



第 2 章

Maven 的安装和配置

本章内容

- ☐ 在 Windows 上安装 Maven
- ☐ 在基于 UNIX 的系统上安装 Maven
- ☐ 安装目录分析
- ☐ 设置 HTTP 代理
- ☐ 安装 m2eclipse
- ☐ 安装 NetBeans Maven 插件
- ☐ Maven 安装最佳实践
- ☐ 小结

数字图书馆
PDG

第1章介绍了 Maven 是什么，以及为什么要使用 Maven，我们将从本章开始实际接触 Maven。本章首先将介绍如何在主流的操作系统下安装 Maven，并详细解释 Maven 的安装文件；其次还会介绍如何在主流的 IDE 中集成 Maven，以及 Maven 安装的最佳实践。

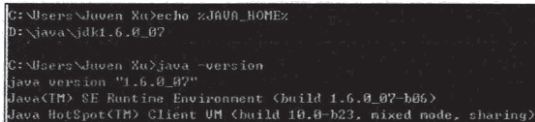
2.1 在 Windows 上安装 Maven

2.1.1 检查 JDK 安装

在安装 Maven 之前，首先要确认你已经正确安装了 JDK。Maven 可以运行在 JDK 1.4 及以上的版本上。本书的所有样例都基于 JDK 5 及以上版本。打开 Windows 的命令行，运行如下命令来检查 Java 安装：

```
C:\Users\Juven Xu>echo %JAVA_HOME%  
C:\Users\Juven Xu>java-version
```

结果如图 2-1 所示：



```
C:\Users\Juven Xu>echo %JAVA_HOME%  
D:\java\jdk1.6.0_07  
  
C:\Users\Juven Xu>java -version  
java version "1.6.0_07"  
Java(TM) SE Runtime Environment (build 1.6.0_07-b06)  
Java HotSpot(TM) Client VM (build 10.0-b23, mixed mode, sharing)
```

图 2-1 Windows 中检查 Java 安装

上述命令首先检查环境变量 JAVA_HOME 是否指向了正确的 JDK 目录，接着尝试运行 java 命令。如果 Windows 无法执行 java 命令，或者无法找到 JAVA_HOME 环境变量，就需要检查 Java 是否安装了，或者环境变量是否设置正确。关于环境变量的设置，请参考 2.1.3 节。

2.1.2 下载 Maven

请访问 Maven 的下载页面：<http://maven.apache.org/download.html>，其中包含针对不同平台的各种版本的 Maven 下载文件。对于首次接触 Maven 的读者来说，推荐使用 Maven 3.0，因此需要下载 apache-maven-3.0-bin.zip。当然，如果你对 Maven 的源代码感兴趣并想自己构建 Maven，还可以下载 apache-maven-3.0-src.zip。该下载页面还提供了 md5 校验和（checksum）文件和 asc 数字签名文件，可以用来检验 Maven 分发包的正确性和安全性。

在编写本书的时候，Maven 2 的最新版本是 2.2.1，Maven 3 基本完全兼容 Maven 2，而且比 Maven 2 的性能更好，还对其中某些功能进行了改进。如果你之前一直使用 Maven 2，现在正犹豫是否要升级，那就大可不必担心了，快点尝试一下 Maven 3 吧！

2.1.3 本地安装

将安装文件解压到指定的目录中，如：

```
D:\bin>jar xvf "C:\Users\Juven Xu\Downloads\apache-maven-3.0 bin.zip"
```

这里的 Maven 安装目录是 D:\bin\apache-maven-3.0，接着需要设置环境变量，将 Maven 安装配置到操作系统环境中。

打开系统属性面板（在桌面上右击“我的电脑”→“属性”），单击高级系统设置，再单击环境变量，在系统变量中新建一个变量，变量名为 M2_HOME，变量值为 Maven 的安装目录 D:\bin\apache-maven-3.0。单击“确定”按钮，接着在系统变量中找到一个名为 Path 的变量，在变量值的末尾加上 %M2_HOME%\bin；。注意：多个值之间需要有分号隔开，然后单击“确定”按钮。至此，环境变量设置完成。详细情况如图 2-2 所示。

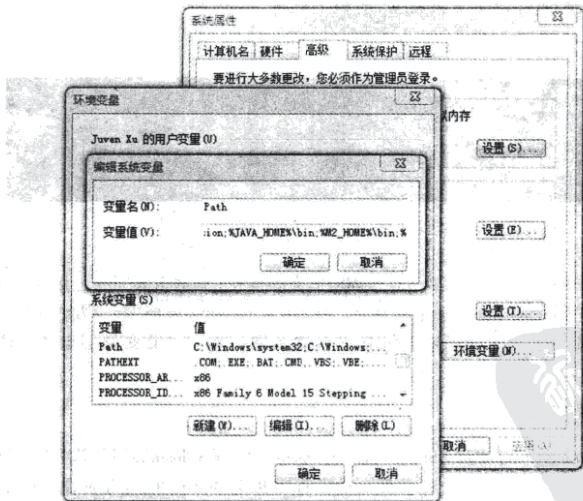


图 2-2 Windows 中系统环境变量配置

值得注意的是 Path 环境变量。当我们在 cmd 中输入命令时，Windows 首先会在当前目录中寻找可执行文件或脚本，如果没有找到，Windows 会接着遍历环境变量 Path 中定义的路径。由于将 %M2_HOME%\bin 添加到了 Path 中，而这里 %M2_HOME% 实际上是引用了前面定义的另一个变量，其值是 Maven 的安装目录。因此，Windows 会在执行命令时搜索目录 D:\bin\apache-maven-3.0\bin，而 mvn 执行脚本的位置就是这里。

了解环境变量的作用之后，现在打开一个新的 cmd 窗口（这里强调新的窗口是因为新的环境变量配置需要新的 cmd 窗口才能生效），运行如下命令检查 Maven 的安装情况：

```
C:\Users\Juven Xu>echo %M2_HOME%
C:\Users\Juven Xu>mvn -v
```

运行结果如图 2-3 所示。

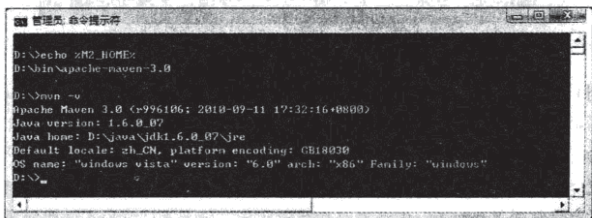


图 2-3 Windows 中检查 Maven 安装

第一条命令 `echo %M2_HOME%` 用来检查环境变量 `M2_HOME` 是否指向了正确的 Maven 安装目录；而 `mvn -v` 执行了第一条 Maven 命令，以检查 Windows 是否能够找到正确的 mvn 执行脚本。

2.1.4 升级 Maven

Maven 更新比较频繁，因此用户往往会需要更新 Maven 安装以获得更多、更酷的新特性，并避免一些旧的 bug。

在 Windows 上更新 Maven 非常简便，只需要下载新的 Maven 安装文件，解压至本地目录，然后更新 `M2_HOME` 环境变量即可。例如，假设 Maven 推出了新版本 3.1，我们将其下载然后解压至目录 `D:\bin\apache-maven-3.1`，接着遵照前一节描述的步骤编辑环境变量 `M2_HOME`，更改其值为 `D:\bin\apache-maven-3.1`。至此，更新就完成了。同理，如果需要使用某一个旧版本的 Maven，也只需要编辑 `M2_HOME` 环境变量指向旧版本的安装目录。

2.2 在基于 UNIX 的系统上安装 Maven

Maven 是跨平台的，它可以在任何一种主流的操作系統上运行。本节将介绍如何在基于 UNIX 的系统（包括 Linux、Mac OS 以及 FreeBSD 等）上安装 Maven。

2.2.1 下载和安装

首先，与在 Windows 上安装 Maven 一样，需要检查 `JAVA_HOME` 环境变量以及 Java 命令，这里对细节不再赘述。命令如下：

```
juven@juven-ubuntu:~$ echo $JAVA_HOME
```

```
juven@ juven-ubuntu:~ $ java -version
```

运行结果如图 2-4 所示。

```
juven@juven-ubuntu:~$ echo $JAVA_HOME
/usr/local/jdk1.6.0_11
juven@juven-ubuntu:~$ java -version
java version "1.6.0_11"
Java(TM) SE Runtime Environment (build 1.6.0_11-b03)
Java HotSpot(TM) Server VM (build 11.0-b16, mixed mode)
```

图 2-4 Linux 中检查 Java 安装

接着到 <http://maven.apache.org/download.html> 下载 Maven 安装文件，如 apache-maven-3.0-bin.tar.gz，然后解压到本地目录：

```
juven@ juven-ubuntu:bin $ tar -xvzf apache-maven-3.0-bin.tar.gz
```

现在已经创建好了一个 Maven 安装目录 apache-maven-3.0。虽然直接使用该目录配置环境变量之后就能使用 Maven 了，但这里的推荐做法是，在安装目录旁平行地创建一个符号链接，以方便日后的升级：

```
juven@ juven-ubuntu:bin $ ln -s apache-maven-3.0 apache-maven
juven@ juven-ubuntu:bin $ ls -l
total 4
lrwxrwxrwx 1 juven juven 18 2009-09-20 15:43 apache-maven -> apache-maven-3.0
drwxr-xr-x 6 juven juven 4096 2009-09-20 15:39 apache-maven-3.0
```

接下来，需要设置 M2_HOME 环境变量指向符号链接 apache-maven-，并且把 Maven 安装目录下的 bin/文件夹添加到系统环境变量 PATH 中：

```
juven@ juven-ubuntu:bin $ export M2_HOME = /home/juven/bin/apache-maven
juven@ juven-ubuntu:bin $ export PATH = $PATH:$M2_HOME/bin
```

一般来说，需要将这两行命令加入到系统的登录 shell 脚本中去，以 Ubuntu 8.10 为例，编辑 ~/.bashrc 文件，添加这两行命令。这样，每次启动一个终端，这些配置就能自动执行。

至此，安装完成。可以运行以下命令检查 Maven 安装：

```
juven@ juven-ubuntu:bin $ echo $M2_HOME
juven@ juven-ubuntu:bin $ mvn -v
```

运行结果如图 2-5 所示。

```
[juven@sonatype02 bin]$ echo $M2_HOME
/home/juven/bin/apache-maven
[juven@sonatype02 bin]$ mvn -v
Apache Maven 3.0.3 (r36106; 2010-09-11 04:13:16-0500)
Java version: 1.6.0_14
Java home: /usr/jdk64/jdk1.6.0_14/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux" version: "2.6.18-128.1.1.el5" arch: "i386" Family: "unix"
[juven@sonatype02 bin]$
```

图 2-5 Linux 中检查 Maven 安装

2.2.2 升级 Maven

在基于 UNIX 的系统上，可以利用符号链接这一工具来简化 Maven 的升级，不必像在 Windows 上那样，每次升级都必须更新环境变量。

前一小节中我们提到，解压 Maven 安装包到本地之后，平行地创建一个符号链接，然后在配置环境变量时引用该符号链接，这样做是为了方便升级。现在，假设需要升级到新的 Maven 3.1 版本，将安装包解压到与前一版本平行的目录下，然后更新符号链接指向 3.1 版的目录便可：

```
juven@ juven-ubuntu:bin$ rm apache-maven
juven@ juven-ubuntu:bin$ ln -s apache-maven-3.1/apache-maven
juven@ juven-ubuntu:bin$ ls -l
total 8
lrwxrwxrwx 1 juven juven 17 2009-09-20 16:13 apache-maven -> apache-maven-3.1/
drwxr-xr-x 6 juven juven 4096 2009-09-20 15:39 apache-maven-3.0
drwxr-xr-x 2 juven juven 4096 2009-09-20 16:09 apache-maven-3.1
```

同理，可以很方便地切换到 Maven 的任意一个版本。现在升级完成了，可以运行 mvn-v 进行检查。

2.3 安装目录分析

前面讲述了如何在各种操作系统中安装和升级 Maven。现在来仔细分析一下 Maven 的安装文件。

2.3.1 M2_HOME

前面讲到设置 M2_HOME 环境变量指向 Maven 的安装目录，本书之后所有使用 M2_HOME 的地方都指代了该安装目录。下面看一下该目录的结构和内容：

```
bin
boot
conf
lib
LICENSE.txt
NOTICE.txt
README.txt
```

- bin: 该目录包含了 mvn 运行的脚本，这些脚本用来配置 Java 命令，准备好 classpath 和相关的 Java 系统属性，然后执行 Java 命令。其中 mvn 是基于 UNIX 平台的 shell 脚本，mvn.bat 是基于 Windows 平台的 bat 脚本。在命令行输入任何一条 mvn 命令时，实际上就是在调用这些脚本。该目录还包含了 mvnDebug 和 mvnDebug.bat 两个文件，同样，前者是 UNIX 平台的 shell 脚本，后者是 Windows 平台的 bat 脚本。那么 mvn 和 mvnDebug 有什么区别和关系呢？打开文件我们就可以看到，两者基本是一样的，只是 mvnDebug 多了一条 MAVEN_DEBUG_OPTS 配置，其作用就是在运行 Maven 时开启 debug，以便调试 Maven 本身。此外，该目录还包含 m2.conf 文件，这是 classworlds

的配置文件，后面会介绍 classworlds。

- ❑ boot: 该目录只包含一个文件，以 maven 3.0 为例，该文件为 `plexus-classworlds-2.2.3.jar`。`plexus-classworlds` 是一个类加载器框架，相对于默认的 `java` 类加载器，它提供了更丰富的语法以方便配置，Maven 使用该框架加载自己的类库。更多关于 `classworlds` 的信息请参考 <http://classworlds.codehaus.org/>。对于一般的 Maven 用户来说，不必关心该文件。
- ❑ conf: 该目录包含了一个非常重要的文件 `settings.xml`。直接修改该文件，就能在机器上全局地定制 Maven 的行为。一般情况下，我们更偏向于复制该文件至 `~/.m2/` 目录下（`~` 表示用户目录），然后修改该文件，在用户范围定制 Maven 的行为。后面将会多次提到 `settings.xml`，并逐步分析其中的各个元素。
- ❑ lib: 该目录包含了所有 Maven 运行时需要的 Java 类库，Maven 本身是分模块开发的，因此用户能看到诸如 `maven-core-3.0.jar`、`maven-model-3.0.jar` 之类的文件。此外，这里还包含一些 Maven 用到的第三方依赖，如 `common-cli-1.2.jar`、`google-collection-1.0.jar` 等。对于 Maven 2 来说，该目录只包含一个如 `maven-2.2.1-uber.jar` 的文件，原本各为独立 JAR 文件的 Maven 模块和第三方类库都被拆解后重新合并到了这个 JAR 文件中。可以说，`lib` 目录就是真正的 Maven。关于该文件，还有一点值得一提的是，用户可以在这个目录中找到 Maven 内置的超级 POM，这一点在 8.5 节详细解释。其他：`LICENSE.txt` 记录了 Maven 使用的软件许可证 `Apache License Version 2.0`；`NOTICE.txt` 记录了 Maven 包含的第三方软件；而 `README.txt` 则包含了 Maven 的简要介绍，包括安装需求及如何安装的简要指令等。

2.3.2 ~/.m2

在讲述该小节之前，我们先运行一条简单的命令：`mvn help:system`。该命令会打印出所有的 Java 系统属性和环境变量，这些信息对我们日常的编程工作很有帮助。这里暂不解释 `help:system` 涉及的语法，运行这条命令的目的是让 Maven 执行一个真正的任务。我们可以从命令行输出看到 Maven 会下载 `maven-help-plugin`，包括 `pom` 文件和 `jar` 文件。这些文件都被下载到了 Maven 本地仓库中。

现在打开用户目录，比如当前的用户目录是 `C:\Users\Juven Xu\`，你可以在 Vista 和 Windows7 中找到类似的用户目录。如果是更早版本的 Windows，该目录应该类似于 `C:\Document and Settings\Juven Xu\`。在基于 UNIX 的系统上，直接输入 `cd` 回车，就可以转到用户目录。为了方便，本书统一使用符号 `~` 指代用户目录。

在用户目录下可以发现 `.m2` 文件夹。默认情况下，该文件夹下放置了 Maven 本地仓库 `.m2/repository`。所有的 Maven 构件都被存储到该仓库中，以方便重用。可以到 `~/.m2/repository/org/apache/maven/plugins/maven-help-plugins/` 目录下找到刚才下载的 `maven-help-plugin` 的 `pom` 文件和 `jar` 文件。Maven 根据一套规则来确定任何一个构件在仓库中的位置，这一点在第 6 章将会详细阐述。由于 Maven 仓库是通过简单文件系统透明地展示给 Maven 用户的，有时

候可以绕过 Maven 直接查看或修改仓库文件，在遇到疑难问题时，这往往十分有用。

默认情况下，`~/.m2` 目录下除了 `repository` 仓库之外就没有其他目录和文件了，不过大多数 Maven 用户需要复制 `M2_HOME/conf/settings.xml` 文件到 `~/.m2/settings.xml`。这是一条最佳实践，我们将在 2.7 小节详细解释。

2.4 设置 HTTP 代理

有时候你所在的公司基于安全因素考虑，要求你使用通过安全认证的代理访问因特网。这种情况下，就需要为 Maven 配置 HTTP 代理，才能让它正常访问外部仓库，以下载所需要的资源。

首先确认自己无法直接访问公共的 Maven 中央仓库，直接运行命令 `ping repol.maven.org` 可以检查网络。如果真的需要代理，先检查一下代理服务器是否畅通。比如现在有一个 IP 地址为 218.14.227.197，端口为 3128 的代理服务，我们可以运行 `telnet 218.14.227.197 3128` 来检测该地址的该端口是否畅通。如果得到出错信息，需要先获取正确的代理服务信息；如果 `telnet` 连接正确，则输入 `ctrl +]`，然后 `q`，回车，退出即可。

检查完毕之后，编辑 `~/.m2/settings.xml` 文件（如果没有该文件，则复制 `$M2_HOME/conf/settings.xml`）。添加代理配置如下：

```
<settings>
...
<proxies>
  <proxy>
    <id>my-proxy </id>
    <active>true </active>
    <protocol>http </protocol>
    <host>218.14.227.197 </host>
    <port>3128 </port>
    <!--
    <username>*** </username>
    <password>*** </password>
    <nonProxyHosts>repository.mycom.com|*.google.com </nonProxyHosts>
    -->
  </proxy>
</proxies>
...
</settings>
```

这段配置十分简单，`proxies` 下可以有多个 `proxy` 元素，如果声明了多个 `proxy` 元素，则默认情况下第一个被激活的 `proxy` 会生效。这里声明了一个 `id` 为 `my-proxy` 的代理，`active` 的值为 `true` 表示激活该代理，`protocol` 表示使用的代理协议，这里是 `http`。当然，最重要的是指定正确的主机名（`host` 元素）和端口（`port` 元素）。上述 XML 配置中注释掉了 `username`、`password`、`nonProxyHost` 几个元素。当代理服务需要认证时，就需要配置 `username` 和 `password`。`nonProxyHost` 元素用来指定哪些主机名不需要代理，可以使用“`|`”符号来分隔多个主机名。此外，该配置也支持通配符，如 `*.google.com` 表示所有以 `google.com` 结尾的域名

访问都不要通过代理。

2.5 安装 m2eclipse

Eclipse 是一款非常优秀的 IDE。除了基本的语法标亮、代码补齐、XML 编辑等基本功能外，最新版的 Eclipse 还能很好地支持重构，并且集成了 JUnit、CVS、Mylyn 等各种流行工具。可惜 Eclipse 默认没有集成对 Maven 的支持。幸运的是，由 Maven 之父 Jason Van Zyl 创立的 Sonatype 公司建立了 m2eclipse 项目。这是 Eclipse 下的一款十分强大的 Maven 插件，可以访问 <http://m2eclipse.sonatype.org/> 了解更多该项目的信息。

本小节将介绍如何安装 m2eclipse 插件，后续的章节会逐步介绍 m2eclipse 插件的使用。

现在以 Eclipse 3.6 为例逐步讲解 m2eclipse 的安装。启动 Eclipse 之后，在菜单栏中选择 Help，然后选择 Install New Software...，接着你会看到一个 Install 对话框。单击 Work with: 字段边上的 Add 按钮，会弹出一个新的 Add Repository 对话框。在 Name 字段中输入 m2e，在 Location 字段中输入 <http://m2eclipse.sonatype.org/sites/m2e>，然后单击 OK 按钮。Eclipse 会下载 m2eclipse 安装站点上的资源信息。等待资源载入完成之后，再将其全部展开，就能看到图 2-6 所示的界面。

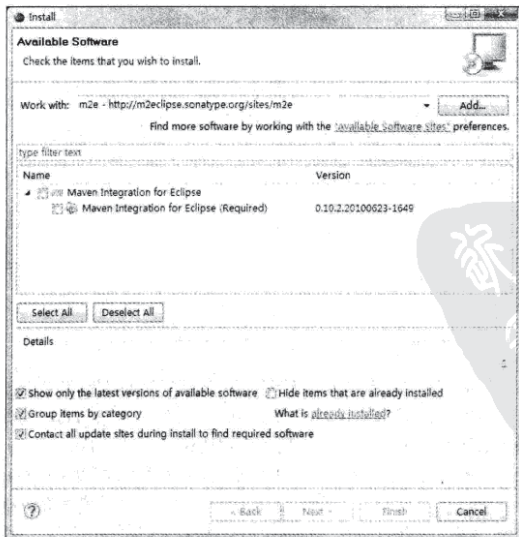


图 2-6 m2eclipse 的核心安装资源列表

图 2-6 显示了 m2eclipse 的核心模块 Maven Integration for Eclipse (Required)，选择后单击 Next 按钮，Eclipse 会自动计算模块间依赖，然后给出一个将被安装的模块列表。确认无误后，继续单击 Next 按钮，这时会看到许可证信息。m2eclipse 使用的开源许可证是 Eclipse Public License v1.0，选择 I accept the terms of the license agreements，然后单击 Finish 按钮，接着就耐心等待 Eclipse 下载安装这些模块，如图 2-7 所示。

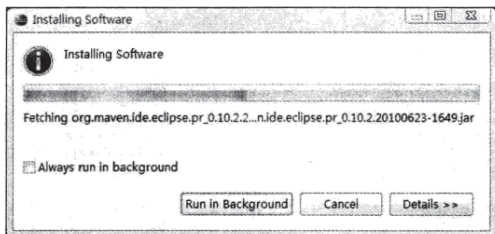


图 2-7 m2eclipse 安装进度

除了核心组件之外，m2eclipse 还提供了一组额外组件，主要是为了方便与其他工具如 Subversion 进行集成，这些组件的安装地址为 <http://m2eclipse.sonatype.org/sites/m2e-extras>。使用前面类似的安装方法，可以看到图 2-8 所示的组件列表。

下面简单解释一下这些组件的用途。

1. 重要的

- ❑ Maven SCM handler for Subclipse (Optional)：Subversion 是非常流行的版本管理工具。该模块能够帮助我们直接从 Subversion 服务器签出 Maven 项目，不过前提是需要首先安装 Subclipse (<http://subclipse.tigris.org/>)。
- ❑ Maven SCM Integration (Optional)：Eclipse 环境中 Maven 与 SCM 集成核心的模块。它利用各种 SCM 工具如 SVN 实现 Maven 项目的签出和具体化等操作。

2. 不重要的

- ❑ Maven issue tracking configurator for Mylyn 3. x (Optional)：该模块能够帮助我们使用 POM 中的缺陷跟踪系统信息连接 Mylyn 至服务器。
- ❑ Maven SCM handler for Team/CVS (Optional)：该模块帮助我们 from CVS 服务器签出 Maven 项目，如果还在使用 CVS，就需要安装它。
- ❑ Maven Integration for WTP (Optional)：使用该模块可以让 Eclipse 自动读取 POM 信息并配置 WTP 项目。
- ❑ M2Eclipse Extensions Development Support (Optional)：用来支持扩展 m2eclipse，一般用户不会用到。

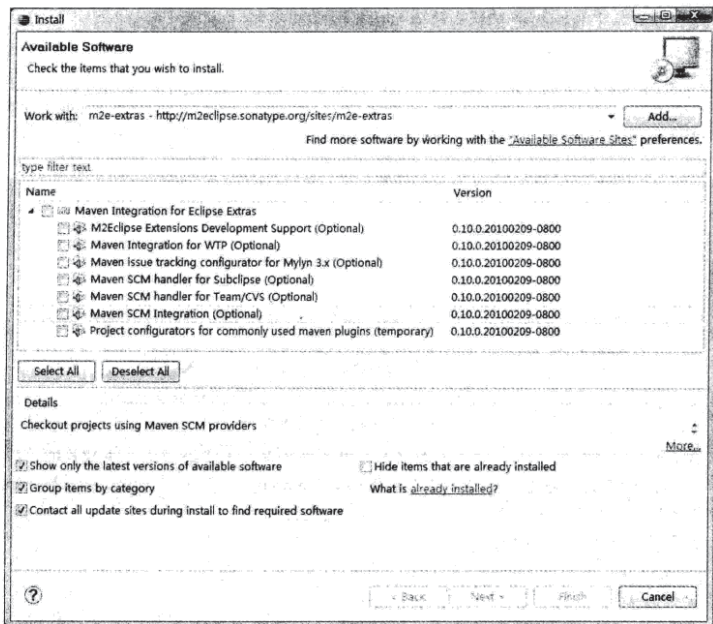


图 2-8 m2eclipse 的额外组件安装资源列表

☐ Project configurators for commonly used maven plugins (temporary): 一个临时的组件，用来支持一些 Maven 插件与 Eclipse 的集成，建议安装。

读者可以根据自己的需要安装相应组件，具体步骤这里不再赘述。

待安装完毕后，重启 Eclipse。现在来验证一下 m2eclipse 是否正确安装了。首先，单击菜单栏中的 Help，然后选择 About Eclipse。在弹出的对话框中，单击 Installation Details 按钮，会得到一个对话框。在 Installed Software 标签中，检查刚才选择的模块是否在这个列表中，如图 2-9 所示。

如果一切没问题，再检查一下 Eclipse 现在是否已经支持创建 Maven 项目。依次单击菜单栏中的 File→New→Other，在弹出的对话框中，找到 Maven 一项，再将其展开，应该能够看到图 2-10 所示的对话框。

如果一切正常，说明 m2eclipse 已经正确安装了。

最后，关于 m2eclipse 的安装需要提醒的一点是，你可能会在使用 m2eclipse 时遇到类似这样的错误：

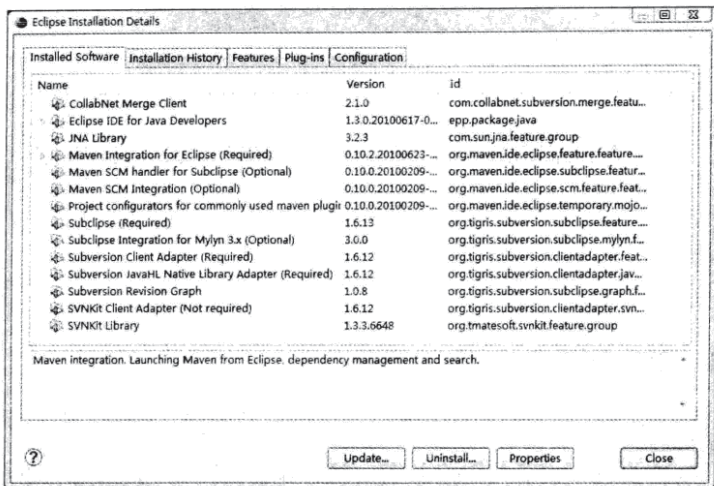


图 2-9 m2eclipse 安装结果

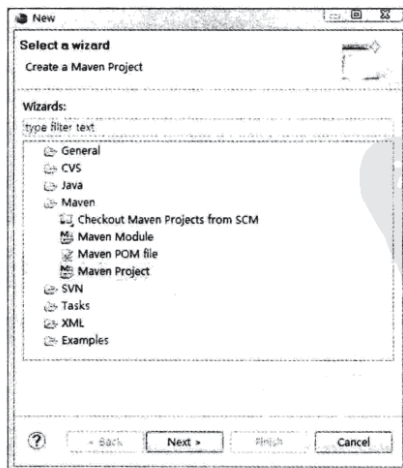


图 2-10 Eclipse 中创建 Maven 项目向导

09-10-6 上午 01 时 14 分 49 秒: Eclipse is running in a JRE, but a JDK is required. Some Maven plugins may not work when importing projects or updating source folders.

这是因为 Eclipse 默认是运行在 JRE 上的, 而 m2eclipse 的一些功能要求使用 JDK。解决方法是配置 Eclipse 安装目录的 eclipse.ini 文件, 添加 vm 配置指向 JDK。例如:

```
--launcher.XXMaxPermSize
256m
-vm
D:\java\jdk1.6.0_07\bin\javaw.exe
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms128m
-Xmx256m
```

2.6 安装 NetBeans Maven 插件

本小节会先介绍如何在 NetBeans 上安装 Maven 插件, 后面的章节中还会介绍 NetBeans 中具体的 Maven 操作。

如果正在使用 NetBeans 6.7 及以上版本, 那么 Maven 插件已经预装了。你可以检查 Maven 插件安装, 单击菜单栏中的工具, 接着选择插件, 在弹出的插件对话框中选择已安装标签, 应该能够看到 Maven 插件, 如图 2-11 所示。

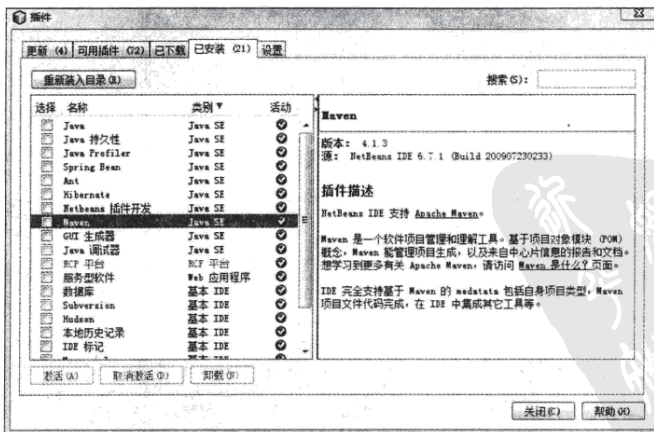


图 2-11 已安装的 NetBeans Maven 插件

如果在使用 NetBeans 6.7 之前的版本,或者由于某些原因 NetBeans Maven 插件被卸载了,那么就需要安装 NetBeans Maven 插件。下面以 NetBeans 6.1 为例,介绍 Maven 插件的安装。

同样,单击菜单栏中的工具,选择插件,在弹出的插件对话框中选择可用插件标签,接着在右边的搜索框内输入 Maven,这时会在左边的列表中看到一个名为 Maven 的插件。选择该插件,然后单击下面的“安装”按钮,如图 2-12 所示。

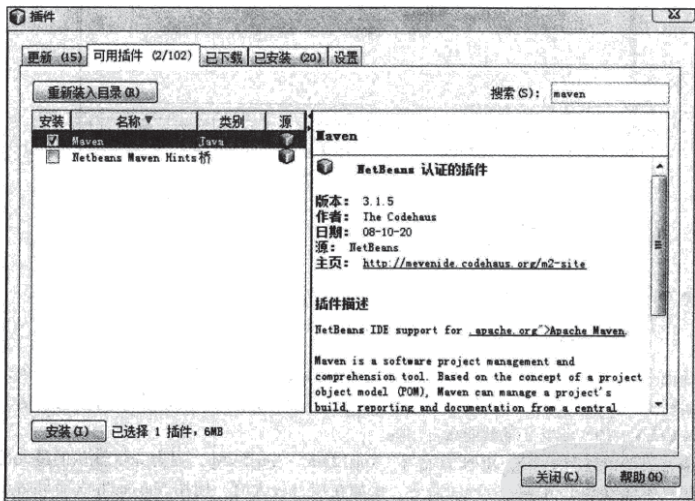


图 2-12 安装 NetBeans Maven 插件

接着在随后的对话框中根据提示操作,阅读相关许可证并接受,NetBeans 会自动帮我们下载并安装 Maven 插件,结束之后会提示安装完成。之后再单击插件对话框中的已安装标签,就能看到已经激活的 Maven 插件。

最后,为了确认 Maven 插件确实已经正确安装了,可以看一下 NetBeans 是否已经拥有创建 Maven 项目的相关菜单。在菜单栏中选择文件,然后选择新建项目,这时应该能够看到项目类别中有 Maven 一项。选择该类别,右边会相应地显示 Maven 项目和基于现有 POM 的 Maven 项目,如图 2-13 所示。

如果能看到类似的对话框,说明 NetBeans Maven 已经正确安装了。

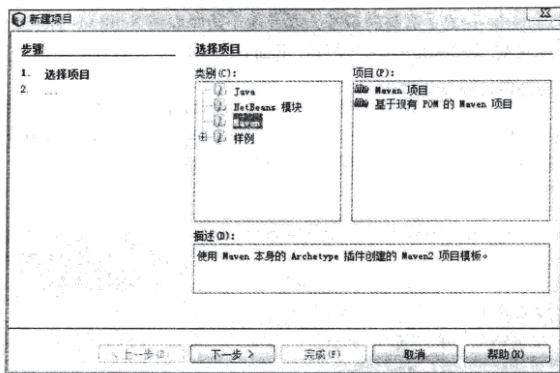


图 2-13 NetBeans 中创建 Maven 项目向导

2.7 Maven 安装最佳实践

本节介绍一些在安装 Maven 过程中不是必须的，但十分有用的实践。

2.7.1 设置 MAVEN_OPTS 环境变量

前面介绍 Maven 安装目录时我们了解到，运行 mvn 命令实际上是执行了 Java 命令，既然是运行 Java，那么运行 Java 命令可用的参数当然也应该在运行 mvn 命令时可用。这个时候，MAVEN_OPTS 环境变量就能派上用场。

通常需要设置 MAVEN_OPTS 的值为 -Xms128m -Xmx512m，因为 Java 默认的最大可用内存往往不能够满足 Maven 运行的需要，比如在项目较大时，使用 Maven 生成项目站点需要占用大量的内存，如果没有该配置，则很容易得到 java.lang.OutOfMemoryError。因此，一开始就配置该变量是推荐的做法。

关于如何设置环境变量，请参考前面设置 M2_HOME 环境变量的做法，尽量不要直接修改 mvn.bat 或者 mvn 这两个 Maven 执行脚本文件。因为如果修改了脚本文件，升级 Maven 时就不得不再次修改，一来麻烦，二来容易忘记。同理，应该尽可能地不去修改任何 Maven 安装目录下的文件。

2.7.2 配置用户范围 settings.xml

Maven 用户可以选择配置 \$ M2_HOME/conf/settings.xml 或者 ~/.m2/settings.xml。前者是全局范围的，整台机器上的所有用户都会直接受到该配置的影响，而后者是用户范围的，只有当前用户才会受到该配置的影响。

推荐使用用户范围的 settings.xml，主要是为了避免无意识地影响到系统中的其他用户。如果有切实的需求，需要统一系统中所有用户的 settings.xml 配置，当然应该使用全局范围的 settings.xml。

除了影响范围这一因素，配置用户范围 settings.xml 文件还便于 Maven 升级。直接修改 conf 目录下的 settings.xml 会导致 Maven 升级不便，每次升级到新版本的 Maven，都需要复制 settings.xml 文件。如果使用 ~/.m2 目录下的 settings.xml，就不会影响到 Maven 安装文件，升级时就不需要触动 settings.xml 文件。

2.7.3 不要使用 IDE 内嵌的 Maven

无论 Eclipse 还是 NetBeans，当集成 Maven 时，都会安装上一个内嵌的 Maven，这个内嵌的 Maven 通常会比较新，但不一定很稳定，而且往往也会和在命令行使用的 Maven 不是同一个版本。这里又会出现两个潜在的问题：首先，较新版本的 Maven 存在很多不稳定因素，容易造成一些难以理解的问题；其次，除了 IDE，也经常还会使用命令行的 Maven，如果版本不一致，容易造成构建行为的不一致，这是我们所不希望看到的。因此，应该在 IDE 中配置 Maven 插件时使用与命令行一致的 Maven。

在 m2eclipse 环境中，单击菜单栏中的 Windows，然后选择 Preferences，在弹出的对话框中，展开左边的 Maven 项，选择 Installation 子项，在右边的面板中，能够看到有一个默认的 Embedded Maven 安装被选中了。单击 Add... 按钮，然后选择 Maven 安装目录 M2_HOME，添加完毕之后选择这一个外部的 Maven，如图 2-14 所示。

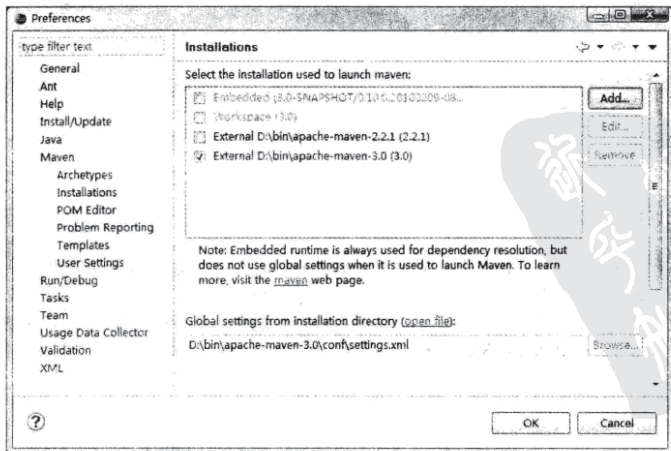


图 2-14 在 Eclipse 中使用外部 Maven

NetBeans Maven 插件默认会侦测 PATH 环境变量，因此会直接使用与命令行一致的 Maven 环境。依次单击菜单栏中的工具→选项→其他→Maven 标签栏，就能看到图 2-15 所示的配置。

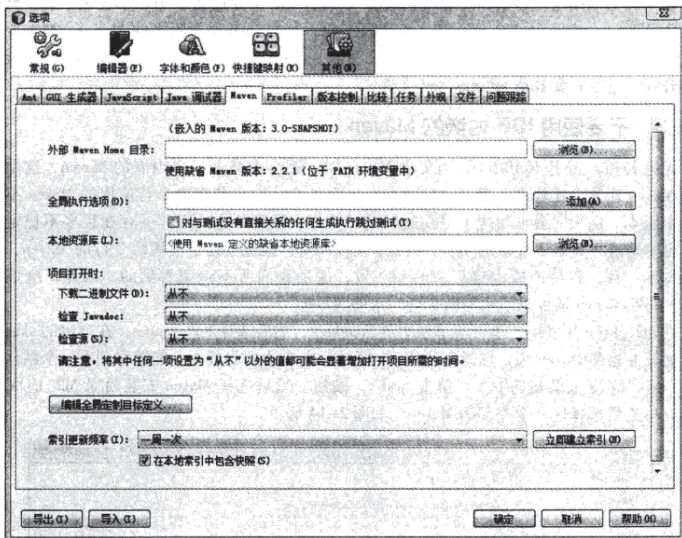


图 2-15 在 NetBeans 中使用外部 Maven

2.8 小结

本章详细介绍了在各种操作系统平台上安装 Maven，并对 Maven 安装目录进行了深入的分析，在命令行的基础上，又进一步介绍了 Maven 与主流 IDE Eclipse 及 NetBeans 的集成。本章最后还介绍了一些与 Maven 安装相关的最佳实践。下一章会创建一个 Hello World 项目，带领读者配置和构建 Maven 项目。