基于 Maven2+Eclipse+WTP+m2eclipse
开发 Java EE 应用程序

谢超良 Jimmy.Shine@Gmail.com
Sep 20,2010

一、 Maven

- **什么是 Maven?**

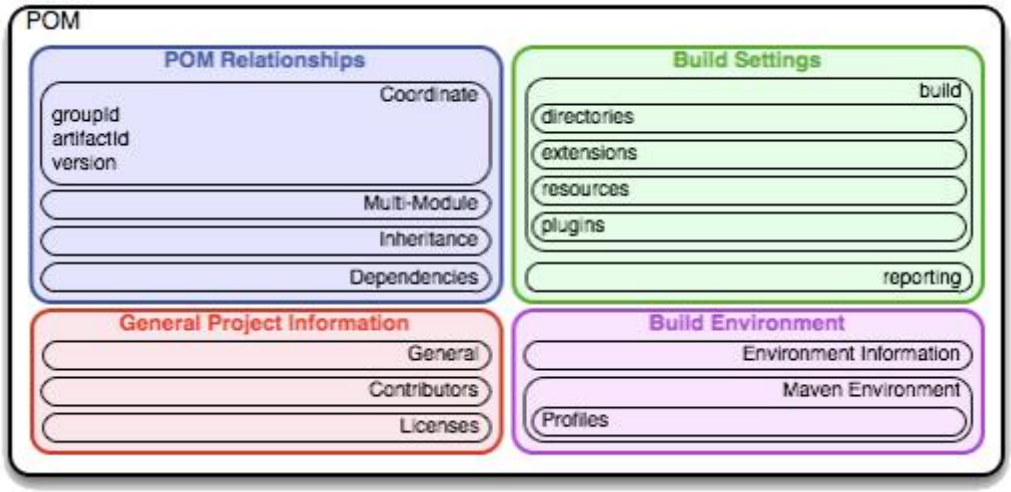
Maven 是标准、存储格式以及一些软件用以管理和描述项目。它为构建、测试、部署项目定义了一个标准的生命周期。它提供了一个框架，允许遵循 Maven 标准的所有项目，方便的重用的构建逻辑。Maven 项目存在的 Apache 软件基金会，是一个开源社区，它开发的软件工具，基于一个通用的软件对象模型（Project Object Model），也就是 POM。

- **约定大于配置**

Maven 遵循约定大于配置的原则。
通过给项目提供默认的行为来减少不必要的配置。

- **POM**

在一个 Maven 项目中，通过声明 POM 来指定项目的相关信息。

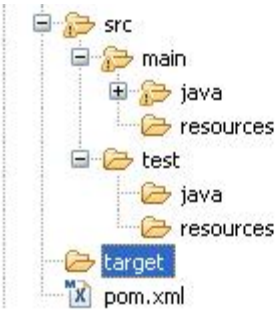


Maven 对于项目的唯一标识条件：
Group ID, artifact ID, version(简称 GAV)

- **Group ID**: An identifier for a collection of related modules. This is usually a hierarchy that starts with the organization that produced the modules, and then possibly moves into the increasingly more specialized project or sub-projects that the artifacts are a part of. This can also be thought of as a *namespace*, and is structured much like the Java package system.
- **Artifact ID**: The Artifact ID is a unique identifier for a given module within a group.
- **Version**: The version is used to identify the release or build number of the project.

- **Maven 项目的结构**

一个 Maven 的项目包括如下结构：



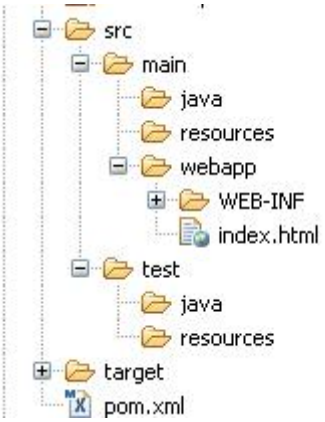
其中：

src 存放源代码，target 存入的是编译后的，pom.xml 是 Maven 用来描述项目的文件。

main 中存放用来发布至生产环境的代码及配置，test 中存放测试时候使用的代码及配置。

java 中存放 java 代码，resource 中存放配置文件。

一个 web 项目结构如下：



一个 web 项目中增加了一个 webapp，用来存放 web 页面代码。

- **Maven 的生命周期**

Maven 默认的生命周期阶段包括：

- **validate** - validate the project is correct and all necessary information is available
- **compile** - compile the source code of the project

- **test** - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package** - take the compiled code and package it in its distributable format, such as a JAR.
- **integration-test** - process and deploy the package if necessary into an environment where integration tests can be run
- **verify** - run any checks to verify the package is valid and meets quality criteria
- **install** - install the package into the local repository, for use as a dependency in other projects locally
- **deploy** - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

Maven2.2 的一个完整的阶段包括：

validate,generate-sources,process-sources,generate-resources,process-resources,compile,process-classes,generate-test-sources,process-test-sources,generate-test-resources,process-test-resources,test-compile,test,prepare-package,package,pre-integration-test,integration-test,post-intgration-test,verify,install,deploy

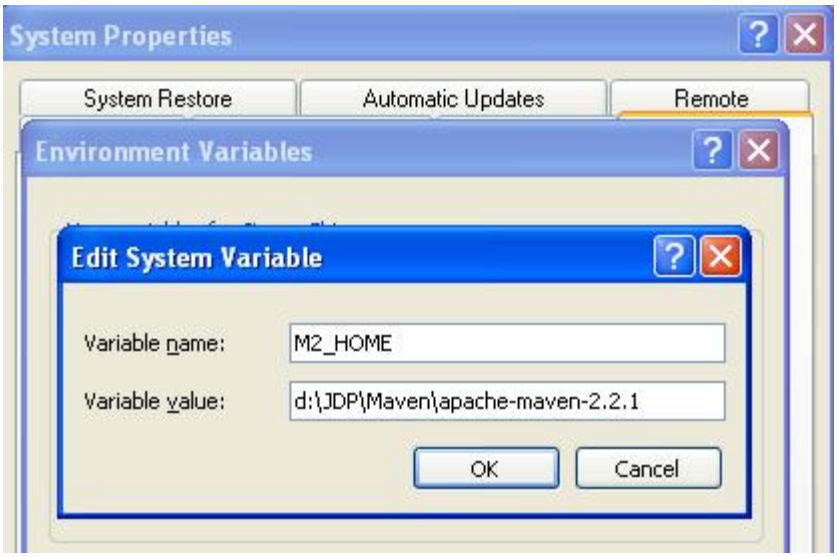
生命周期阶段	描述
validate	验证项目是否正确，以及所有为了完整构建所需要的信息是否可用
generate-sources	生成所有需要包含在编译过程中的源代码
process-sources	处理源代码，比如过滤一些值
generate-resources	生成所有需要包含在打包过程中资源文件
process-resources	复制并处理资源文件至目标目录
compile	编译项目的源代码
process-classes	后处理编译生成的文件 例如对 Java 类进行一些代码增强(bytecode)
generate-test-sources	生成所有包含在测试编译过程的测试源码
process-test-sources	处理测试源码，比如过滤一些值
generate-test-resources	生成测试需要的资源文件
process-test-resources	复制并处理测试资源文件至于测试目录
test-compile	编译测试源码至测试目标目录
test	使用合适的单元测试框架运行测试
prepare-package	在真正的打包之前，执行一些准备打包必要的操作
package	将编译好的代码打包成可分发的格式，如 JAR,WAR,EAR.
pre-integration-test	执行一些在集成测试运行之前需要的动作，如建立执行测试需要的环境
integration-test	处理包并发布至集成测试可以运行的环境
post-integration-test	执行一些在集成环境运行之后需要的动作，如清理集成测试环境
Verify	执行所有的检查，验证包是有效的，符合质量规范
Install	安装包至本地仓库，以备其它的项目做为依赖使用
Deploy	复制最终的包至远程仓库，共享给其它开发人员和项目（通常和一次正式的发布相关）

- **如何执行 Maven**

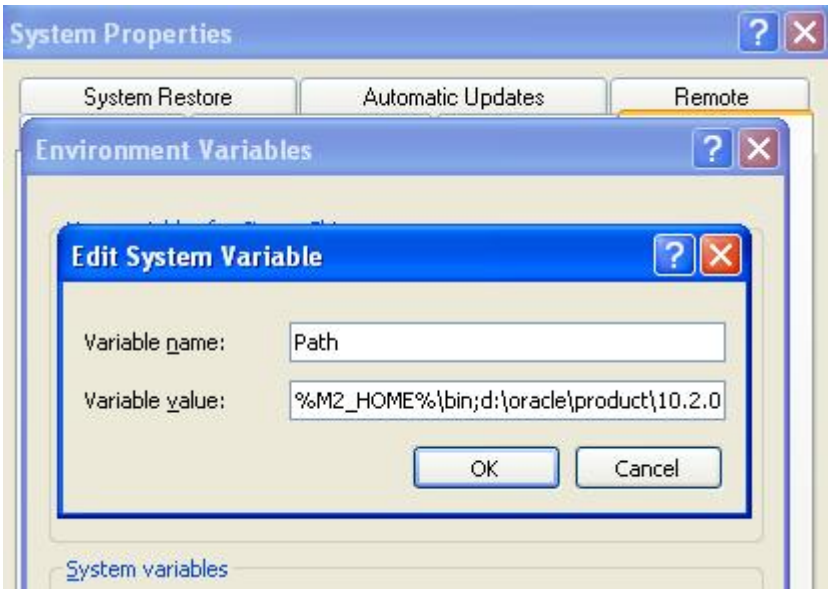
1. **安装 Maven**

（1） 下载 Maven，并解压

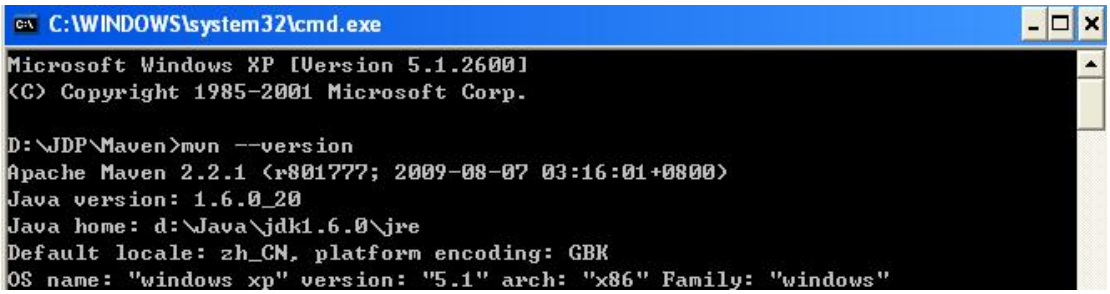
（2） 在系统环境变量中设置 M2_HOME



（3） 将%M2_HOME%/bin;加入至系统环境变量 path 中。



（4） 在 DOS 窗口中，输入 mvn -version，若显示版本号相关信息，则安装成功。



2. 配置 Maven

Maven 的配置是通过%M2_HOME%/conf/setting.xml 来进行配置。

setting.xml 用来配置一些全局性的配置，包括代理服务器的地址，本地的存储库的地址等。

常用的配置包括：

（1） 本地 Maven 存储库的地址（localRepository）

用来定义本地的 Maven 的库的存储地址，即从 Maven 仓库中下载的文件的存储的位

置。

如：

```
<localRepository>d:/Maven/repository</localRepository>
```

（2） Maven 服务器地址（server）

在发布构件至本地的情况下需要。

如：

```
<servers>
    <server>
        <id>nexus-releases</id>
        <username>admin</username>
        <password>admin123</password>
    </server>
    <server>
      <id>nexus-snapshots</id>
      <username>admin</username>
      <password>admin123</password>
    </server>
</servers>
```

（3） 镜像服务器（Mirror）

定义镜像服务器来替代指定的仓库，即通过匹配相关的远程的仓库的 id 从镜像的服务器上查找构件，而不从远程的服务上面查找。

如：

```
<mirrors>
    <mirror>
        <id>nexus</id>
        <mirrorOf>*</mirrorOf>
        <name>Nexus Mirror for all.</name>
        <url>http://127.0.0.1/nexus/content/groups/public</url>
    </mirror>
</mirrors>
```

（4） 配置（Profile）

Profile 用来定义仓库的地址和插件仓库的地址，用来发布、获取构件。

如：

```
<profiles>
   <profile>
     <id>nexus</id>
     <repositories>
        <repository>
          <id>nexus</id>
          <name>本地开发库</name>

<url>http://127.0.0.1/nexus/content/groups/public</url>
          <layout>default</layout>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </repository>
      </repositories>
     <pluginRepositories>
        <pluginRepository>
          <id>nexus</id>

<url>http://127.0.0.1/nexus/content/groups/public</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
</profiles>
```

（5） 激活配置

指定哪些配置被激活。

如：

```
<activeProfiles>
    <activeProfile>nexus</activeProfile>
</activeProfiles>
```

3. 使用 Maven

（1） 创建 Maven 项目可以通过执行相应的 maven 命令来进行。

执行方式包括：mvn pluginId:goalId 或者是 mvn phase。

mvn pluginId:goalId 是执行相关的插件的目标。

如：

```
mvn -archetype:create -DgroupId=com.vaalhaai -DartifactId=common
-DpackageName=com.vaalhaai.common
```

将创建一个新的项目。

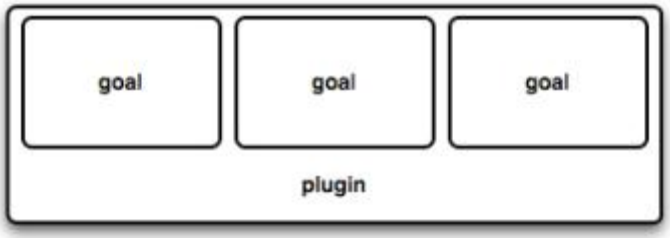
其中：

mvn 是 Maven 的命令。

archetype:create 称为 Maven 的目标。archetype 实际上是一个插件 id，create 称之为目标 id。

-Dname=value 是将会传入到目标中的参数，使用-D 属性的形式，类似于通过命令行向虚拟机传递系统属性。

一个 Maven 插件是一组或者多个目标的集合。



mvn phase 是执行 maven 的生命周期的阶段

如：

```
mvn package
```

将执行打包的操作。

可以将插件的目标绑定到 Maven 的生命周期上，每个阶段可以绑定零个或者是多个目标。

Maven 在执行生命周期的时候，会有序的执行前置的所有阶段，直到命令指定的生命周期。

- **Maven 的依赖管理**

一个复杂项目会包含有很多的依赖，也可能是包含有依赖于其它构件的依赖。Maven 支持依赖传递（transitive dependencies），如果你的项目依赖于一个库，而这个库又依赖于多个其它的库，你只需要加上你依赖的库即可，Maven 会自动将依赖的库的依赖加入你的项目中。

Maven 中的对于项目的依赖的配置在 POM.xml 中，通过配置 dependency 来指定

如：

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.8.1</version>
        <scope>test</scope>
```

```
    </dependency>
</dependencies>
```

添加 junit 依赖。

其中：

groupId,artifactId 指定依赖库的 groupId 和 artifactId；

version 指定构件的版本；

scope 指定库依赖的范围，test 指在测试阶段需要（不会打包到生产环境中去）。

scope	描述
compile	缺省值，适用于所有阶段，会随着项目一起发布
test	只在测试时使用，用于编译和运行测试代码。不会随项目发布
runtime	只在运行时使用，如 JDBC 驱动，适用运行和测试阶段
provided	类似 compile，期望 JDK、容器或使用者会提供这个依赖。如 servlet.jar
system	类似 provided，需要显式提供包含依赖的 jar，Maven 不会在 Repository 中查找它
import	它只使用在<dependencyManagement>中，表示从其它的 pom 中导入 dependency 的配置

二、 Eclipse

Eclipse 作为开源的集成开发环境，是目前使用最广泛的 IDE。

三、 WTP

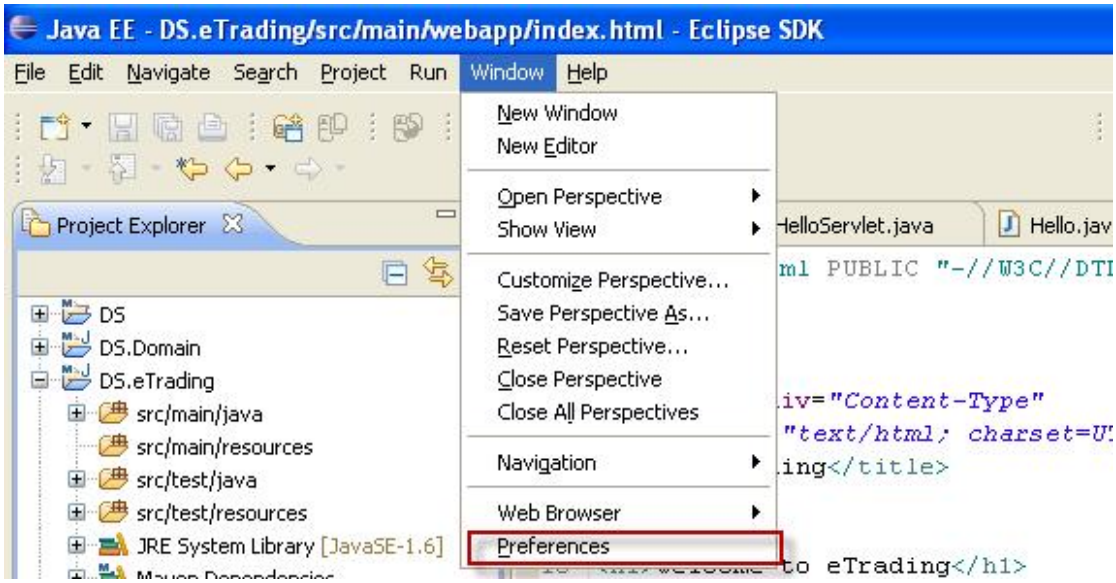
WTP（Web Tools Platform）是基于 Eclipse 开发 Web 组件的插件。

四、 m2eclipse

m2eclipse 为 Eclipse 提供了与 Maven 的集成，提供了简易的使用 Maven 的方式。

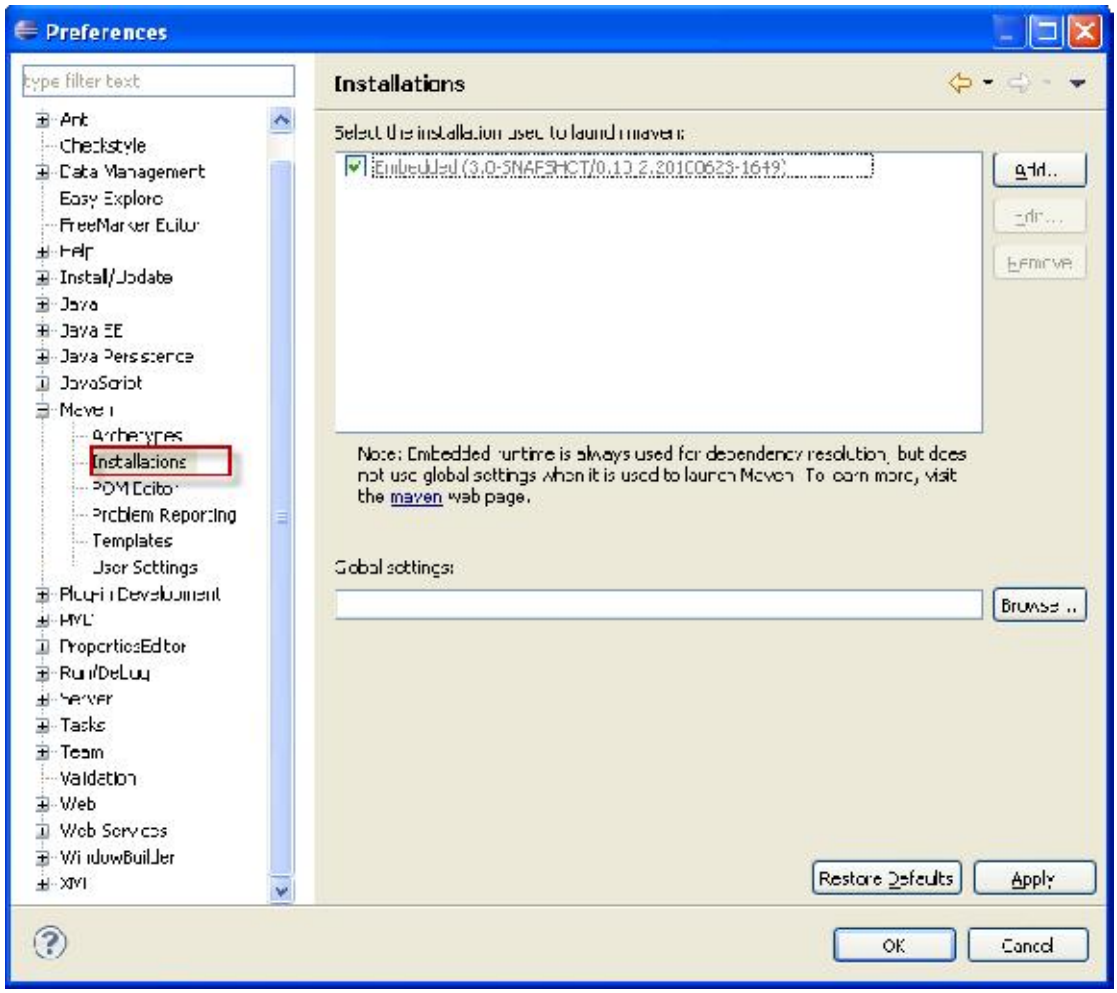
- **配置 m2eclipse**

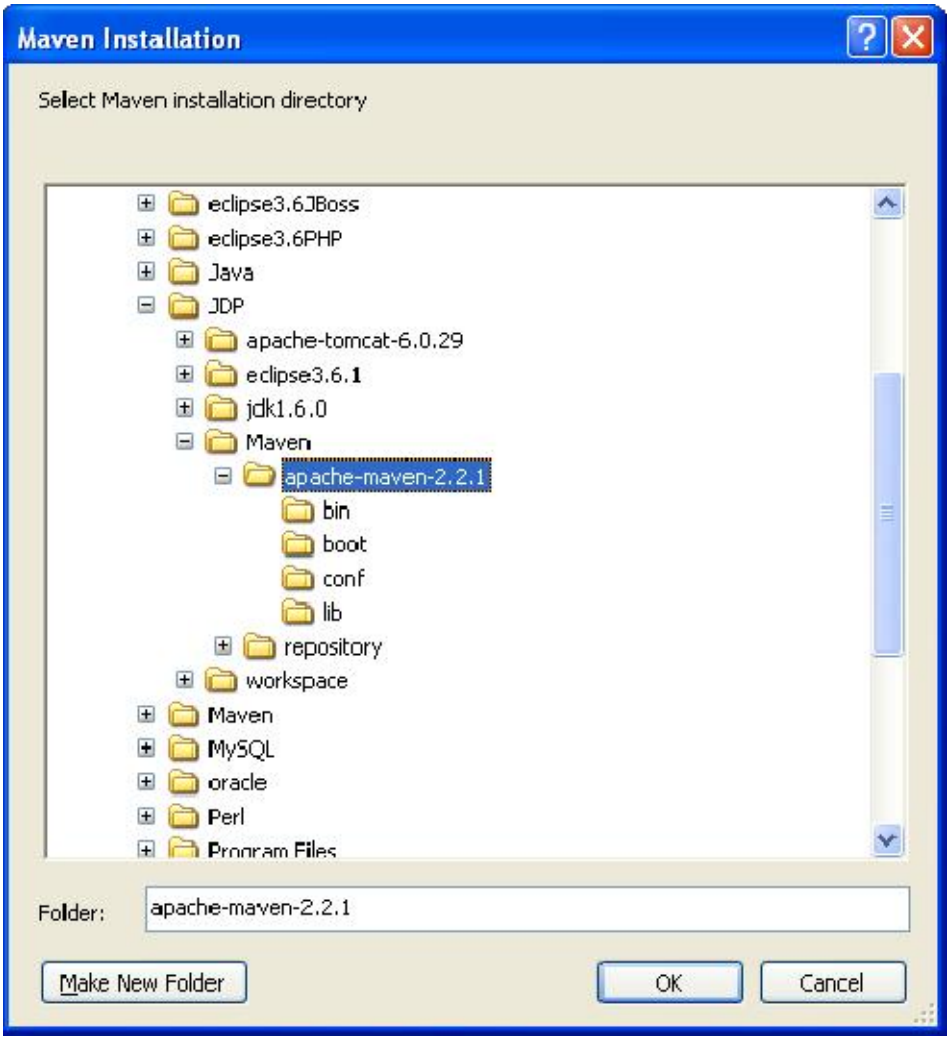
  Window → Preferences

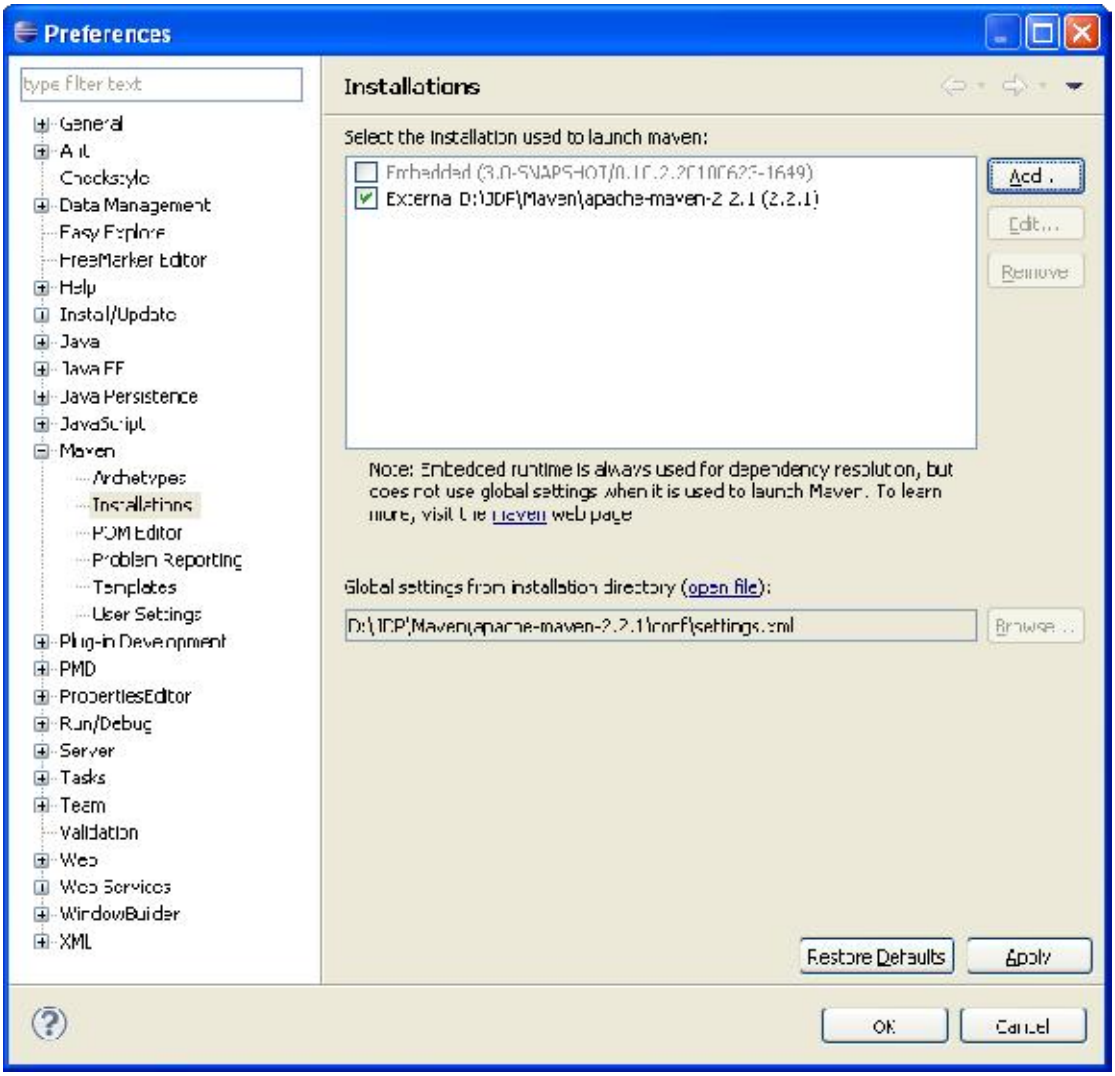


1． 配置 Maven 安装路径

Maven → Installations → Add,选择本地的 Maven 安装路径。

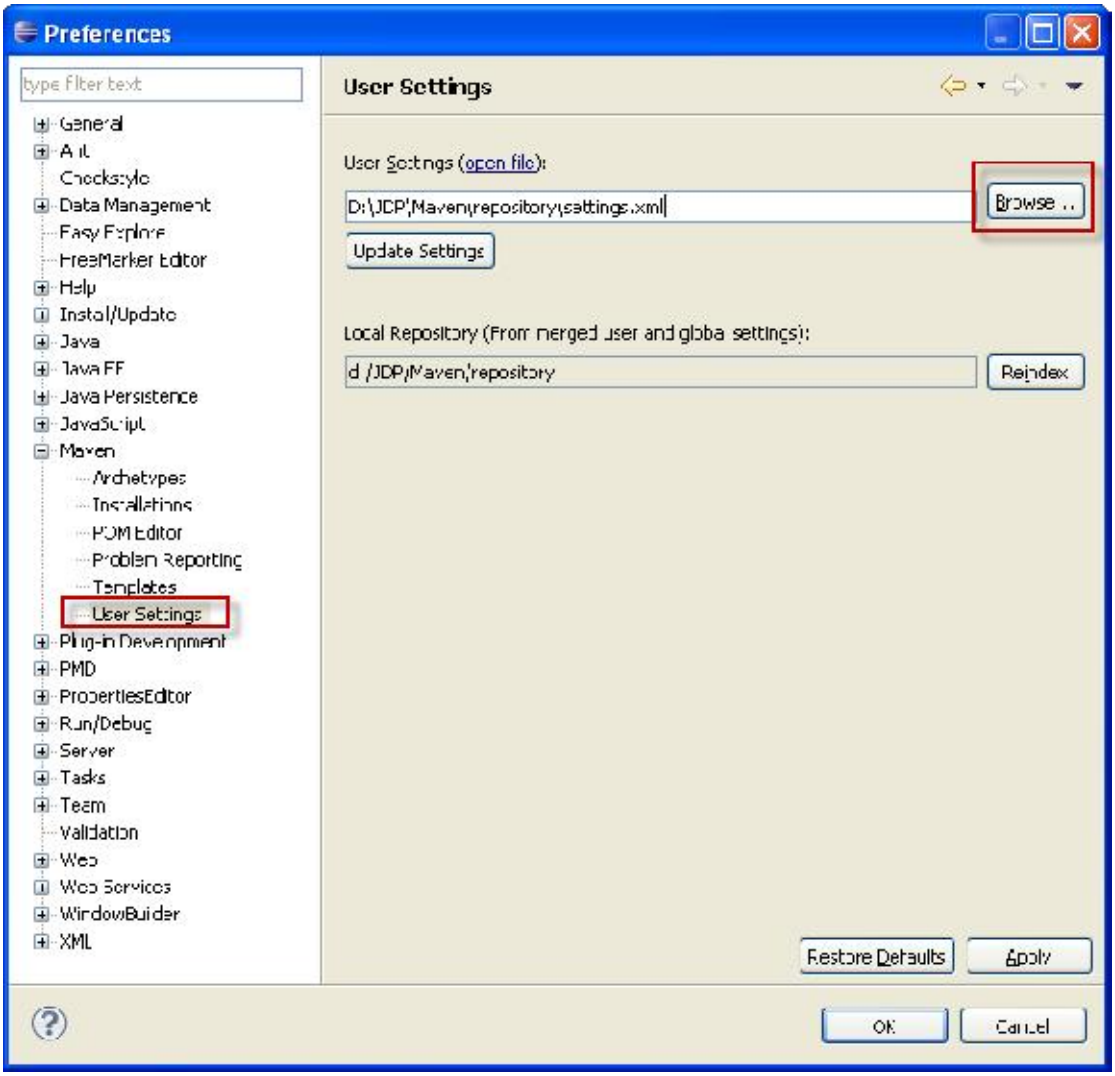



Maven Installation
Select Maven installation directory
eclipse3.6JBoss
eclipse3.6PHP
Java
JDP
apache-tomcat-6.0.29
eclipse3.6.1
jdk1.6.0
Maven
apache-maven-2.2.1
bin
boot
conf
lib
repository
workspace
Maven
MySQL
oracle
Perl
Program Files
Folder:
apache-maven-2.2.1
Make New Folder
OK
Cancel



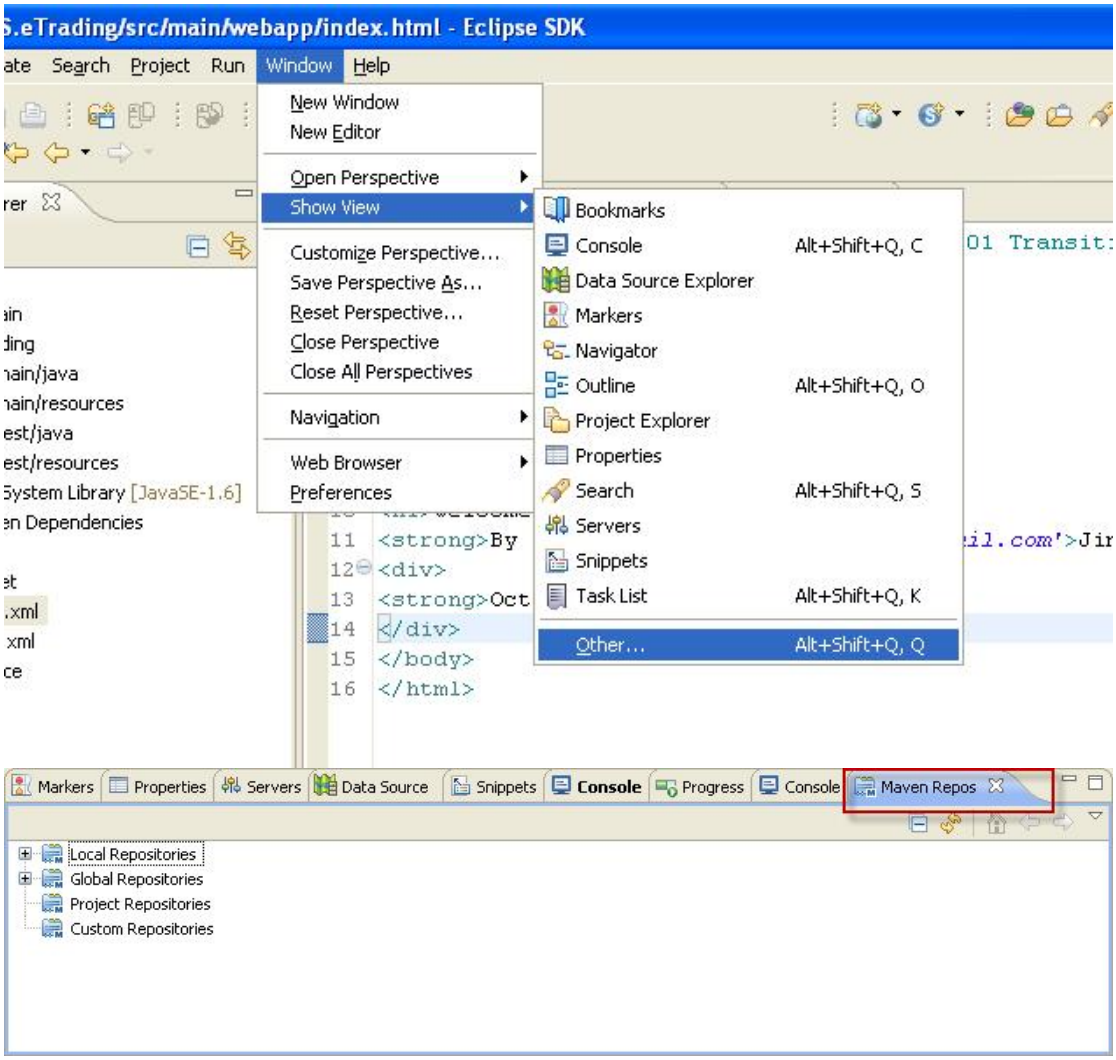
2. 配置 Maven 的 User setting.

Maven → User Settings → Brower…,选择 settings.xml.(settings.xml 配置参见配置 Maven)



3. 打开 Maven Repositories 视图

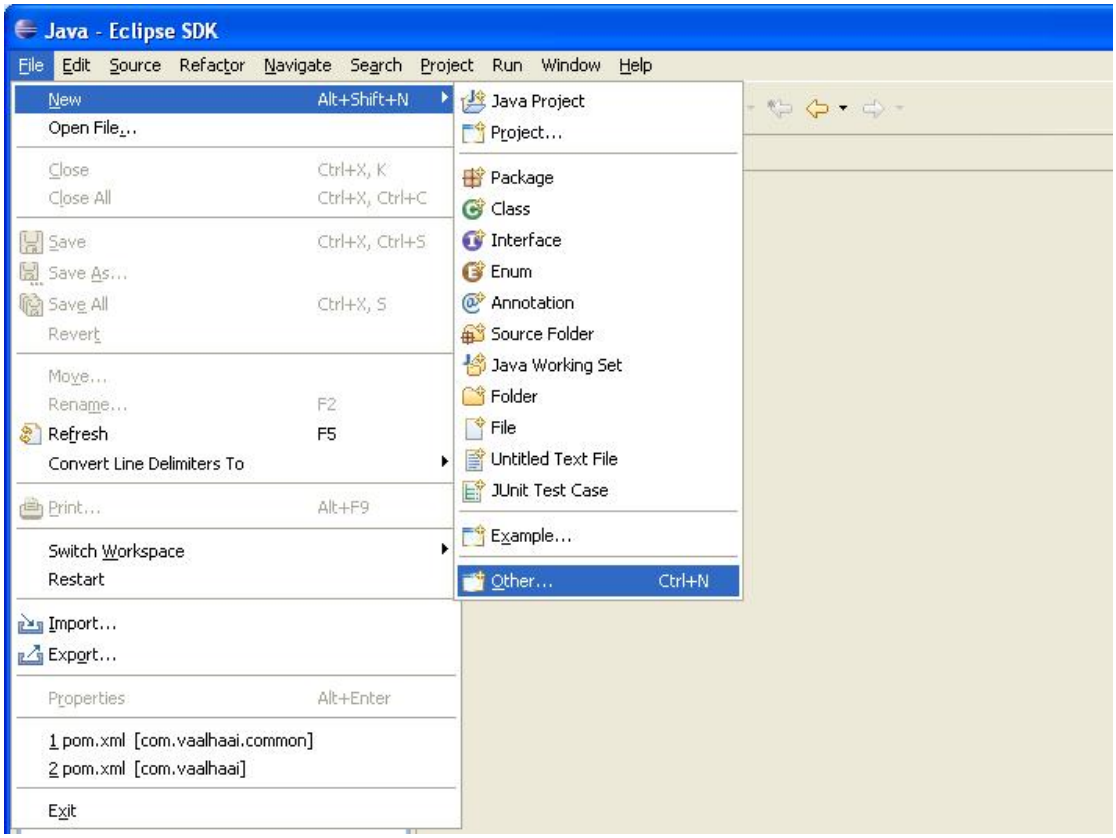
Window → Show View → Other,选择 Maven → Maven Repositories.



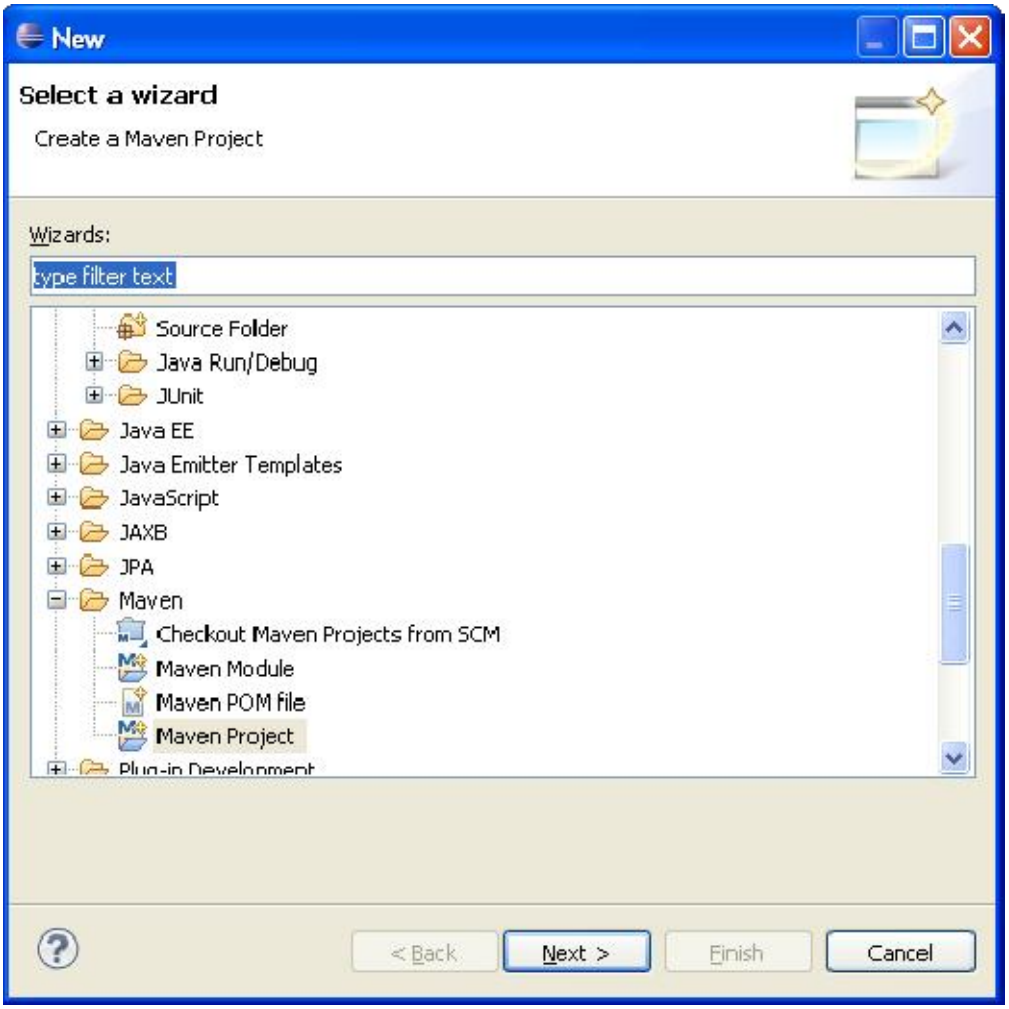
- **使用 m2eclipse 进行开发**

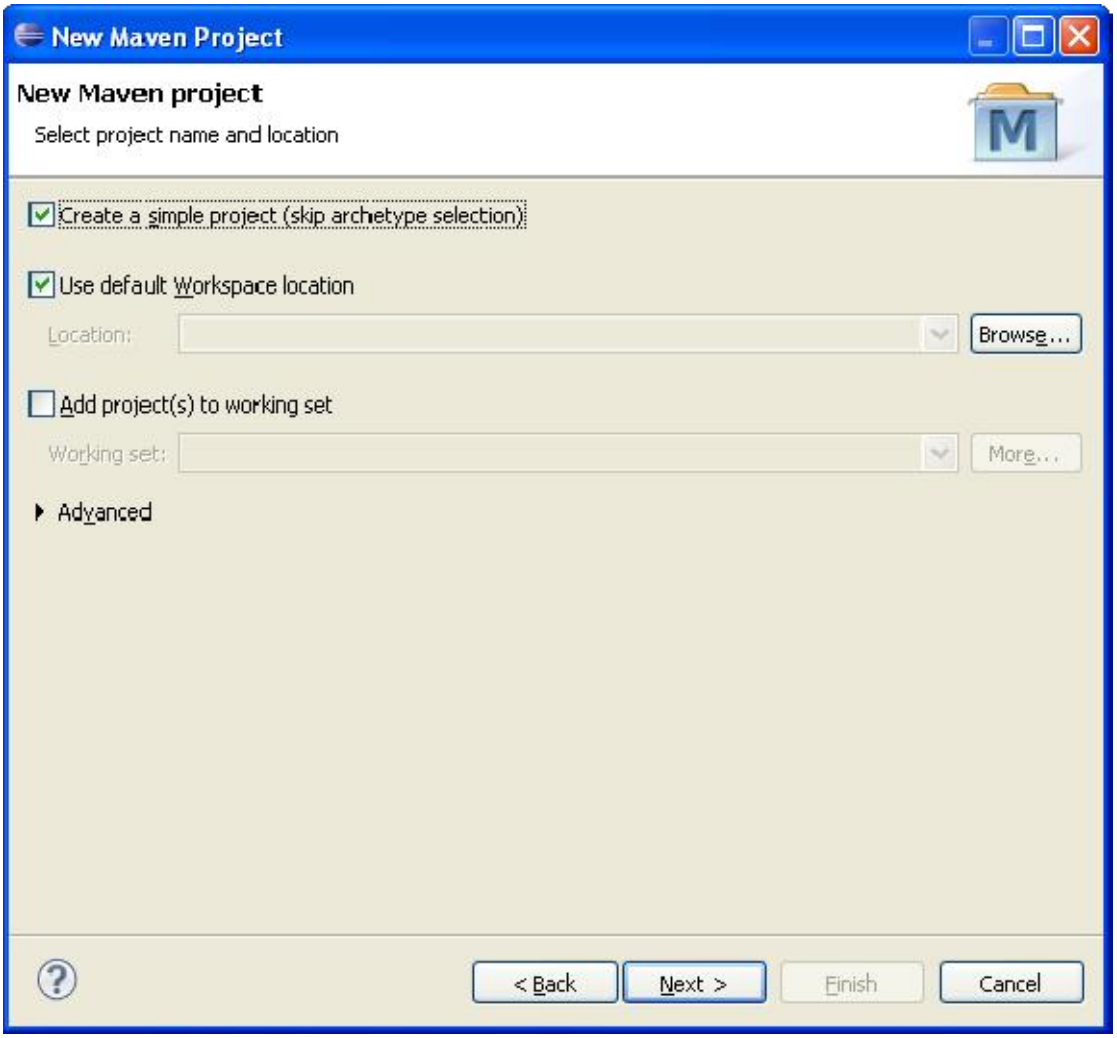
1. 创建 Maven 项目

当安装了 m2eclipse 后，在选择 File→New→Other 后，在出现在向导中，会出现 Maven 的相关的。选择 Maven Project，就可以创建 Maven 项目。

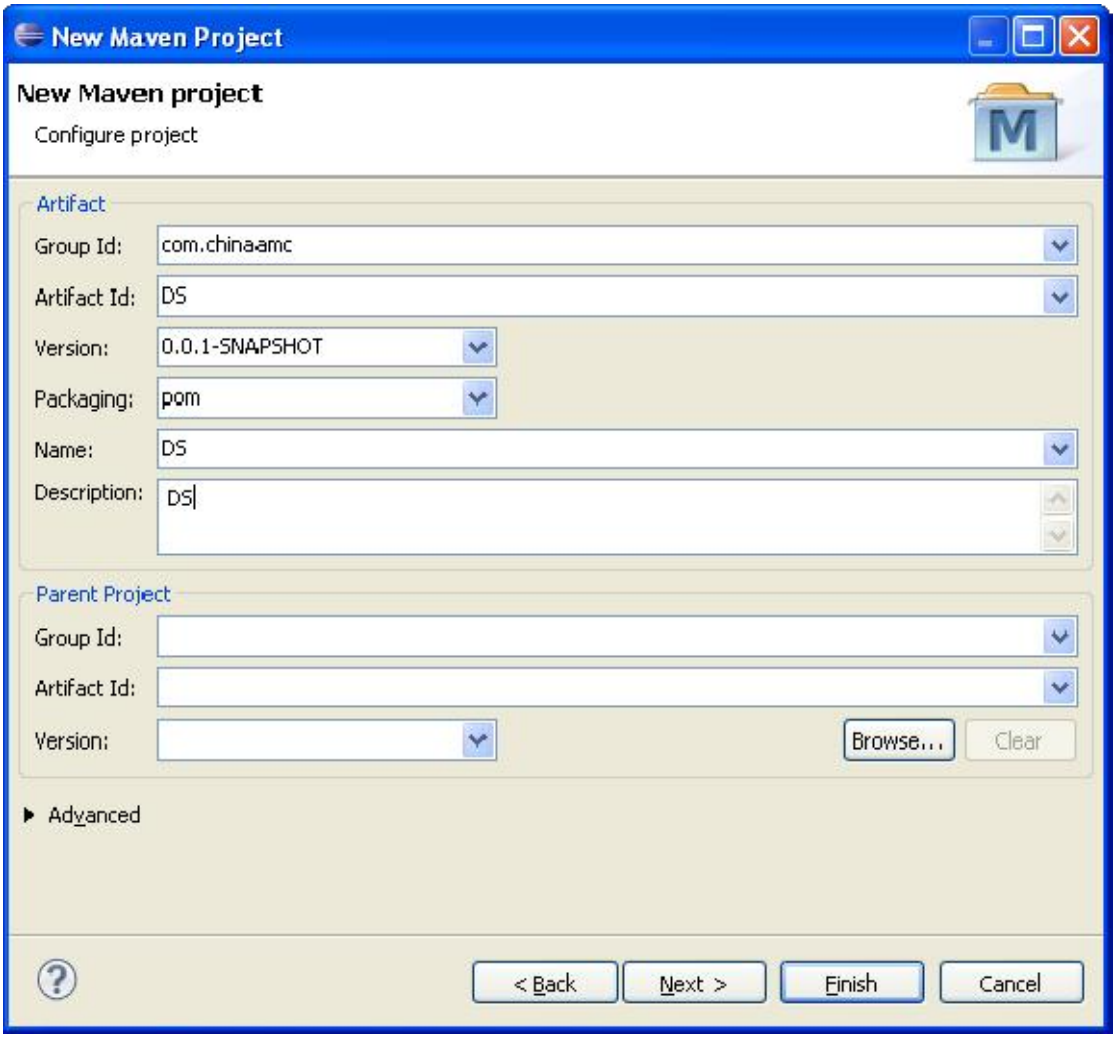

Java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+X, K
Close All Ctrl+X, Ctrl+C
Save Ctrl+X, Ctrl+S
Save As...
Save All Ctrl+X, S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Alt+F9
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 pom.xml [com.vaalhaai.common]
2 pom.xml [com.vaalhaai]
Exit
Java Project
Project...
Package
Class
Interface
Enum
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
JUnit Test Case
Example...
Other... Ctrl+N

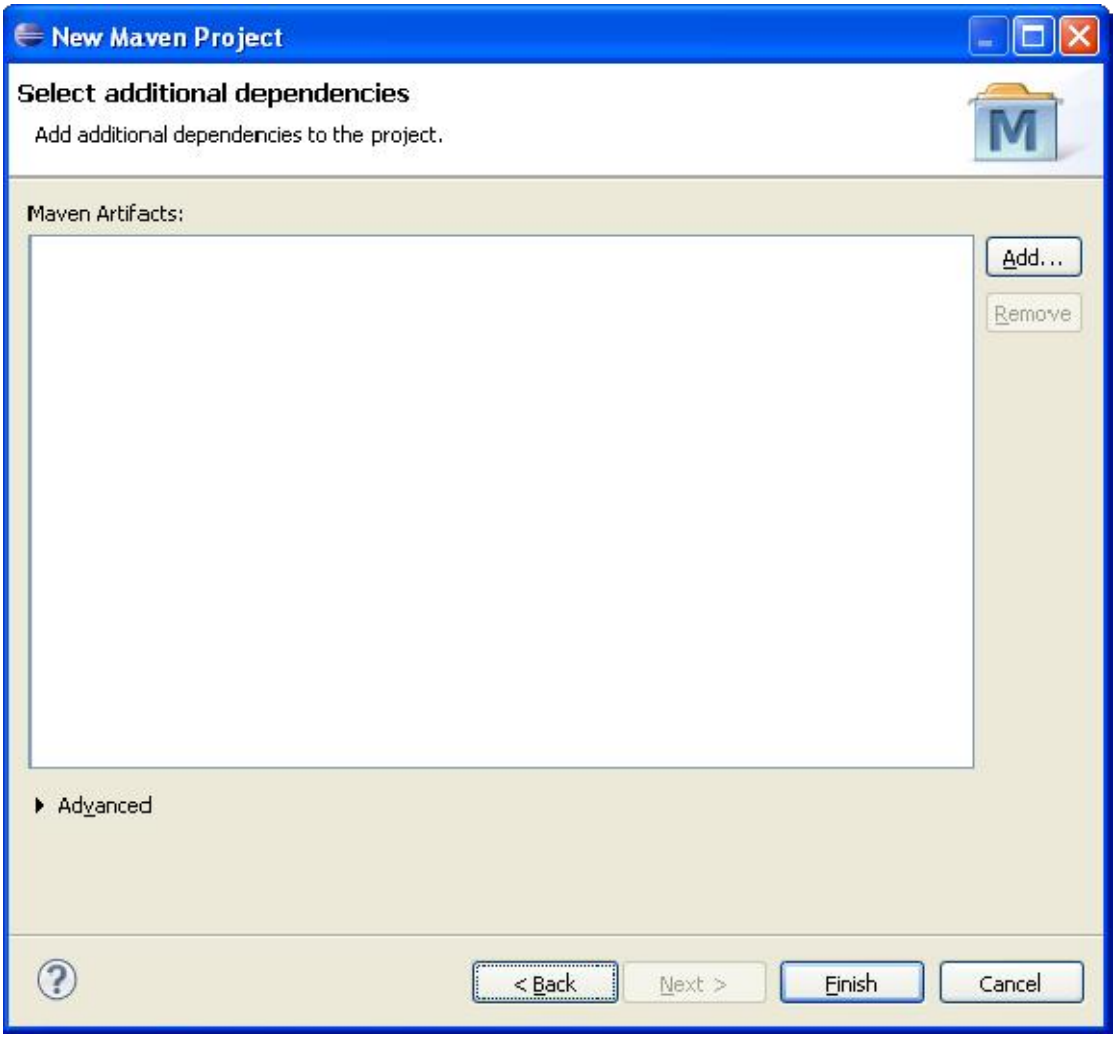
选择 Maven 项目


New
Select a wizard
Create a Maven Project
Wizards:
type filter text
Source Folder
Java Run/Debug
JUnit
Java EE
Java Emitter Templates
JavaScript
JAXB
JPA
Maven
Checkout Maven Projects from SCM
Maven Module
Maven POM file
Maven Project
< Back
Next >
Finish
Cancel
选择简单的项目

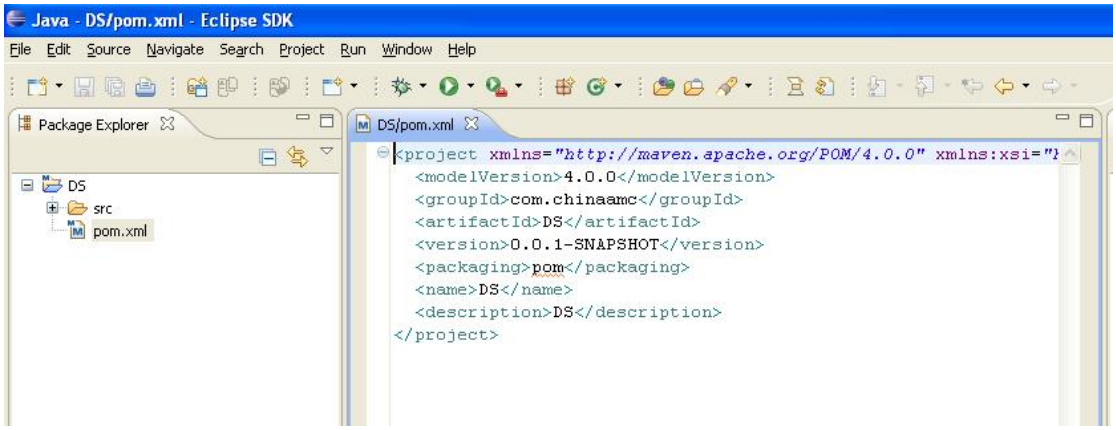

New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

输入 GAV 信息

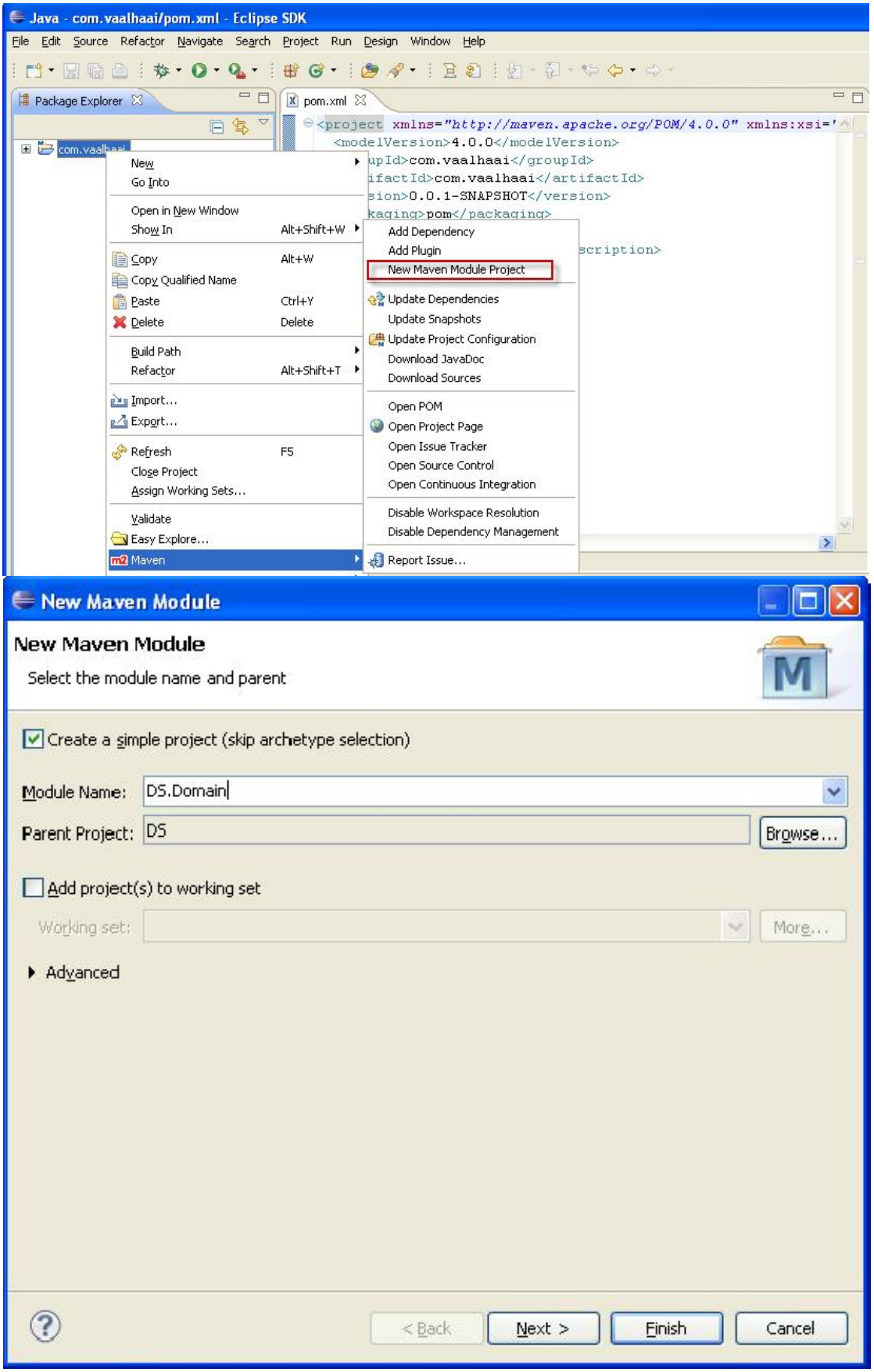

New Maven Project
New Maven project
Configure project
Artifact
Group Id:
com.chinaamc
Artifact Id:
DS
Version:
0.0.1-SNAPSHOT
Packaging:
pom
Name:
DS
Description:
DS
Parent Project
Group Id:
Artifact Id:
Version:
Browse...
Clear
Advanced
< Back
Next >
Finish
Cancel

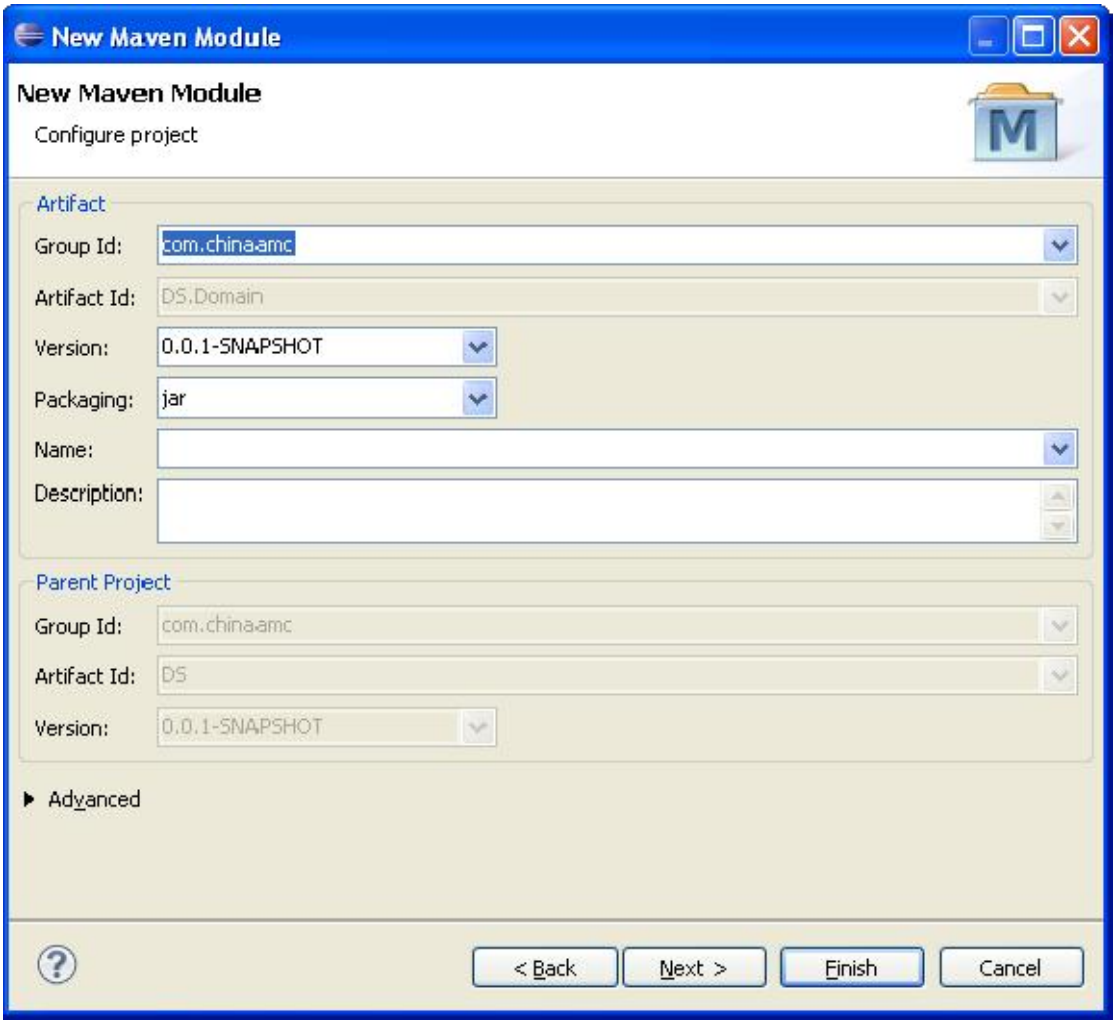


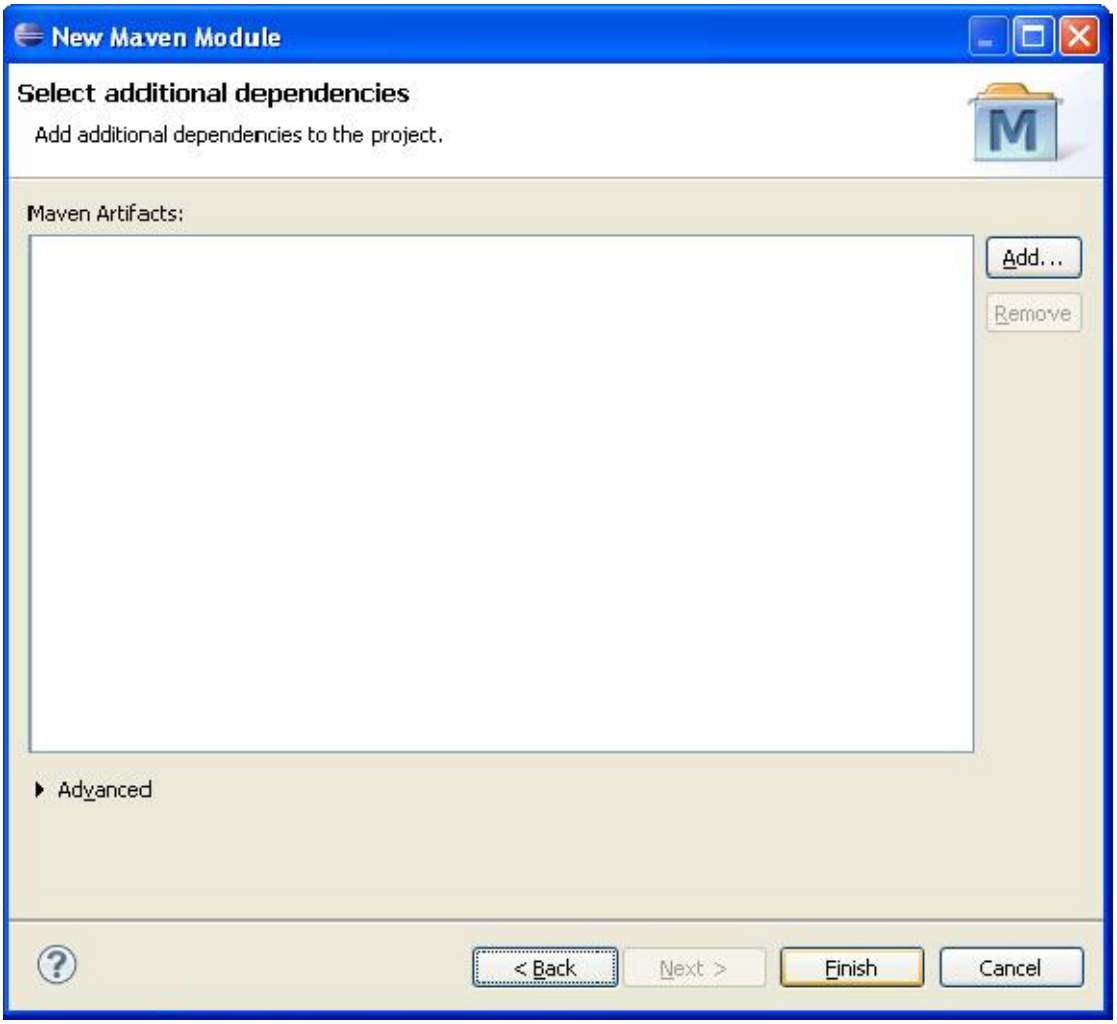
如：我们通过 m2clipse 创建了一个 POM 项目



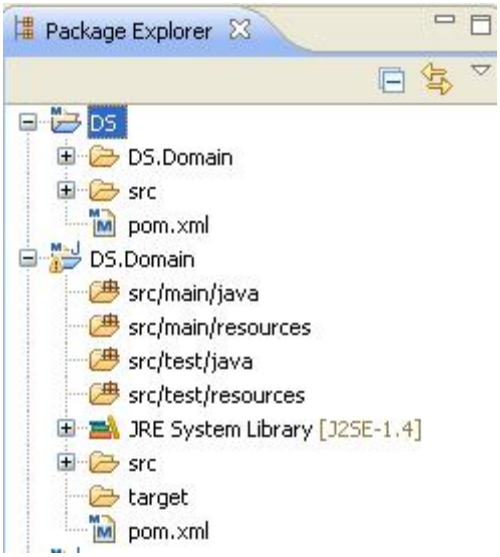
若创建 POM 项目的 Module 项目，在右键菜单中选择 Maven→New Maven Module Project


Java - com.vaalhaai/pom.xml - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Design Window Help
Package Explorer
pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi='
<modelVersion>4.0.0</modelVersion>
upId>com.vaalhaai</groupId>
ifactId>com.vaalhaai</artifactId>
sion>0.0.1-SNAPSHOT</version>
kaging>pom</packaging>
scription>
New
Go Into
Open in New Window
Show In Alt+Shift+W
Copy Alt+W
Copy Qualified Name
Paste Ctrl+Y
Delete Delete
Build Path
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Validate
Easy Explore...
Maven
Add Dependency
Add Plugin
New Maven Module Project
Update Dependencies
Update Snapshots
Update Project Configuration
Download JavaDoc
Download Sources
Open POM
Open Project Page
Open Issue Tracker
Open Source Control
Open Continuous Integration
Disable Workspace Resolution
Disable Dependency Management
Report Issue...
New Maven Module
New Maven Module
Select the module name and parent
Create a simple project (skip archetype selection)
Module Name: DS.Domain
Parent Project: DS
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

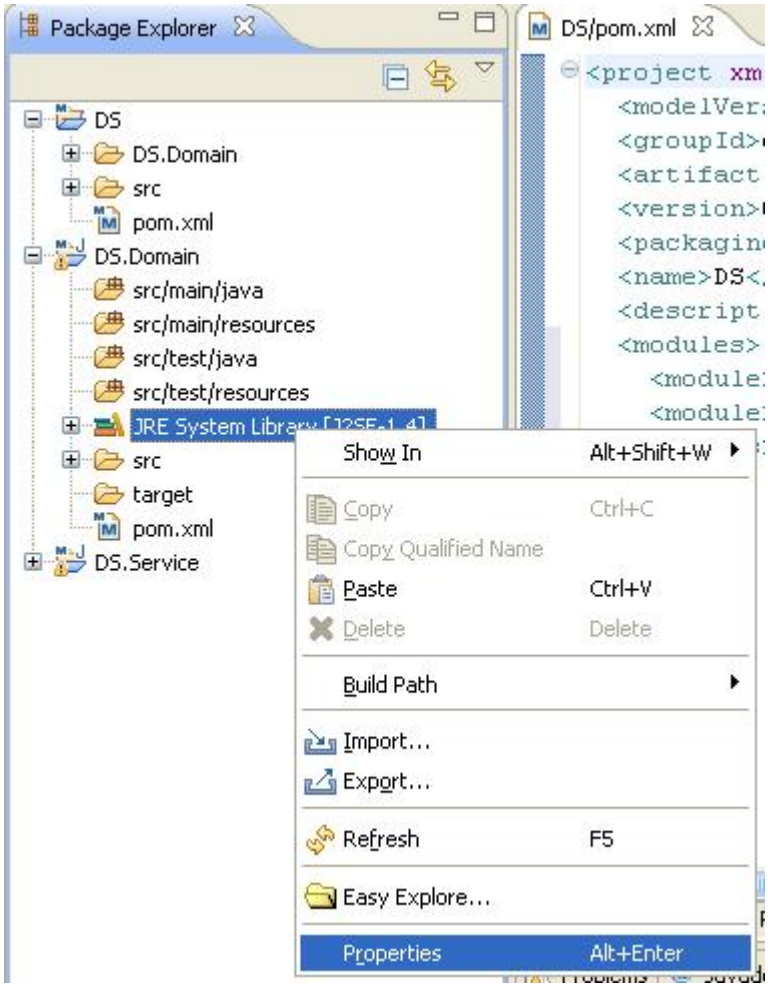

New Maven Module
New Maven Module
Configure project
Artifact
Group Id:
com.chinaamc
Artifact Id:
DS.Domain
Version:
0.0.1-SNAPSHOT
Packaging:
jar
Name:
Description:
Parent Project
Group Id:
com.chinaamc
Artifact Id:
DS
Version:
0.0.1-SNAPSHOT
Advanced
< Back
Next >
Finish
Cancel



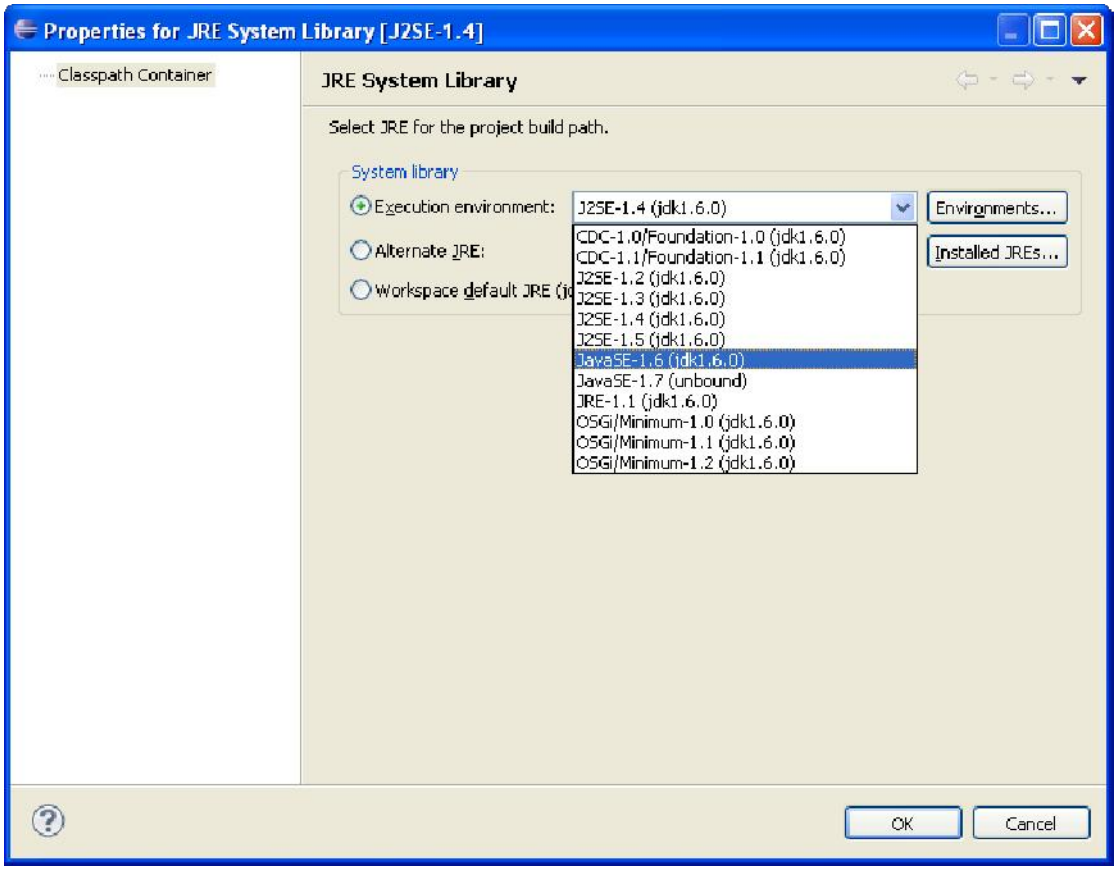
通过向导创建了一个新项目如下：



由于默认的创建为运行环境为 `1.4`，我们需要修改为 `1.6`.

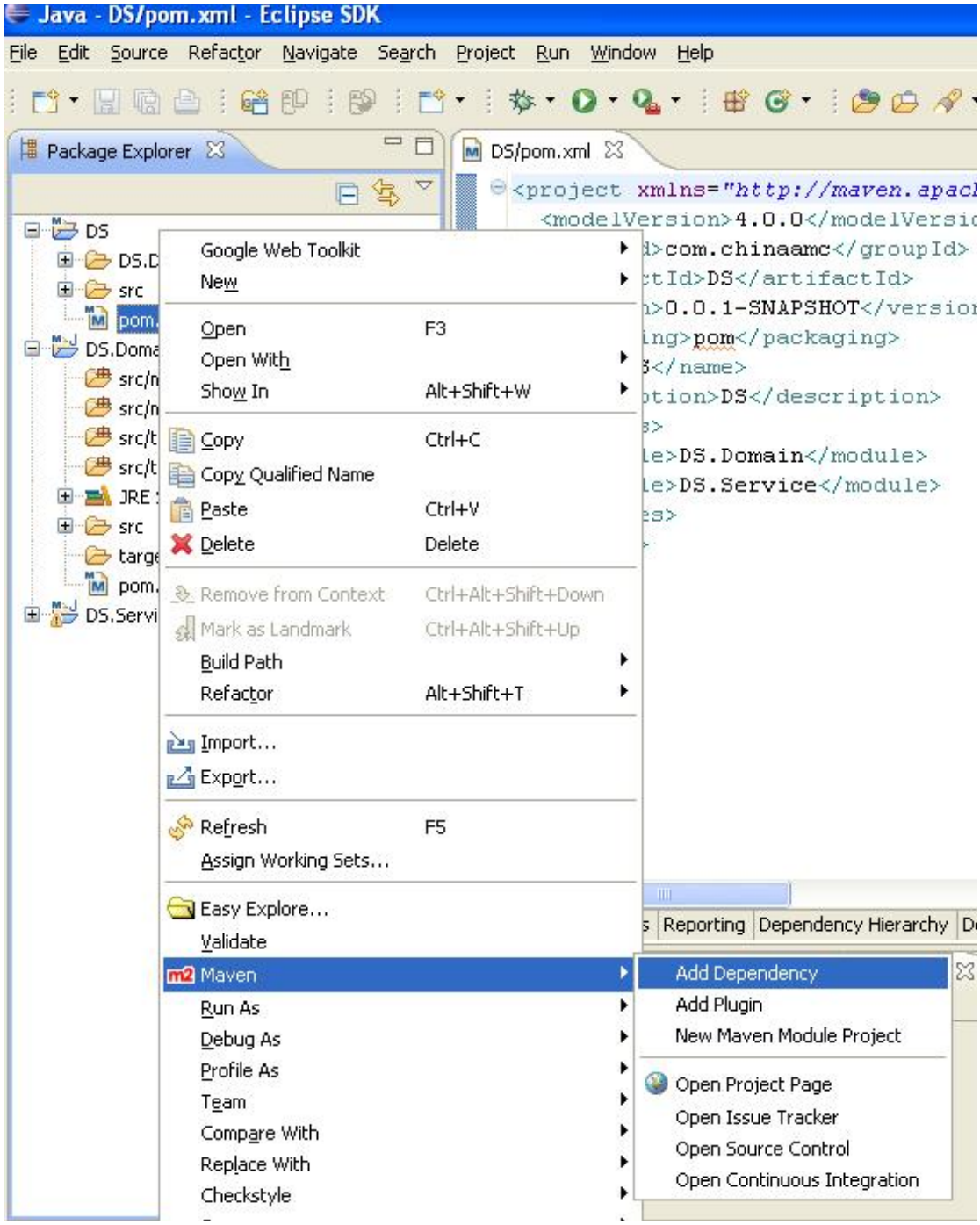

Package Explorer
DS
DS.Domain
src
pom.xml
DS.Domain
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library
src
target
pom.xml
DS.Service
DS/pom.xml
<project xm
<modelVer
<groupId>
<artifact
<version>
<packagin
<name>DS<
<descript
<modules>
<module
<module
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Import...
Export...
Refresh F5
Easy Explore...
Properties Alt+Enter

修改运行环境为 `1.6`

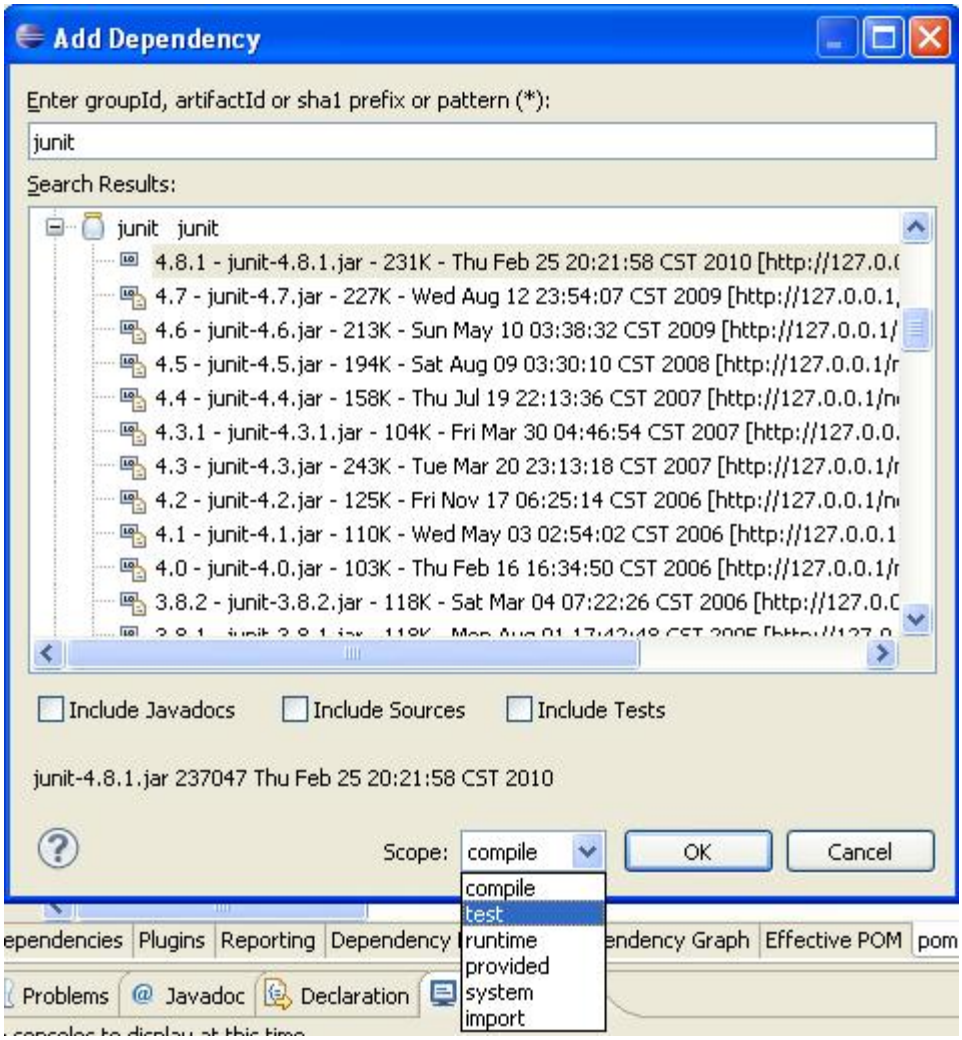


2. 添加依赖

通过右键菜单 `Maven→Add dependency` 来添加依赖库。

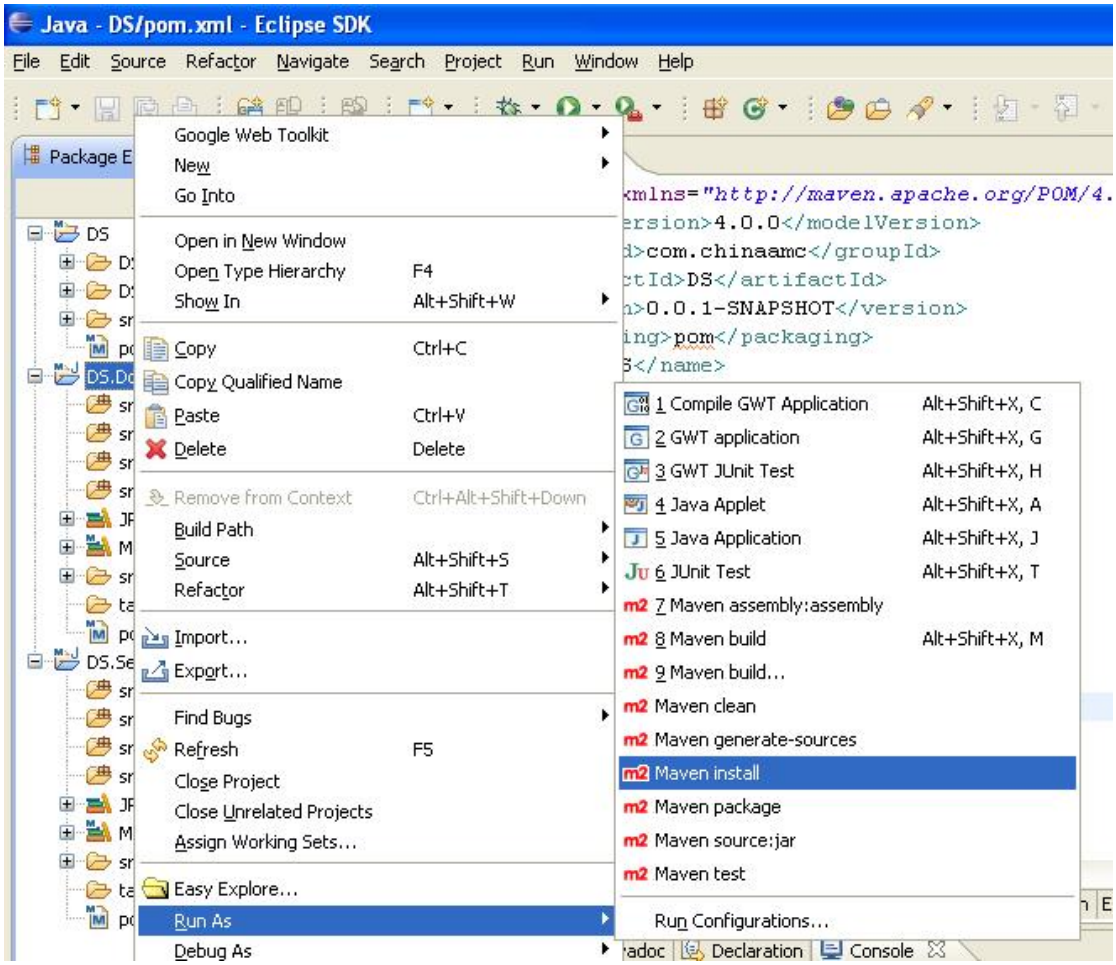


输入依赖库的名称，选择构件，并选择 `scope`。



3. 运行项目

通过右键菜单 `Run as`→ 选择相应目标来完成



1、

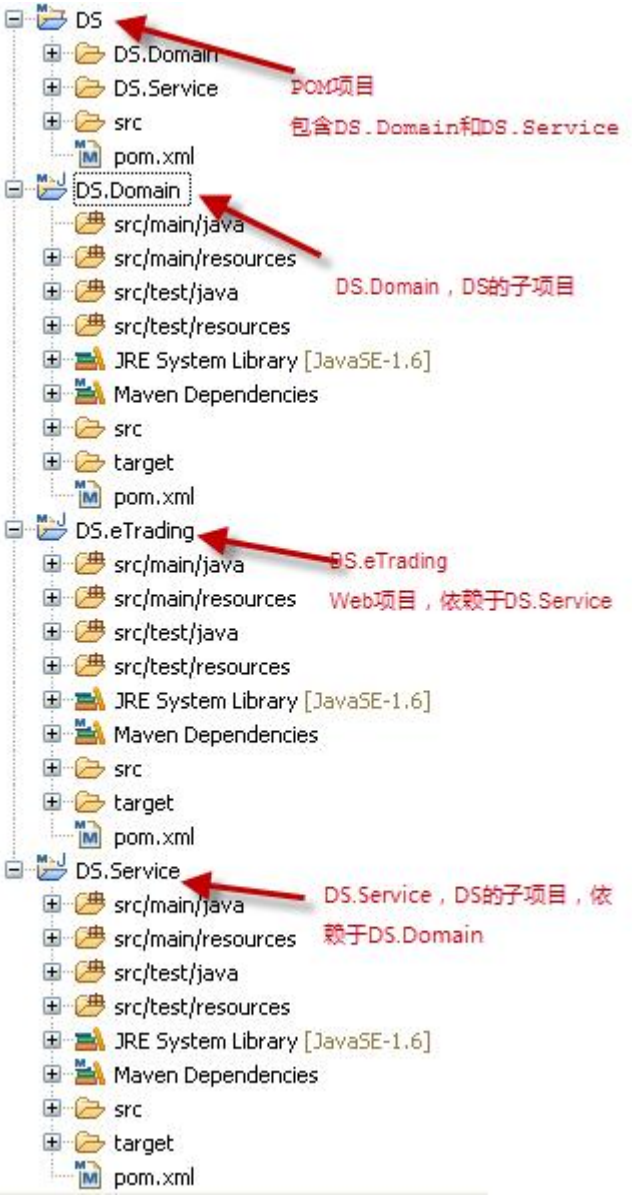
五、 如何进行 Maven Java EE 项目的开发调试

1. 项目划分

依据 Maven 的最佳实践，对于大型项目，我们一般按照分层进行划分。

创建一个 Maven 的 POM 项目，然后在里面再创建子模块。

一个典型的 Java EE 项目结构如下：



建立 POM 父项目 DS，包含二个子项目 DS.Domain(打包方式为 jar)以及 DS.Service(打包方式为 jar)。

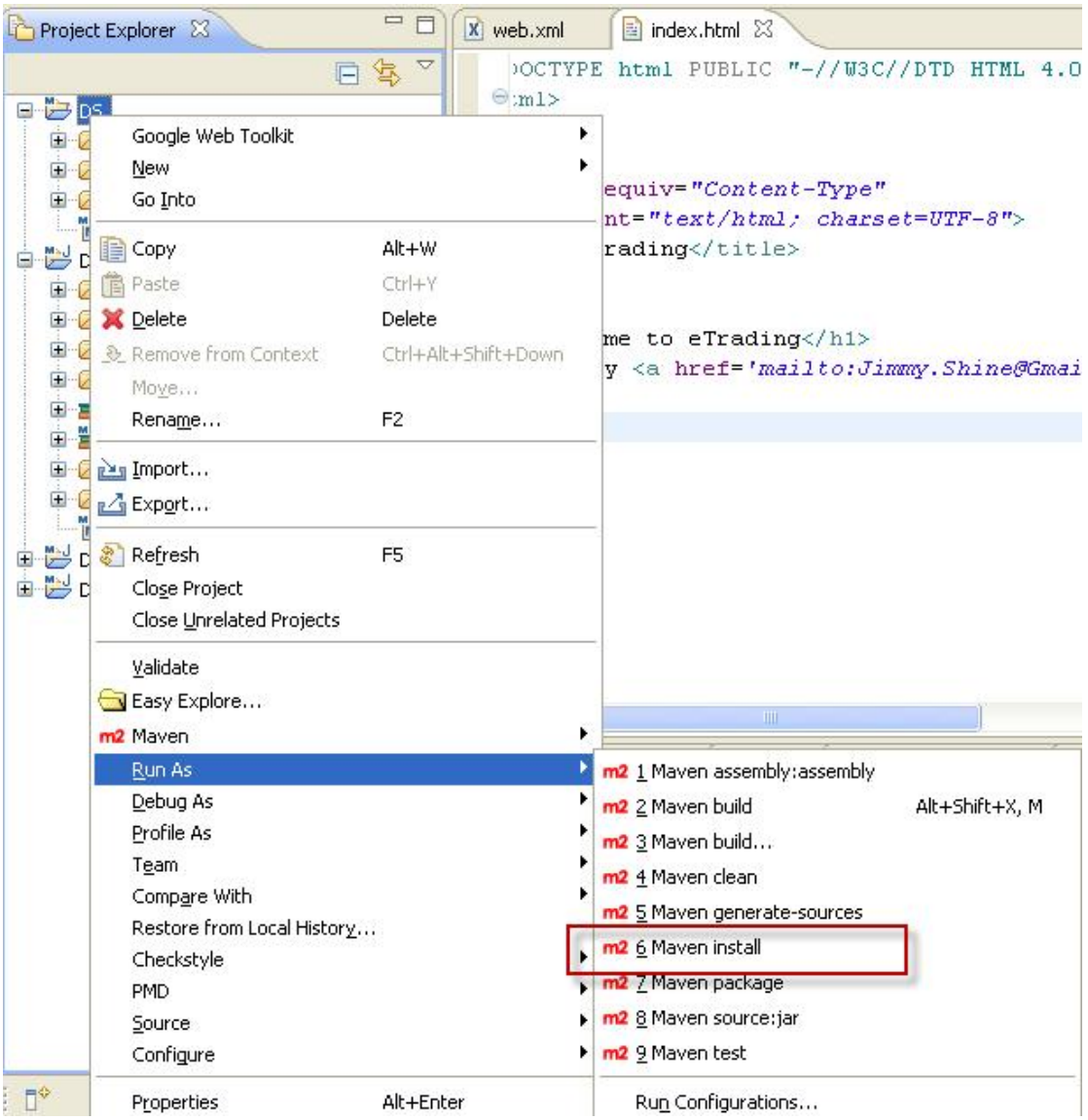
Service 依赖于 Domain.

独立的 Maven 项目 DS.eTrading(打包方式为 war)，依赖于 DS.Service.

2. 如何运行项目

(1) mvn install DS

在 POM 项目点击右键，Run As→ Maven install



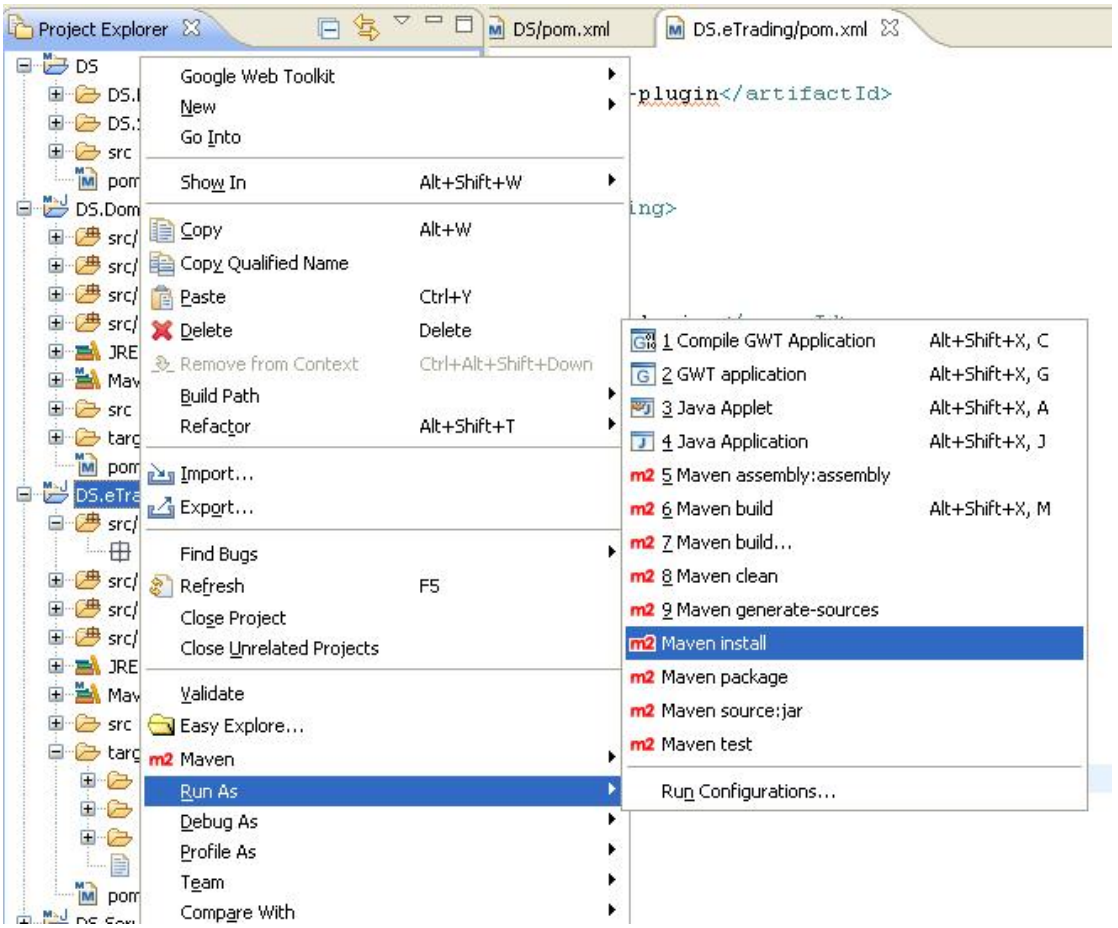
（2） mvn install eTrading

为了保证版本号不至于影响至调试，也便于访问，我们修改打包后的文件名为 eTrading。在 POM.xml 中增加 build 的子元素 finalName。

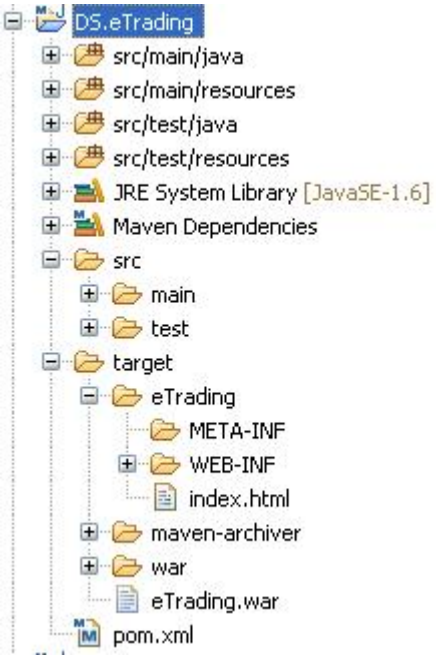
如下：

```
<finalName>eTrading</finalName>
```

在 ds.eTrading 项目点击右键，Run As→ Maven install

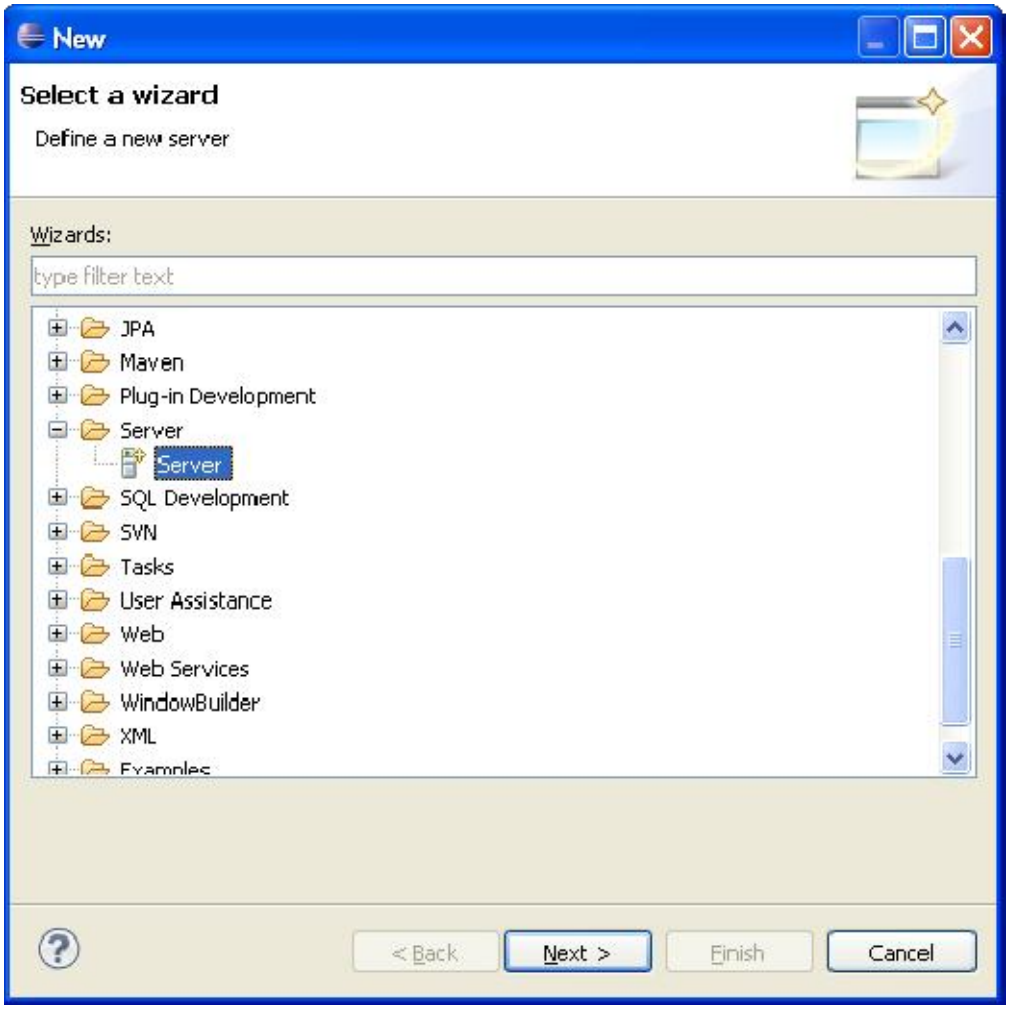


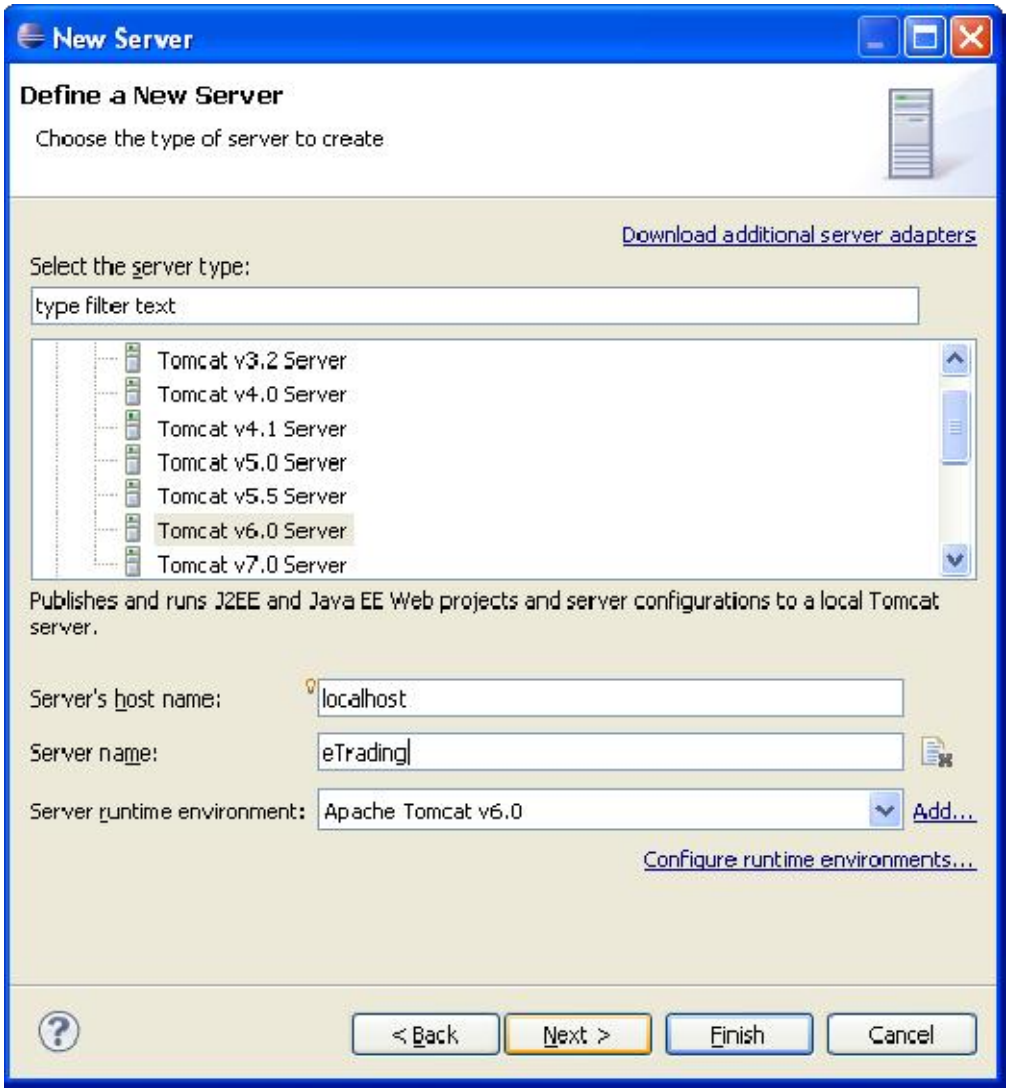
可以看到目录如下：

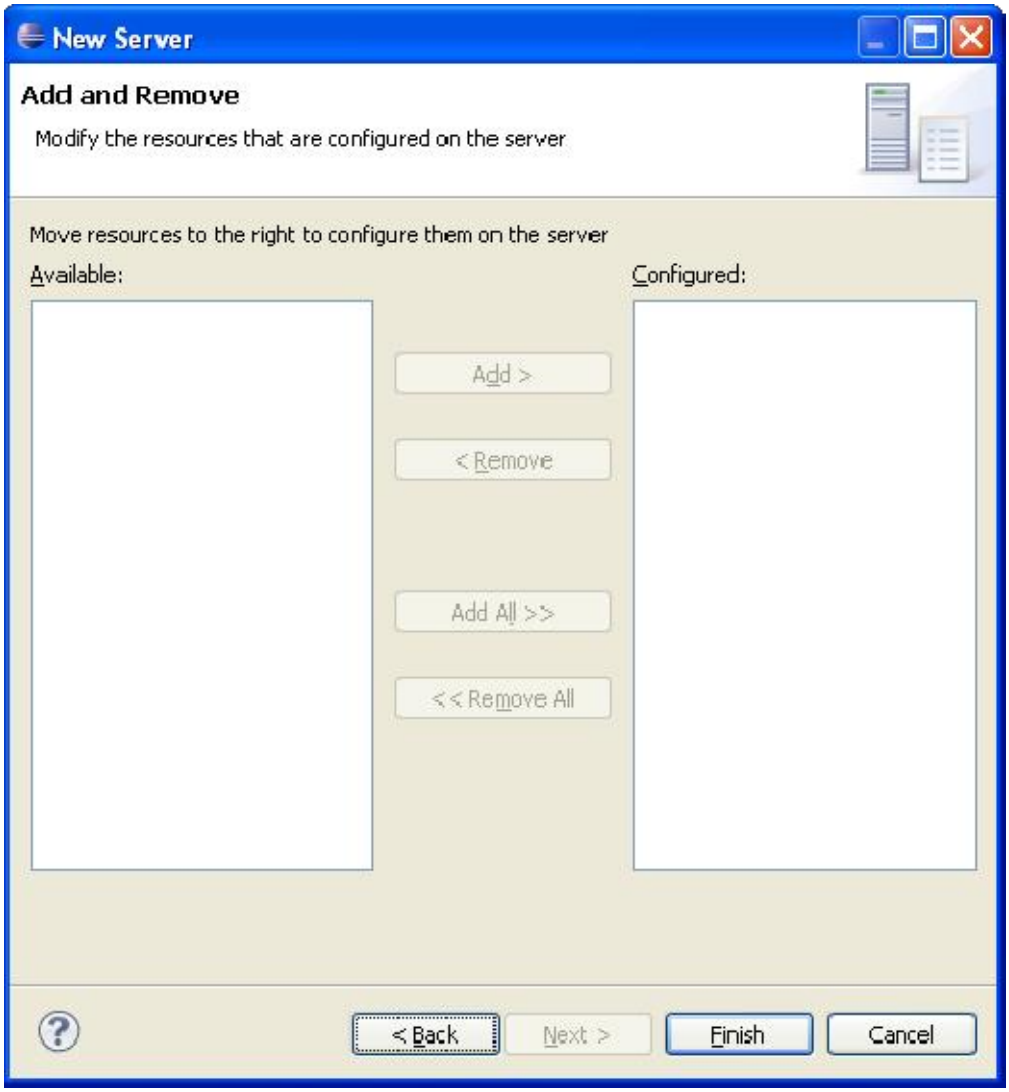


（3） 新建 WTP Server

File → New → Other,选择 Server, Next,修改 server name, Finish

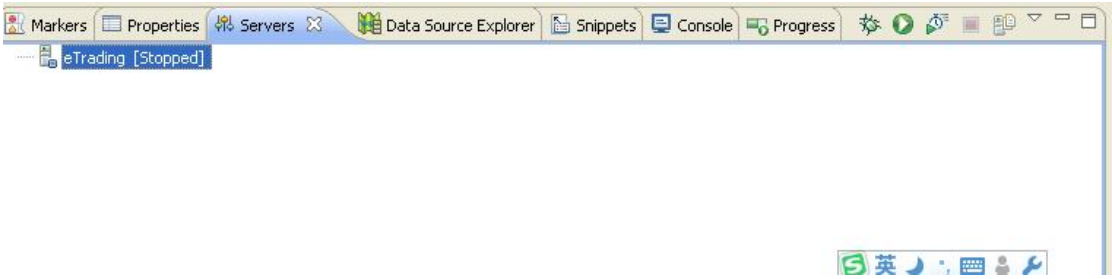

New
Select a wizard
Define a new server
Wizards:
type filter text
JPA
Maven
Plug-in Development
Server
Server
SQL Development
SVN
Tasks
User Assistance
Web
Web Services
WindowBuilder
XML
Examples
< Back
Next >
Finish
Cancel


New Server
Define a New Server
Choose the type of server to create
Download additional server adapters
Select the server type:
type filter text
Tomcat v3.2 Server
Tomcat v4.0 Server
Tomcat v4.1 Server
Tomcat v5.0 Server
Tomcat v5.5 Server
Tomcat v6.0 Server
Tomcat v7.0 Server
Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.
Server's host name:
localhost
Server name:
eTrading
Server runtime environment:
Apache Tomcat v6.0
Add...
Configure runtime environments...
< Back
Next >
Finish
Cancel

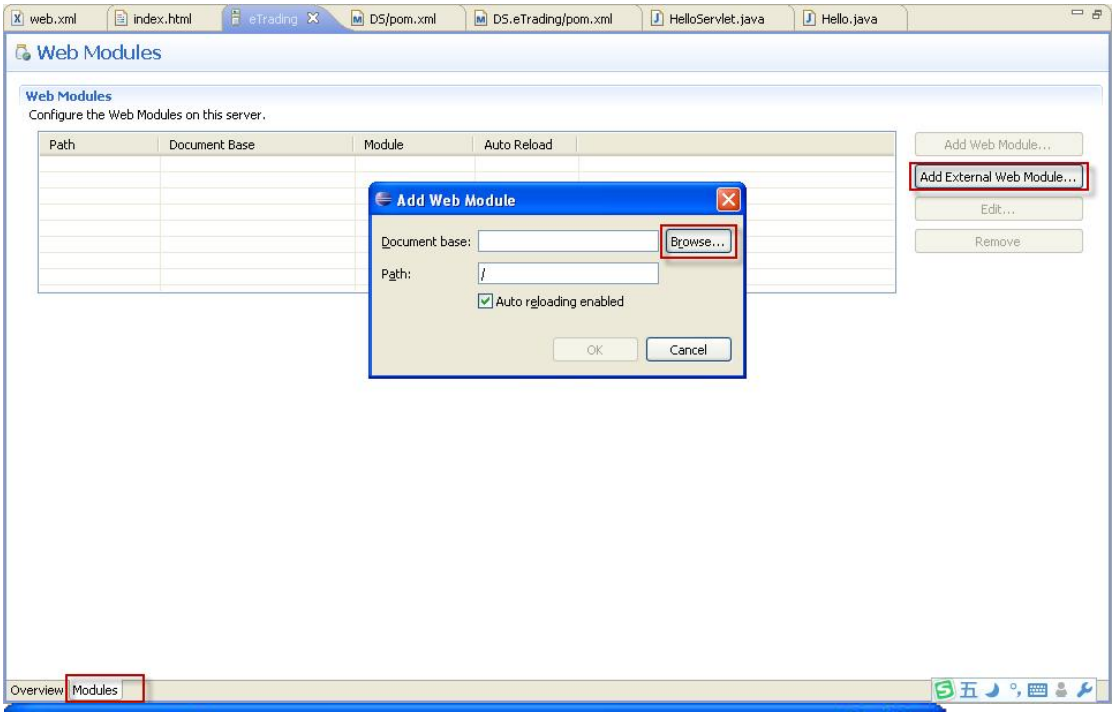


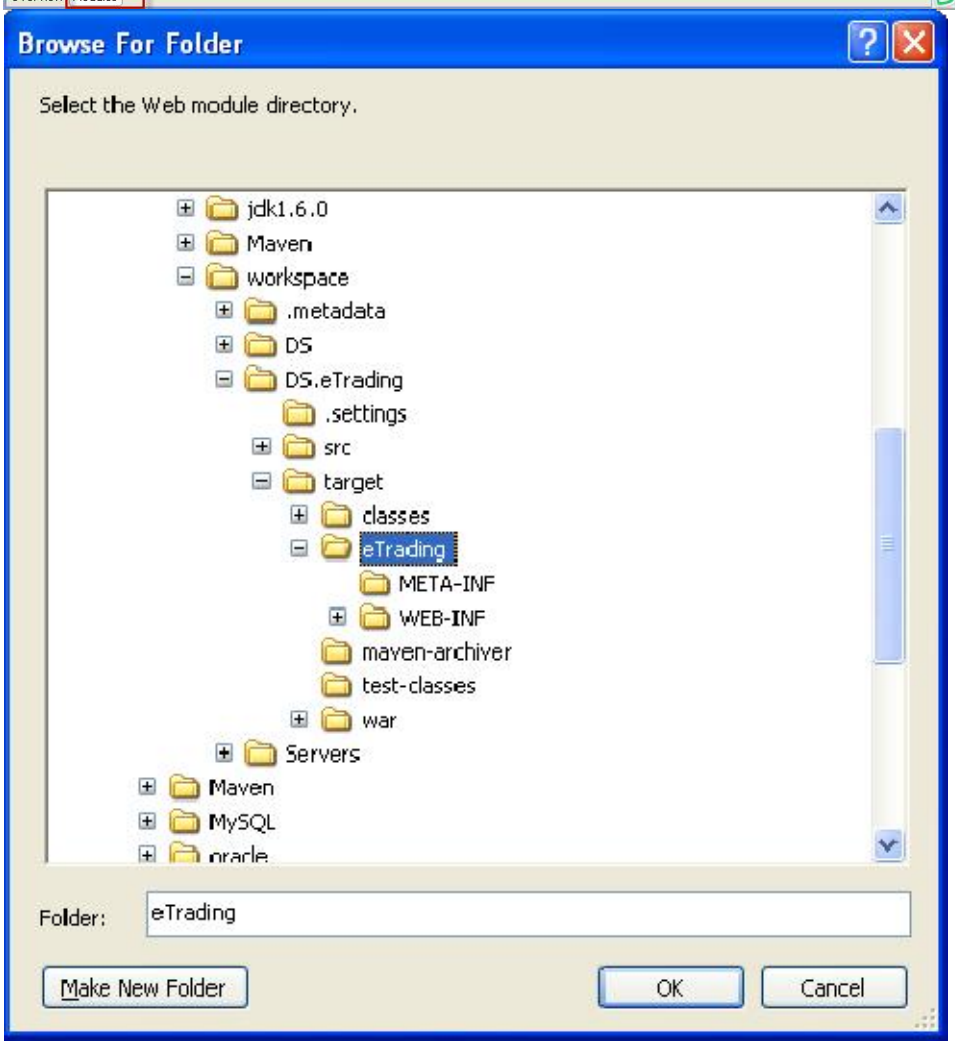
（4） 添加 eTrading 项目至 WTP Server

打开 WTP 的 server 的配置页，在双击 servers 下面的 server(eTrading)



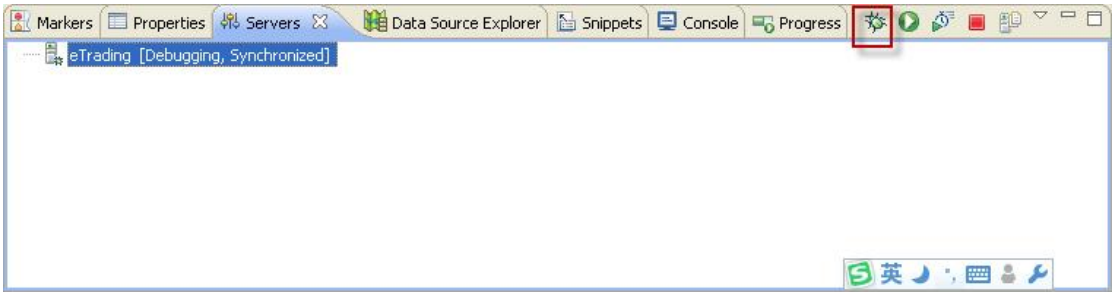
选择 Modules→Add External Web Module→ Brower… 选择 ds.eTrading 项目下的 target 目录下的 eTrading 目录.修改 Path 为 eTrading，保存。


web.xml
index.html
eTrading
DS/pom.xml
DS.eTrading/pom.xml
HelloServlet.java
Hello.java
Web Modules
Web Modules
Configure the Web Modules on this server.
Path
Document Base
Module
Auto Reload
Add Web Module...
Add External Web Module...
Edit...
Remove
Add Web Module
Document base:
Browse...
Path:
/
Auto reloading enabled
OK
Cancel
Overview
Modules


Browse For Folder
Select the Web module directory.
jdk1.6.0
Maven
workspace
.metadata
DS
DS.eTrading
.settings
src
target
classes
eTrading
META-INF
WEB-INF
maven-archiver
test-classes
war
Servers
Maven
MySQL
oracle
Folder:
eTrading
Make New Folder
OK
Cancel

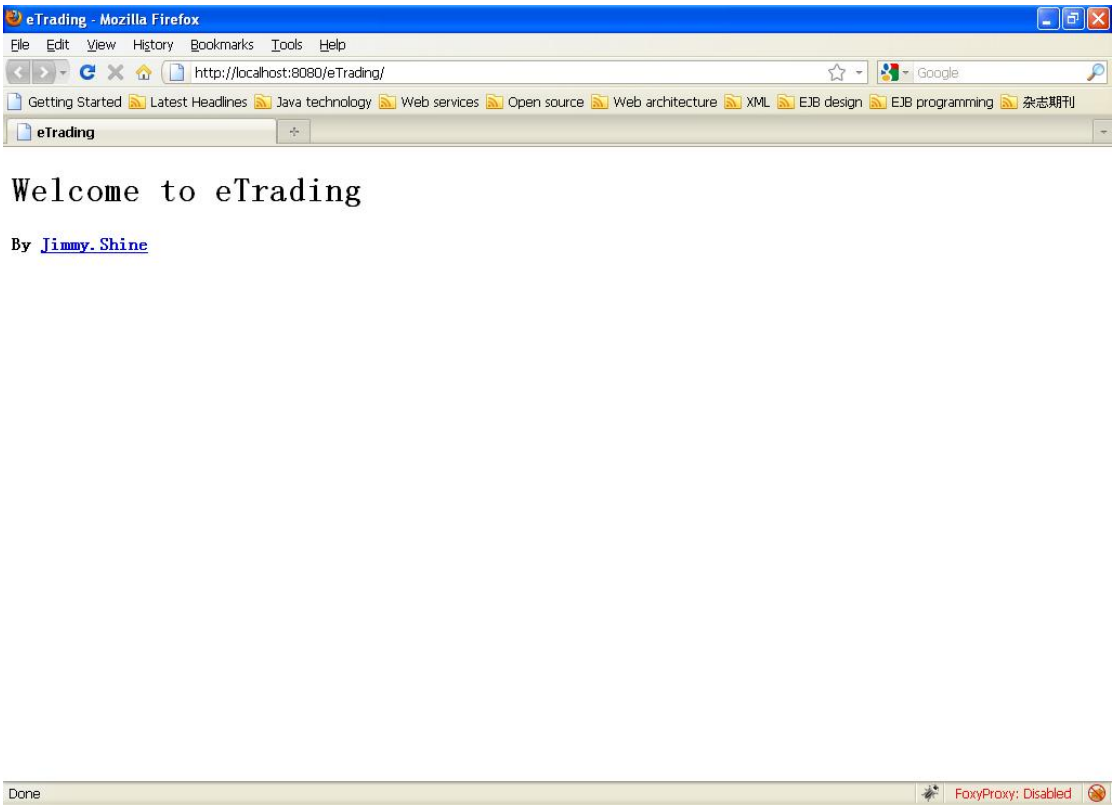
（5） 以 debug 模式启动 WTP Server

选择 eTrading，点击 debug.



（6） 访问 Web 应用

在浏览器中输入网址，访问

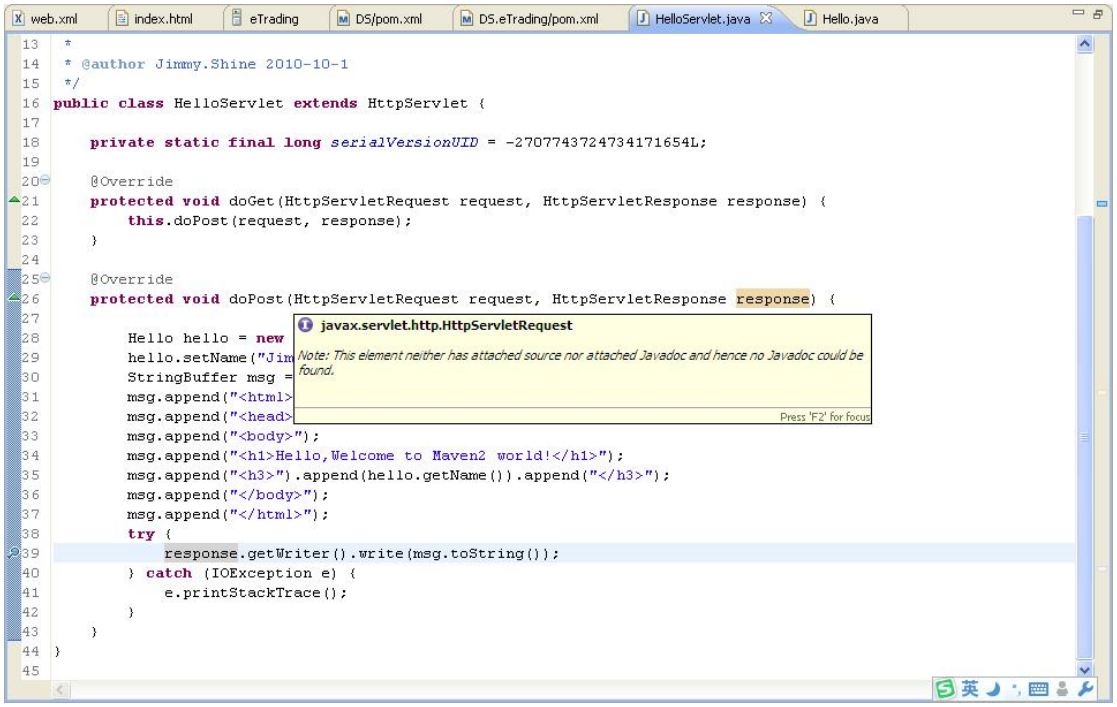


访问我们开发的 Servlet(参见源码)

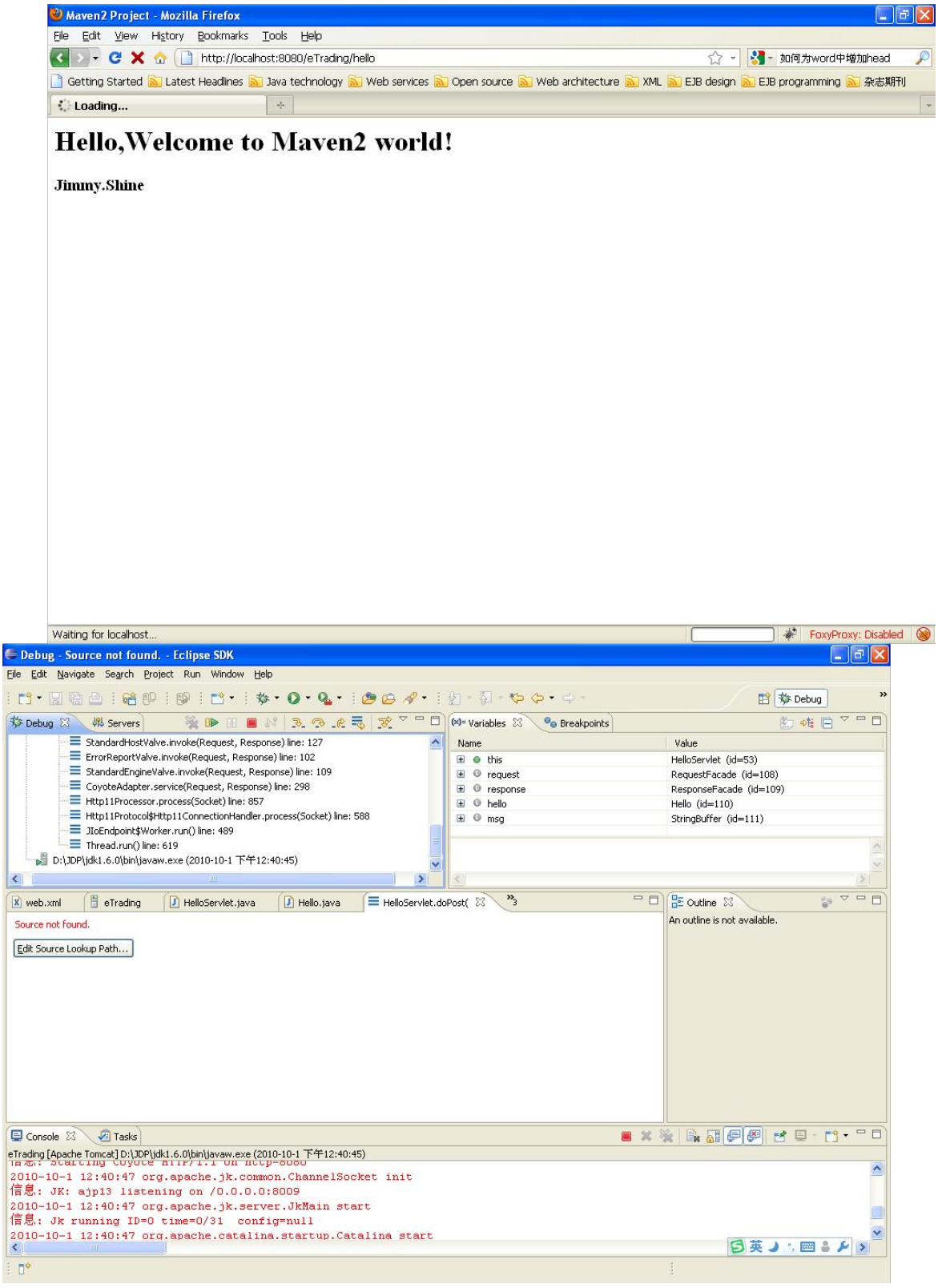
3. 调试项目

（1） 设置断点

在 Java 源码左边，双击，设置断点。



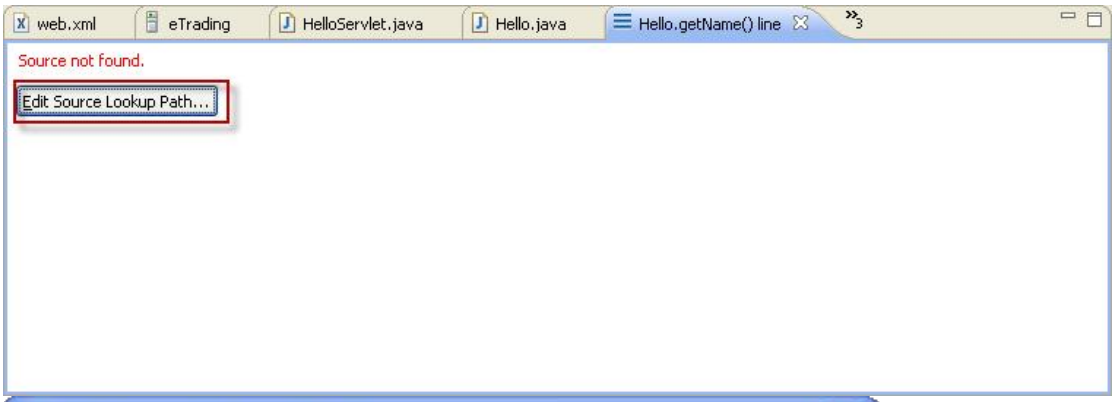
（2） 访问

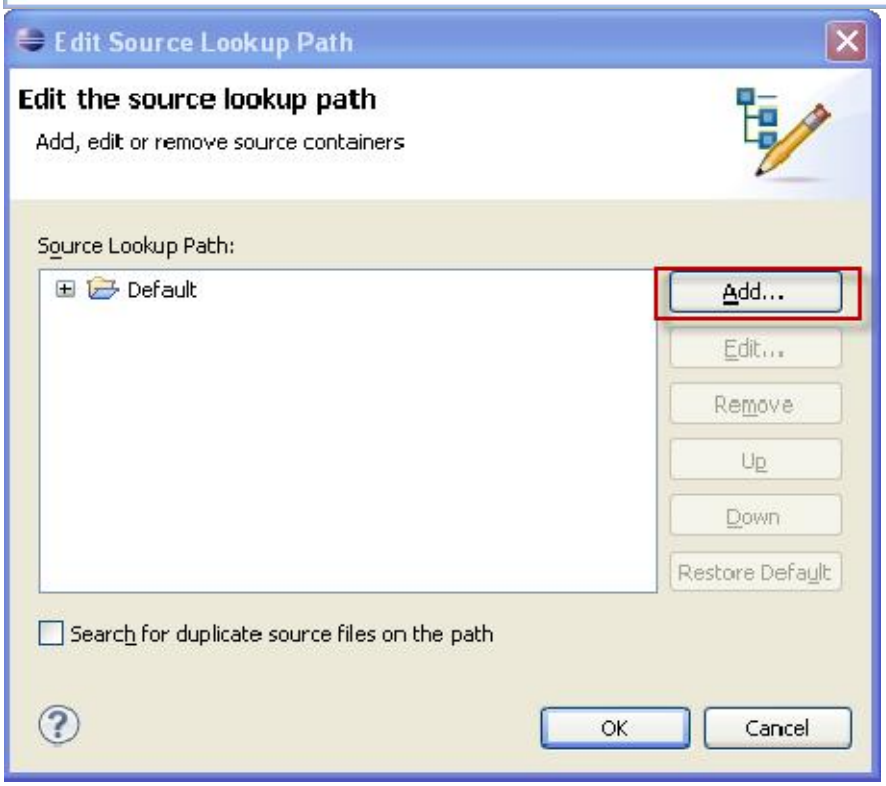


（3） 附加源码

Edit Source Lookup Path，在弹出的页面上点击 Add，在弹出的页面上选择 Java Project，选择项

目 ds.eTrading,确定。





同理，可以为 ds.Domain 和 ds.Service 附加源码。

可以看到调试视图定位到了断点处。

（4） 修改 Java 源代码，实时调试

对 Java 源代码进行修改，服务器可是以实现实时加载(具体的原理可以参见 JDPA ,可以利用 JDPA 机制，实现对于代码的断点调试)。

结束 Java 源代码的调试修改后，应当执行 mvn install，以使修改的代码更新至仓库中。

（5） 修改非 Java 源代码

修改非 Java 源代码时候，应当执行 mvn install，将修改的代码更新到仓库中。

参考资料：

1. 对于 Maven 生命周期的介绍：
   http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
2. 《Apache Maven2 Effective Implementation》
3. 《Maven 权威中文指南中文版》
4. 《深入 Java 调试体系.docx》