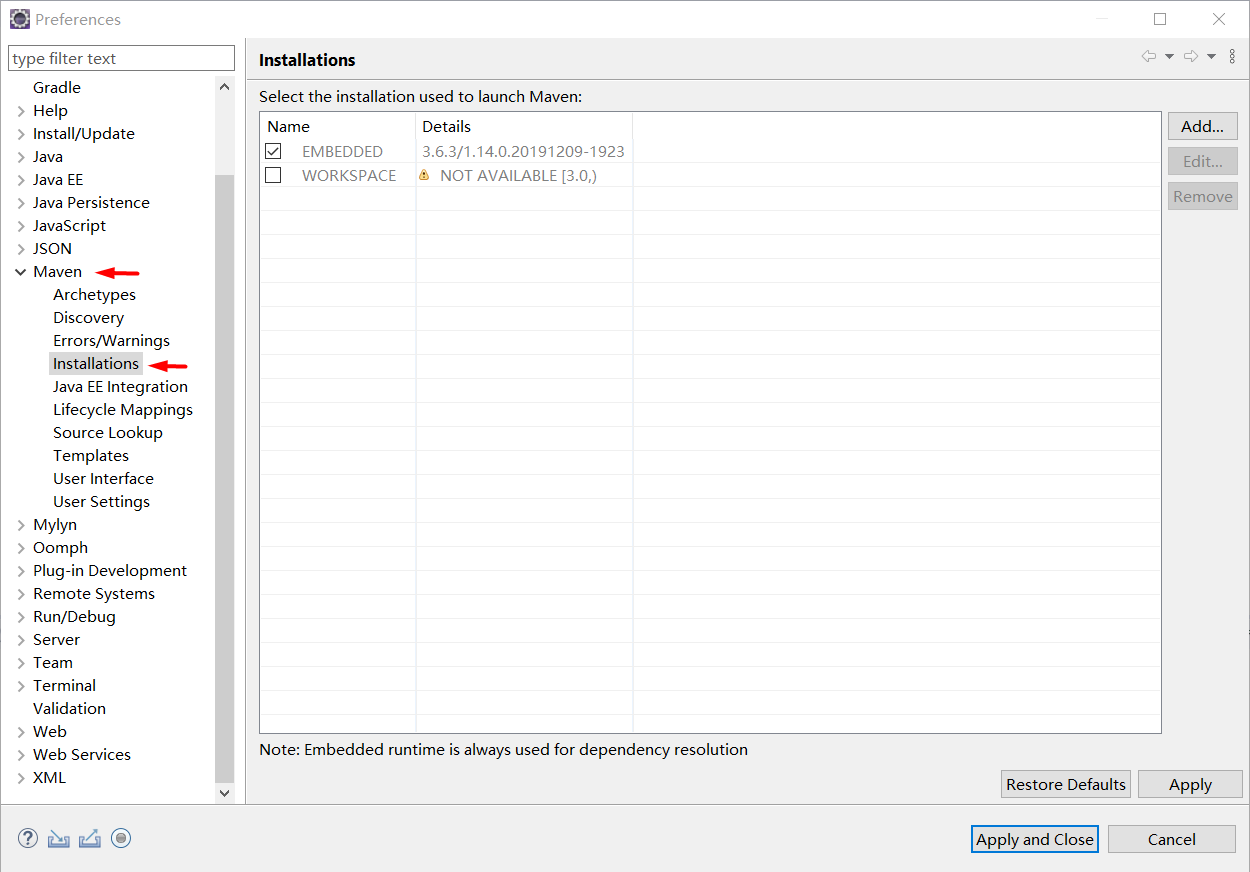
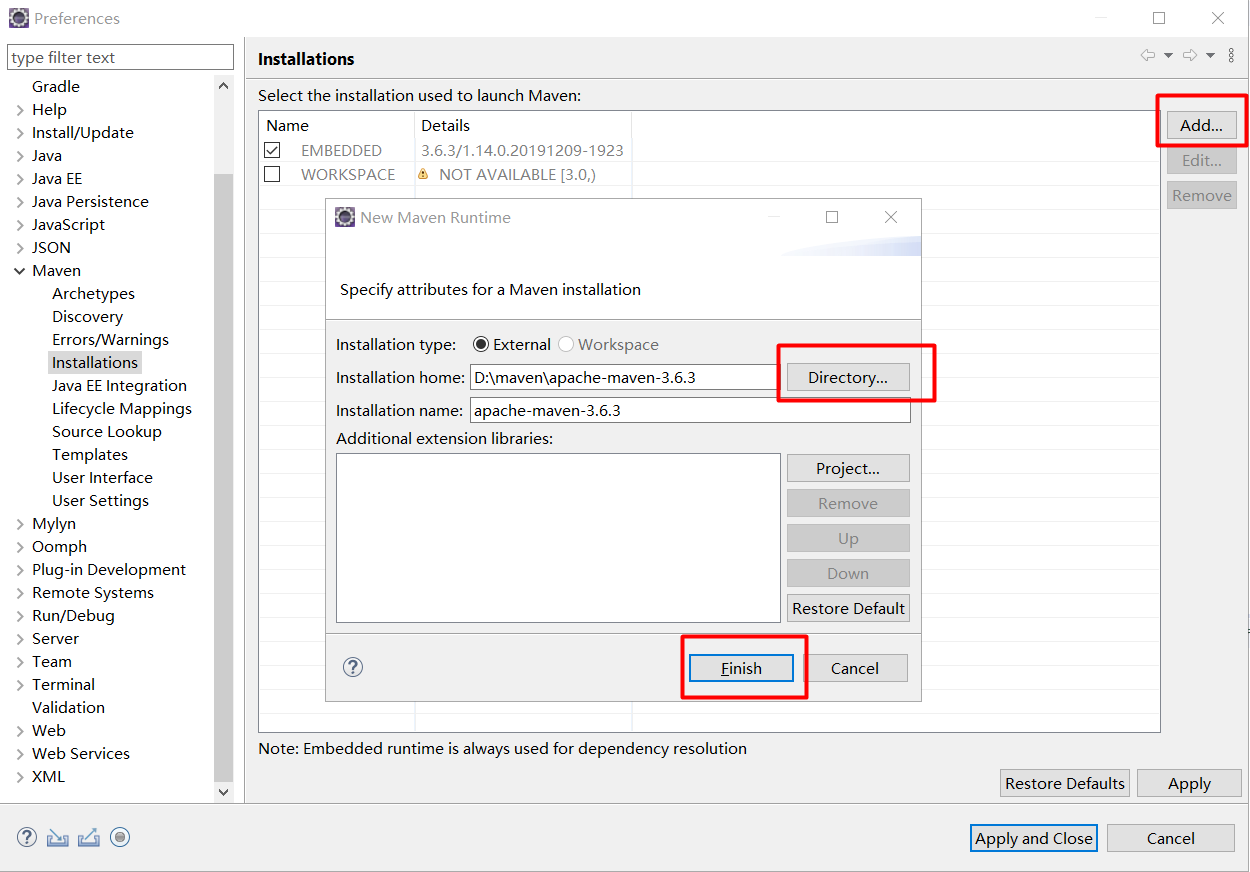
# Eclipse中使用Maven

## 配置Maven

1. 打开主菜单【Window】下的【Preferences】，在左侧找到【Maven】下的【Installations】，如下：

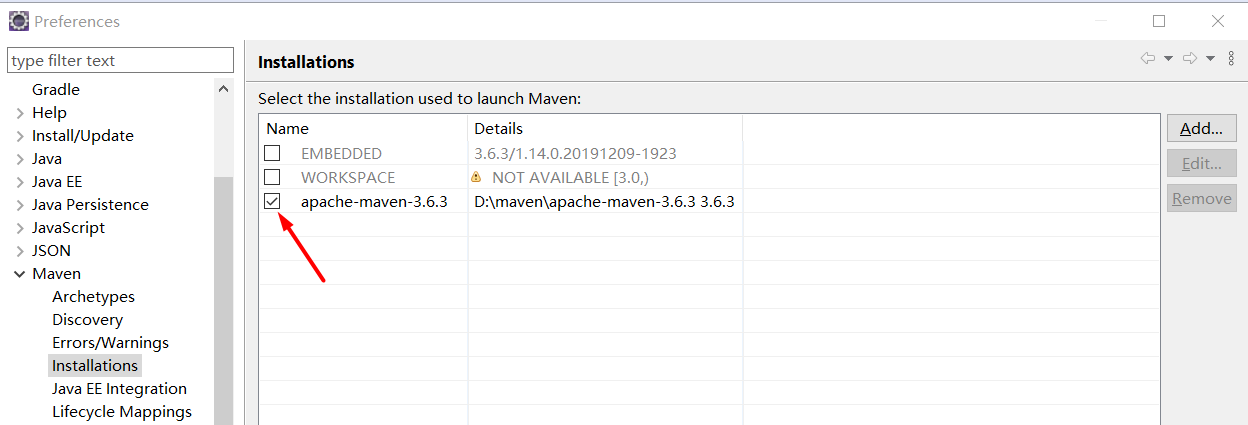


1. 点击右侧的【Add】，出现【New Maven Runtime】窗口，在右侧点击【Directory】浏览选择maven的安装目录，如下：

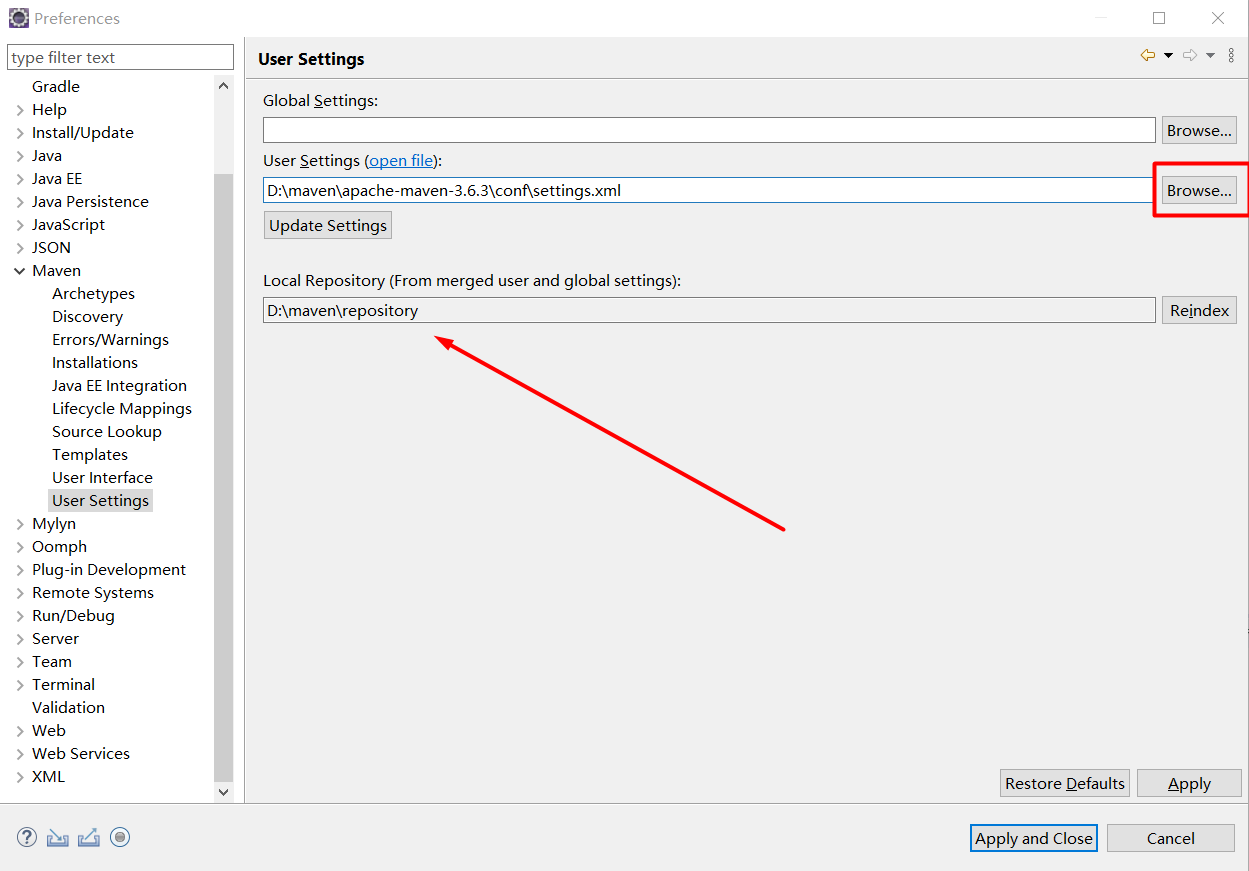


选择好之后，点击【Finish】。

1. 点击【Finish】后，记得把√上刚刚所添加的Maven。如下：



1. 继续在右侧选项中选择【Maven】下的【User Settings】。点击右侧【Browse】选择maven安装目录下conf文件夹中的settings.xml配置文件。如下：

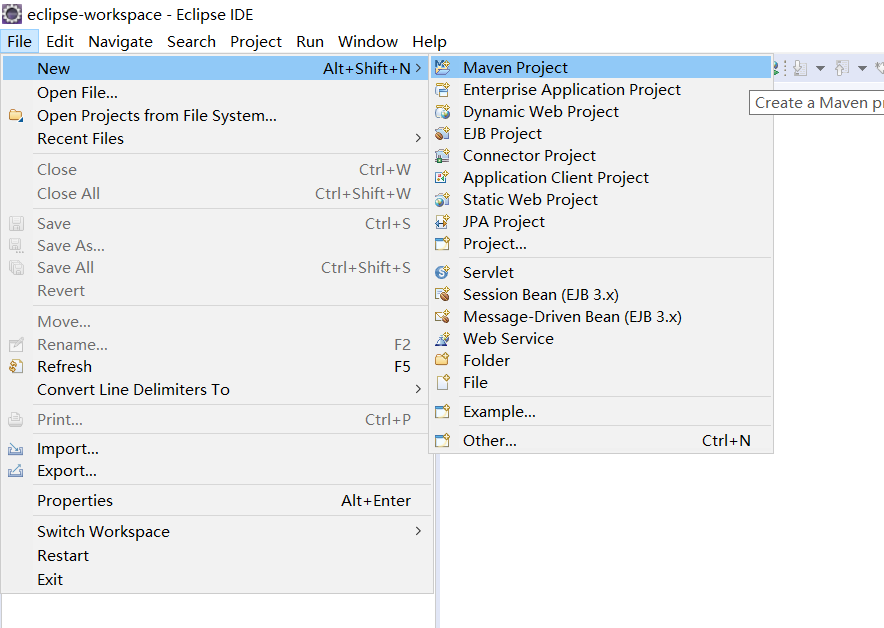


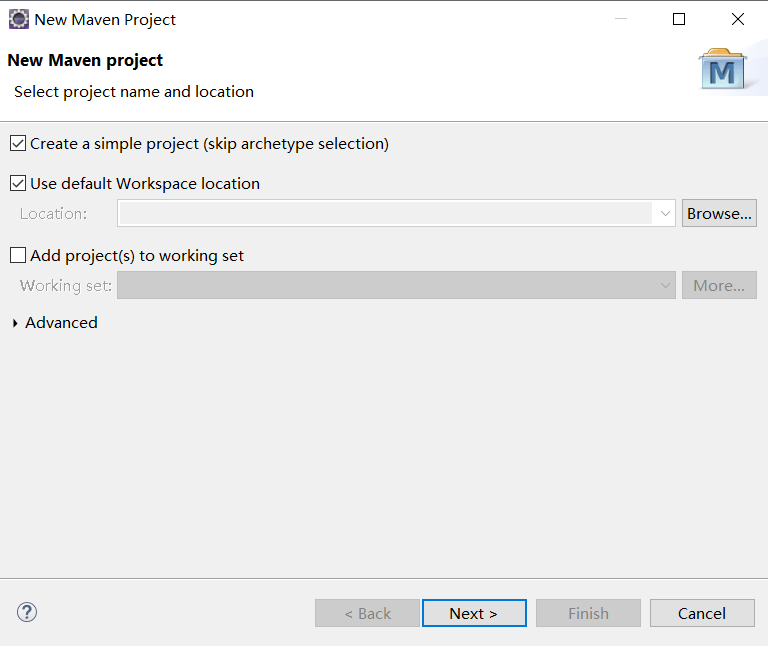
由于你在settings.xml文件中配置了本地仓库，所以它会自动地寻找到本地仓库路径。点击【Apply and Close】保存即可。

## Eclipse创建Maven项目

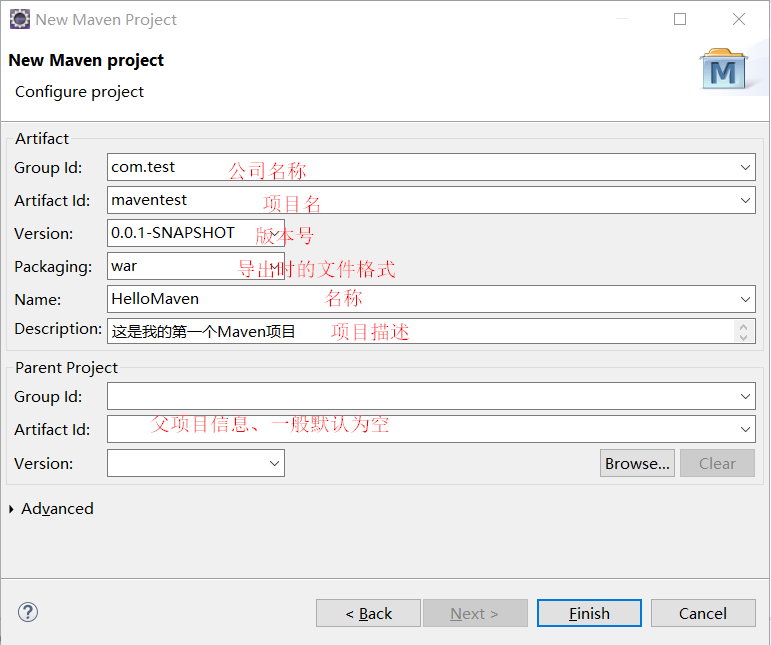
1. 在主界面中，选择主菜单【File】->【New】--【Maven Project】创建Maven项目。

如下：



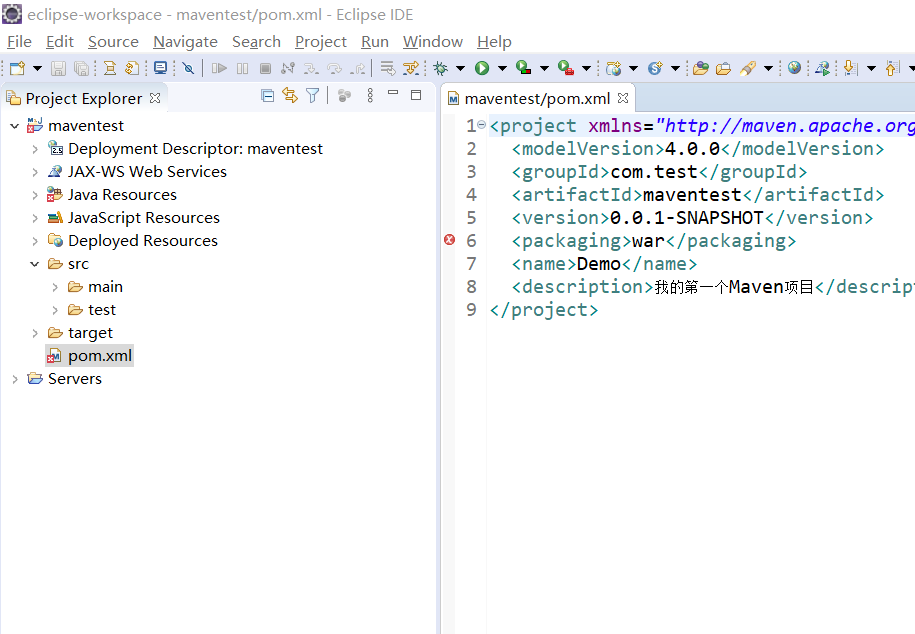


1. 点击【Next】，填写项目的相关信息。如下：



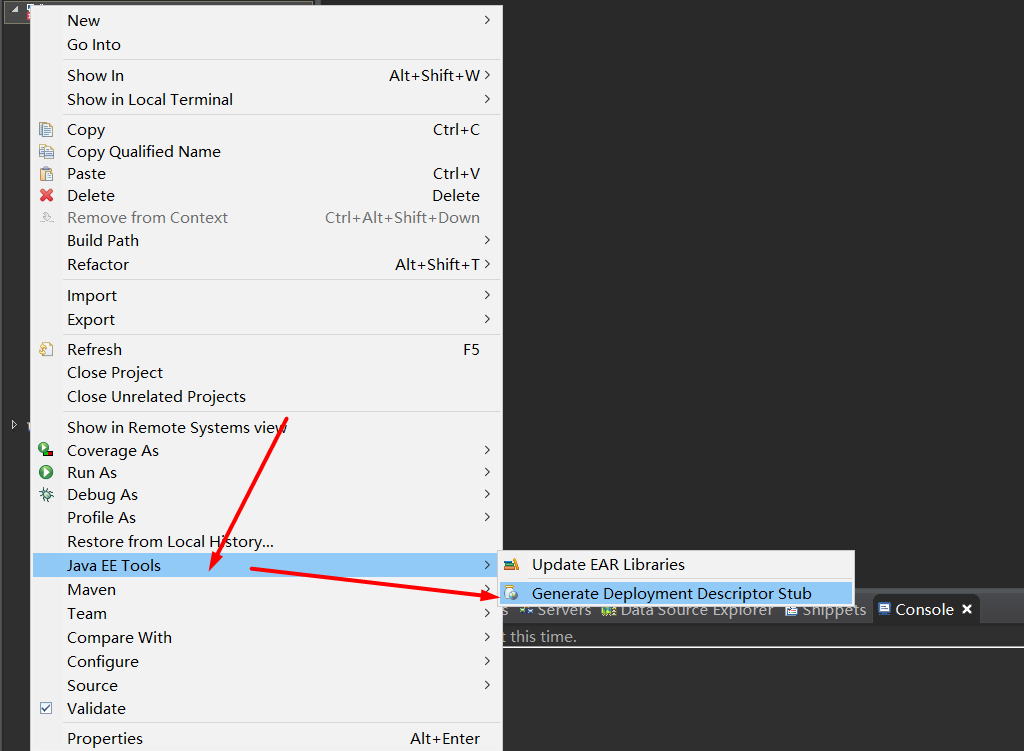
填写完项目信息后，点击【Finish】完成。

1. 创建好项目后，会报错。如下：



这是因为缺失了web.xml文件，还有可能就是没有修改编译环境。

1. 选中Maven项目右键点击【Java EE Tools】下的【Generate……】选项，自动生成web.xml文件。如下：

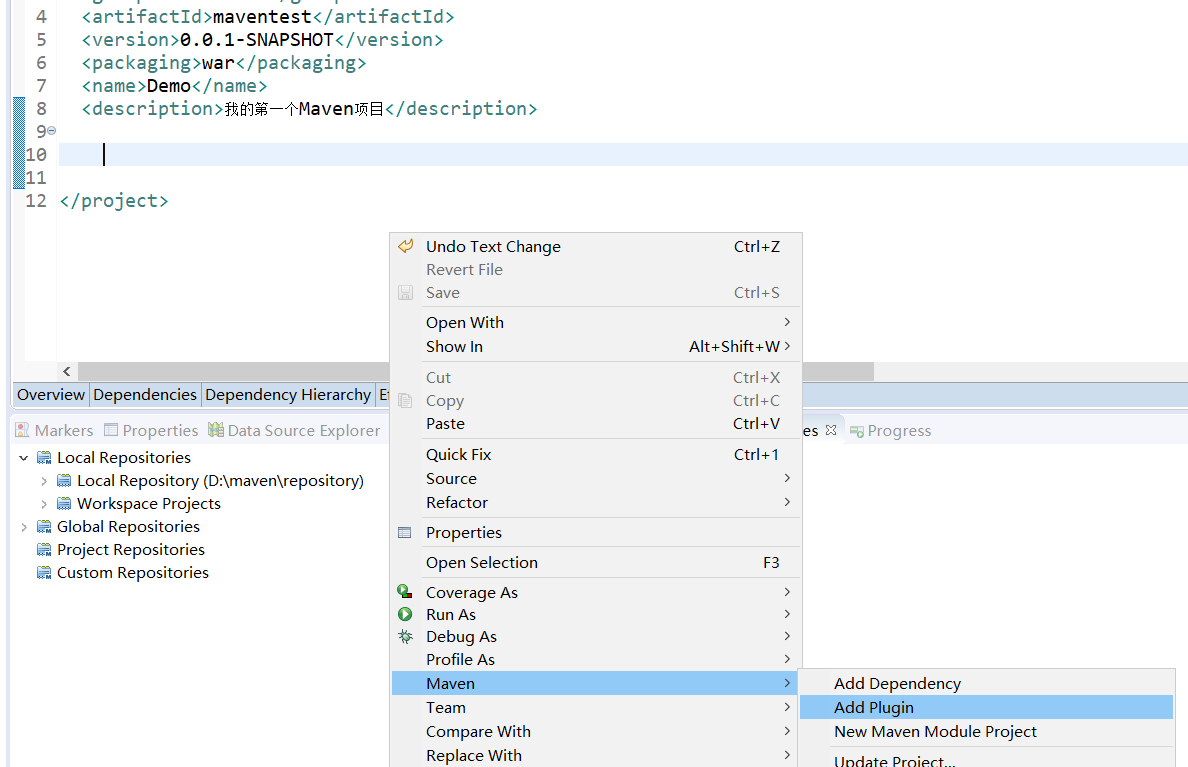


这样错误就会消失。

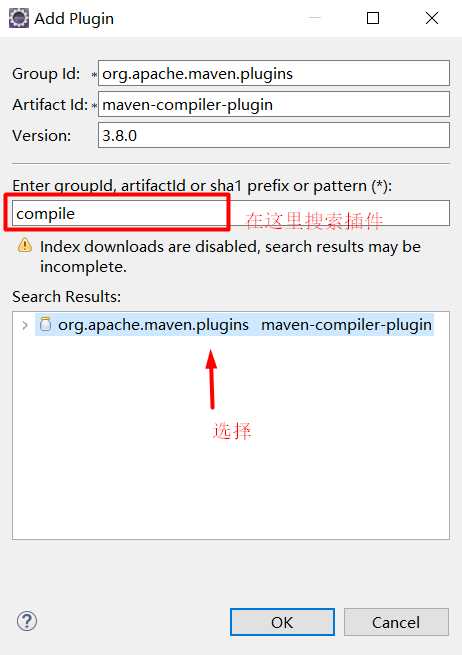
## 修改JDK版本

为什么要修改JDK插件呢？因为Maven默认的JDK版本一般很低，而如果你的源码中是使用了新版本的JDK，且使用了该版本JDK的新特性时，此时Maven默认版本JDK版本很低，一些新特性难免不支持。修改JDK版本有两种方式：一是在pom.xml文件中添加指定版本的JDK插件，二是在maven安装目录/conf/settings.xml文件中进行全局配置。

1. **在pom.xml文件中添加JDK插件**
2. 在pom.xml文件中鼠标右键选择【Maven】-->【Add plugin】。如下：

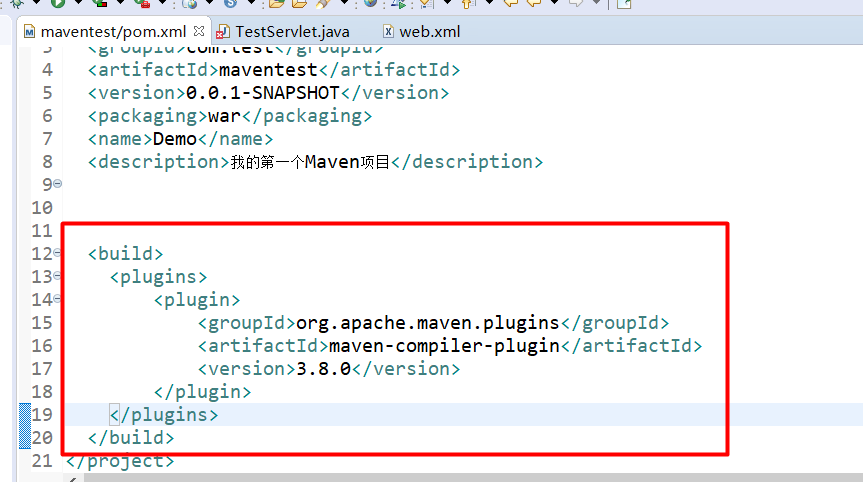


1. 在【Add Plugin】窗口中搜索compile插件，选中它点击【OK】进行添加。如下：

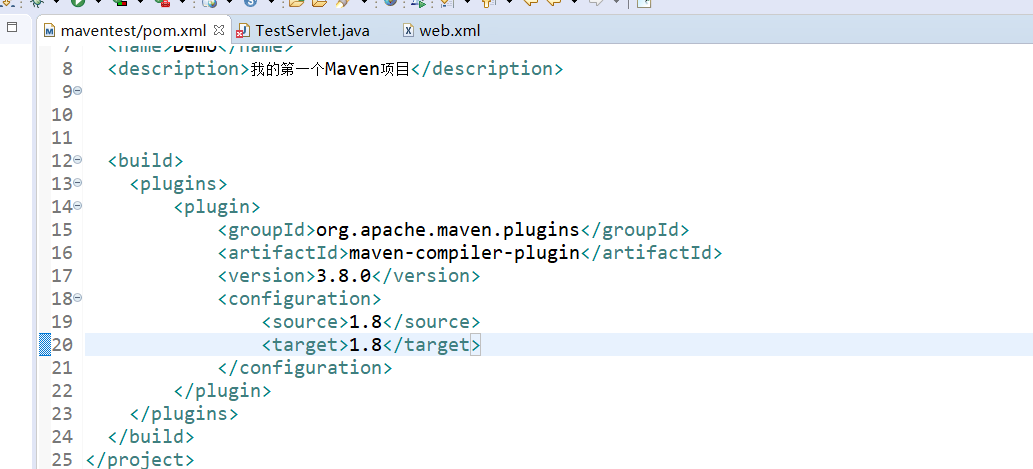


如果第一次引入时仓库中没有，则需要手写插件代码。

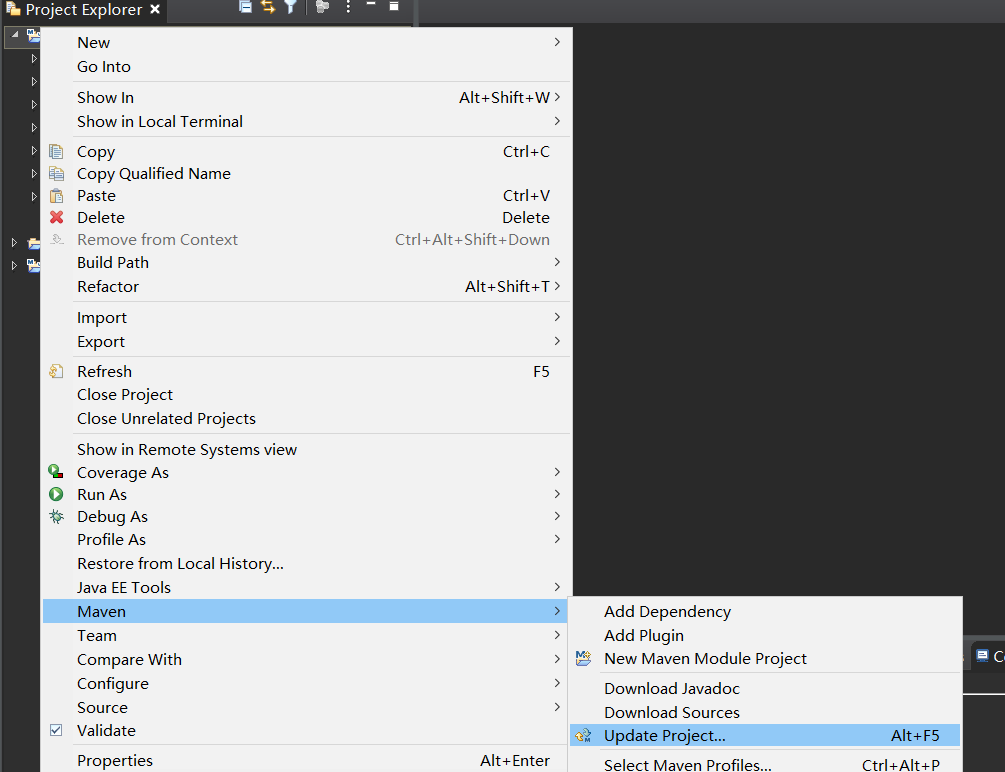
之后我们可以看到在pom.xml文件中生成了插件。



1. 还需要对这个插件进行一些修改。如下：



配置好JDK插件后，需要更新一下这个Maven项目。



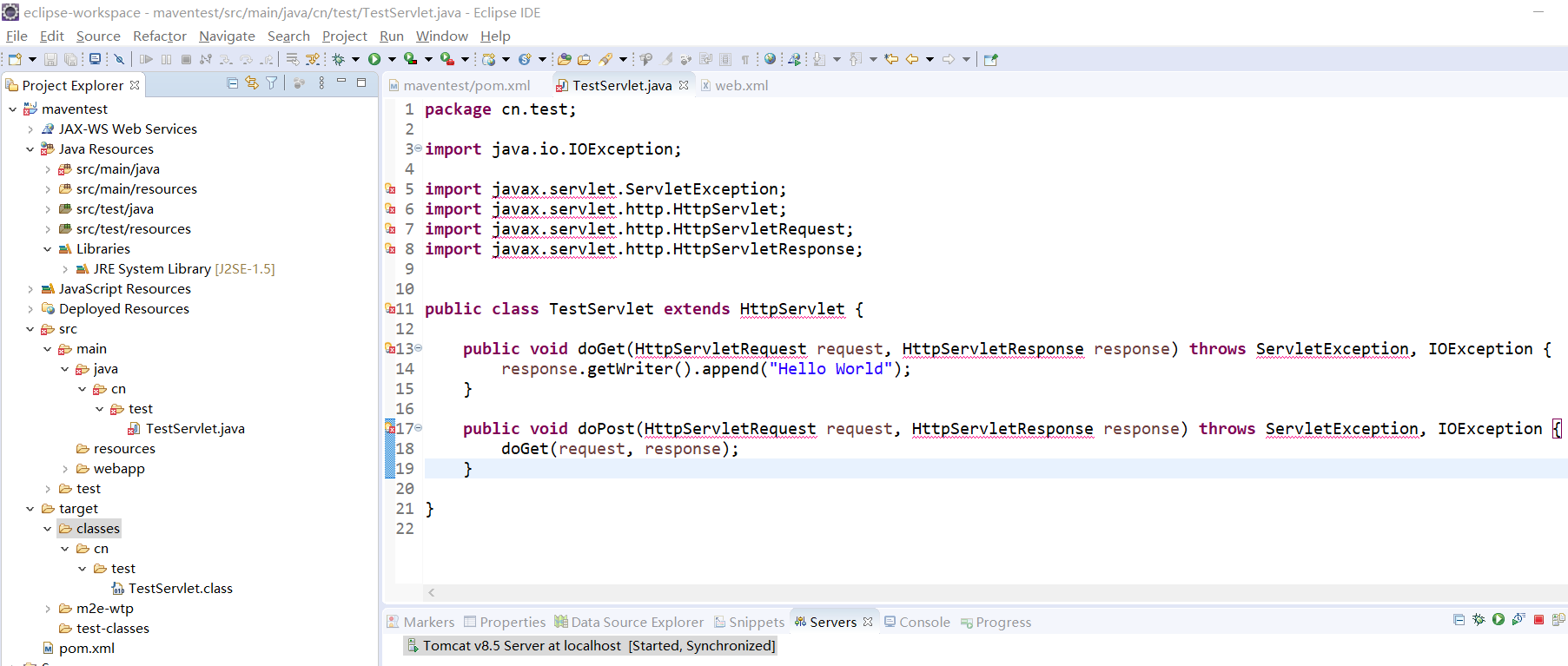
1. **全局配置JDK版本**除了可以在pom.xml文件中通过添加JDK插件的方式来修改JDK版本，但这种方式是局部的，每创建一个Maven项目都需要到pom.xml中修改版本。为此，可以通过修改Maven配置文件的方式进行全局的统一JDK版本，这样就不需要在每个Maven项目中的pom.xml文件中通过添加插件的方式来修改JDK版本了。在Maven安装目录/conf/settings.xml文件中添加全局配置代码：



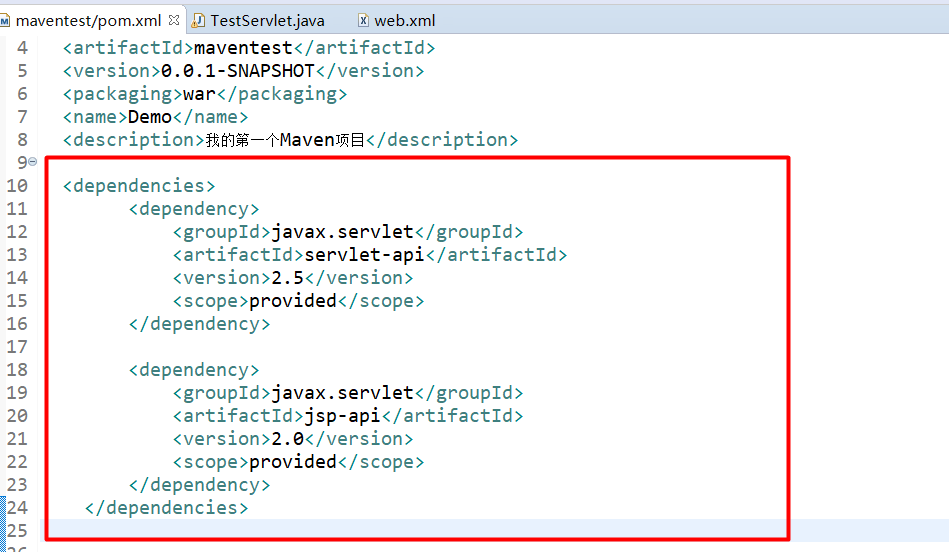
以上表示创建的Maven项目都默认使用JDK1.8版本的JRE，这样，以后每次创建的Maven项目就会默认使用1.8版本的JDK和JRE了。

## 项目中引入jar包

1. 在maven项目通过依赖引入jar包。例如，现在再src/main/java下创建一个Servlet，如下：

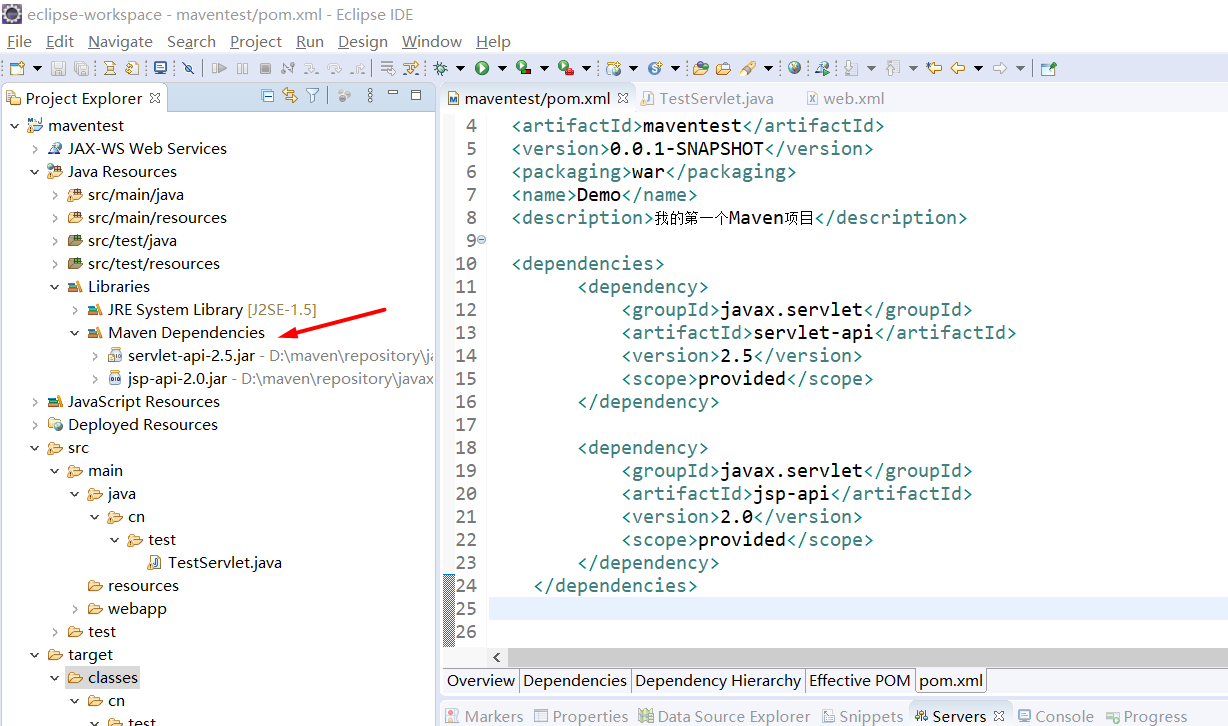


1. 这时，会发现引入的类不存在。因为我们没有引入jar包。所以我们需要在项目根目录下的pom.xml文件中添加依赖管理从本地仓库中引入jar包，如下：



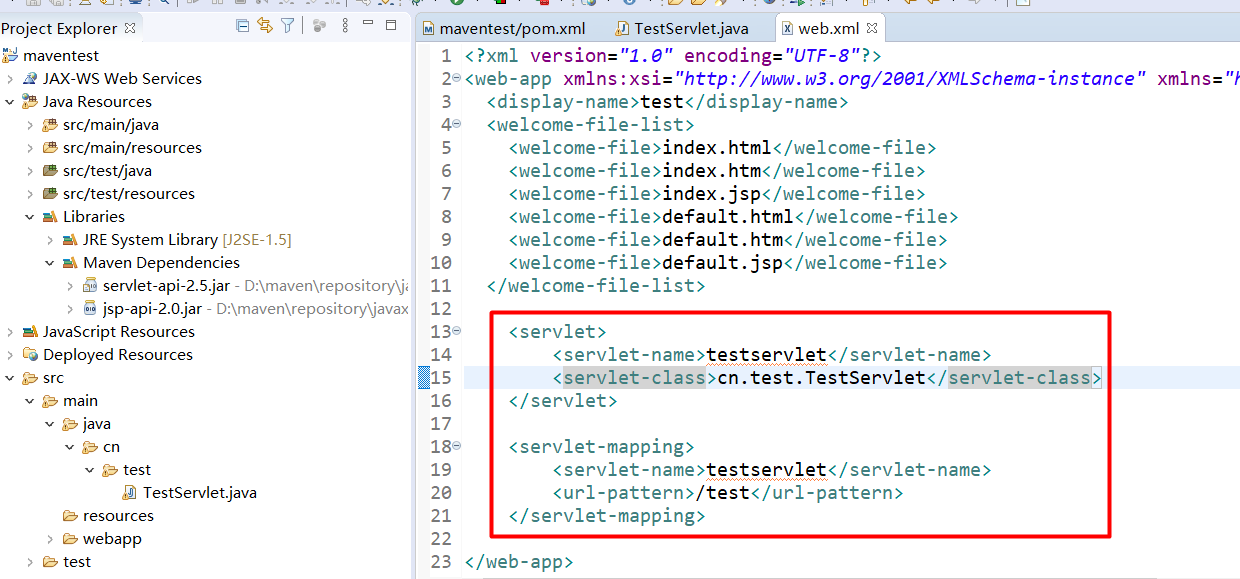
在这里引入了两个jar包，之所以该jar包的作用域是provided，是因为不需要将这些jar包进行打包，如果要将此项目安装到Tomcat上，Tomcat本身就存在这些jar包（在Tomcat安装目录中的lib目录中）。关于依赖的作用域后面会讲到。

1. 引入jar包后，会在Library目录下生成Maven Dependencies目录，此目录中为该项目引入的jar包。如下：

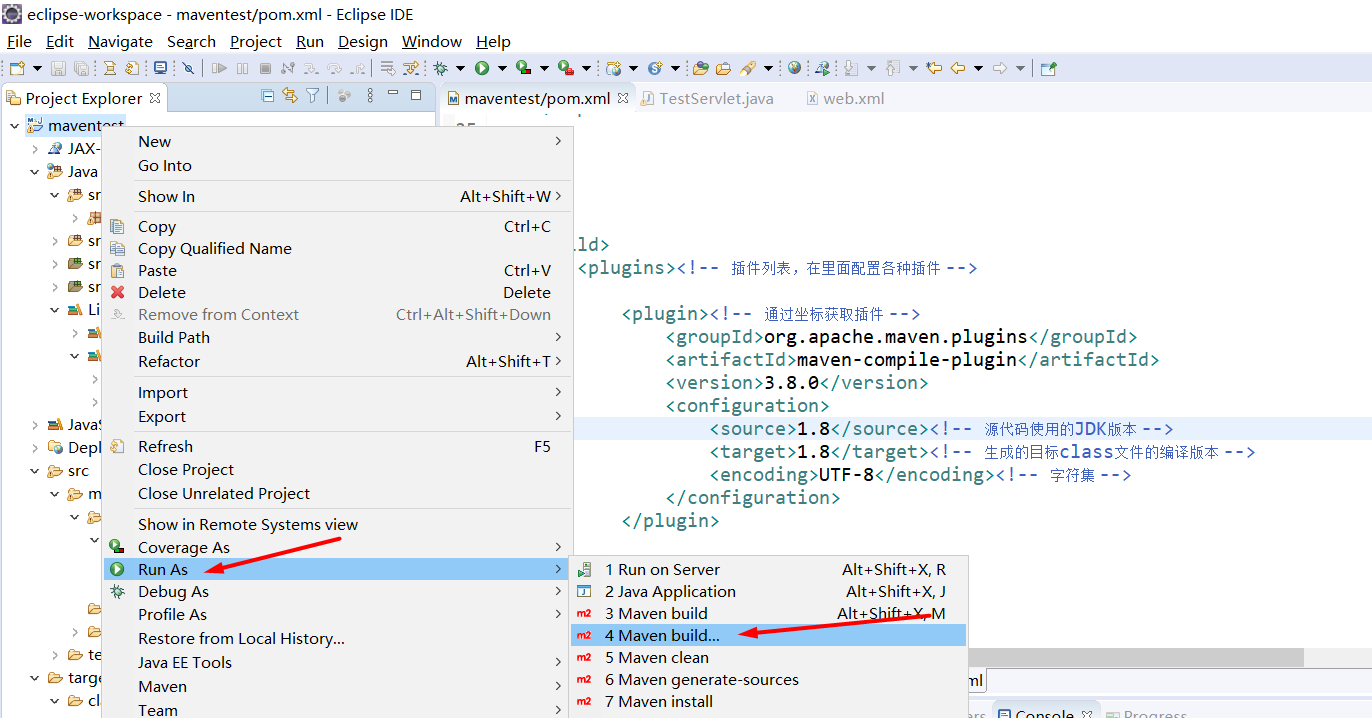


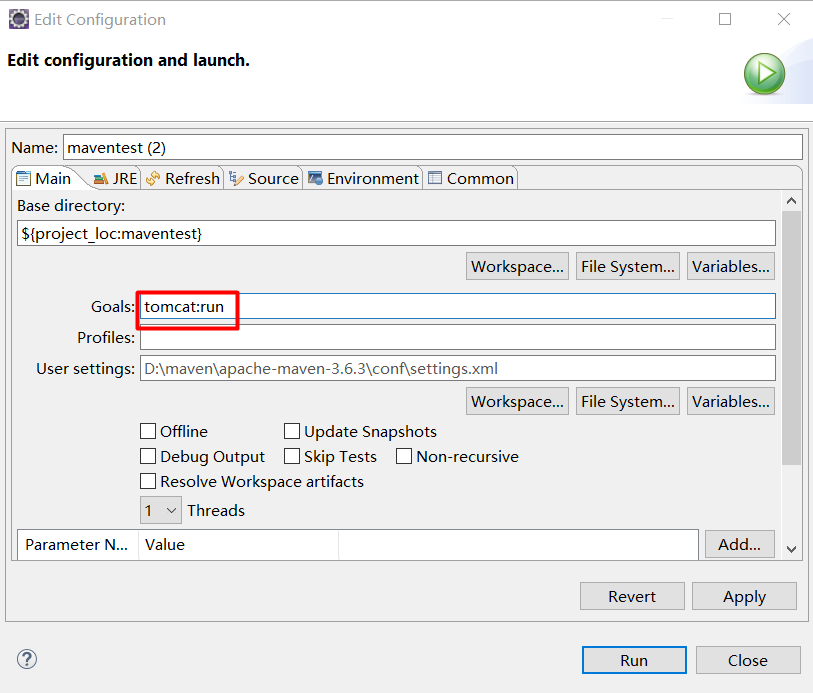
## 运行项目

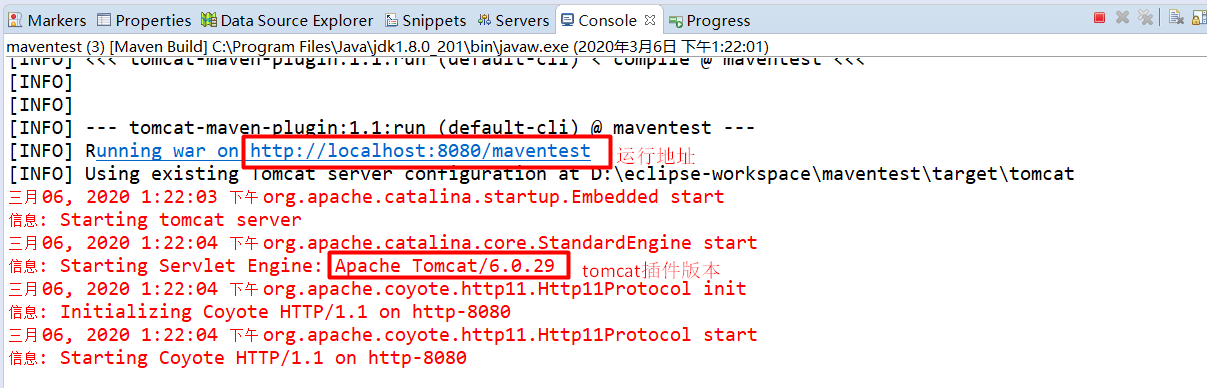
1. jar包引入后，在web.xml文件中配置该Servlet。如下：



1. 配置好Servlet后，运行一下这个Maven项目。你可以通过两种方式来运行这个Maven项目：一是通过本地tomcat服务器运行、二是通过Maven的tomcat插件运行。在这里通过Maven的tomcat插件运行。鼠标右击Maven项目。点击【Run As】下的【Maven build…】。如下：

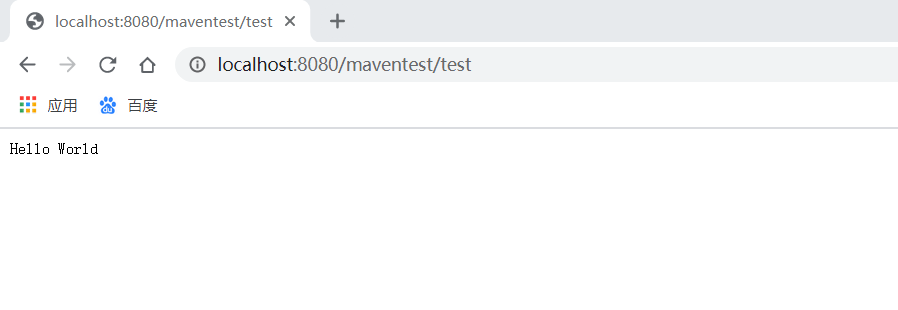


1. 在出现的窗口中填写命令。如下：
2. 点击【Run】后，在Console窗口显示tomcat插件启动日志。如下：



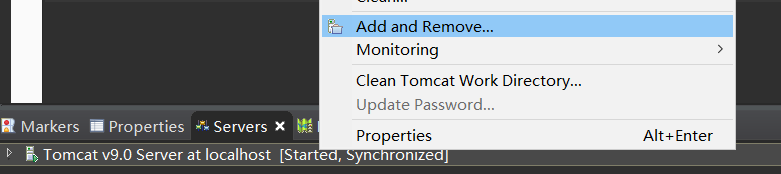
在这里会显示运行地址以及tomcat插件版本信息等。

1. 复制运行地址，在浏览器地址栏中输入地址，运行项目。如下：



项目成功运行。

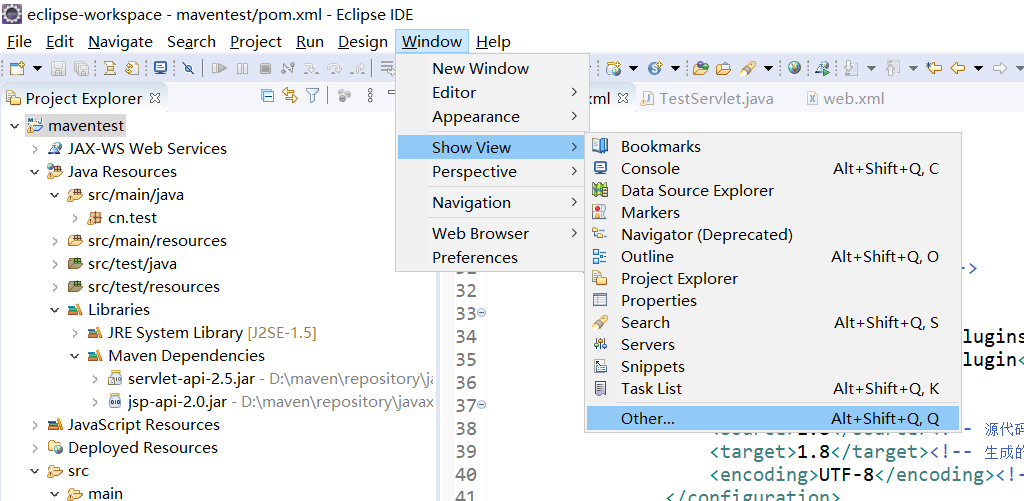
6）如果要通过本地的Tomcat运行，将项目添加到配置的Tomcat服务器中，

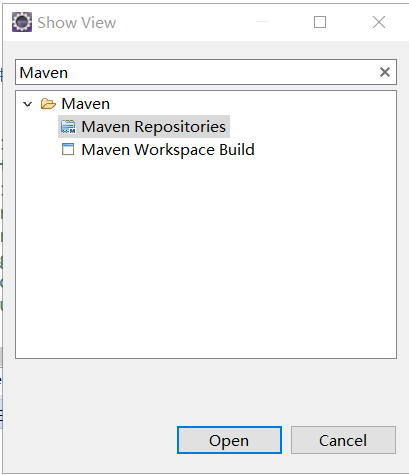


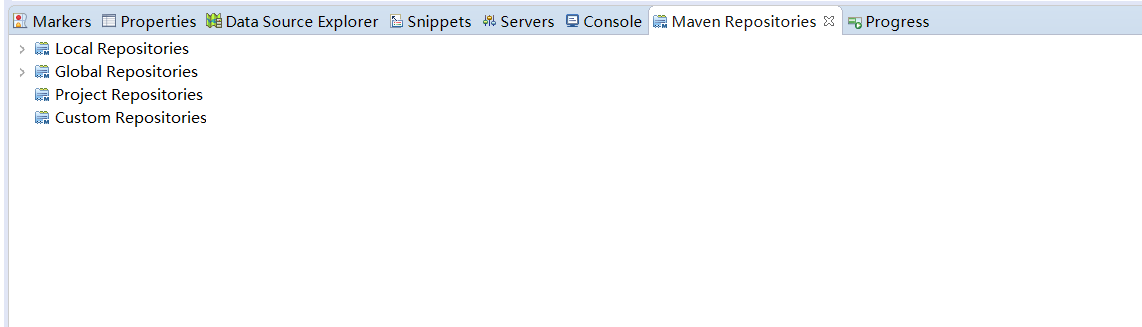
## 为本地仓库建立或更新索引

为本地仓库建立索引能够提高jar包引入效率。

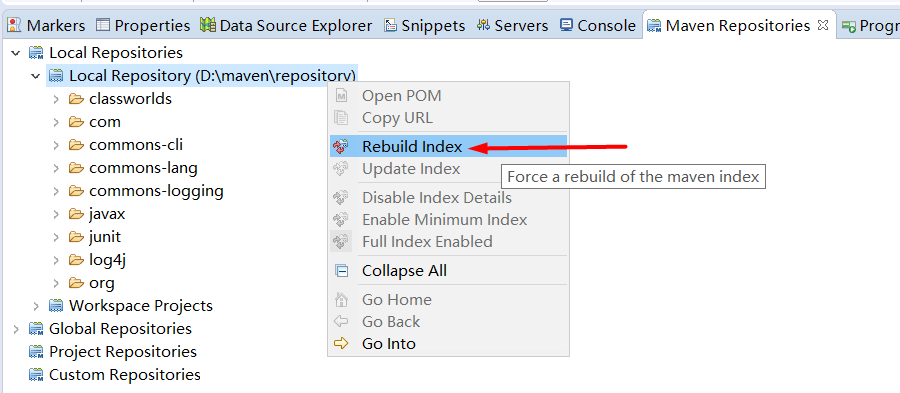
1. 在主界面【Window】-->【Show View】-->【other】找到【Maven Repositories】，将该视图显示。如下：







1. 其中，Local Repositories为本地仓库。我们鼠标右击为其添加索引。如下：



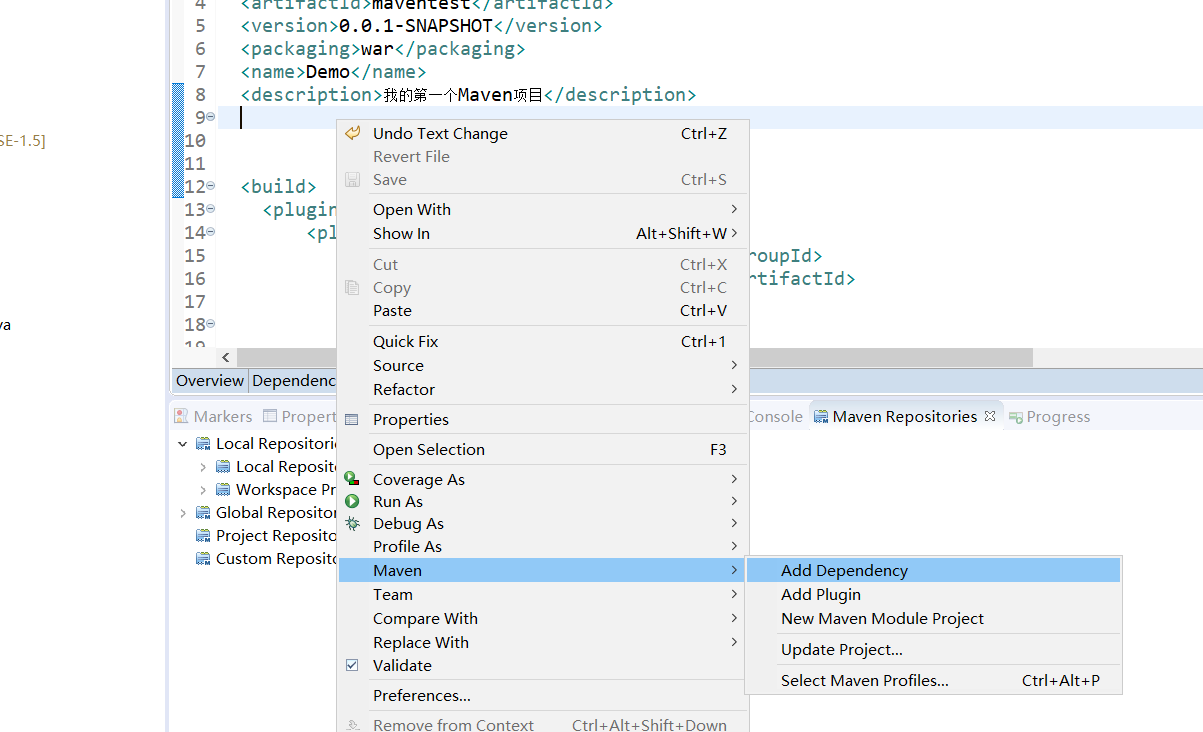
在右下方会显示进度。



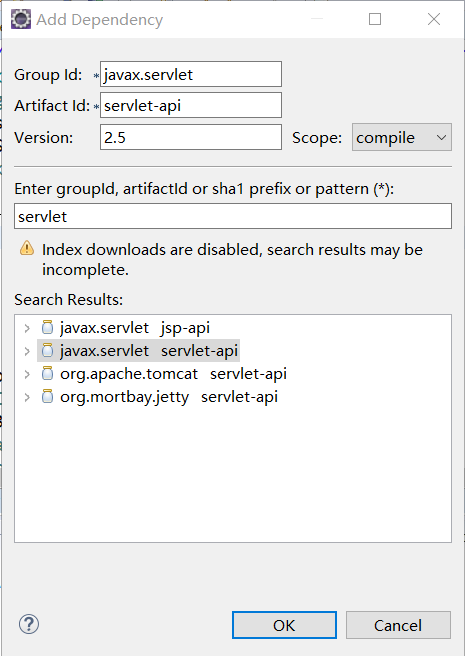
## 添加依赖

添加依赖指的是为项目添加jar包。

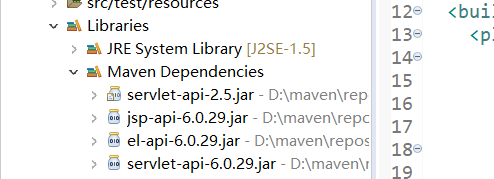
1）在pom.xml的空白处鼠标右键。选择【Maven】-->【Add Dependency】。如下



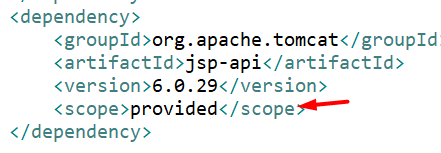
1. 在打开的【Add Dependency】窗口中，搜索想要的jar包，选中点击【OK】，即可在pom.xml中生成依赖；如果jar包有多个版本的话，你也可以选择jar包的相应的版本。如下：







添加依赖管理后，依赖默认的作用域为compile，我们一般还需要手动给它添加作用域。如下：

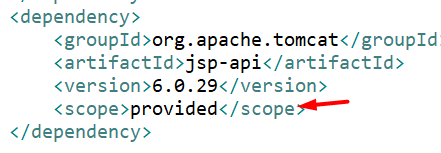


关于作用域后面会进行讲解。

## 依赖的作用域

依赖作用域也称为依赖范围。

在pom.xml文件中添加依赖后，默认的作用域是compile，我们一般需要对其修改，也就是通过<scope>标签指定作用域。如下：



它的取值有5种：

1. compile：默认的作用域，在编译、运行、测试、打包时有效；
2. runtime：测试、运行、打包时有效
3. provided：编译、测试时有效，运行、打包无效；
4. test：仅在测试时有效
5. system：使用system作用域时不会去本地仓库寻找依赖，要指定本地路径。（不推荐使用）



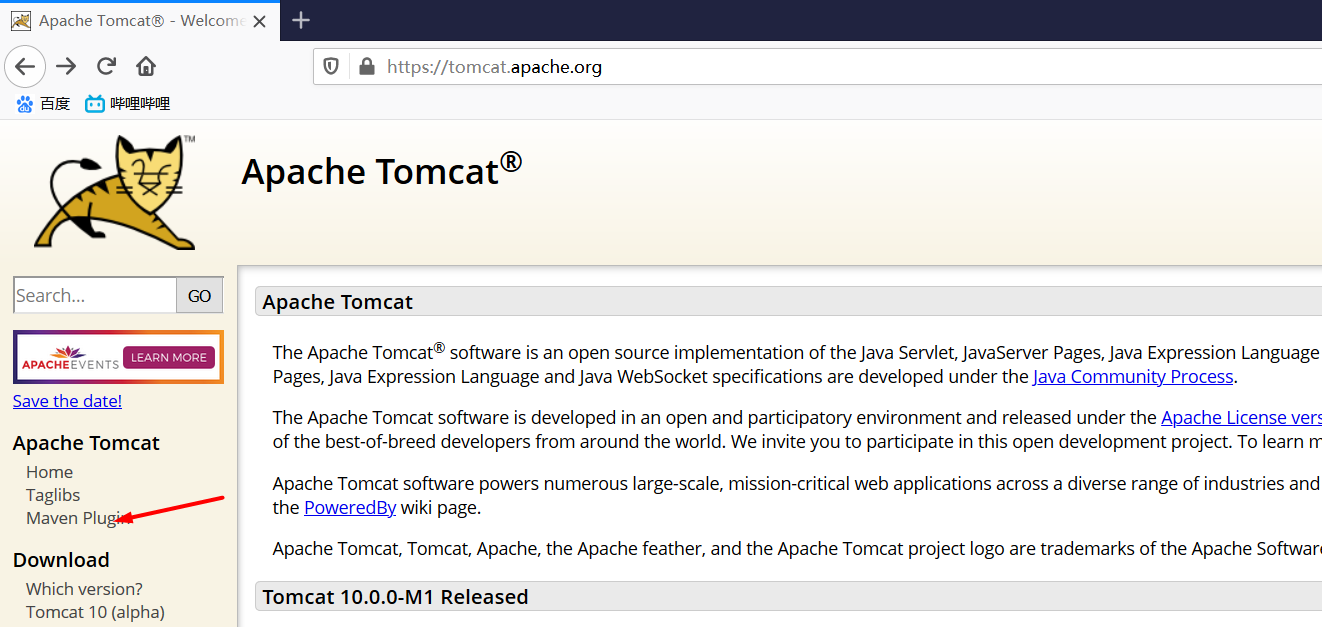
例如某个依赖的作用域为compile，则该项目打war包时会将该依赖打入WEB-INF下的lib目录中。

## 配置tomcat7插件

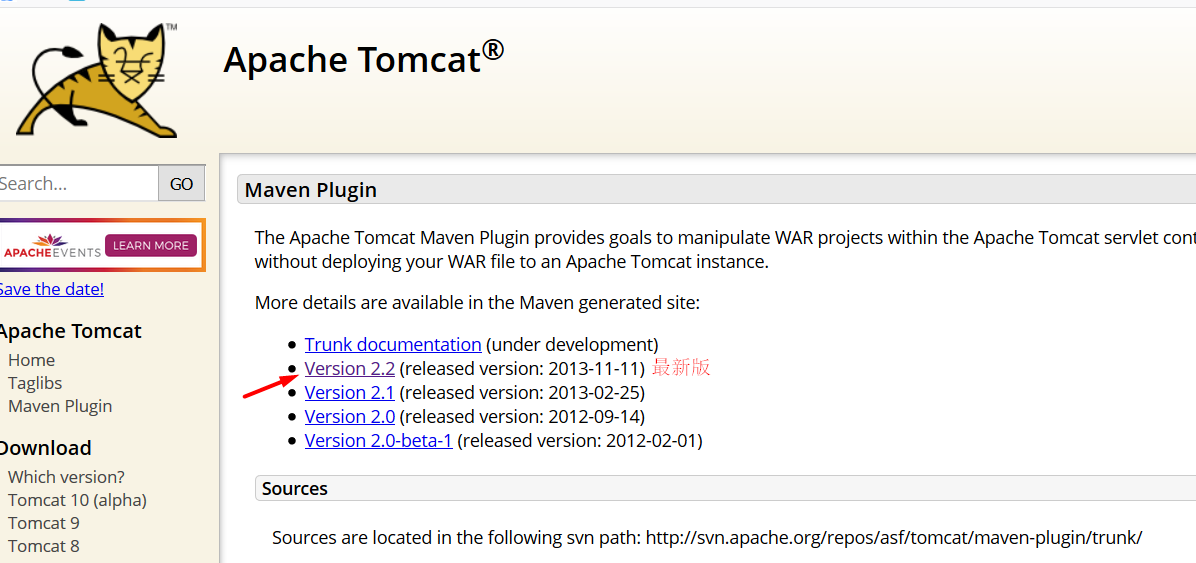
通过tomcat插件，我们可以将tomcat服务内嵌到web项目中，直接运行webapp项目。

如果没有配置tomcat7插件的话，使用插件的方式运行程序，则会默认使用maven默认版本的tomcat插件来运行，但maven默认的tomcat插件版本一般都很低。所以，如果有必要配置一个tomcat7插件，因为目前最新的内嵌tomcat插件就是tomcat7。

1. 打开tomcat.apache.org官网，在左侧进入Maven Plugin（Maven插件）页面。如下：

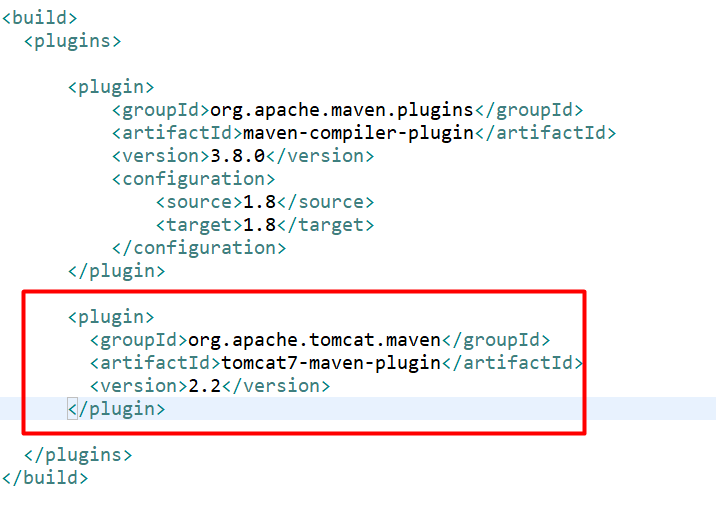


1. 进入最新版的tomcat插件页，如下：



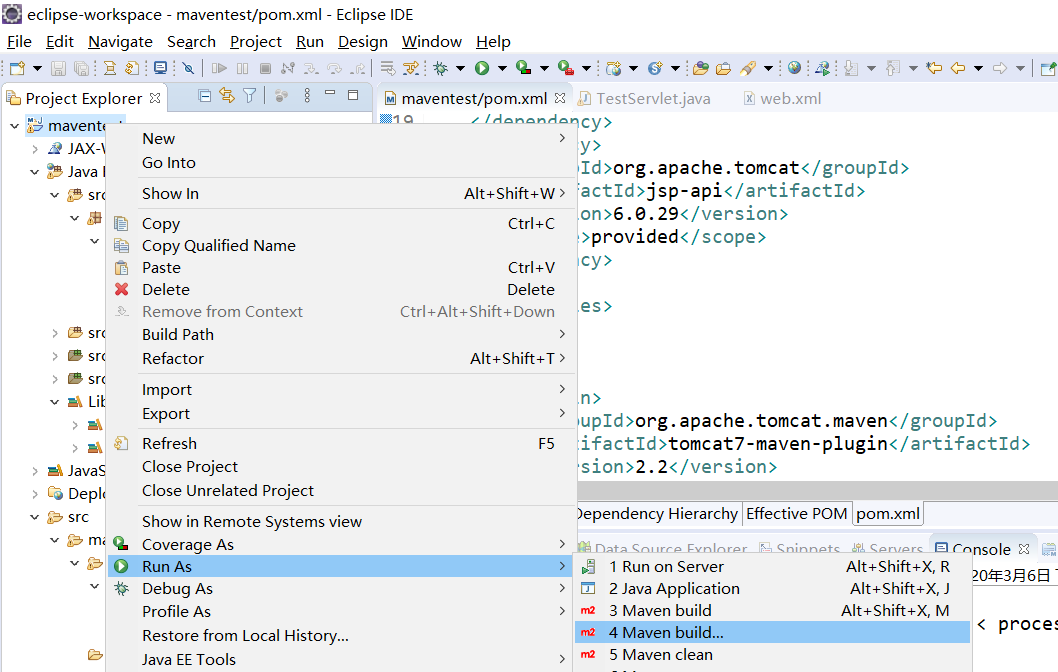
1. 复制插件代码到pom.xml插件列表中。如下：



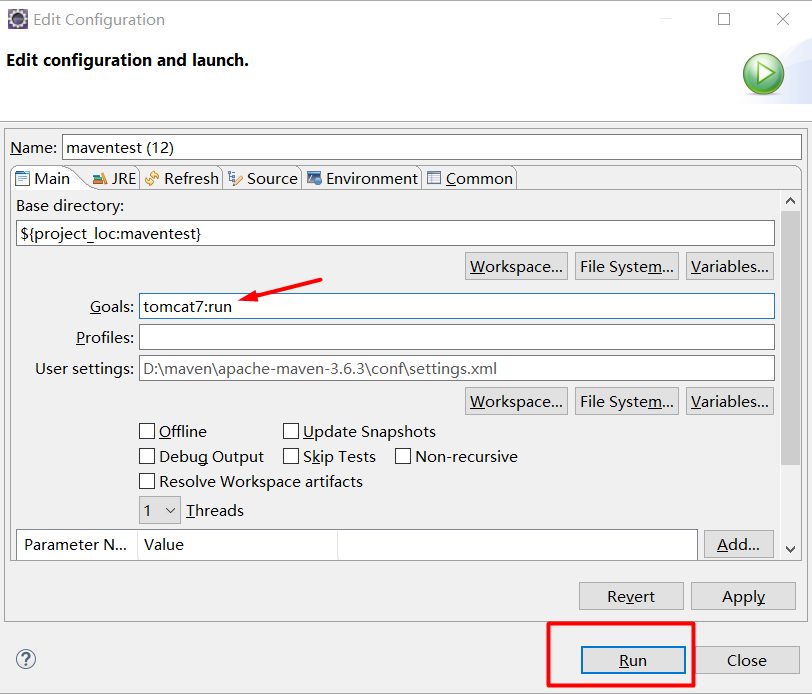


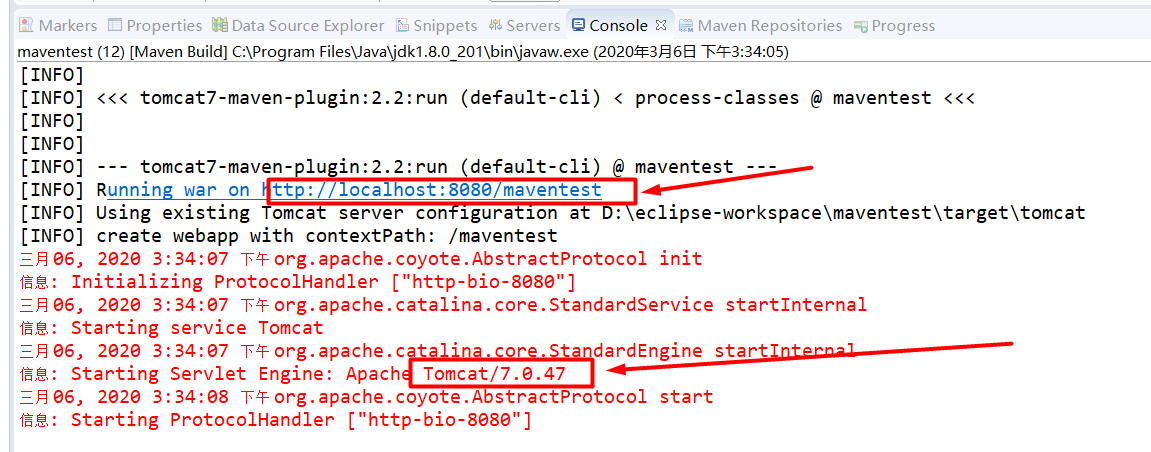
这样，tomcat7插件就配置成功了。

1. tomcat7插件配置好了，如果要使用tomcat插件运行程序，在项目右键打开【Run As】下的【Maven build…】。如下：

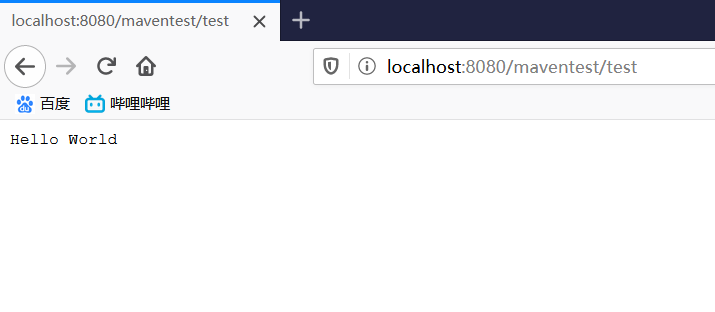


1. 在这里我们输入tomcat7:run，表示使用配置的tomcat7插件运行程序了，如果还是用的tomcat:run，则使用的还是Maven默认的tomcat插件运行程序。点击【Run】，便会在【Console】打印运行日志。如下：



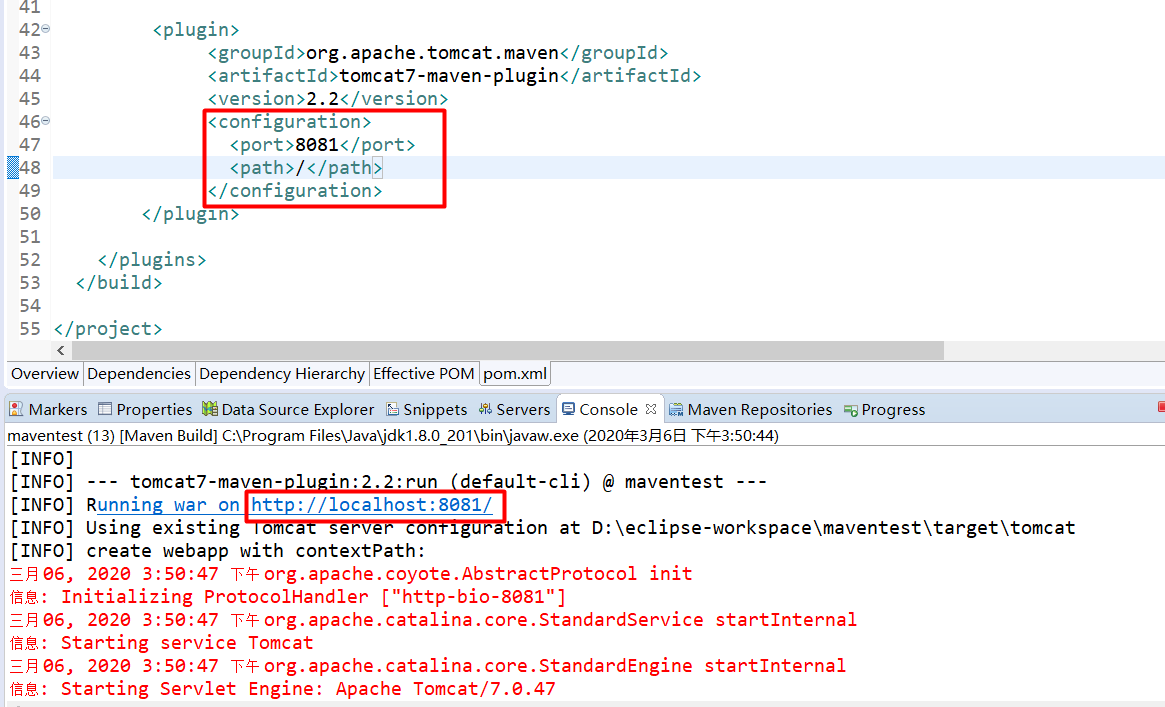


在浏览器输入地址访问。如下：

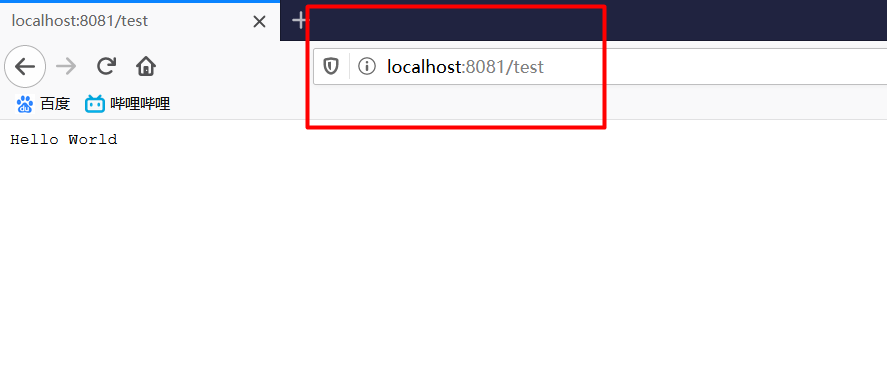


访问成功。

1. 此外，还可以在pom.xml中对tomcat插件进行细节设置，如端口号、路径等。如下：



配置<path>后，其访问路径如下：

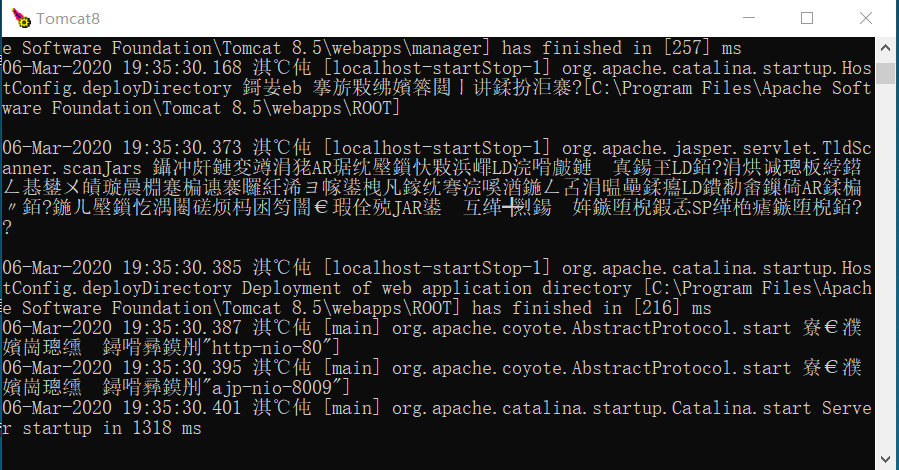


# tomcat热部署

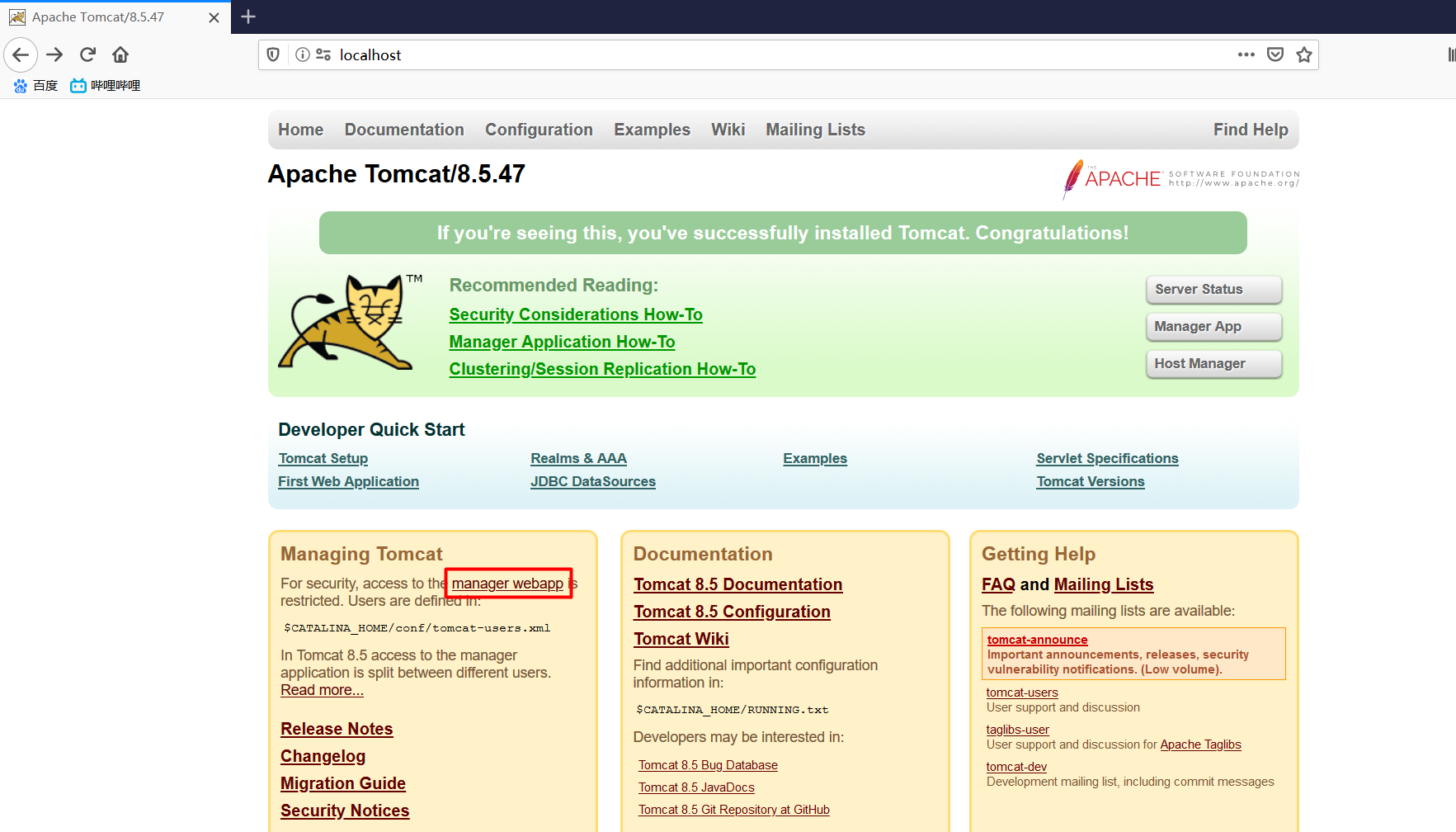
热部署指的是在tomcat服务器运行时进行项目的部署。以前都是先部署好项目再启动服务器。热部署的位置是本地tomcat的，所以在进行热部署前请先确保，

## 在tomcat应用管理页进行热部署

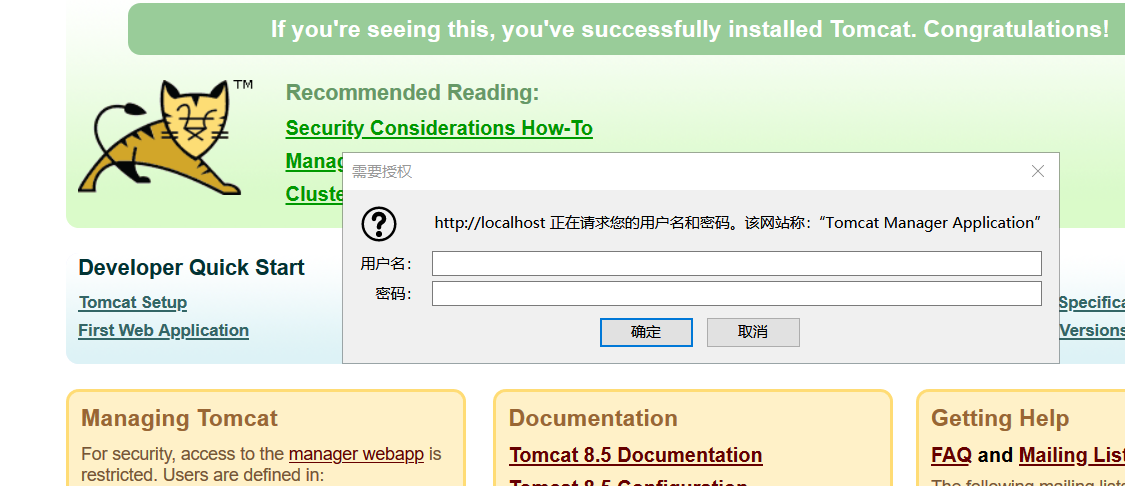
1. 启动本地tomcat服务器。如下：



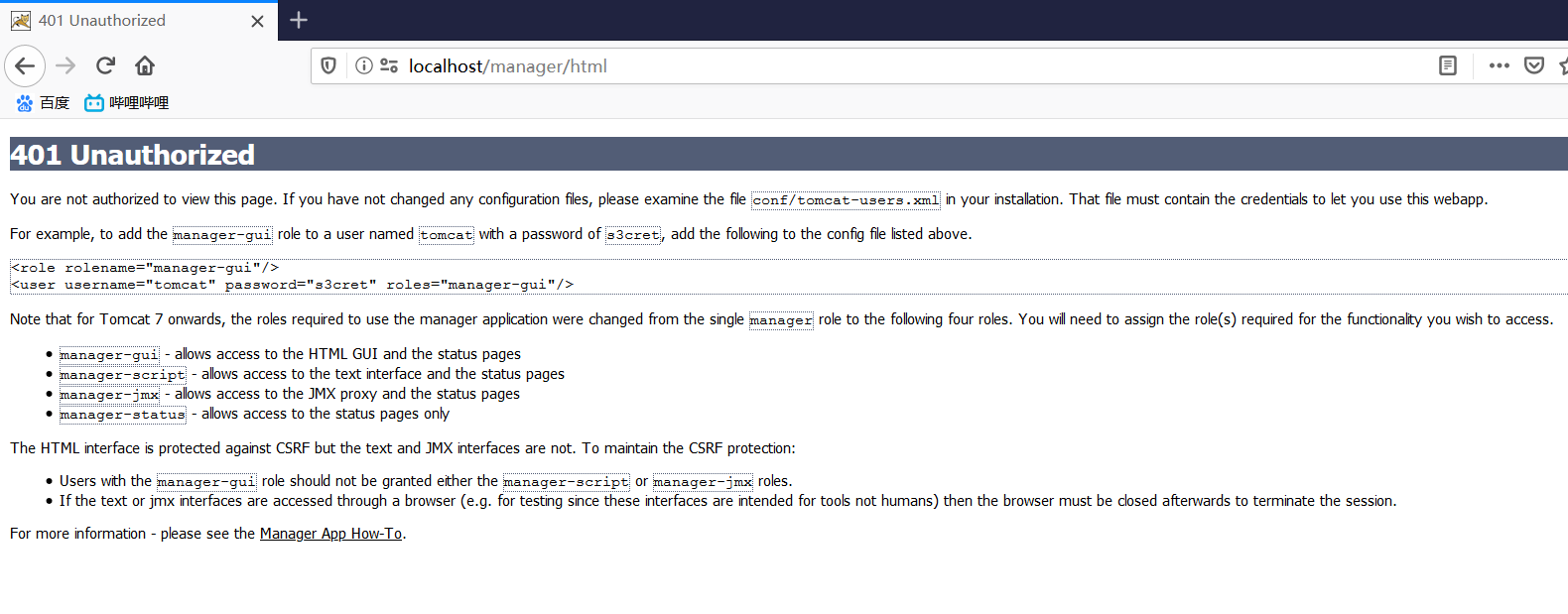
1. 在浏览器中输入localhost，进入tomcat服务页，进入webapp管理入口。如下：



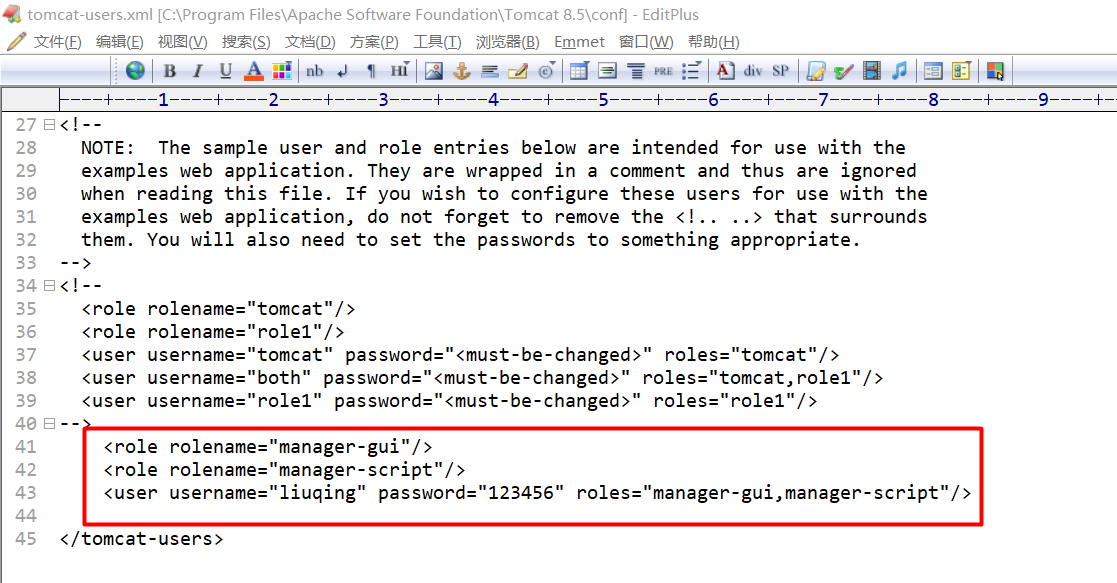
1. 在进入webapp管理页面时，会提示你输入用户名和密码。如下：



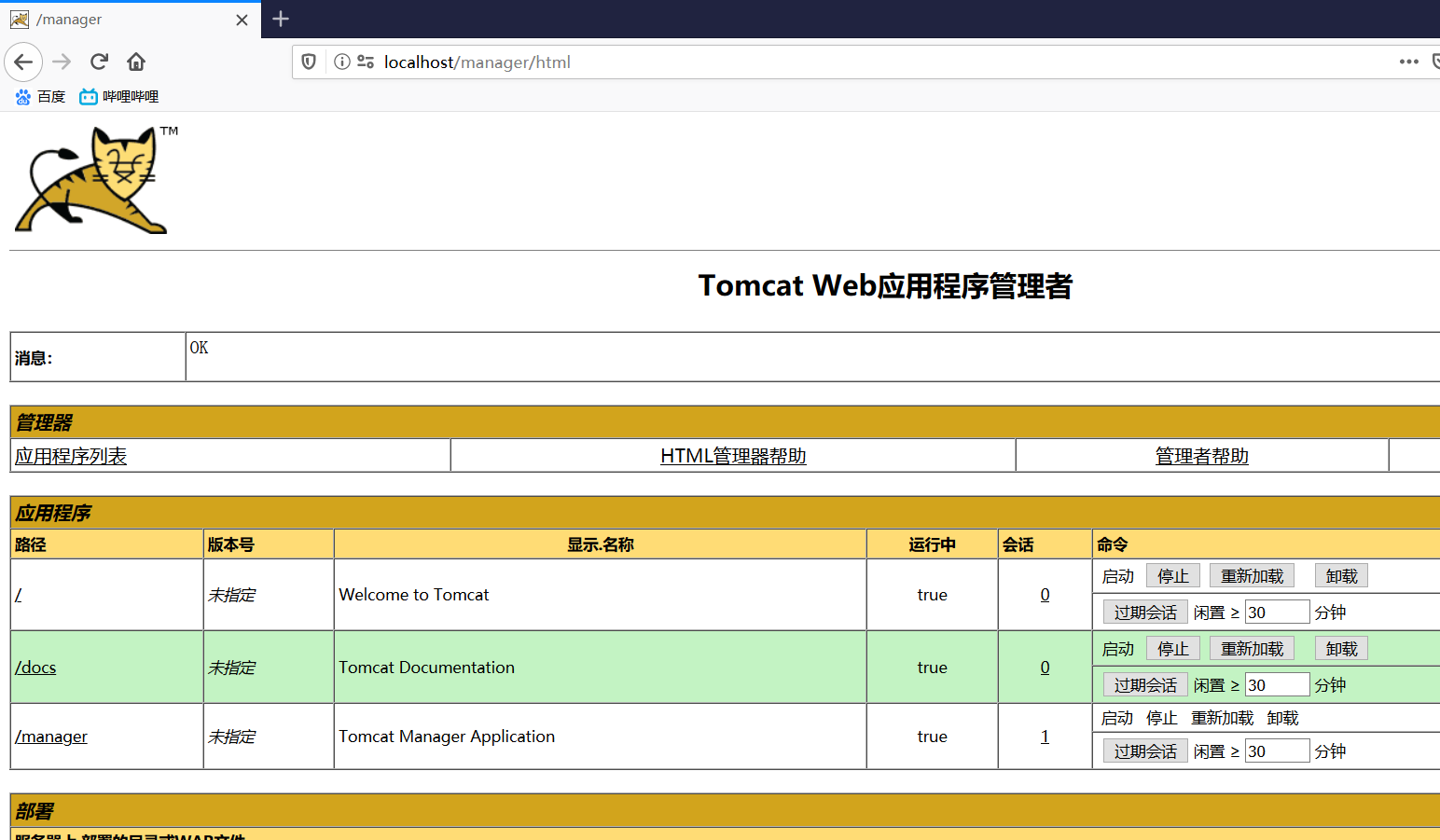
1. 如果你已经配置用户名和密码了，则可以进行登录。如果没有配置，点击取消验证后，会进入到一个错误页。如下：

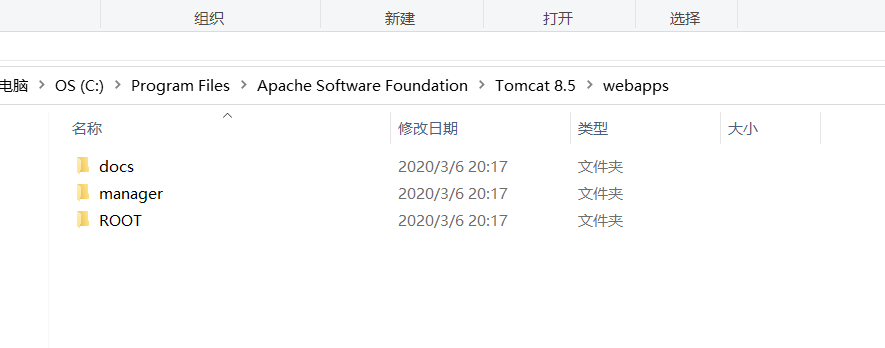


1. 进入到tomcat安装目录，找到conf下的tomcat-users.xml文件，将其打开。配置用户名和密码。如下：

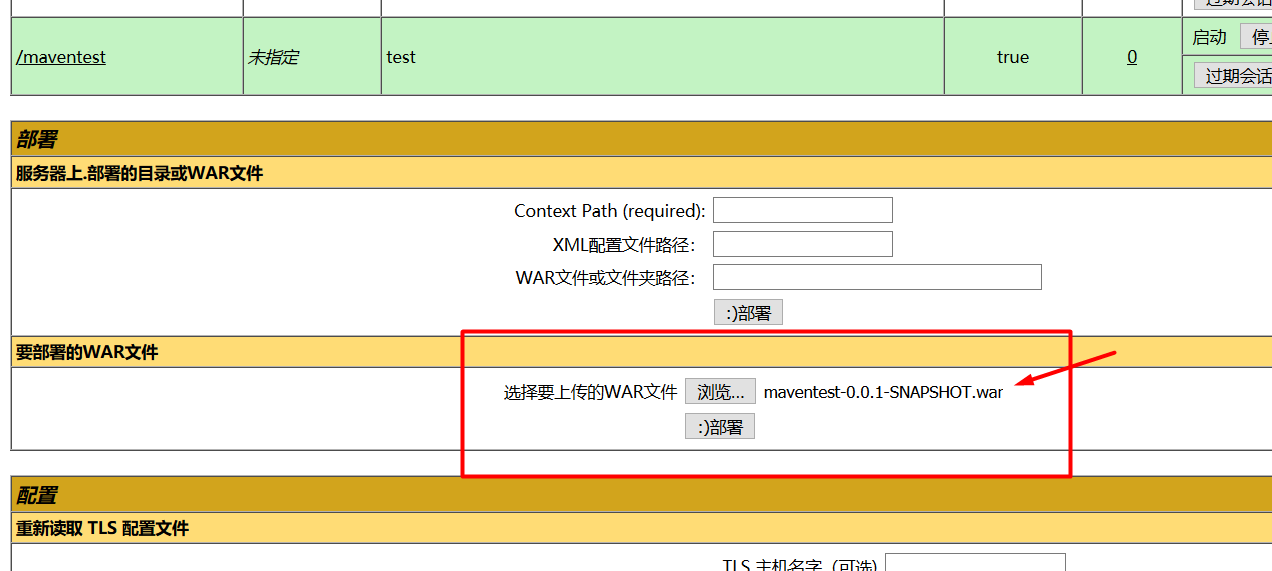


1. 配置完成后，重启tomcat服务。进入tomcat管理页，输入刚刚配置的用户和密码，进入webapp管理界面，它默认有3个应用，这些应用是tomcat服务，与tomcat安装目录下webapp目录中的3个文件相对应。如下：





1. 在下方浏览选择要上传的war包（war包可以进入到项目目录使用mvn package命令生成，也可以通过eclipse生成）。如下：

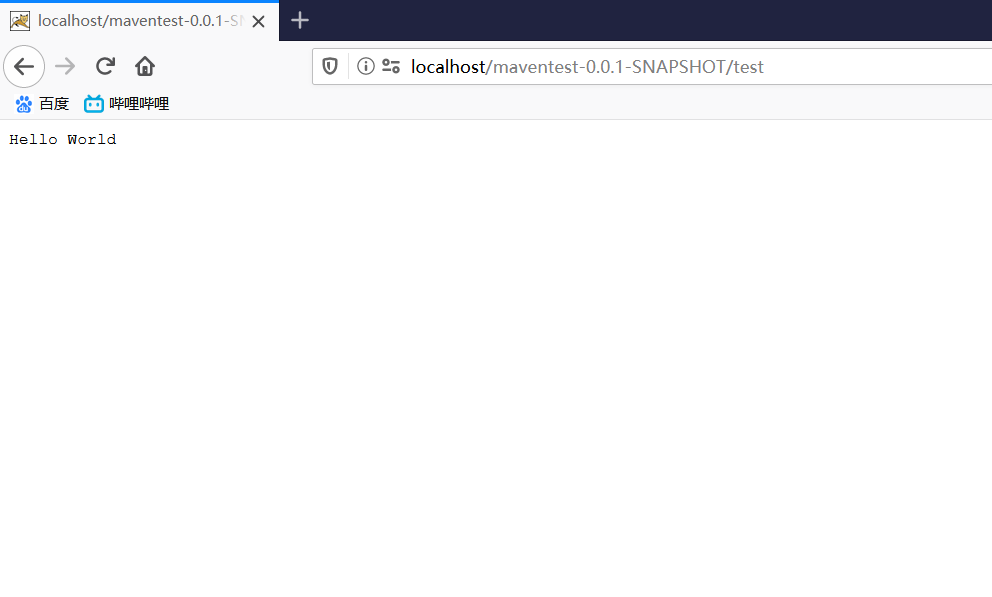


选择war包后，点击部署。

1. 部署war包后，会在上面的应用程序列表中出现你上传war包项目，如下：



同时你可以在浏览器地址栏输入地址进行访问。如下：



## 在Eclipse中进行热部署

暂无。