第 16 章

m2eclipse

本章内容

- m2eclipse 简介
- 新建 Maven 项目
- 导入 Maven 项目
- 执行 mvn 命令
- 访问 Maven 仓库
- 管理项目依赖
- 其他实用功能
- 小结

由于 Eclipse 是非常流行的 IDE，为了方便用户，日常开发使用的各种工具都会提供相应的 Eclipse 插件。例如，Eclipse 默认就集成了 JUnit 单元测试框架、CVS 版本控制工具以及 Mylyn 任务管理框架。Eclipse 插件的数量非常多，读者可以访问 Eclipse Marketplace[⊖]以了解各种各样的 Eclipse 插件。m2eclipse 就是一个在 Eclipse 中集成 Maven 的插件，有了该插件，用户可以方便地在 Eclipse 中执行 Maven 命令、创建 Maven 项目、修改 POM 文件等。本章将详细介绍 m2eclipse 的使用。

16.1 m2eclipse 简介

和 Nexus 一样，m2eclipse 也是 Sonatype 出品的一款开源工具。它基于 Eclipse Public License-v.10 开源许可证发布，用户可以免费下载并使用，还可以查看其源代码。m2eclipse 的官方站点地址为 http://m2eclipse.sonatype.org/。

m2eclipse 为 Eclipse 环境提供了全面丰富的 Maven 集成。它的主要功能如下：

- ❑ 创建和导入 Maven 项目
- ❑ 管理依赖并与 Eclipse 的 classpath 集成
- ❑ 自动下载依赖
- ❑ 自动解析依赖的 sources 与 javadoc 包
- ❑ 使用 Maven Archetype 创建项目
- ❑ 浏览与搜索远程 Maven 仓库
- ❑ 从 Maven POM 具体化一个项目
- ❑ 从 SCM 仓库签出 Maven 项目
- ❑ 自动适配嵌套的多模块 Maven 项目至 Eclipse
- ❑ 集成 Web Tools Projects（WTP）
- ❑ 集成 Subclipse
- ❑ 集成 Mylyn
- ❑ 可视化 POM 编辑
- ❑ 图形化依赖分析

16.2 新建 Maven 项目

m2eclipse 的安装已经在 2.5 节中详细介绍，这里不再赘述。在 m2eclipse 中新建一个 Maven 十分简单，在菜单栏中依次选择 File→New→Other，这时可以看到图 16-1 所示的向导。

选择 Maven Project 之后，向导会提示用户选择是否跳过 archetype 而创建一个最简单的

⊖ 网址为：http://marketplace.eclipse.org/

Maven 项目（Create a simple project）。这个最简单项目将只包含最基本的 Maven 项目目录结构，读者可以根据自己的需要进行选择。如果选择使用 Archetype 创建项目，单击 Next 按钮之后，向导会提示用户选择 Archetype，如图 16-2 所示。

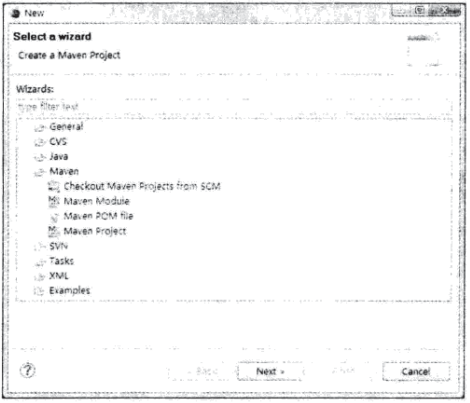

图 16-1　新建 Maven 项目向导

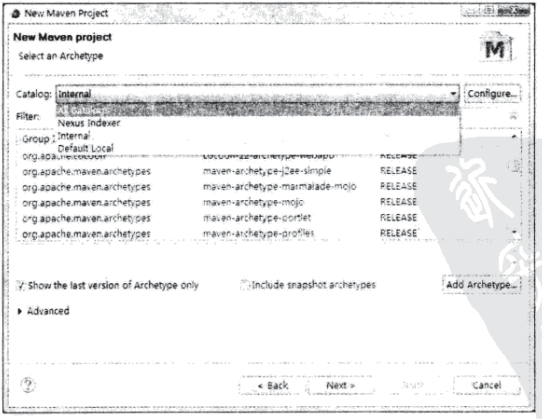

图 16-2　选择创建项目的 Archetype

图 16-2 中有 4 个 Archetype Catalog 可供用户选择，包括 maven-archetype-plugin 内置的 Internal、本地仓库的 Default Local、m2eclipse 下载到仓库索引中包含的 Nexus Indexer，以及所有这 3 个合并得到的 All Catalogs。如果对 Archetype Catalog 不是很清楚，可以参考 18.3 节。一般来说，只需要选择 Internal，然后再选择一个 Archetype（如 maven-archetype-quick-start），最后单击 Next 按钮。

接下来要做的就是输入项目坐标 Group Id、Artifact Id、Version 以及包名。这一个步骤与在命令行中使用 Archetype 创建项目类似，如果 Archetype 有其他可配置的属性，用户也可以在这里一并配置，如图 16-3 所示。

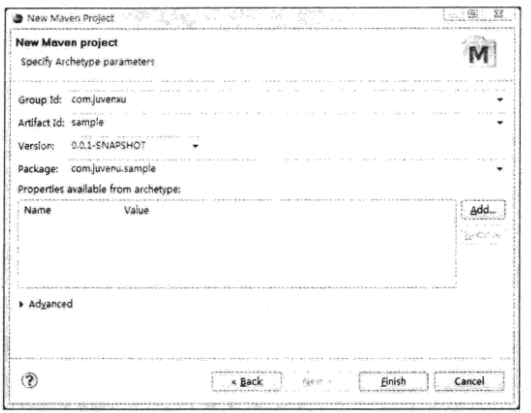

图 16-3 为项目输入坐标和包名

单击 Finish 按钮之后，m2eclipse 就会快速地在工作区创建一个 Maven 项目，这同时也是一个 Eclipse 项目。

16.3 导入 Maven 项目

较之于创建新的 Maven 项目，实际工作中更常见的是导入现有的 Maven 项目。m2eclipse 支持多种导入的方式，其中最常用的是导入本地文件系统的 Maven 项目以及导入 SCM 仓库中的 Maven 项目。

单击菜单栏中的 File，然后选择 Import 开始导入项目，如图 16-4 所示。

从图 16-4 中可以看到在 Maven 类中有 4 种导入方式，常用的就是第一种和第二种，即导入 SCM 仓库中的 Maven 项目和导入本地文件系统的 Maven 项目。

图 16-4 中的 Install or deploy an artifact to a Maven repository 能让用户将任意的文件安装

到 Maven 的本地仓库。如果该文件没有对应的 POM，则需要为其定义 Maven 坐标。

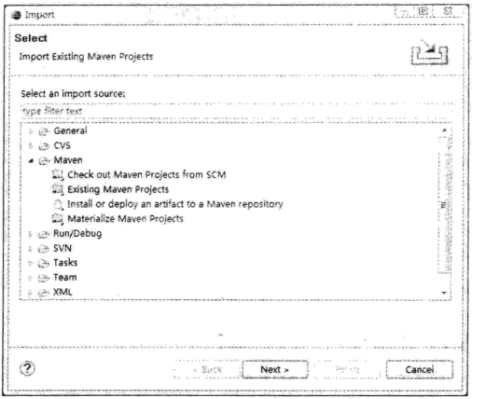

图 16-4　开始导入 Maven 项目

图 16-4 中的 Materialize Maven Projects 能让用户导入第三方的 Maven 项目，用户只需要提供一些关键字如 nexus-api，然后选择要导入的项目，m2eclipse 就能基于索引找到其对应的 POM 信息。如果该 POM 中包含了 SCM 信息，m2eclipse 就能直接下载该项目的源码并导入到 m2eclipse 中。当用到某个第三方类库，同时想研究其源码的时候，这一特性就非常有用，你不再需要打开浏览器去寻找该项目的信息，简单地在 m2eclipse 中操作几步就能完成第三方项目的导入。当然，这一特性的前提是第三方类库提供了正确的 SCM 信息。大多数开源项目在往 Maven 中央仓库提交构件的时候都会提供完整的信息，但也有例外，为了避免信息不完整的项目进入 Maven 中央仓库，最新的规则已经强制要求提交者提供完备的信息，如 SCM、许可证以及源码包等。这无疑能帮助 m2eclipse 表现得更好。

16.3.1　导入本地 Maven 项目

现在详细介绍一下如何导入本地 Maven 项目。选择图 16-4 中的 Existing Maven Projects 项，然后在弹出的对话框中选择本地项目所在的目录，如图 16-5 所示。

m2eclipse 能够自动识别出目录中所包含的 Maven 项目，如果发现是多模块项目，则会列出所有的模块。用户可以根据自己的需要选择要导入的模块，然后单击 Finish 按钮。m2eclipse 会执行导入项目信息、更新下载项目依赖，以及重建工作区等操作。根据实际项目的情况，这个过程可能花费几十秒到十几分钟。

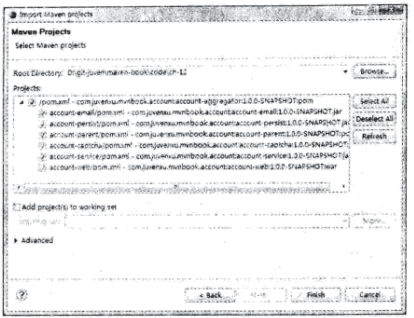

图 16-5 导入现有 Maven 项目

16.3.2 从 SCM 仓库导入 Maven 项目

通常我们的项目源代码都存储在 SCM 仓库中，例如 Subversion 仓库，读者当然可以使用 Subversion 命令将项目源码签出到本地，然后再导入到 m2eclipse 中。但 m2eclipse 支持用户直接从 SCM 仓库中导入 Maven 项目。

要从 SCM 导入 Maven 项目，首先需要确保安装了集成 SCM 的 Eclipse 插件，如 Subclipse，还需要 m2eclipse 的附属组件 Maven SCM Integration 以及对应的 SCM handler，如集成 Subclipse 的 Maven SCM handler for Subclipse。

如果这些组件都得以正确安装，就可以选择图 16-4 中的 Check out Maven Projects from SCM，在单击 Next 按钮之后，选择 SCM 类型并输入 SCM 地址，如图 16-6 所示。

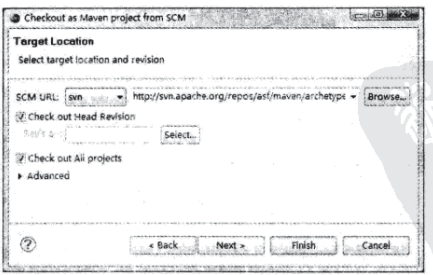

图 16-6 从 SCM 仓库导入 Maven 项目

单击 Next 按钮之后用户可以选择项目导入的本地位置，然后单击 Finish 按钮，m2eclipse 就会在后台使用 SCM 工具签出项目并执行 Maven 构建。用户可以单击 Eclipse 右

下角的状态栏查看后台进程的状态，如图 16-7 所示。

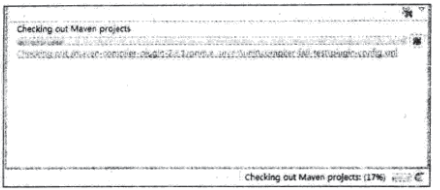

图 16-7　m2eclipse 在后台签出项目

同样地，根据项目大小以及网络的健康状况，这个过程可能花费几十秒到几十分钟不等。

16.3.3　m2eclipse 中 Maven 项目的结构

一个典型的 Maven 项目在 m2eclipse 中的结构如图 16-8 所示。

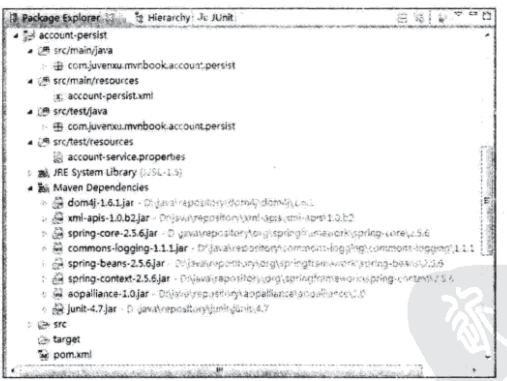

图 16-8　m2eclipse 中的 Maven 项目的结构

Maven 项目的主代码目录 src/main/java/、主资源目录 src/main/resources/、测试代码目录 src/test/java/和测试资源目录 src/test/resources/都被自动转换成了 Eclipse 中的源码文件夹（Source Folder）。Maven 的依赖则通过 Eclipse 库（Libraries）的方式引入，所有 Maven 依赖都在一个名为 Maven Dependencies 的 Eclipse 库中。需要注意的是，这些依赖文件并没有被复制到 Eclipse 工作区，它们只是对 Maven 本地仓库的引用。所有的源码文件夹和 Maven 依赖都在 Eclipse 项目的构建路径（Build Path）中。当然，用户还可以直接访问项目根目录下的 pom.xml 文件。此外，代码目录和资源目录之外的其他目录不会被转换成 Eclipse

的源码文件夹，它们不会被加入到构建路径中，但用户还是可以在 Eclipse 中访问它们。

注意：如果用户更改了 POM 内容且导致项目结构发生变化，例如添加了一个额外的资源目录，m2eclipse 可能无法自动识别。这时用户需要主动让 m2eclipse 更新项目结构：在项目或者 pom.xml 上单击鼠标右键，选择 Maven，再选择 Update Project Configuration。

16.4 执行 mvn 命令

到目前为止，大家已经了解了如何在 m2eclipse 中创建 Maven 项目和导入 Maven 项目，下一步要做的就是构建这些项目，或者说在这些项目中执行 mvn 命令。当然，大家还是可以在命令行的对应目录下执行 mvn 命令，不过这里要讲的是如何在 m2eclipse 中直接执行 mvn 命令。

要在 m2eclipse 中执行 mvn 命令，首先要做的是打开 m2eclipse 的 Maven 控制台。一般来说，Eclipse 窗口的下方会有一个终端（Console）视图，打开该视图后，可以在视图的右上角选择打开 Maven 终端，如图 16-9 所示。

图 16-9 打开 Maven 终端

Maven 终端视图中会显示 m2eclipse 中所有 mvn 命令的输出。现在可以在 Maven 项目中执行 mvn 命令。直接在项目上或者 pom.xml 上单击鼠标右键，选择 Run As 选项，就能看到如图 16-10 所示的菜单。

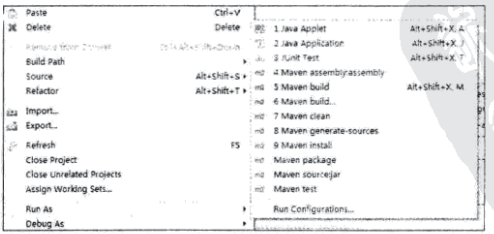

图 16-10 执行 Maven 构建命令

在图 16-10 中可以看到，菜单预置了很多构建命令，包括 clean、test、package 以及 install 等，直接单击就能让 m2eclipse 执行相应的 Maven 构建。

如果想要执行的 mvn 命令并没有被预置在这个菜单中该怎么办呢？这时可以选择

图 16-10 中的 Maven build 项来自定义 mvn 命令。图 16-11 显示的是单击 Maven Build... 项后显示的自定义 mvn 命令配置对话框。

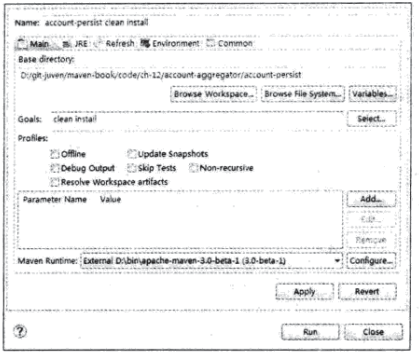

图 16-11　自定义 mvn 命令

图 16-11 为该配置提供了 Maven 目标 clean install，还定义了一个 account-persist clean install 的名称以方便日后重用。读者可以看到该配置页面能让用户自定义很多内容，例如是否更新 Snapshots、是否跳过测试、是否开启 Debug 输出，还包括添加额外的运行参数，等等。配置完成后，单击 Run 按钮就能执行该 mvn 命令了。读者可以在 Maven 终端查看运行输出。

使用上述的方法可以自定义任意多的 mvn 命令，而且这些配置都是可以被重用的。要再次运行自定义的 mvn 命令，单击图 16-10 中的 Maven build（注意没有省略号），然后就能看到如图 16-12 所示的对话框。

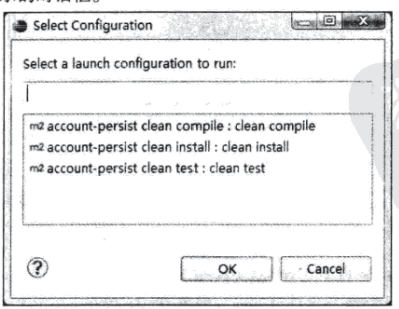

图 16-12　重用自定义 mvn 命令

如图 16-12 所示，读者可以选择并直接运行之前配置过的自定义 mvn 命令。需要注意的是，如果只配置了一个自定义 mvn 命令，m2eclipse 会跳过该选择框并直接运行，如果还没有配置任何自定义的 mvn 命令，m2eclipse 则会提供配置对话框让读者定义（第一次）mvn 命令。

16.5 访问 Maven 仓库

有了 m2eclipse，用户可以直接在 Eclipse 中浏览本地和远程的 Maven 仓库，并且能够基于这些仓库的索引进行构件搜索和 Java 类搜索。这样就免去了离开 Eclipse 访问本地文件系统或者浏览器的麻烦，提高了日常开发的效率。

16.5.1 Maven 仓库视图

m2eclipse 提供了 Maven 仓库视图，能让用户方便地浏览本地及远程仓库的内容，不过默认情况下该视图不被开启。要开启 Maven 仓库视图，依次选择 Eclipse 菜单栏中的 Windows、Show View、Other 选项，Eclipse 会弹出一个对话框让用户选择要打开的视图。选择 Maven 类下的 Maven Repositories，如图 16-13 所示。

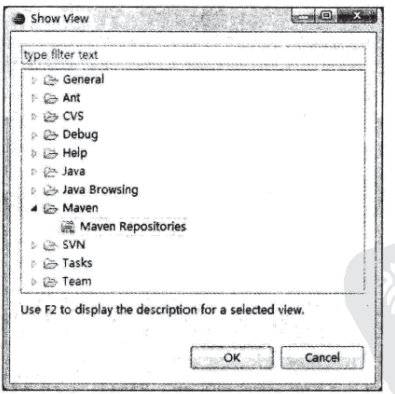

图 16-13 打开 Maven 仓库视图

这时可以在 Eclipse 窗口下方看到 Maven 仓库视图，这个视图中包含了 3 类 Maven 仓库，分别为本地仓库、全局仓库以及项目仓库，如图 16-14 所示。

其中本地仓库包含了 Maven 的本地仓库以及当前 Eclipse 工作区的项目；全局仓库默认是 Maven 中央仓库，但是如果在 settings.xml 中设置了镜像，全局仓库就会自动变更为镜像

仓库。最后，如果当前 Maven 项目的 pom. xml 中配置了其他仓库，它们就会被自动加入到项目仓库这一类中。这些仓库的信息来源于用户的 settings. xml 文件和工作区中 Maven 项目的 pom. xml 文件。

图 16-14 Maven 仓库视图

用户可以以树形结构快速浏览仓库的内容，双击叶子节点，打开构件对应的 POM 文件，如图 16-15 所示。

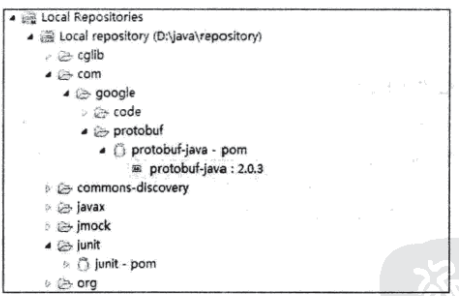

图 16-15 浏览 Maven 仓库内容

大家可能已经猜到，m2eclipse 其实不会真正地去存储所有仓库的内容，那样需要消耗大量的磁盘及网络带宽。因此与 Nexus 一样，m2eclipse 使用 nexus-indexer 索引仓库内容的信息。以全局仓库 central 为例，用户在首次使用 m2eclipse 的仓库浏览及搜索功能之前，需要构建该仓库的索引，在如图 16-16 所示的仓库上右击。

快捷菜单中的 Rebuild Index 让 m2eclipse 重新下载完整的远程索引，由于当前仓库是 central，索引文件较大，因此重建该索引会消耗比较长的时间。Update Index 则让 m2eclipse 以增量的方式下载索引文件。如果是本地仓库，Update Index 将无法使用，而 Rebuild Index 的效果是重新遍历本地仓库的文件建立索引。

图 16-16 中的菜单还有几个选项，Disable Index Details 让 m2eclipse 关闭该仓库的索引，

从而用户将无法浏览该仓库的内容，或者对其进行搜索。Minimum Index Enabled 表示只对仓库内容的坐标进行索引，而 Enable Full Index 不仅索引仓库内容的坐标，还索引这些文件所包含的 Java 类信息，从而能够支持用户搜索仓库中的 Java 类。

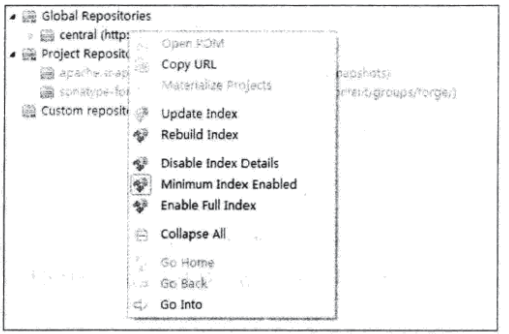

图 16-16 构建仓库索引

16.5.2 搜索构件和 Java 类

有了仓库索引之后，用户就可以通过关键字搜索 Maven 构件了。单击 Eclipse 菜单栏中的 Navigate，再选择 Open Maven POM 选项，就能得到构件搜索框。输入关键字后就能得到一个结果列表，还可以点击列表项进一步展开以查看版本信息，如图 16-17 所示。双击某个具体版本的构件能让 m2eclipse 直接打开对应的 POM 文件。

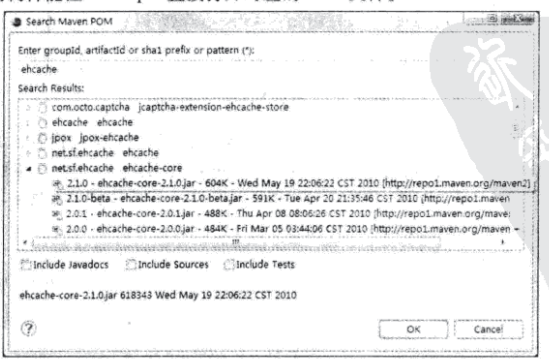

图 16-17 搜索 Maven 构件

如果为仓库开启了 Enable Full Index 选项，也就是说索引中包含了 Java 类型信息，则就可以通过 Java 类名的关键字寻找构件。单击 Eclipse 菜单栏中的 Navigate，再选择 Open Type from Maven，就能得到类搜索框。输入关键字后就能得到图 16-18 所示的搜索结果。同样，用户可以单击列表项展开其版本，还可以双击具体版本打开其 POM。

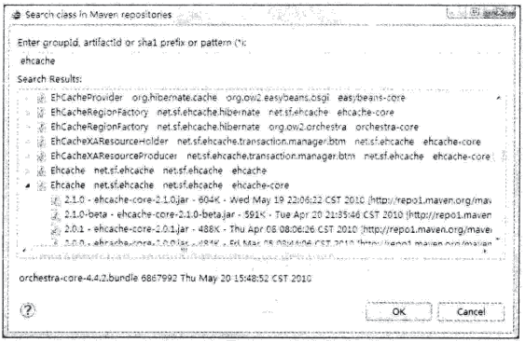

图 16-18 搜索 Java 类

不用离开 Eclipse，用户就能随时搜索想要使用的类库以及 Java 类，m2eclipse 仅仅要求用户提供一些必要的关键字，这无疑是非常方便的。

16.6 管理项目依赖

添加 Maven 依赖的传统做法是先搜索得到依赖的坐标，然后配置项目的 pom.xml 文件，加入 dependency 元素。当然，在 m2eclipse 中也可以这样做，不过 m2eclipse 提供了更方便的添加依赖的方法，用户直接根据关键字搜索依赖并从结果中选择即可。此外，m2eclipse 还提供了丰富的可视化界面帮助用户分析项目中的各种依赖以及它们之间的关系。

16.6.1 添加依赖

在 m2eclipse 中有多种添加依赖的方法，直接编辑 pom.xml 是一种，不过这里要讲的是另外两种更方便的做法。

首先用户可以在项目上或者 pom.xml 上右击，然后选择 Maven，再选择 Add Dependency 添加依赖，如图 16-19 所示。

在弹出的对话框中，用户只需要输入必要的关键字，然后选择要添加的依赖及版本，并且设定正确的依赖范围，单击 OK 按钮之后，依赖就被自动加入到 pom.xml 中。图 16-20 所示就为项目添加了 javax.servlet:servlet-api:2.5 这样一个依赖，并且在图的下方选择了

provided 这样一个依赖范围。

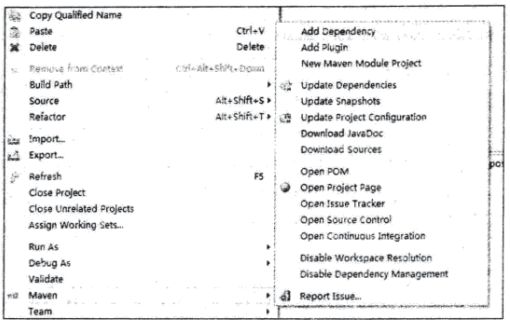

图 16-19 在项目上添加依赖

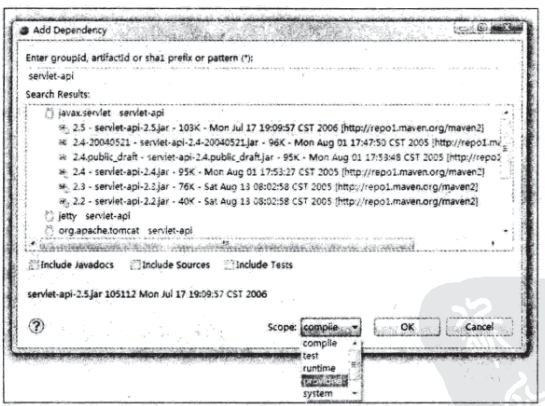

图 16-20 为项目添加 servlet-api 依赖

第二种快速添加依赖的方式是使用 m2eclipse 的 POM 编辑器。默认情况下，用户双击项目的 pom. xml 就能打开 POM 编辑器。POM 编译器下方有很多选项卡，包括概览、依赖、插件、报告、依赖层次、依赖图、Effective POM 等。其中，依赖（Dependencies）一项可以用来添加、删除和编辑依赖，如图 16-21 所示。

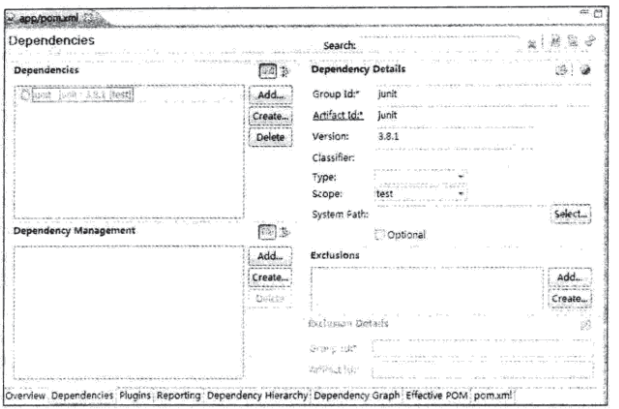

图 16-21 POM 编辑器中的依赖管理项

单击图 16-21 中上方的 Add 按钮就能得到如图 16-20 所示的添加依赖对话框。此外，从图中还可以看到，用户可以查看依赖的细节并对其进行编辑。

添加项目依赖之后，如果 m2eclipse 没有自动将依赖更新至项目的构建路径，用户可以强制要求 m2eclipse 更新，方法是在项目或者 pom. xml 上右击，选择 Maven，再选择 Update Dependencies。

16. 6. 2 分析依赖

5. 9. 3 节介绍了如何使用 maven-dependency-plugin 分析并优化项目的依赖，Maven 用户可以在命令行以树状的形式查看项目的依赖以及它们之间的关系。有了 m2eclipse，这种可视化的分析将更为清晰和直观。

开启 POM 编辑器中的依赖层次项（Dependency Hierarchy），就能看到图 16-22 所示的依赖层次图。

图 16-22 中左边列表显示了项目的树形依赖层次，右边列表则是所有 Maven 最终解析得到的依赖。默认情况下，两个列表都会显示依赖的 artifact、version 以及 scope。要查看依赖的 groupId，可以单击列表上方右起第二个按钮——Show GroupId。

有了这样一个依赖层次图，用户就能很清晰地看到所有依赖是如何进入到项目中来的，可能这是个直接依赖，那么在左边的它就是个顶层节点；可能这是个传递性依赖，那么这个树形层次就能够告诉用户传递路径是什么。如果这个依赖是同一 Maven 项目的另外一个模块，那么它的图标将与其他依赖不同，而是一个文件夹的样子。如果用户单击右边已解析依赖列表中的任意一项，左边就会自动更新为该依赖的传递路径，如图 16-23 所示。

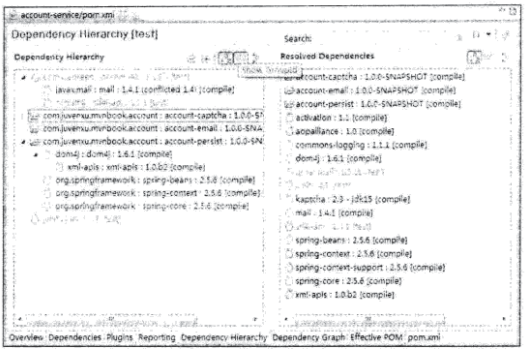

图 16-22 依赖层次列表

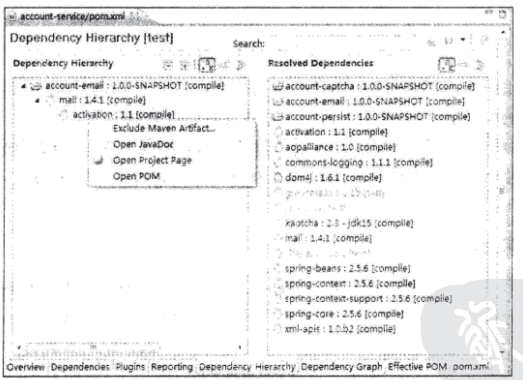

图 16-23 查看已解析依赖的传递路径

从图 16-23 中我们知道，activation 这样一个依赖是通过 account-email 依赖的 mail 依赖引入的。

此外，从图 16-23 中还能看到，在任何一个依赖上右击，可以执行打开依赖的 POM 和排除依赖等操作。尤其是排除依赖这一操作，比编辑 POM 更加直观和方便。

除了依赖层次列表，POM 编辑器还提供了一个更为图形化、更为直观的依赖图，如图 16-24 所示。

在这个依赖图中，每个依赖都是一个圆角矩形，用户可以随意拖动每个依赖，被选择依赖与其他依赖的连接线会被标亮。用户也可以在依赖上右击，选择显示 groupId，以及执

行打开 POM 和排除依赖等操作。

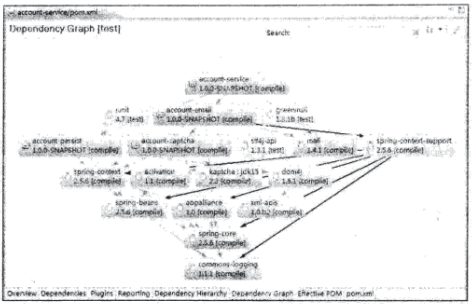

图 16-24　依赖图

16.7　其他实用功能

到目前为止，本章介绍了 m2eclipse 最主要的几个功能，包括新建项目、导入项目、执行 mvn 命令、访问 Maven 仓库和管理项目依赖。m2eclipse 还有很多琐碎的功能，由于其中有一些在实际中很少用到，笔者不计划逐一详细介绍。本章剩余的内容讲述几个 m2eclipse 非常实用的小特性。

16.7.1　POM 编辑的代码提示

m2eclipse 的 POM 编辑器能让用户以表单的形式编辑 pom.xml 文件，但很多时候这总没有直接编辑 XML 文件来得直接。有了 m2eclipse，用户在编辑 pom.xml 的时候就能得到即时的代码提示帮助，如图 16-25 所示。

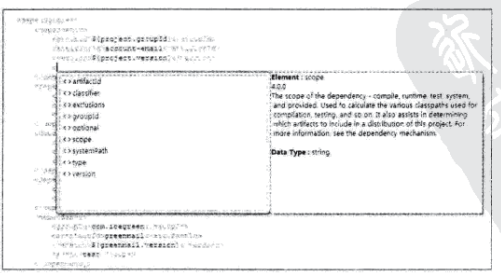

图 16-25　POM 编辑的代码提示

在图 16-25 中可以看到，当用户在 < dependency > 元素下输入左尖括号想要添加一个子元素的时候，会得到可用元素的列表（用户也可以使用 Alt + / 主动调出代码提示）。在该例中，< dependency > 下可用的子元素有 artifactId、classifier、exclusions 以及 scope 等。使用键盘的上下键可以选择查看某个元素，列表右边就会显示出该元素的解释。该例中右边显示了 scope 元素的解释。选择想要输入的元素后按 Enter 键，m2eclipse 就会自动填上元素标签，用户只需要输入元素的值即可。对于不熟悉 POM 结构的用户来说，这种代码提示帮助他们免去查阅文档的麻烦。对于熟悉 POM 的用户来说，代码提示也可以帮助他们节省输入时间。

16.7.2 Effective POM

我们都知道，任何一个项目的 POM 都至少继承自 Maven 内置的超级 POM，有些项目中用户还会配置自己的继承层次。也就是说，单从当前的 POM 是无法全面了解项目信息的，你必须同时查看所有父 POM。Maven 有一个 Effective POM 的概念，它表示一个合并整个继承结构所有信息的 POM。假设项目 A 继承自项目 B，而 B 又隐式地继承自超级 POM，那么 A 的 Effective POM 就包含了所有 A、B 以及超级 POM 的配置。有了 Effective POM，用户就能一次得到完整的 POM 信息。

Maven 用户可以直接从命令行获得 Effective POM：

```
$ mvn help:effective-pom
```

在 m2eclipse 的 POM 编辑器中，有一项专门的 Effective POM，用户可以直接查看当前项目的 Effective POM，如图 16-26 所示。当然，由于这是一个由其他 POM 合并而来的文件，你将无法对其直接进行修改。

图 16-26 Effective POM

16.7.3　下载依赖源码

m2eclipse 能够自动下载并使用依赖的源码包，当你需要探究第三方开源依赖的细节，或者在调试应用程序的时候，这一特性非常有用。当然，该功能的前提是依赖提交了相应的源码包至 Maven 仓库，通常这个源码包是一个 classifier 为 sources 的 jar 文件。例如 junit-4.8.1.jar 就有一个对应的 junit-4.8.1-sources.jar 源码包。

m2eclipse 用户可以在项目上或者 pom.xml 上右击，选择 Maven，再选择 Download Sources 让 m2eclipse 为当前项目的依赖下载源码包。也可以设置 Maven 首选项让 m2eclipse 默认自动下载源码包。方法是单击 Eclipse 菜单中的 Window 并选择 Preferences，然后在弹出的对话框左边选择 Maven，接着在右边选上 Download Artifact Sources，如图 16-27 所示。

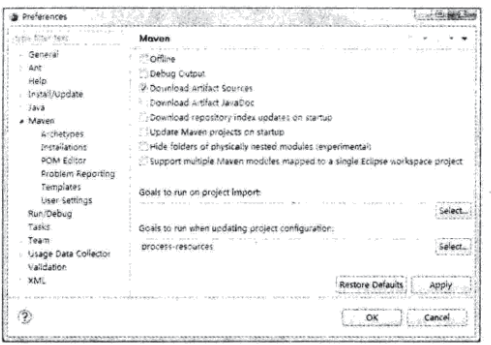

图 16-27　开启源码包下载

从图 16-27 中读者还可以看到，Maven 首选项允许配置很多 m2eclipse 的默认行为，包括是否开启 Debug 输出、是否打开 Eclipse 就下载索引等。左边的 Maven 子项还允许用户做更多的配置，包括配置 m2eclipse 使用的 Maven 安装、自定义 settings.xml 文件等。读者可以根据自己的实际需要进行调整，这里不再赘述。

16.8　小结

笔者不推荐在不熟悉 Maven 命令行的情况下就使用 m2eclipse，如果不理解 Maven 的基本概念和命令行操作，华丽的 IDE 界面只能给你带来更多的困惑，尤其是当遇到问题的时候，由于牵扯了更多的非 Maven 因素，排疑会变得更加困难。

如果你已经熟悉了 Maven 的基本概念和命令行，并且你日常使用的 IDE 是 Eclipse，那么就大胆使用 m2eclipse 吧。你可以在 m2eclipse 中直接创建 Maven 项目，也可以从本地或者

SCM 仓库导入 Maven 项目，在 m2eclipse 中执行 mvn 命令也很方便，你还可以自定义并保存 mvn 命令。m2eclipse 还集成了 Maven 仓库客户端的功能，不用离开 IDE，用户就可以浏览和搜索 Maven 仓库，并且随时添加依赖。m2eclipse 提供的依赖分析功能也比命令行更加直观和清晰。除了这些主要特性，m2eclipse 还能让用户享受便捷的 POM 编辑代码提示，可以直接查看 Effective POM，以及自动下载使用依赖的源码包，这些功能都能大大提高日常开发的效率。