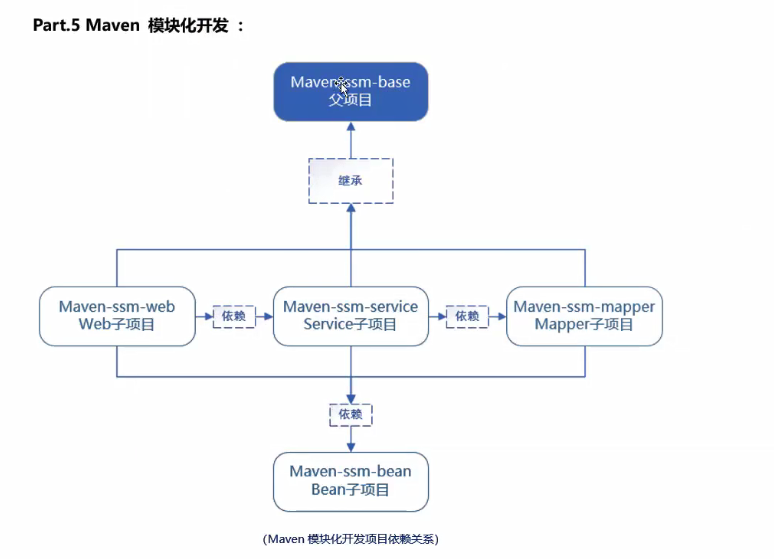
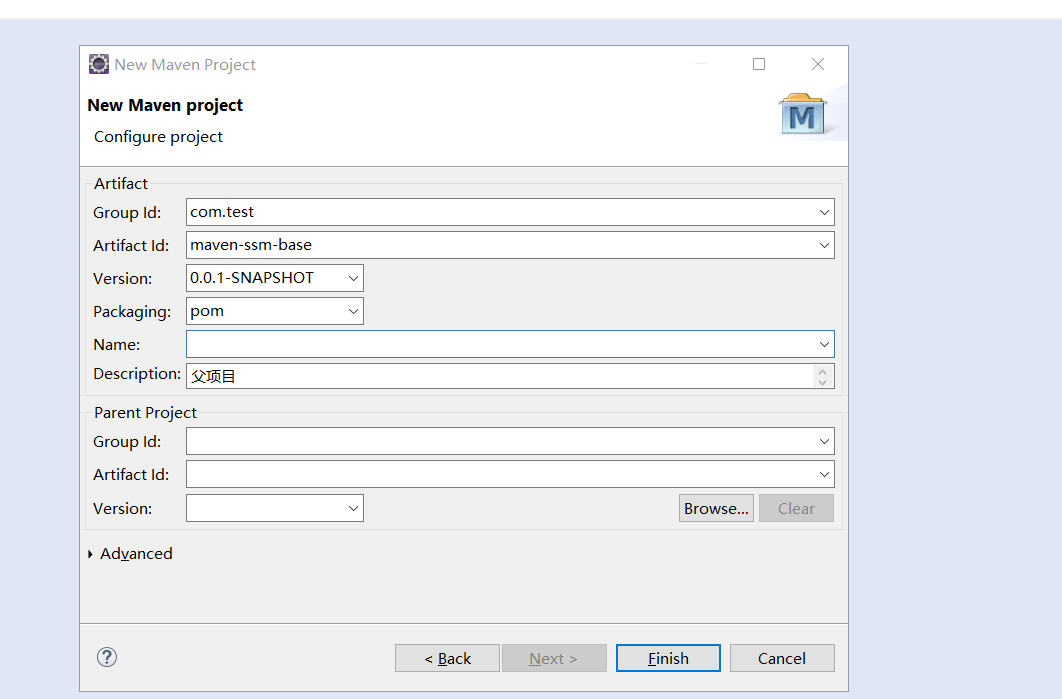
# 创建父子项目

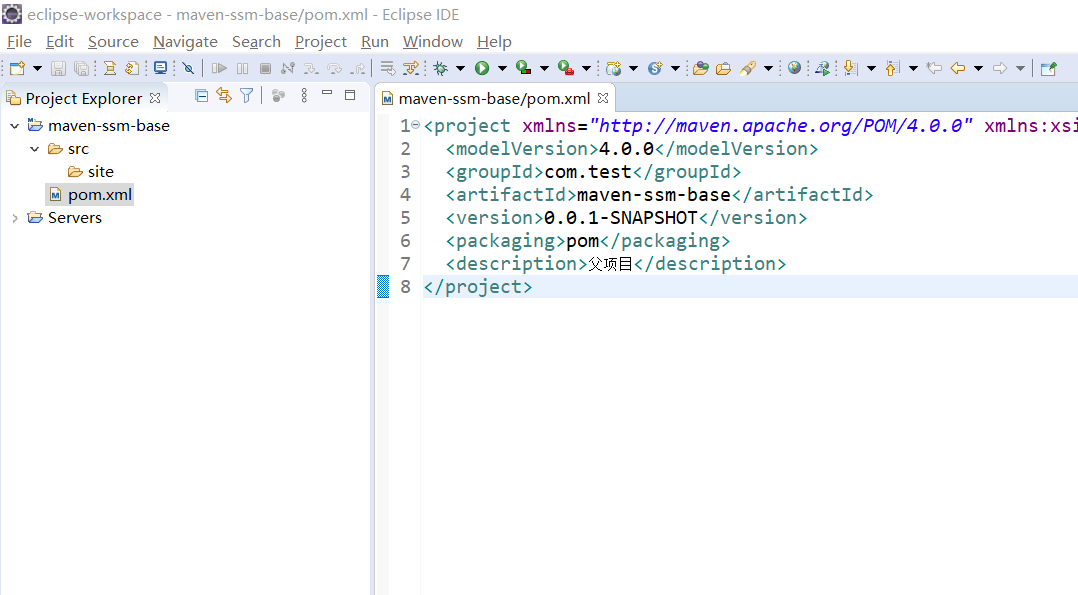
使用Maven能够将一个项目分为若干个子项目或模块，方便协作，提高开发效率。对于一个项目，如果分模块划分，我们一般将其分为4个模块。即：bean、controller、service、dao。它们分别表示的是Bean模块，控制层、业务层、持久层。划分模块后，我们需要明白一个问题，也就是模块间的依赖问题。这4个模块都依赖一个父项目，mapper、service、controller都依赖bean，controller依赖service、service依赖mapper。



## 创建父项目

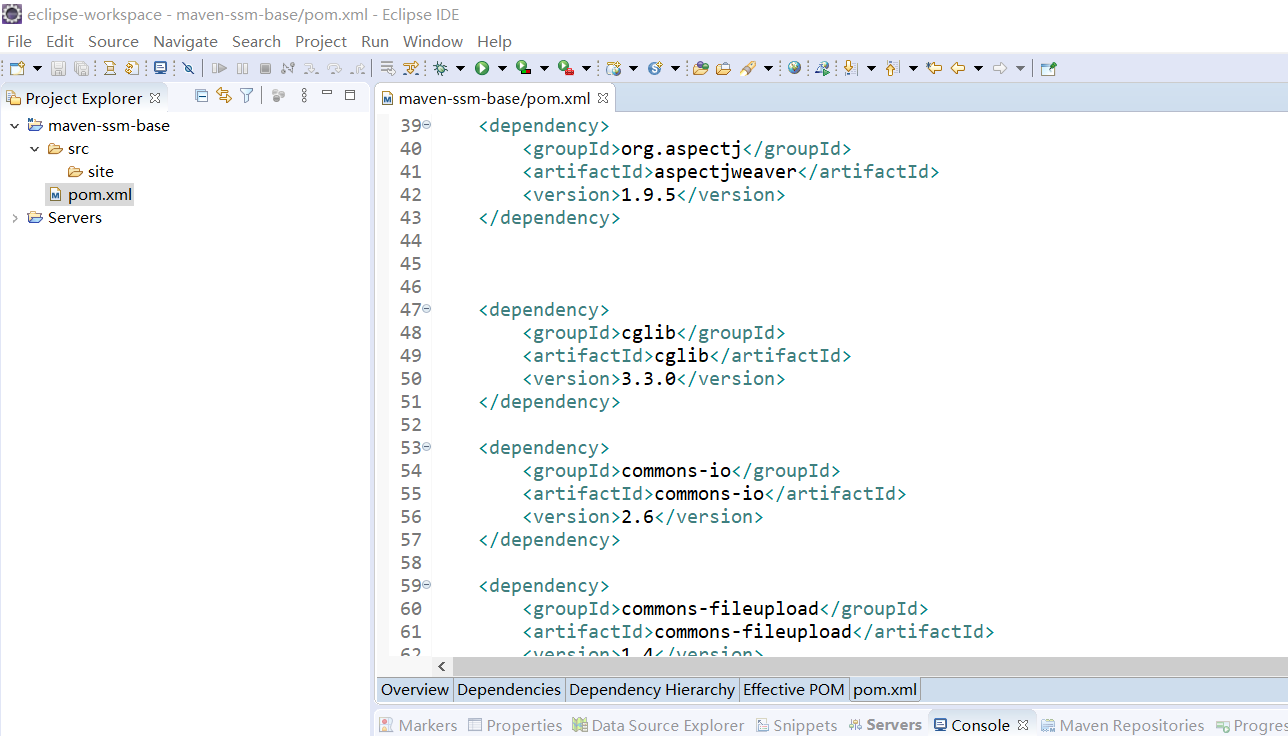
1）创建父项目，简单类型的Maven项目，打包方式为pom。如下：





2）在父子项目开发中，父项目中pom.xml文件能够被子项目继承；同时子项又可以配

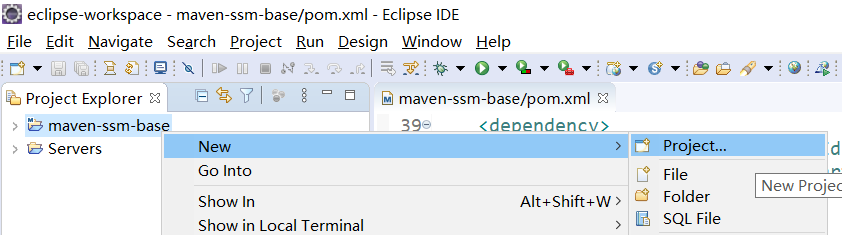
置自己的pom.xml文件。父项目建好后，一般会修改pom.xml文件（如添加依赖等 ）让子项目继承、在父项目中管理所有的jar包等。



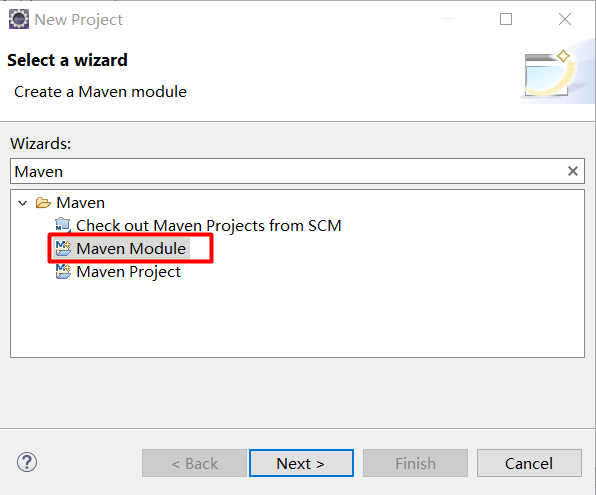
之后将父项目打包到本地仓库或远程仓库，供其他模块或项目依赖使用。

## 创建mapper模块

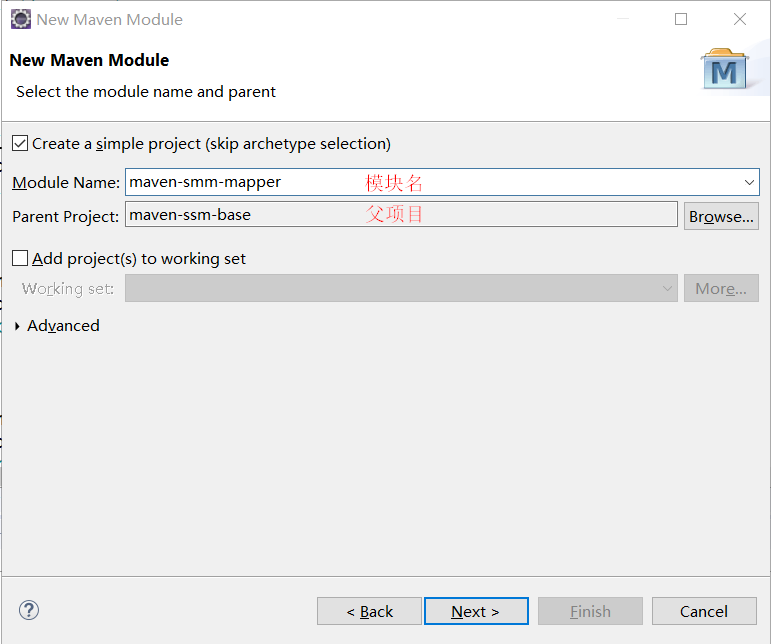
1）在父项目上创建子项目，为mapper模块。如下：



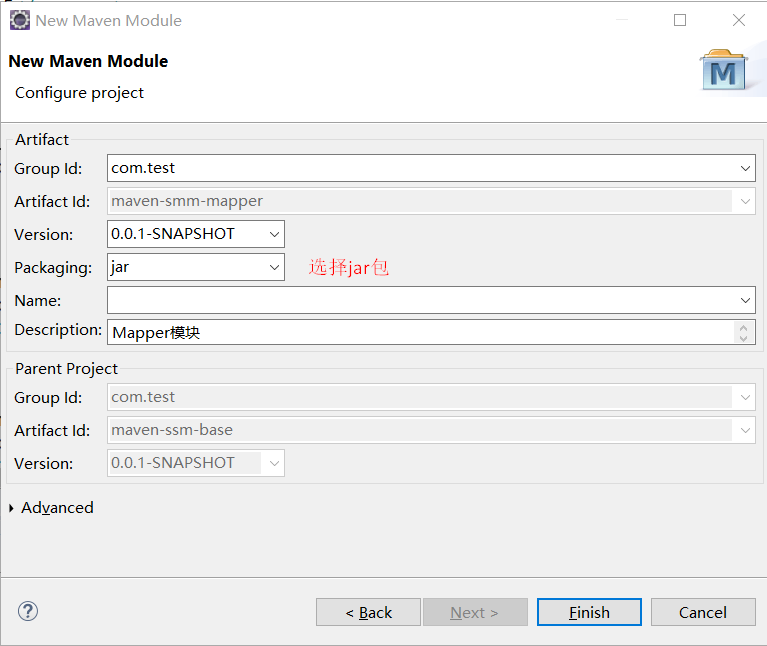
选择【Maven Module】，点击【Next】



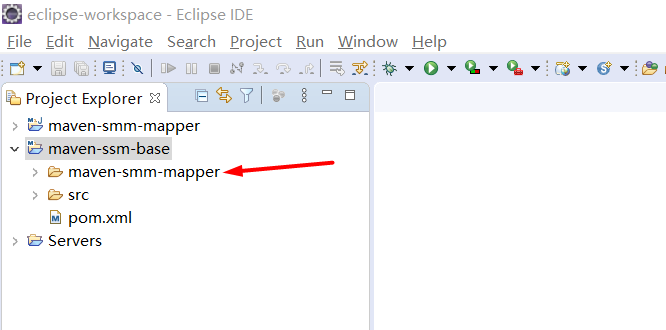
继续点击【Next】



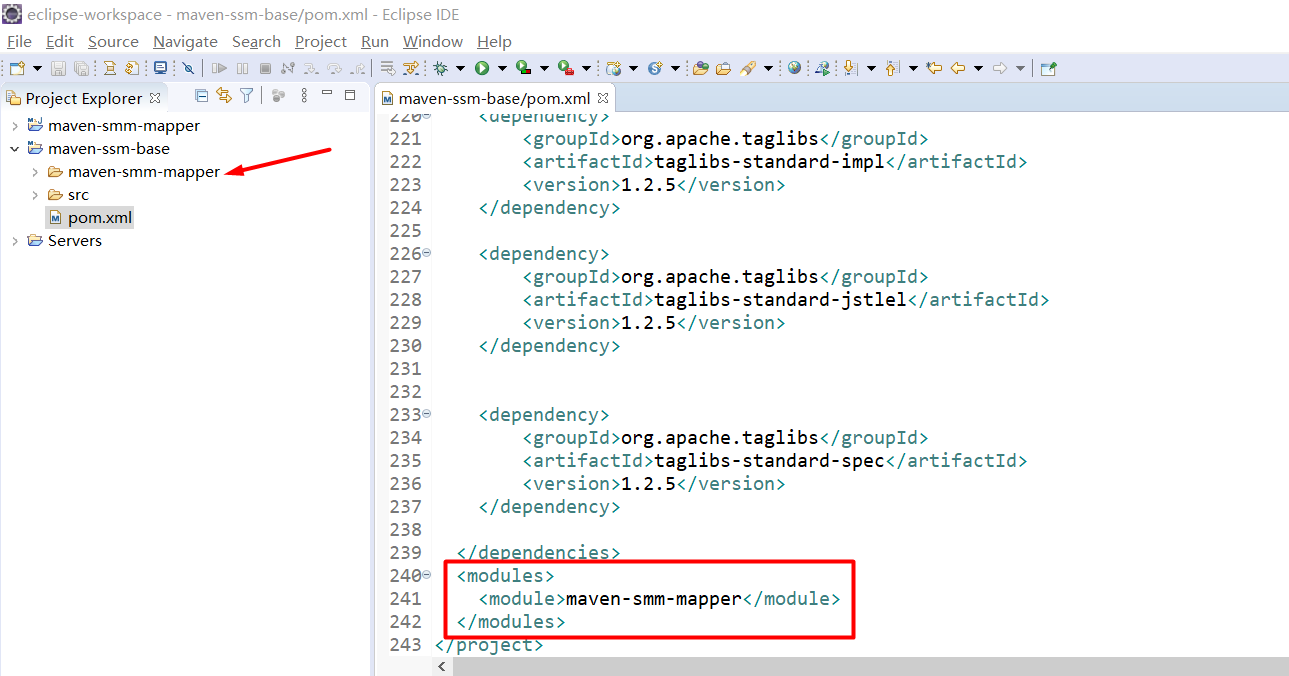
点击【Finish】



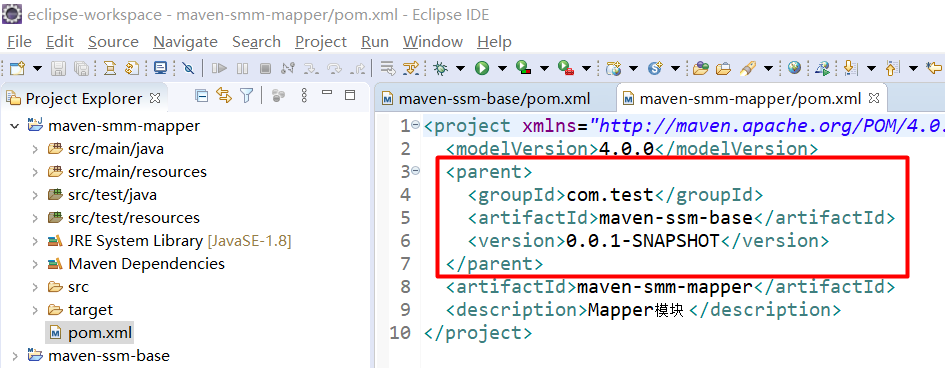
子项目创建完成后，即可在父项目中看到子项目。如下：



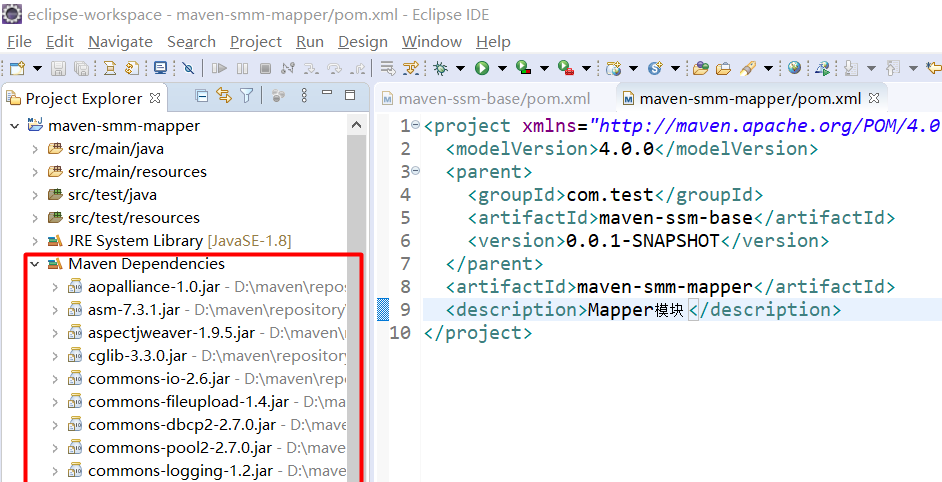
并且，创建子模块后，父项目的pom.xml文件中会标记子模块。如下：



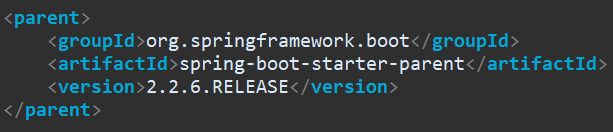
同时，在子项目pom.xml文件中也会出现父项目信息。如下：



虽然子项目没有添加依赖，但是它继承了父项目的pom.xml文件。由于父项目pom.xml文件中配置了依赖，所以子项目会引入父项目的依赖。如下：

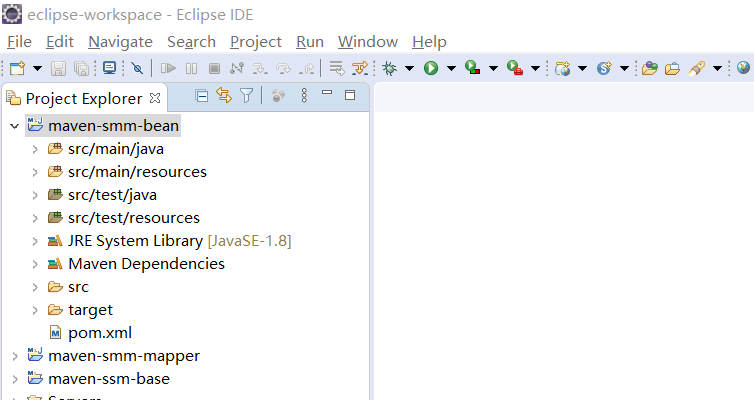


因为是学习，所以父工程和子模块都建立在一台计算机上，但实际上，并不在一台计算机上，李四创建父工程后，会将其打包到远程仓库，然后张三在子模块中通过pom.xml文件引入父模块。比如：

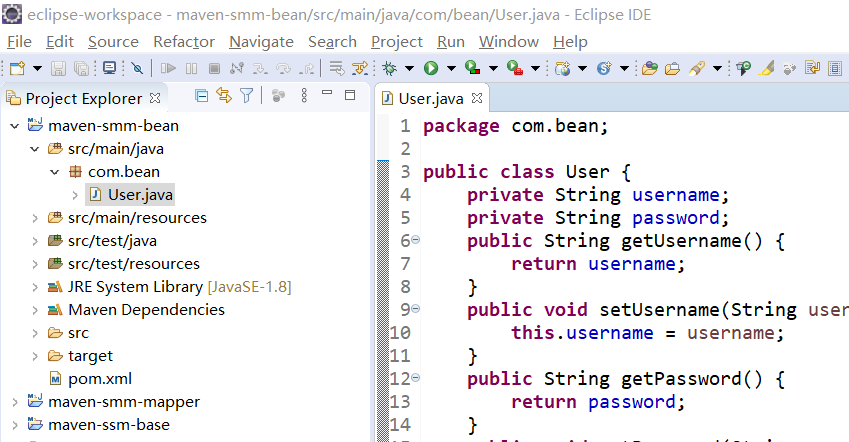


## 创建bean模块

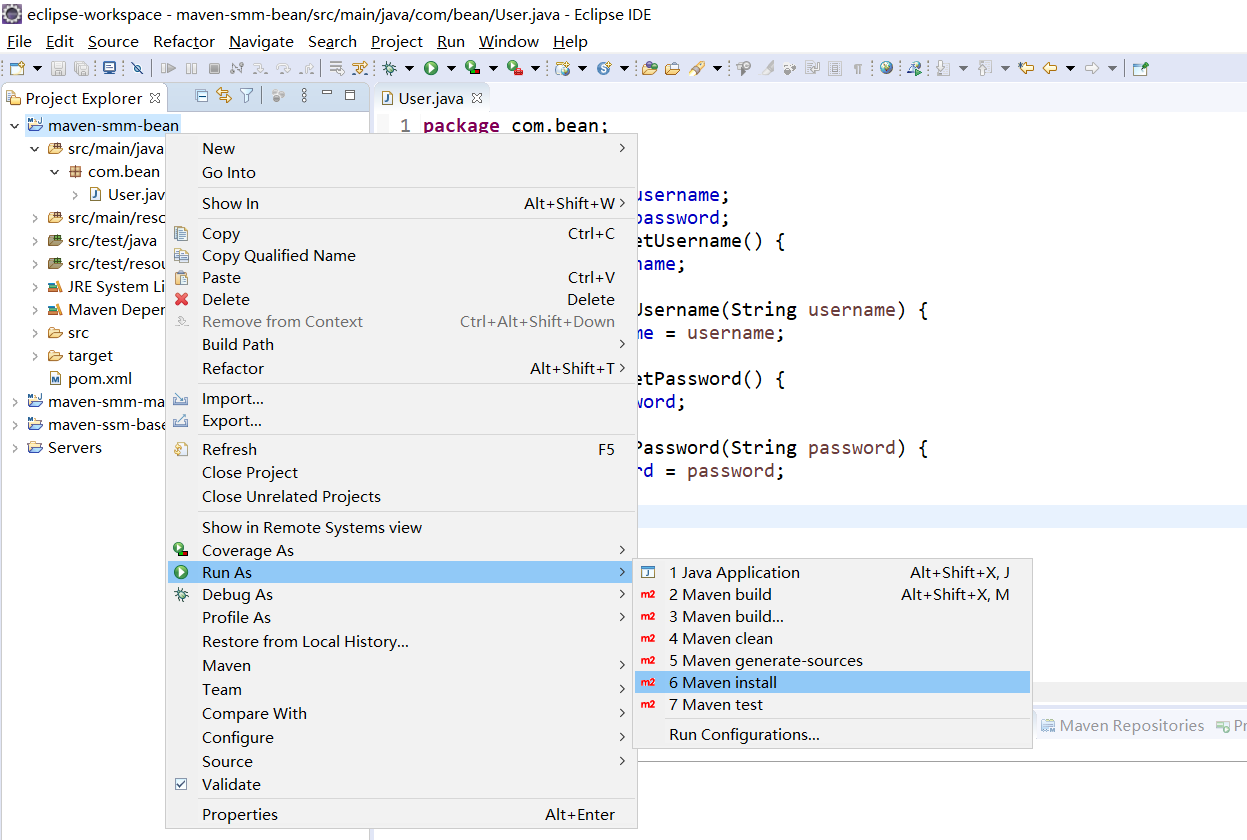
5）创建bean子项目。如下



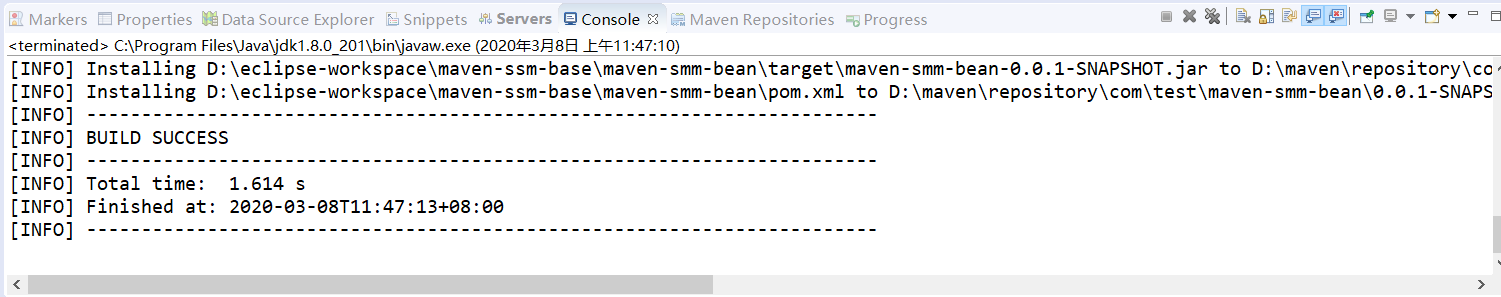
在bean模块中创建需要的bean。如下：



将bean模块打成jar包放入本地仓库中。如下：

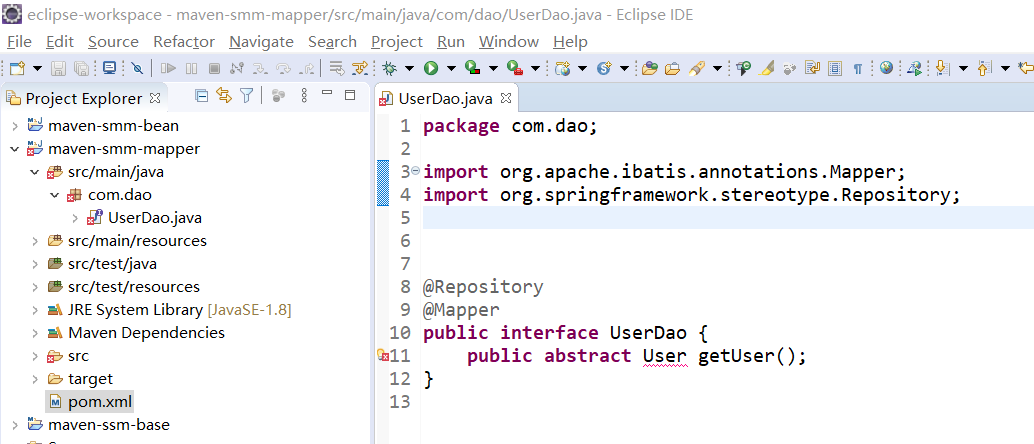


打包成功。



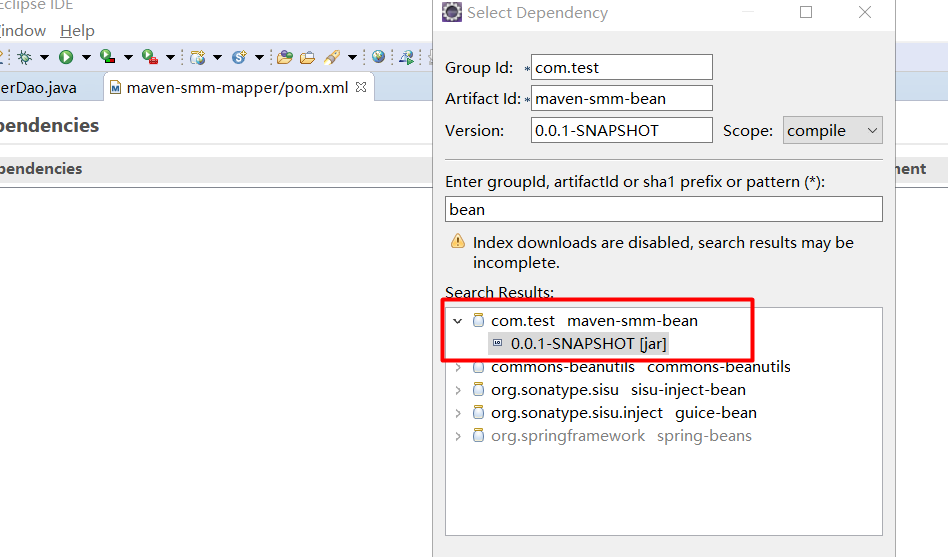
## mapper引入bean

在maven-ssm-mapper模块中创建映射接口。如下：



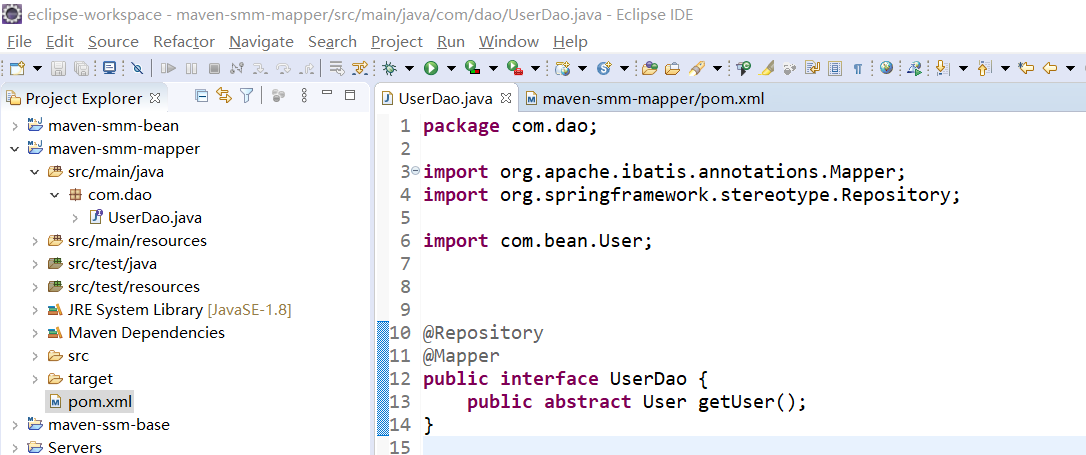
但是，由于mapper模块需要依赖bean模块，所以需要引入bean模块。

bean模块已经被打成jar包存入本地仓库了，所以可以在mapper模块的pom.xml文件中添加bean模块的依赖。如下：

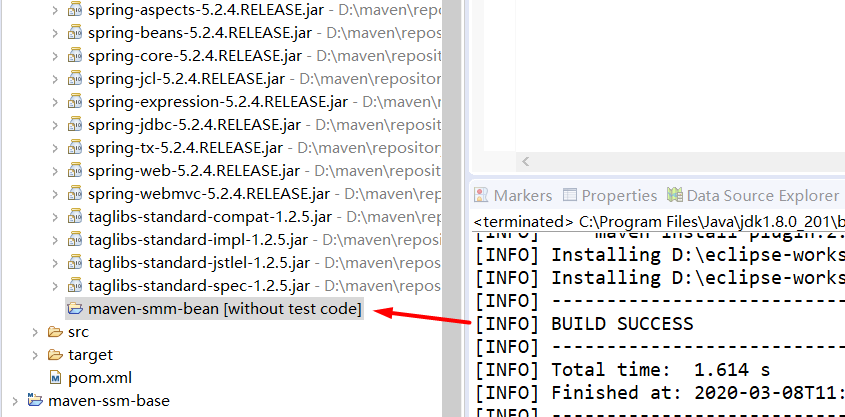




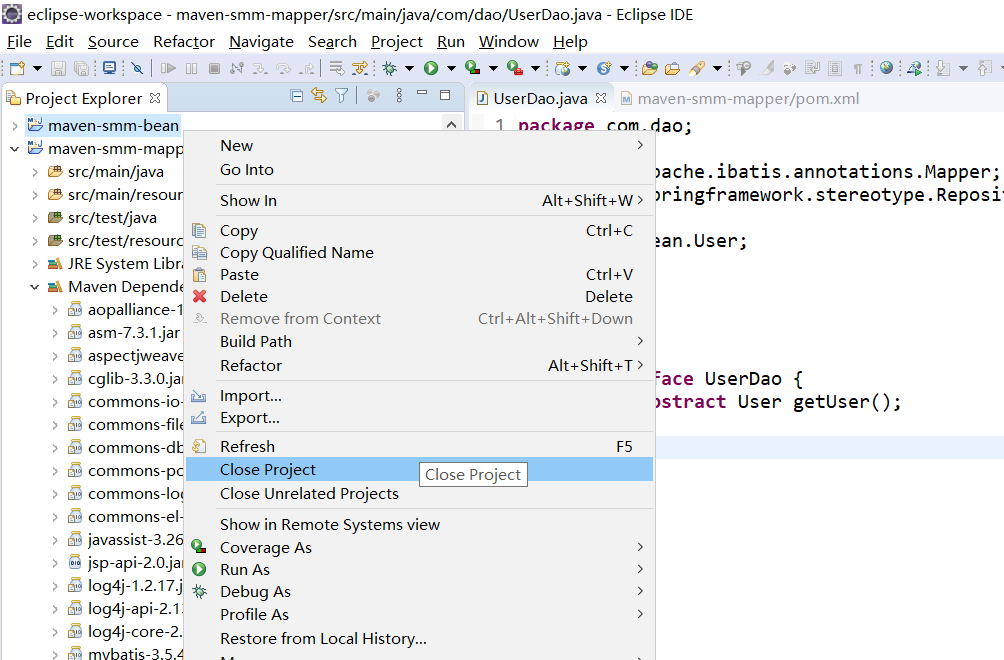
在映射器接口中引入。如下：



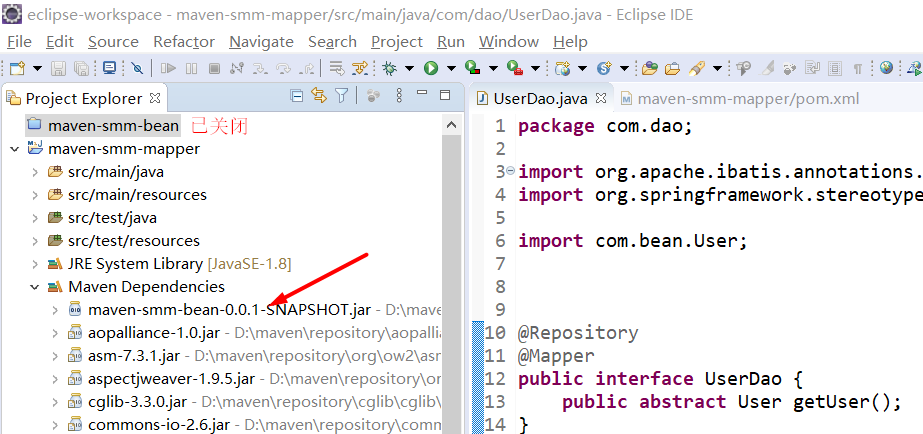
在maven-ssm-mapper模块依赖中，会出现bean模块的依赖，如下：



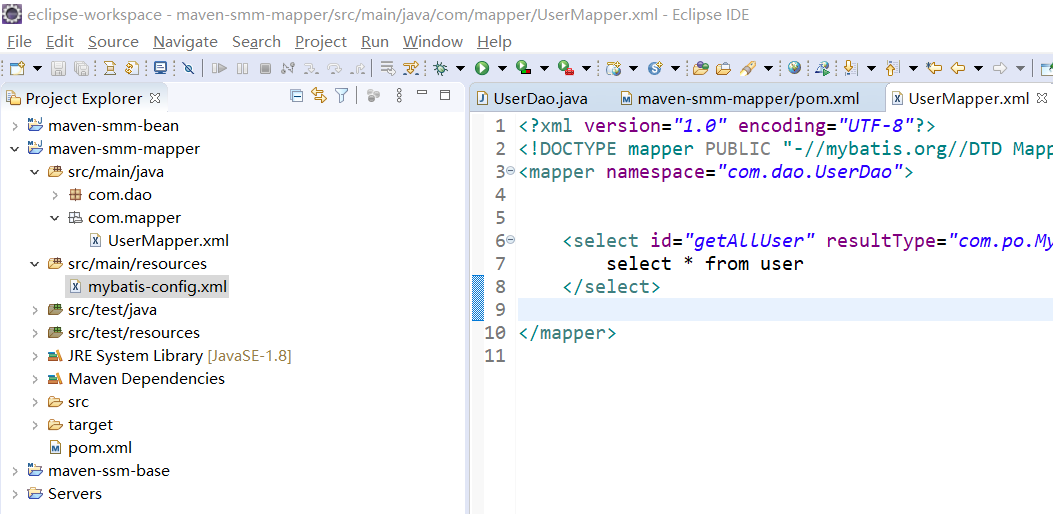
但并不是一个jar包，而是以工程的形式存在。这是因为我们将这些模块放入了同一个Eclipse工程中，在实际开发中，模块通常会由不同的人开发。所以不会以这样的形式存在，而是以jar包的形式存在。你可以将bean工程关闭，如下；

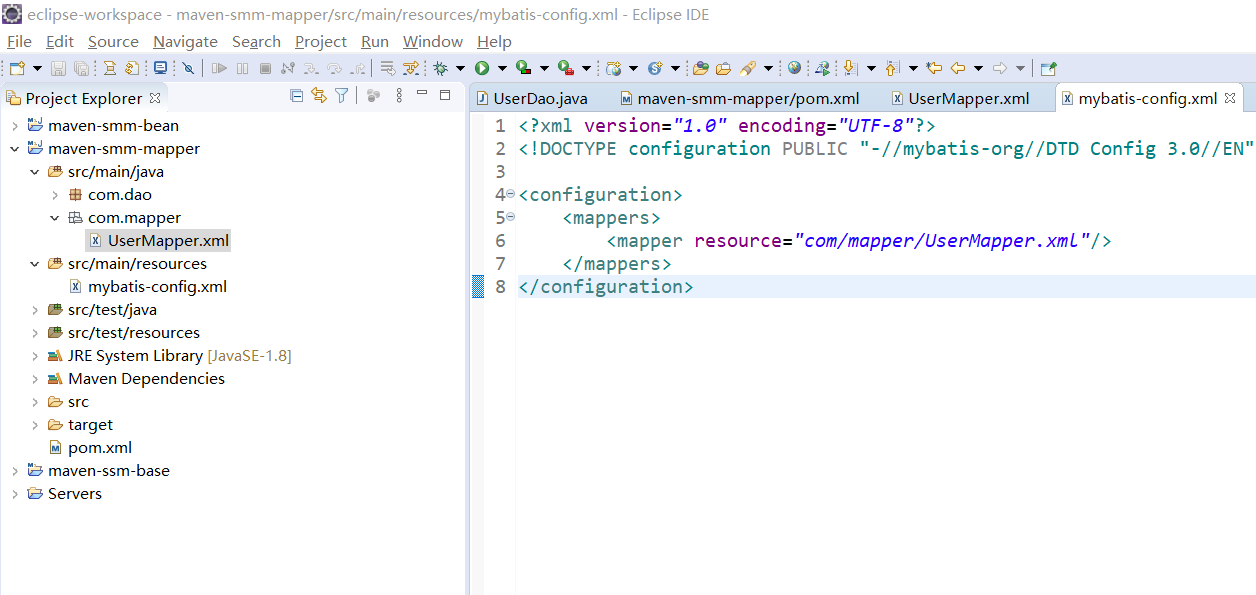


随后，它便以jar包的形式存在，如下：



接着，你可以继续在mapper模块中创建mapper映射器和mybatis-config.xml配置文件。如下：

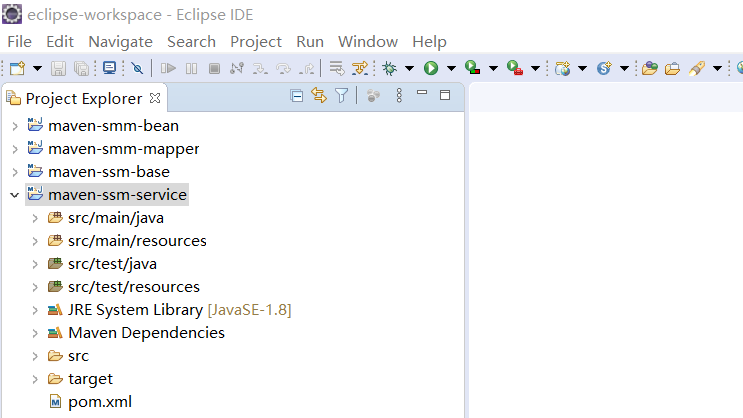




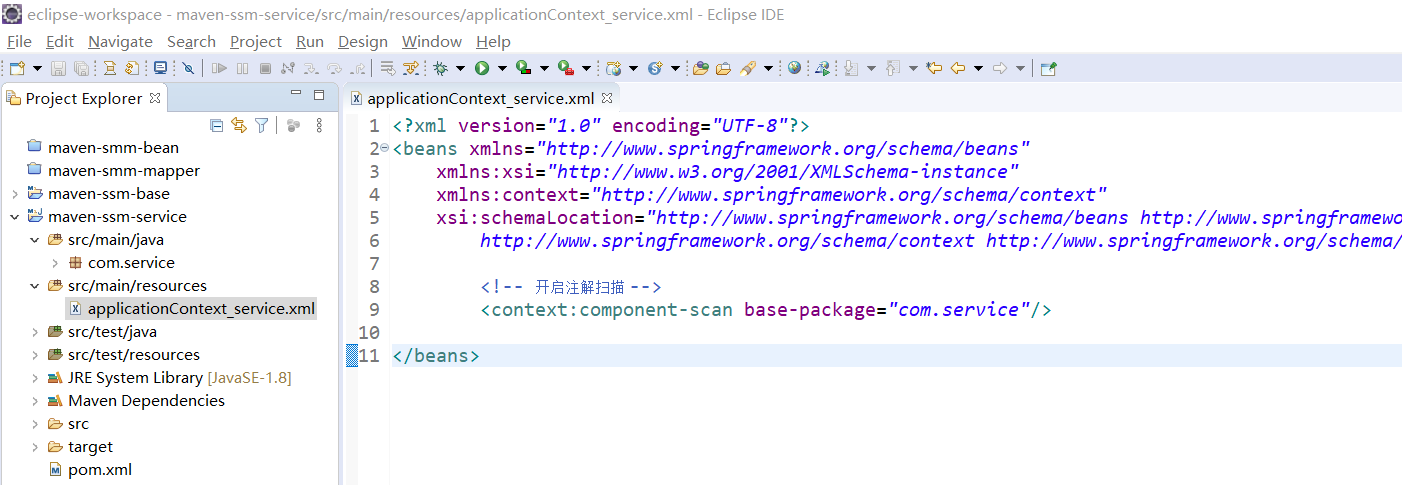
随后，你可以继续进行持久层的开发了。

## 创建service模块

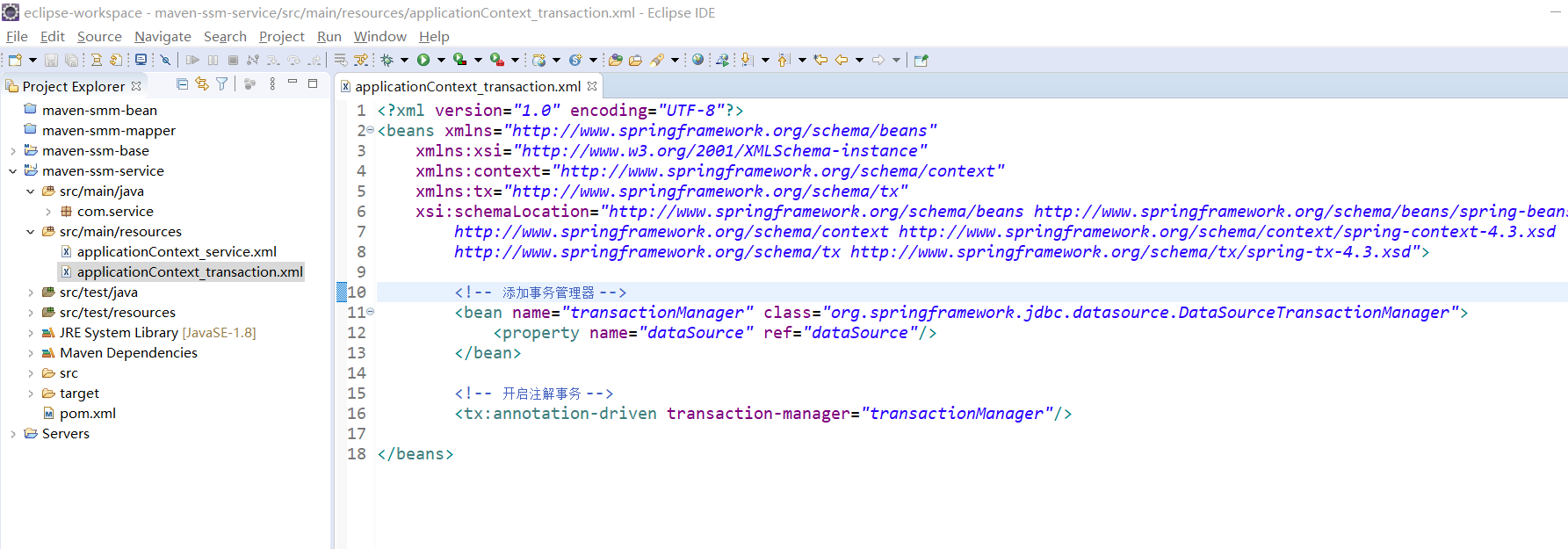
1. 在父项目的基础上创建Service模块。如下：



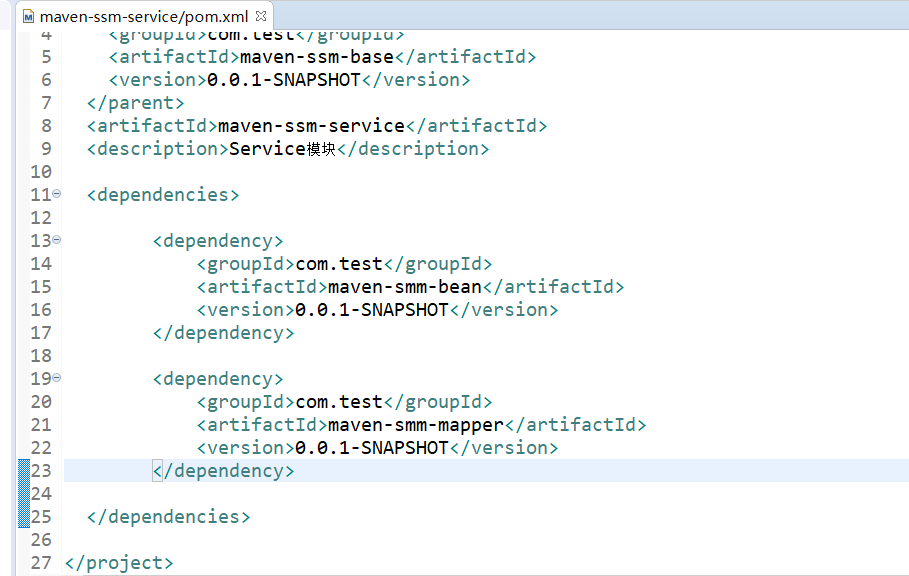
1. 在src/main/resources添加Service层所需要的配置文件applicationContext\_service.xml文件。在文件中扫描包。如下：



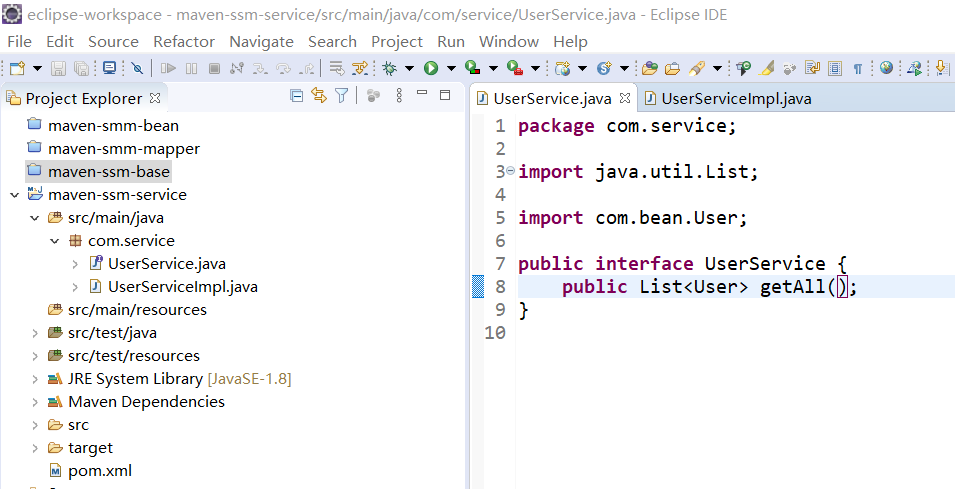
1. 由于该模块是业务层模块，需要使用到事务。在src/mian/resources下创建applicationContext\_transaction.xml文件。如下：

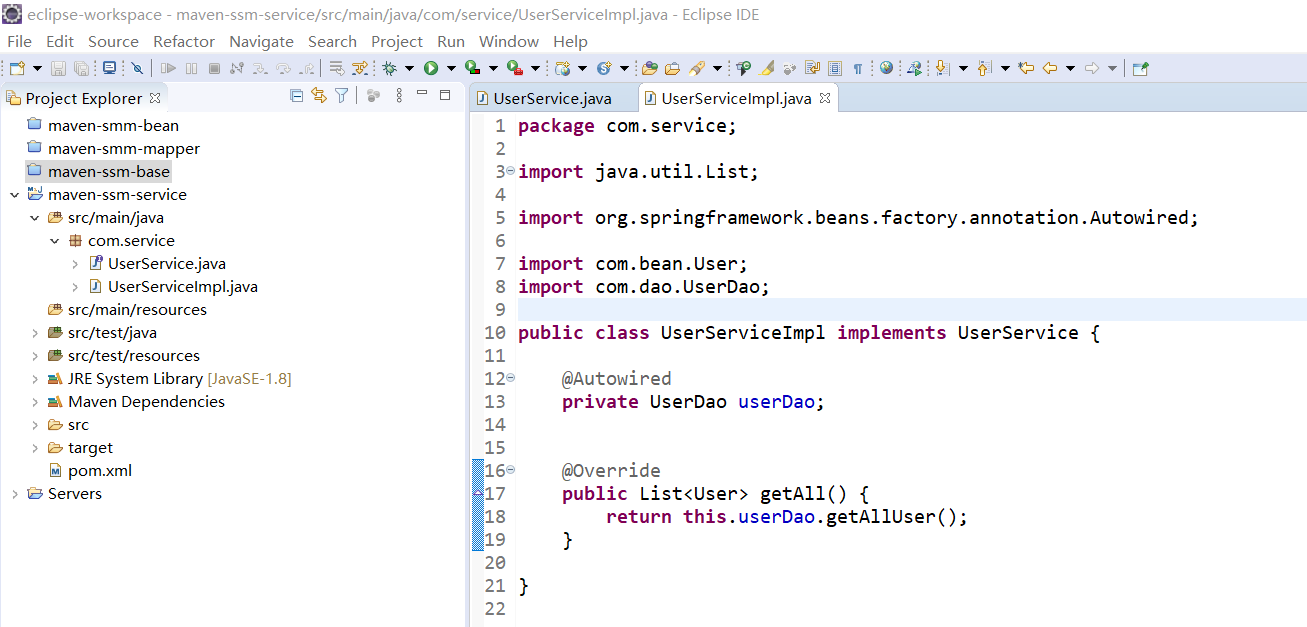


1. 因为service模块依赖bean模块和mapper模块。所以需要通过依赖引入Mapper模块打成的jar包和bean模块打成的jar包。



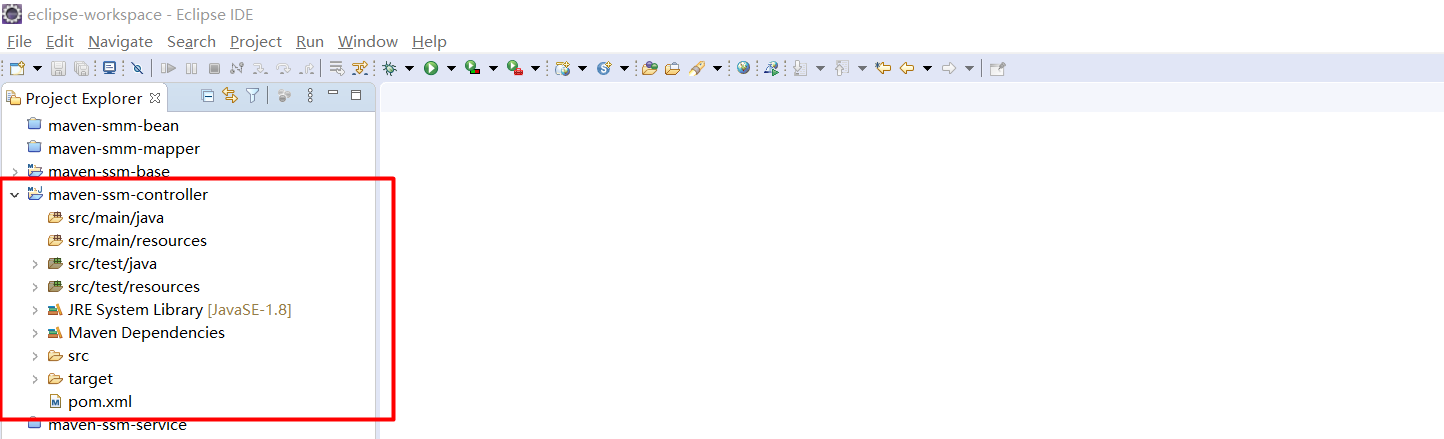
1. 编写业务层代码。如下：



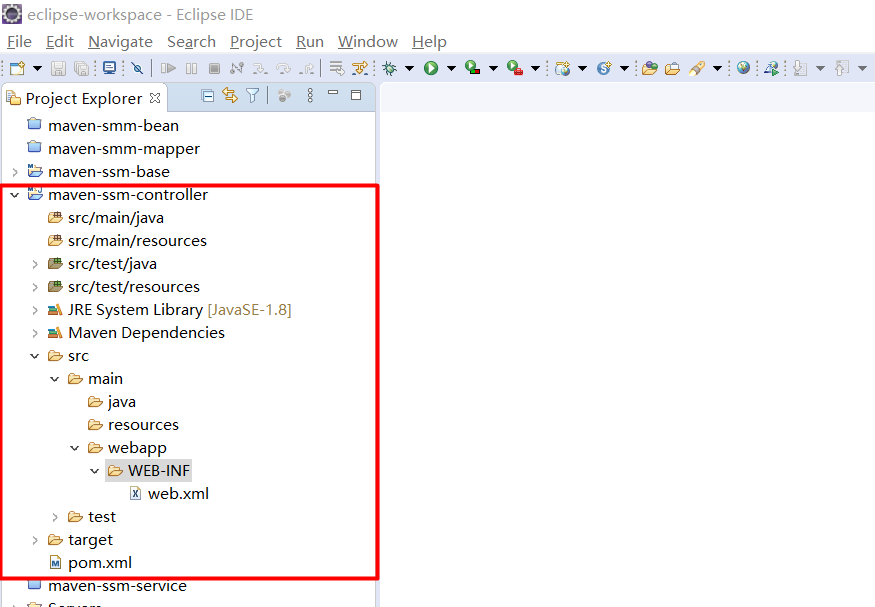


## 创建Controller模块

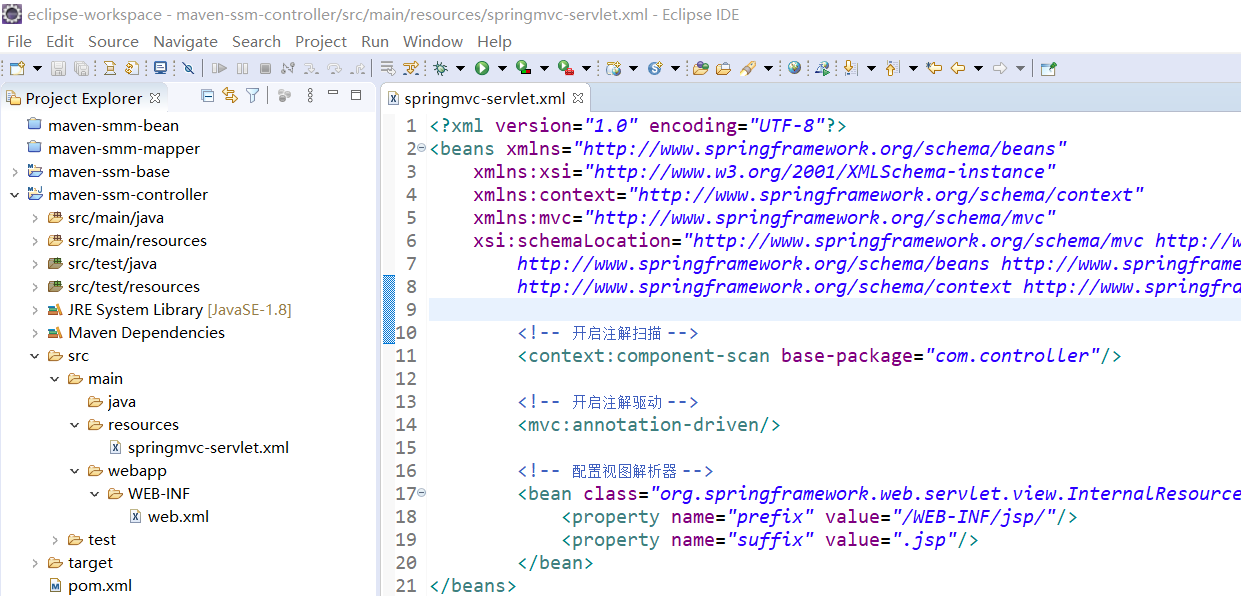
1. 在父项目的基础上创建controller模块。如下：



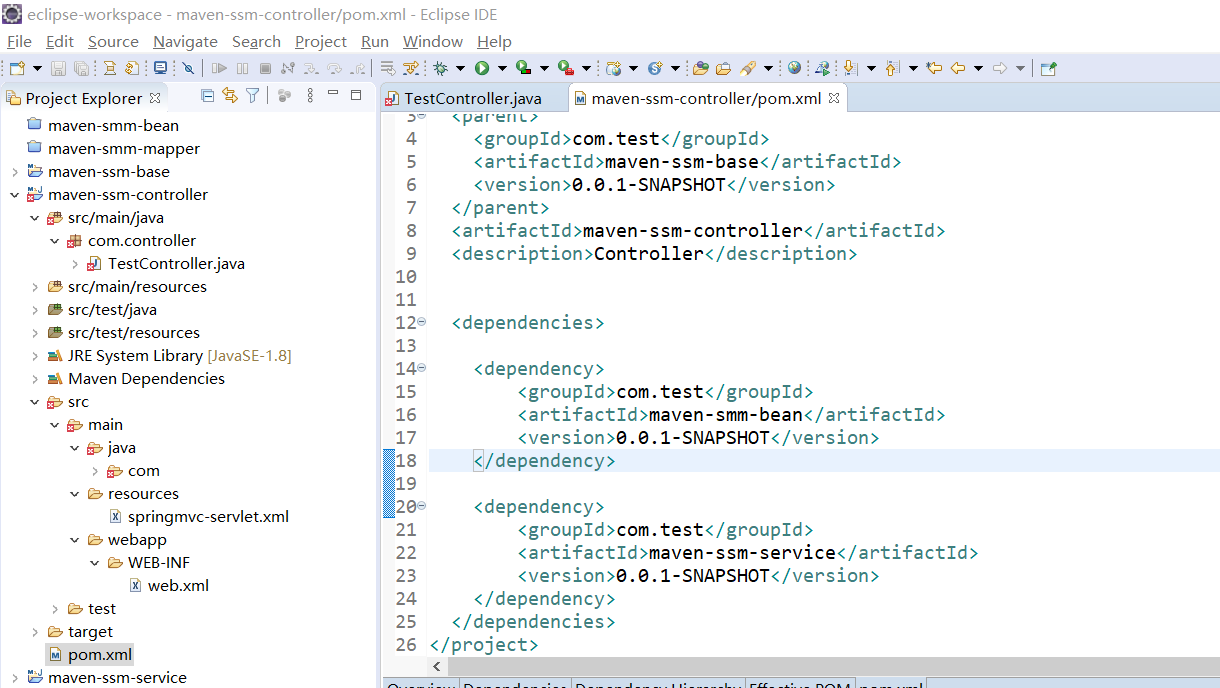
1. 在开发中一般是将webapp目录也放入controller模块中，所以需要在src/man/java/下创建webapp，并对其进行完善。（也可以将controller模块创建为web形式的Maven模块）如下：



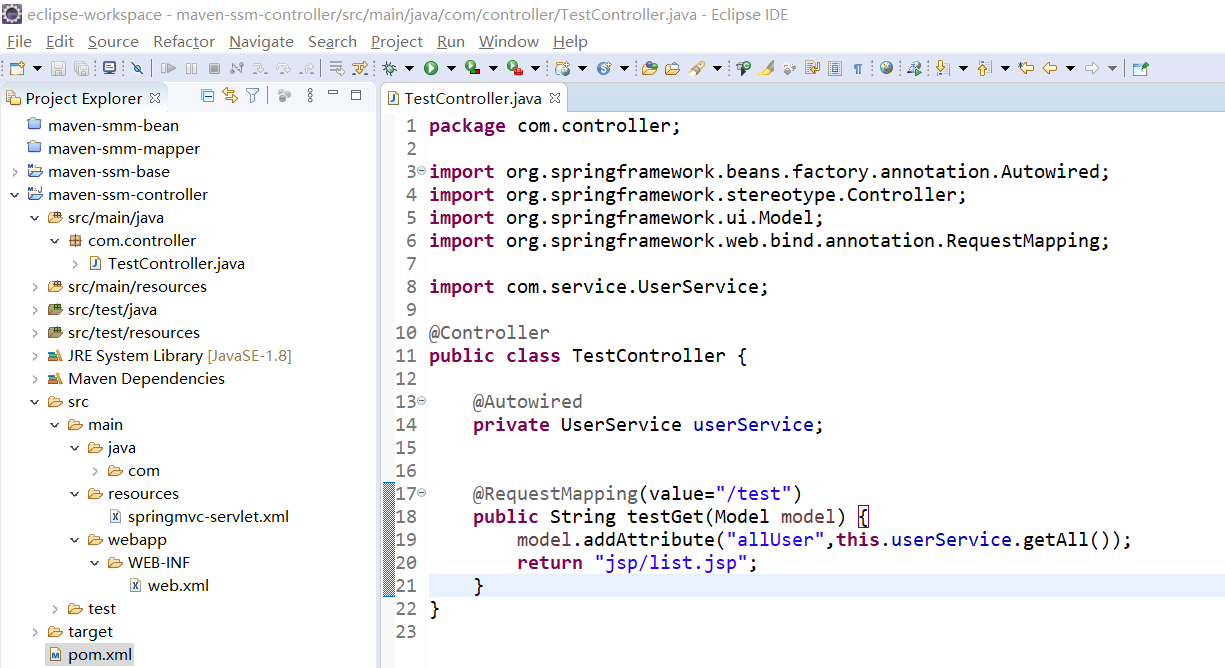
1. 在src/main/resources目录下创建springmvc-servlet.xml文件。并对其进行完善。如下：



1. 因为controller模块依赖bean模块和service模块，所以需要在pom.xml中通过依赖引入bean模块打成的jar包和service模块打成的jar包。如下：

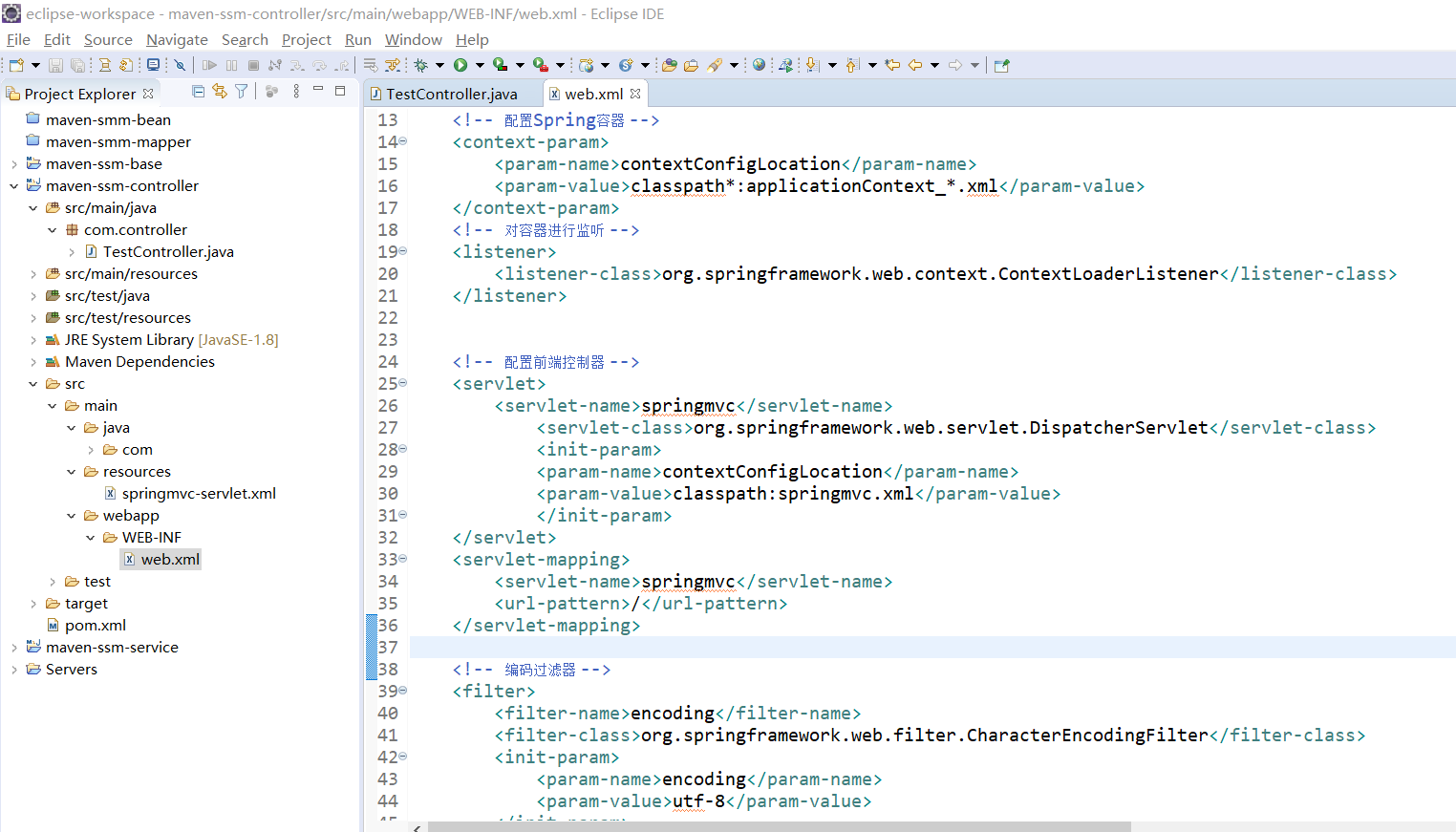


1. 编写控制层代码。如下：



别忘了在springmvc-servlet.xml文件中对其扫描。

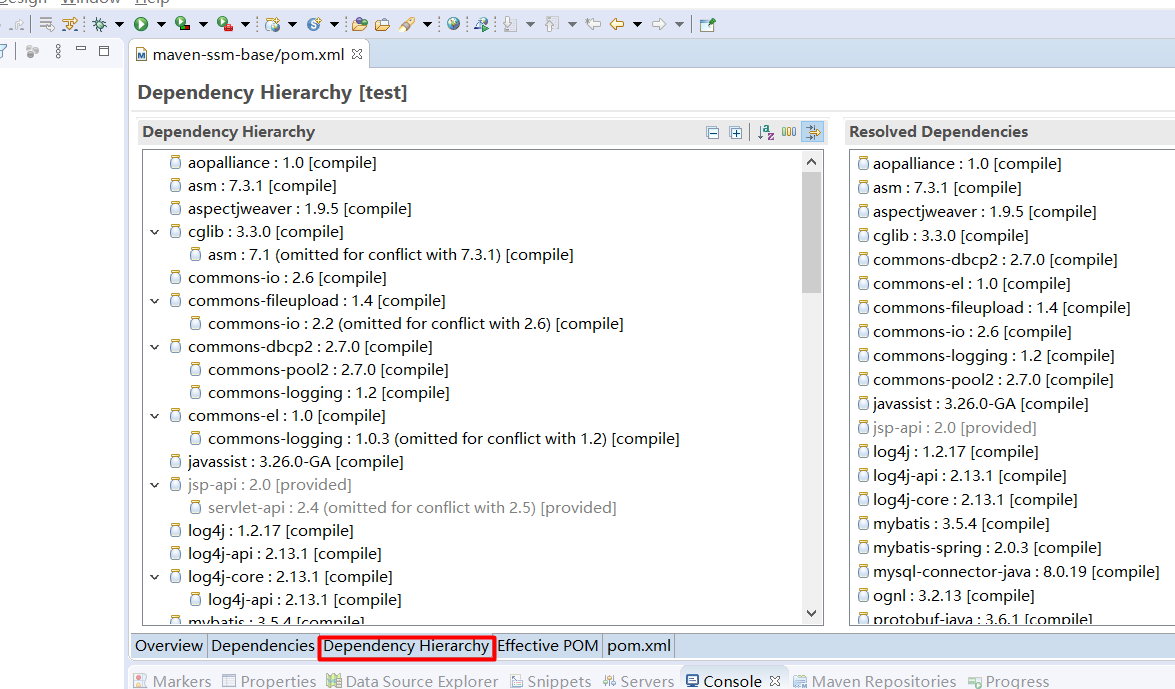
1. 完善web.xml文件。如下：



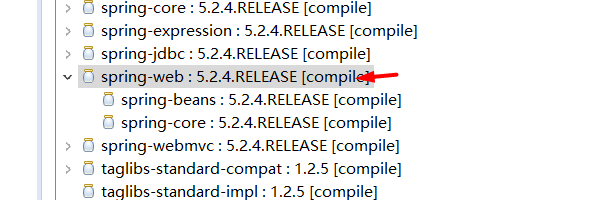
# 依赖传递与依赖冲突

## 依赖传递

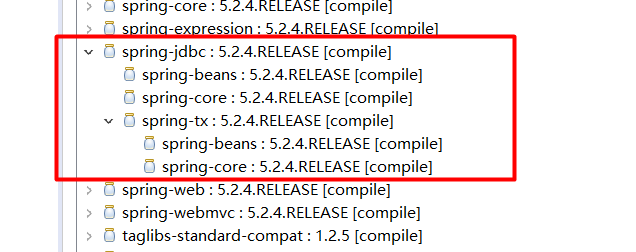
在Eclipse中以Dependency Hierarchy的方式打开pom.xml文件。如下：



在左边它会以jar包的形式列出该pom.xml配置的所有依赖，但是有的jar包上有个小箭头，而有的却没有；展开小箭头，小箭头下为该jar包要依赖的jar包。以spring-web为例，它依赖spring-bean以及spring-core包。



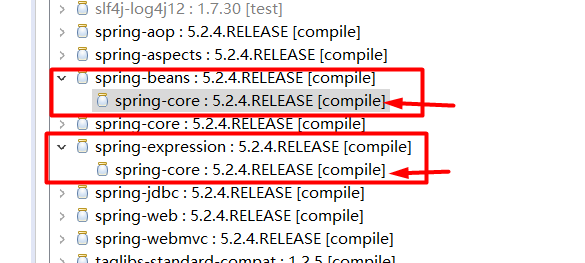
有的依赖关系不仅是单层，还有多层的。例如：



在前面的学习中，我们在导入一个依赖时，如果该jar包还依赖其他jar包时，则也会将其导入进来。这就是依赖传递。

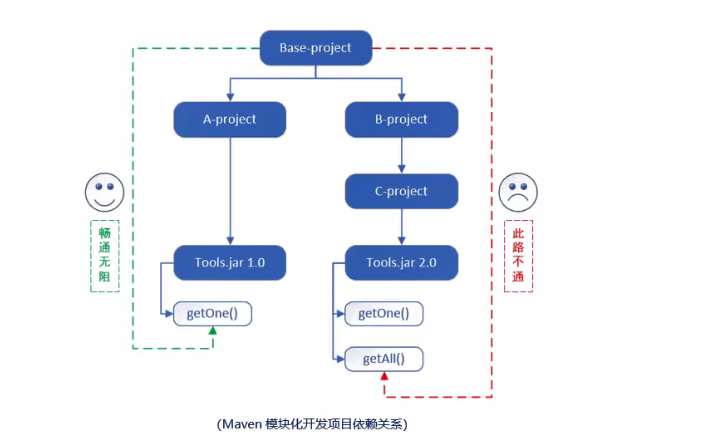
## 依赖冲突

但依赖传递存在一个问题，如下：



可以看出spring-beans依赖spring-core，而spring-expression也依赖springcore。在这里我们用的是一个框架，所以暂时还没有出现冲突问题。

但如果此时存在一个base-project项目，由三个子项目组成，其中A-project项目依赖tools.jar 1.0版本，C-project项目依赖tools.jar 2.0版本，根据依赖传递规则，B-project将同时依赖tools.jar的两个版本，但这就造成了依赖冲突。根据就近原则，该项目最终依赖tools.jar 1.0版本，导致C-project项目无法调用getAll( )方法从而报错，因此我们要手动选择一个合适项目的版本。



## 解决依赖冲突

那么如何解决依赖冲突呢？解决依赖冲突有3种方法：

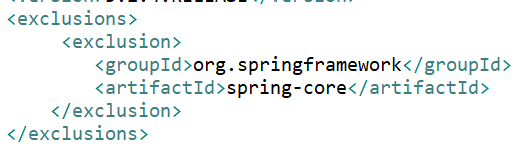
1. 直接添加依赖。

例如，父项目base-project中配置了spring-core 2.0版本，那么在子项目a-project中不再使用父项目的spring-core 2.0版本，而是在子项目自己的pom.xml中配置另个一个版本的依赖，如spring-core 3.0。

从中，我们也可以发现，子项目可以添加另一个版本的依赖覆盖掉父项目已 有的依赖。

1. 使用排除。

如果引入的某个jar包依赖其他jar包，此时可以使用排除，表示不引入其依赖的jar包。

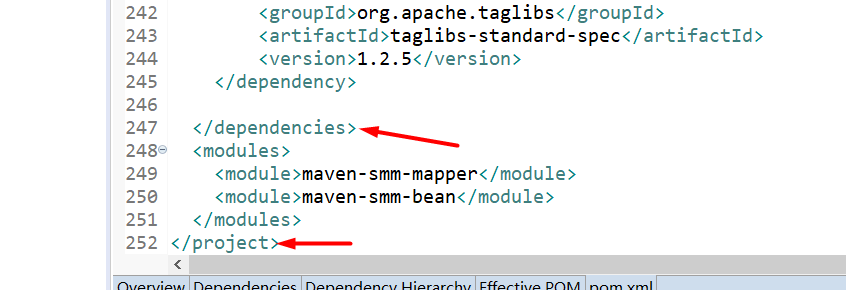


1. 版本锁定。

版本锁定指的是在父工程的pom.xml中使用<dependencyManagement>来统一管理版本。（详细操作在下节）

## 版本锁定

解决依赖冲突的第三种方式是通过版本锁定，通过版本锁定不仅能够锁定版本，还可以方便统一的管理依赖版本。在没有使用版本锁定前，所有的依赖都是声明在<project>下<dependencies>标签中的，声明在<dependencies>标签的依赖会将jar包引入。如下：

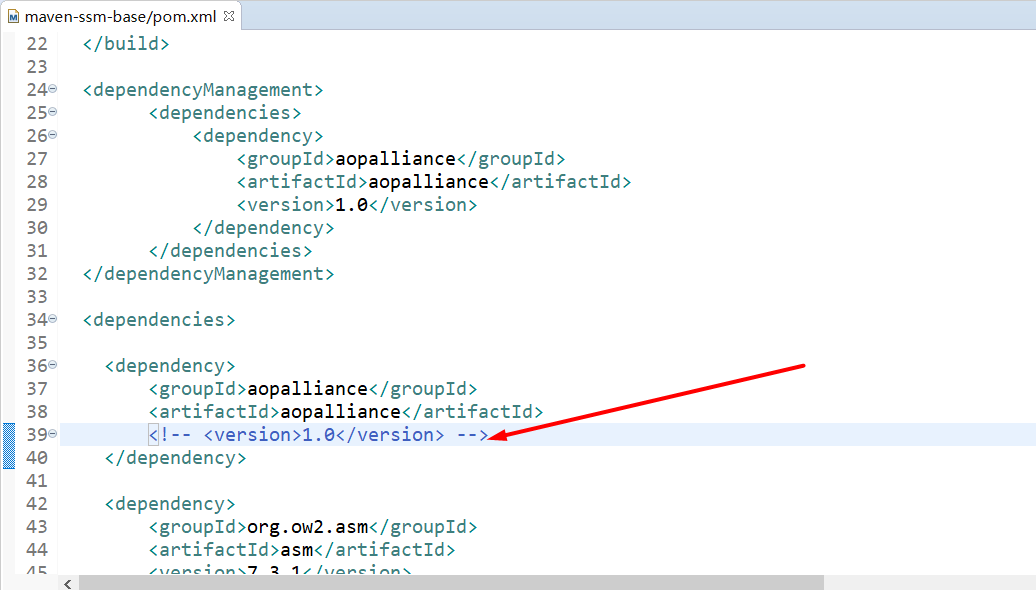


而版本锁定，将所有的依赖声明在<project>下的<dependencyManagement>下的<dependencies>标签中。但它只是声明，并不会引入jar包，真正引入jar包还是需要在<project>下<dependencies>标签中声明引入，它只是起到了一个锁定版本的功能。如下：



由于对其进行了版本锁定，所以在声明依赖引入jar包时如果配置了版本信息则会出现警告，我们可以版本信息删除。

如下：



前面的学习已经得知，父项目pom.xml文件会被子项目继承，所以子项目依赖引入jar时也必须遵循父类中定义的版本锁定规则。并且在实际开发中，我们会将子项目通用的依赖jar包及其版本锁定都配置在父项目中的pom.xml文件中，以方便统一管理与维护。

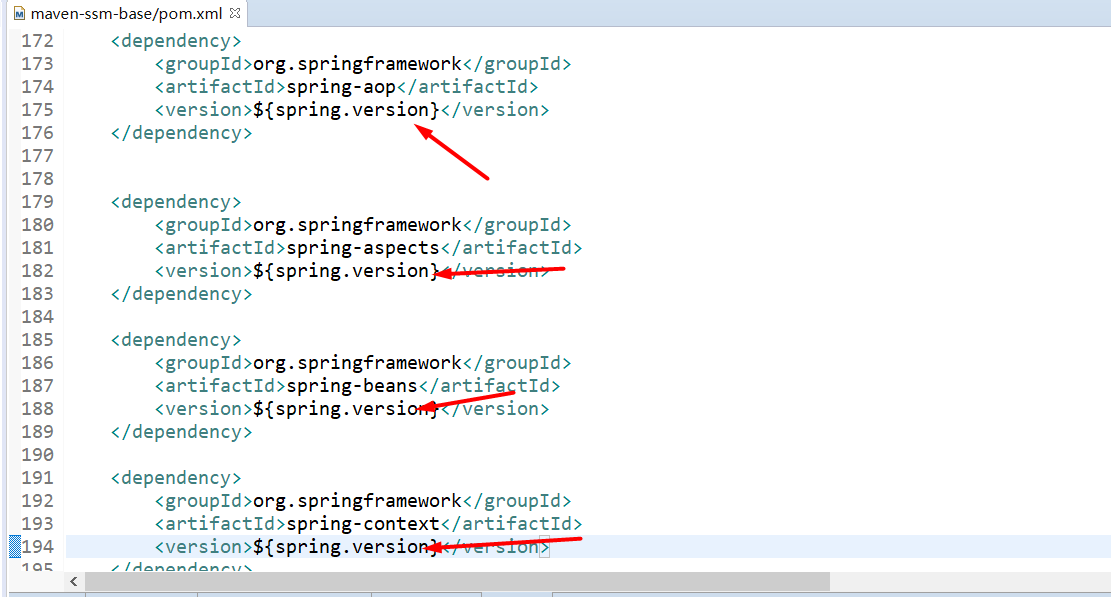
## 属性定义与引入

我们还可以在pom.xml文件中使用<properties>标签来声明属性，并且通过EL进行引用。

如统一规定版本号，在声明依赖时，如果涉及到版本信息的话，直接使用EL引用即可。如下：



再例如，现在Spring有很多版本，我们这时可以在<properties>标签中声明一个标签来规定Spring的版本，让所以属于Spring的依赖通过EL引用定义的版本信息，以此达到统一Spring所有jar包的版本。如下：



如果要修改某个版本，直接修改<properties>标签中定义版本信息即可。这样一来，我们就可以非常方便的管理jar包的版本问题了。