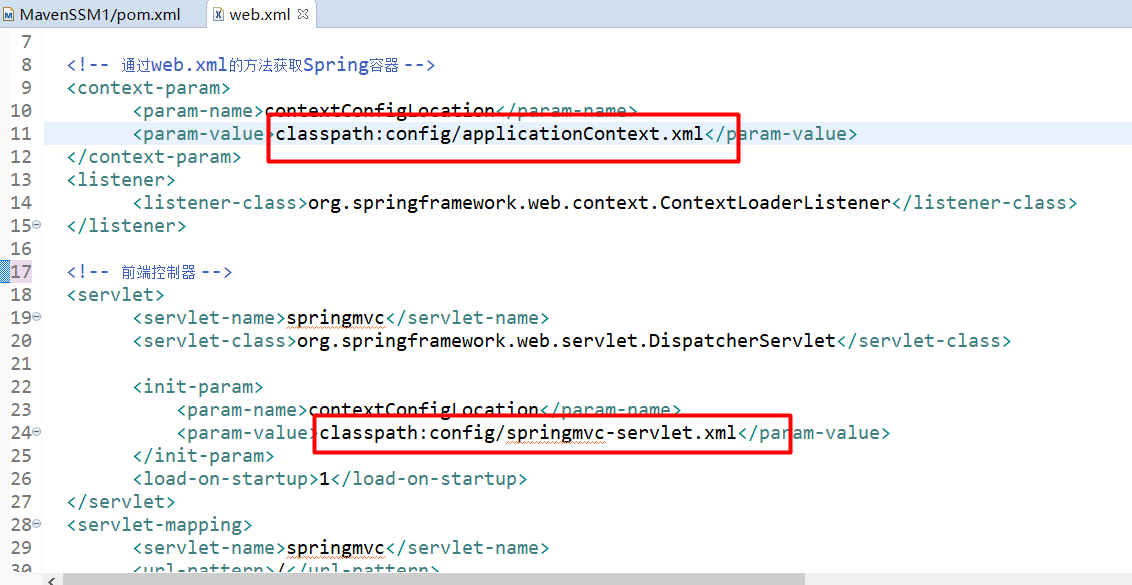


因为classes目录表示的就是classpath路径。

如果你的配置文件并不是直接放在src/main/resources根目录下，而是在src/main/resource下创建一个包（文件夹），将配置文件放入该包中。如下：

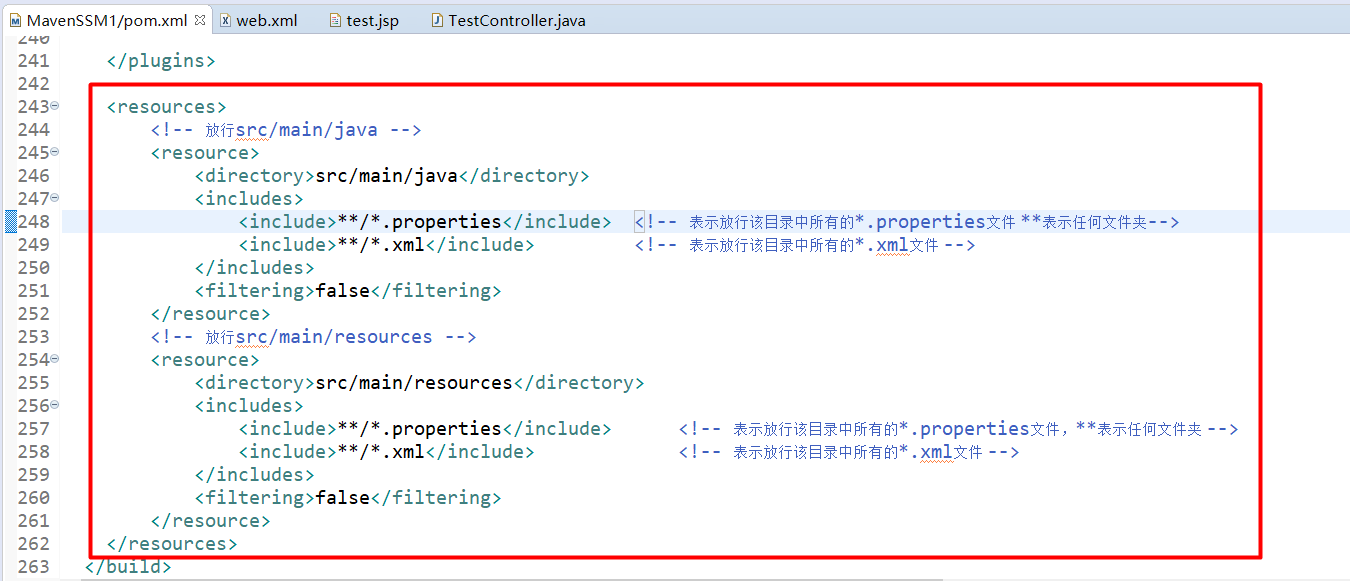


那么你在web.xml文件中引用时，就必须加上配置文件所在的包名称。如下：

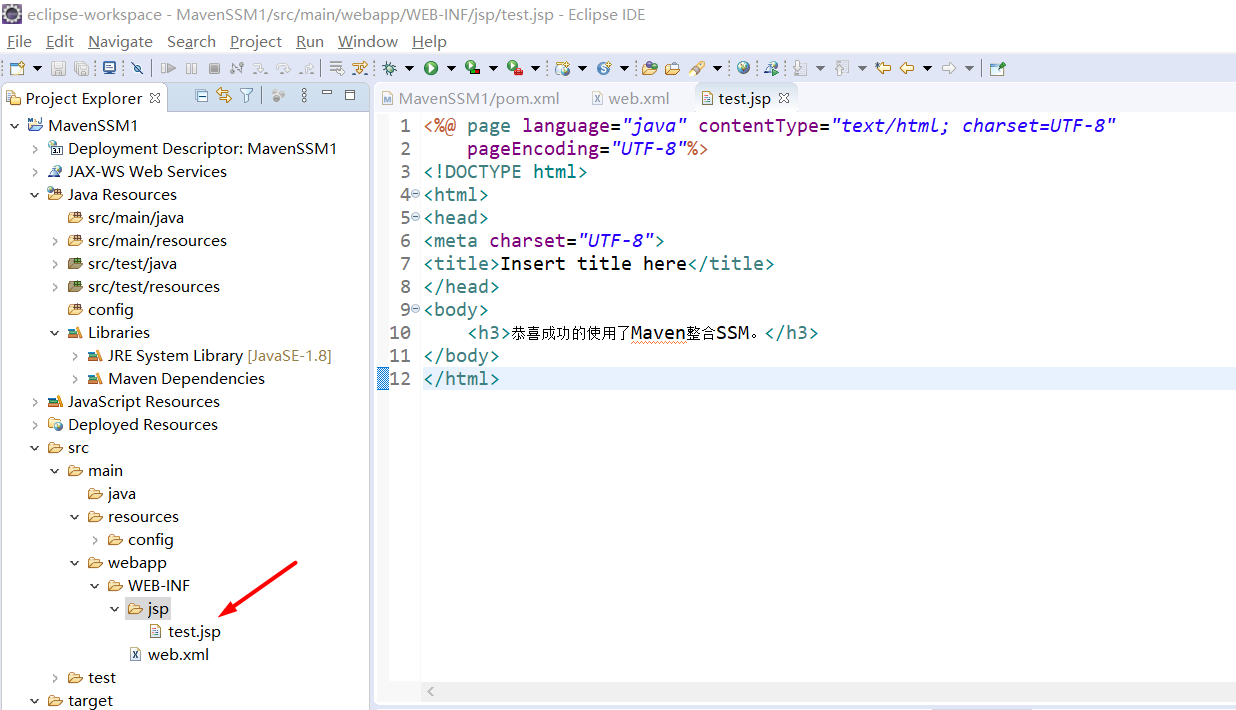


但不管怎么样，配置文件不是放在src/main/resources根目录下就是放在其子目录下。

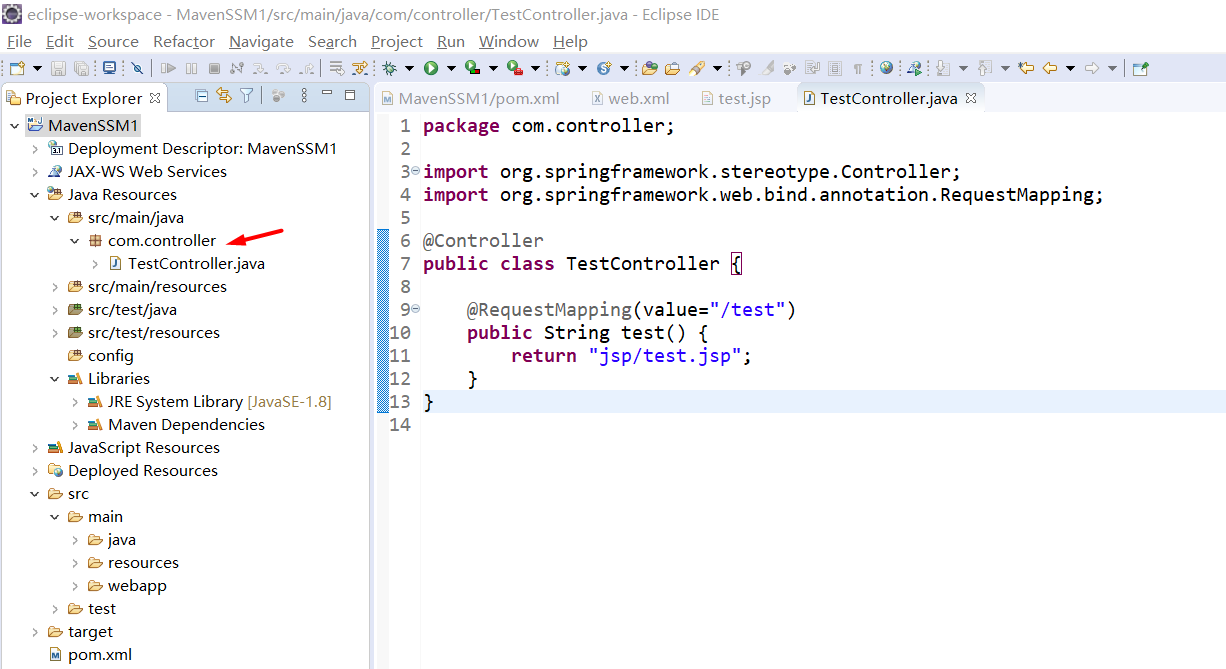
1. 在pom.xml中放行配置文件。如下：



1. 在WEB-INF目录下创建jsp目录，并在jsp目录中创建一个jsp文件。如下：



1. 在src/main/java根目录下创建包，编写控制器，访问该jsp文件。如下：



（注意：在前面已经配置文件中已配置了视图解析器和扫描器。）

1. 运行服务器，访问jsp文件。如下：

