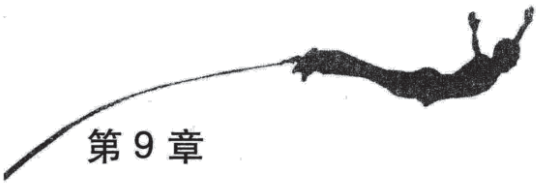

第 9 章

使用 Nexus 创建私服

本章内容

- Nexus 简介
- 安装 Nexus
- Nexus 的仓库与仓库组
- Nexus 的索引与构件搜索
- 配置 Maven 从 Nexus 下载构件
- 部署构件至 Nexus
- Nexus 的权限管理
- Nexus 的调度任务
- 其他私服软件
- 小结

私服不是Maven的核心概念，它仅仅是一种衍生出来的特殊的Maven仓库，本书已经在6.3.4节解释了其概念和用途，然而这还不够。通过建立自己的私服，就可以降低中央仓库负荷、节省外网带宽、加速Maven构建、自己部署构件等，从而高效地使用Maven。

有三种专门的Maven仓库管理软件可以用来帮助大家建立私服：Apache基金会的Archiva、JFrog的Artifactory和Sonatype的Nexus。其中，Archiva是开源的，而Artifactory和Nexus的核心也是开源的，因此读者可以自由选择使用。笔者作为Nexus开发团队的成员，自然十分推崇Nexus。事实上，Nexus也是当前最流行的Maven仓库管理软件。

本章将介绍Nexus的主要功能，并结合大量图片帮助读者快速地建立起自己的Maven私服。

9.1 Nexus简介

2005年12月，Tamas Cservenak由于受不了匈牙利电信ADSL的低速度，开始着手开发Proximity——一个很简单的Web应用。它可以代理并缓存Maven构件，当Maven需要下载构件的时候，就不需要反复依赖于ADSL。到2007年，Sonatype邀请Tamas参与创建一个更酷的Maven仓库管理软件，这就是后来的Nexus。

Nexus团队的成员来自世界各地，它也从社区收到了大量反馈和帮助，在写本书的时候，Nexus刚发布1.7.2版本，它也正健康快速地成长着。

Nexus分为开源版和专业版，其中开源版本基于GPLv3许可证，其特性足以满足大部分Maven用户的需要。以下是一些Nexus开源版本的特性：

- 较小的内存占用（最少仅为28MB）
- 基于ExtJS的友好界面
- 基于Restlet的完全REST API
- 支持代理仓库、宿主仓库和仓库组
- 基于文件系统，不需要数据库
- 支持仓库索引和搜索
- 支持从界面上传Maven构件
- 细粒度的安全控制

Nexus专业版本是需要付费购买的，除了开源版本的所有特性之外，它主要包含一些企业安全控制、发布流程控制等需要的特性。感兴趣的读者可以访问该地址了解详情：http://www.sonatype.com/products/nexus/community。

9.2 安装Nexus

Nexus是典型的Java Web应用，它有两种安装包，一种是包含Jetty容器的Bundle包，另一种是不包含Web容器的war包。

9.2.1 下载Nexus

首先从http://nexus.sonatype.org/downloads/下载最新版本的Nexus，在本书编写的时候，

Nexus 的最新版本为 1.7.2。读者可以根据需要下载 Bundle 包 nexus-webapp-1.7.2-bundle.tar.gz 和 nexus-webapp-1.7.2-bundle.zip，或者 war 包 nexus-webapp-1.7.2.war。

9.2.2 Bundle 方式安装 Nexus

Nexus 的 Bundle 自带了 Jetty 容器，因此用户不需要额外的 Web 容器就能直接启动 Nexus。首先将 Bundle 文件解压（例如笔者将其解压到 D:\bin\目录），这时就会得到如下两个子目录：

- **nexus-webapp-1.7.2/**：该目录包含了 Nexus 运行所需要的文件，如启动脚本、依赖 jar 包等。
- **sonatype-work/**：该目录包含 Nexus 生成的配置文件、日志文件、仓库文件等。

其中，第一个目录是运行 Nexus 所必需的，而且所有相同版本 Nexus 实例所包含的该目录内容都是一样的。而第二个目录不是必须的，Nexus 会在运行的时候动态创建该目录，不过它的内容对于各个 Nexus 实例是不一样的，因为不同用户在不同机器上使用的 Nexus 会有不同的配置和仓库内容。当用户需要备份 Nexus 的时候，默认备份 sonatype-work/目录，因为该目录包含了用户特定的内容，而 nexus-webapp-1.7.2 目录下的内容是可以从安装包直接获得的。

用户只需要调用对应操作系统的脚本就可以启动 Nexus，这里介绍主流的在 Windows 和 Linux 平台上启动 Nexus 的方式。

在 Windows 操作系统上，用户需进入 nexus-webppp-1.7.2/bin/jsw/windows-x86-32/子目录，然后直接运行 nexus.bat 脚本就能启动 Nexus。如果看到如下输出，就说明启动成功了：

```
jvm 1        | 2010-09-02 15:27:11 INFO  [er_start_runner] -o.s.n.DefaultNexus -
Started Nexus (version 1.7.2 OSS)
jvm 1        | 2010-09-02 15:27:11 INFO  [er_start_runner] -o.s.n.p.a.DefaultAt ~
-Attribute storage directory does not exists, creating it here
: . \.. \sonatype-work \nexus \proxy \attributes
jvm 1        | 2010-09-02 15:27:11 WARN [er_start_runner] -o.s.s.m.s.FileModel ~ -
No configuration file in place, copying the default one and c
ontinuing with it.
jvm 1        | 2010-09-02 15:27:11 INFO [er_start_runner] -o.s.s.m.s.FileModel ~ -
Loading Security configuration from D:\bin\nexus-oss-webapp-1
.7.2\. \.. \sonatype-work \nexus \conf \security.xml
jvm 1        | 2010-09-02 15:27:11 INFO [er_start_runner] -o.s.s.w.PlexusConfi ~ -
SecurityManager with role ='org.sonatype.security.PlexusSecuri
tyManager'and roleHint ='web'found in Plexus.
jvm 1        | 2010-09-02 15:27:12 INFO [er_start_runner] -org.mortbay.log -Star-
ted SelectChannelConnector@ 0.0.0.0:8081
```

这时，打开浏览器访问 http://localhost:8081/nexus/就能看到 Nexus 的界面，如图 9-1 所示。

要停止 Nexus，可以在命令行按 Ctrl + C 键。

在 nexus-webppp-1.7.2/bin/jsw/windows-x86-32/目录下还有其他一些脚本：

- **Installnexus.bat**：将 Nexus 安装成 Windows 服务。
- **Uninstallnexus.bat**：卸载 Nexus Windows 服务。
- **Startnexus.bat**：启动 Nexus Windows 服务。

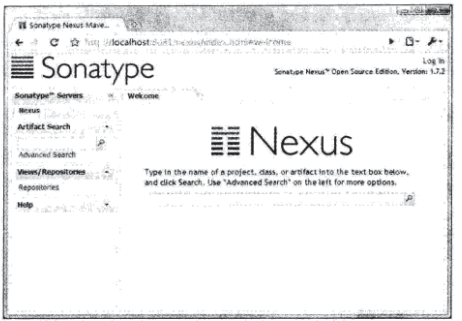

图 9-1　Nexus 的初始界面

❑ **Stopnexus. bat**：停止 Nexus Windows 服务。

❑ **Pausenexus. bat**：暂停 Nexus Windows 服务。

❑ **Resumenexus. bat**：恢复暂停的 Nexus Windows 服务。

借助 Windows 服务，用户就可以让 Nexus 伴随着 Windows 自动启动，非常方便。

在 Linux 系统上启动 Nexus 也非常方便，例如笔者使用 Ubuntu 32 位系统，那么只需要进入到 nexus-webapp-1. 7. 2/bin/jsw/linux-x86-32/，然后运行如下命令：

```
$ ./nexus console
```

同样地，读者可以看到 Nexus 启动的命令行输出，并且可以使用 Ctrl + C 键停止 Nexus。除了 console 之外，Nexus 的 Linux 脚本还提供如下的命令：

❑ **./nexus start**：在后台启动 Nexus 服务。

❑ **./nexus stop**：停止后台的 Nexus 服务。

❑ **./nexus status**：查看后台 Nexus 服务的状态。

❑ **./nexus restart**：重新启动后台的 Nexus 服务。

关于 Bundle 安装的一个常见问题是端口冲突。Nexus Bundle 默认使用的端口是 8081，如果该端口已经被其他应用程序占用，或者你想使用 80 端口开放 Nexus 服务，则编辑文件 nexus-webapp-1. 7. 2/conf/plexus. properties，找到属性 application-port，按需要将默认值 8081 改成其他端口号，然后保存该文件，重启 Nexus 便可。

9. 2. 3　WAR 方式安装 Nexus

除了 Bundle，Nexus 还提供一个可以直接部署到 Web 容器中的 war 包。该 war 包支持主流的 Web 容器，如 Tomcat、Glassfish、Jetty 和 Resin。

以 Tomcat 6 为例，笔者在 Vista 机器上的目录为 D:\bin\apache-tomcat-6. 0. 20\，那么只需要复制 Nexus war 包至 Tomcat 的部署目录 D:\bin\apache-tomcat-6. 0. 20\webapps\nex-

us. war，然后转到 D:\bin\apache-tomcat-6.0.20\bin\目录，运行 startup.bat。这时，读者可以从 Tomcat 的 console 输出中看到它部署 nexus. war。

待 Tomcat 启动完成后，访问 http://localhost:8080/nexus/就能看到 Nexus 的界面了。

9.2.4 登录 Nexus

Nexus 拥有全面的权限控制功能，默认的 Nexus 访问都是匿名的，而匿名用户仅包含一些最基本的权限，要全面学习和管理 Nexus，就必须以管理员方式登录。可以单击界面右上角的 Log In 进行登录，Nexus 的默认管理员用户名和密码为 admin/admin123，如图 9-2 所示。

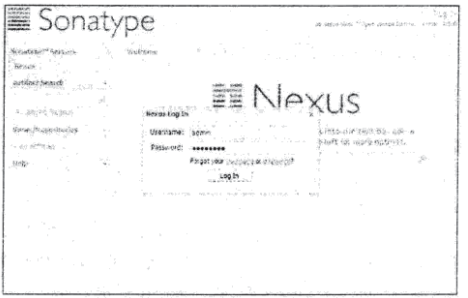

图 9-2 使用默认用户名登录 Nexus

9.3 Nexus 的仓库与仓库组

作为 Maven 仓库服务软件，仓库自然是 Nexus 中最重要的概念。Nexus 包含了各种类型的仓库概念，包括代理仓库、宿主仓库和仓库组等。每一种仓库都提供了丰富实用的配置参数，方便用户根据需要进行定制。

9.3.1 Nexus 内置的仓库

在具体介绍每一种类型的仓库之前，先浏览一下 Nexus 内置的一些仓库。单击 Nexus 界面左边导航栏中的 Repositories 链接，就能在界面右边看到如图 9-3 所示的内容。

这个列表已经包含了所有类型的 Nexus 仓库。从中可以看到仓库有四种类型：group（仓库组）、hosted（宿主）、proxy（代理）和 virtual（虚拟）。每个仓库的格式为 maven2 或者 maven1。此外，仓库还有一个属性为 Policy（策略），表示该仓库为发布（Release）版本仓库还是快照（Snapshot）版本仓库。最后两列的值为仓库的状态和路径。

下面解释一下各个仓库的用途。由于本书不涉及 Maven 1 的内容，maven1 格式的仓库会被省略。此外，由于虚拟类型仓库的作用实际上是动态地将仓库内容格式转换，换言之

也是为了服务 maven1 格式，因此也被省略。

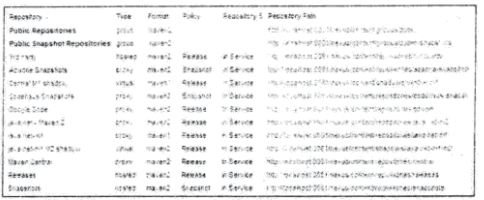

图 9-3　Nexus 内置的仓库列表

- **Maven Central**：该仓库代理 Maven 中央仓库，其策略为 Release，因此只会下载和缓存中央仓库中的发布版本构件。
- **Releases**：这是一个策略为 Release 的宿主类型仓库，用来部署组织内部的发布版本构件。
- **Snapshots**：这是一个策略为 Snapshot 的宿主类型仓库，用来部署组织内部的快照版本构件。
- **3rd party**：这是一个策略为 Release 的宿主类型仓库，用来部署无法从公共仓库获得的第三方发布版本构件。
- **Apache Snapshots**：这是一个策略为 Snapshot 的代理仓库，用来代理 Apache Maven 仓库的快照版本构件。
- **Codehaus Snapshots**：这是一个策略为 Snapshot 的代理仓库，用来代理 Codehaus Maven 仓库的快照版本构件。
- **Google Code**：这是一个策略为 Release 的代理仓库，用来代理 Google Code Maven 仓库的发布版本构件。
- **java. net - Maven 2**：这是一个策略为 Release 的代理仓库，用来代理 java. net Maven 仓库的发布版本构件。
- **Public Repositories**：该仓库组将上述所有策略为 Release 的仓库聚合并通过一致的地址提供服务。
- **Public Snapshot Repositories**：该仓库组将上述所有策略为 Snapshot 的仓库聚合并通过一致的地址提供服务。

举一个简单的例子。假设某公司建立了 Maven 项目 X，公司内部建立了 Nexus 私服，为所有 Maven 项目提供服务。项目 X 依赖于很多流行的开源类库如 JUnit 等，这些构件都能从 Maven 中央仓库获得，因此 Maven Central 代理仓库会被用来代理中央仓库的内容，并在私服上缓存下来，X 还依赖于某个 Google Code 的项目，其构件在中央仓库中不存在，只存在于 Google Code 的仓库中，因此上述列表中的 Google Code 代理仓库会被用来代理并缓存这样的构件。X 还依赖于 Oracle 的 JDBC 驱动，由于版权的因素，该类库无法从公共仓库获得，因此公司管理员将其部署到 3rd party 宿主仓库中，供 X 使用。X 的快照版本构件成功后，

会被部署到 Snapshots 宿主仓库中，供其他项目使用。当 X 发布正式版本的时候，其构件会被部署到 Release 宿主仓库中。由于 X 用到了上述列表中的很多仓库，为每个仓库声明 Maven 配置又比较麻烦，因此可以直接使用仓库组 Public Repositories 和 Public Snapshot Repositories，当 X 需要 JUnit 的时候，它直接从 Public Repositories 下载，Public Repositories 会选择 Maven Central 提供实际的内容。

9.3.2 Nexus 仓库分类的概念

为了帮助读者理解宿主仓库、代理仓库和仓库组的概念，图 9-4 用更为直观的方式展现了它们的用途和区别。

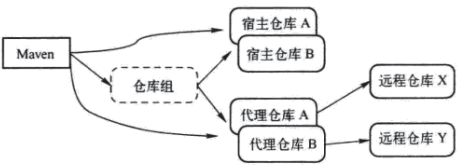

图 9-4 各种类型的 Nexus 仓库

从图 9-4 中可以看到，Maven 可以直接从宿主仓库下载构件；Maven 也可以从代理仓库下载构件，而代理仓库会间接地从远程仓库下载并缓存构件；最后，为了方便，Maven 可以从仓库组下载构件，而仓库组没有实际内容（图中用虚线表示），它会转向其包含的宿主仓库或者代理仓库获得实际构件的内容。

9.3.3 创建 Nexus 宿主仓库

要创建一个宿主仓库，首先单击界面左边导航栏中的 Repositories 链接，在右边的面板中，选择 Add，接着在下拉菜单中选择 Hosted Repository，就会看到图 9-5 所示的配置界面。

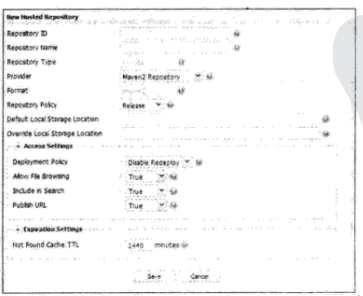

图 9-5 创建 Nexus 宿主仓库

根据自己的需要填入仓库的ID和名称，下一字段Repository Type表示该仓库的类型。Provider用来确定该仓库的格式。一般来说，选择默认的Maven2 Repository。然后是Repository Policy，读者可以根据自己的需要来配置该仓库是发布版构件仓库还是快照版构件仓库。Default Local Storage Location表示该仓库的默认存储目录，图中该字段的值为空，待仓库创建好之后，该值就会成为基于sonatype-work的一个文件路径，如sonatype-work/nexus/storage/repository-id/，Override Local Storage Location可以用来配置自定义的仓库目录位置。

在Access Settings小组中，Deployment Policy用来配置该仓库的部署策略，选项有只读（禁止部署）、关闭重新部署（同一构件只能部署一次）以及允许重新部署。Allow File Browsing表示是否允许浏览仓库内容，一般选True。每个仓库（包括代理仓库和仓库组）都有一个Browse Storage选项卡，用户以树形结构浏览仓库存储文件的内容，如图9-6所示。Include in Search表示是否对该仓库进行索引并提供搜索，我们会在9.4节详细讨论索引和搜索。Publish URL用来控制是否通过URL提供服务，如果选False，当访问该仓库的地址时，会得到HTTP 404 Not Found错误。配置中最后的Not Found Cache TTL表示当一个文件没有找到后，缓存这一不存在信息的时间。以默认值1440分钟为例，如果某文件不存在，那么在之后的1440分钟内，如果Nexus再次得到该文件的请求，它将直接返回不存在信息，而不会查找文件系统。这么做是为了避免重复的文件查找操作以提升性能。

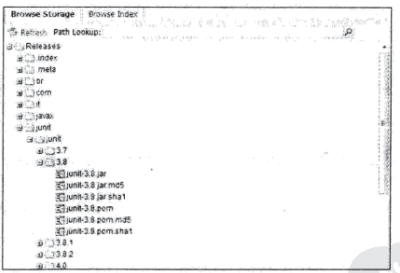

图9-6 浏览Nexus仓库内容

9.3.4 创建Nexus代理仓库

首先单击界面左边导航栏中的Repositories链接，在右边的面板中，选择Add...，接着在下拉菜单中选择Proxy Repository，就会看到图9-7所示的配置界面。

仓库的ID、名称、Provider、Format、Policy、默认本地存储位置和覆盖本地存储位置等配置前面都已提过，这里不再赘述。需要注意的是，这里的Repository Type的值为proxy。

对于代理仓库来说，最重要的是远程仓库的地址，即Remote Storage Location，用户必须在这里输入有效的值。Download Remote Indexes表示是否下载远程仓库的索引，有些远程

仓库拥有索引，下载其索引后，即使没有缓存远程仓库的构件，用户还是能够在本地搜索和浏览那些构件的基本信息。Checksum Policy 配置校验和出错时的策略，用户可以选择忽略、记录警告信息或者拒绝下载。当远程仓库需要认证的时候，这里的 Authentication 配置就能派上用处。

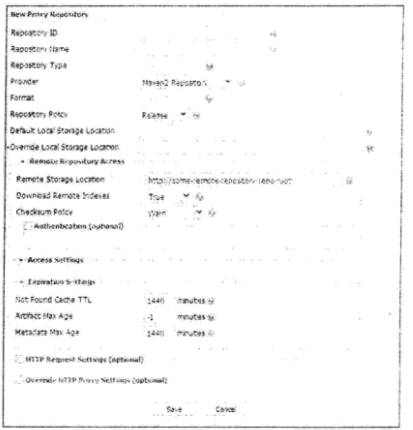

图 9-7 创建 Nexus 代理仓库

Access Settings 的配置与宿主仓库类似，在此不再赘述。Expiration Settings 较宿主仓库多了 Artifact Max Age 和 Metadata Max Age。其中，前者表示构件缓存的最长时间，后者表示仓库元数据文件缓存的最长时间。对于发布版仓库来说，Artifact Max Age 默认值为 -1，表示构件缓存后就一直保存着，不再重新下载。对于快照版仓库来说，Artifact Max Age 默认值为 1440 分钟，表示每隔一天重新缓存代理的构件。

配置中最后两项为 HTTP Request Settings 和 Override HTTP Proxy Settings，其中前者用来配置 Nexus 访问远程仓库时 HTTP 请求的参数，后者用来配置 HTTP 代理。

9.3.5 创建 Nexus 仓库组

要创建一个仓库组，首先单击界面左边导航栏中的 Repositories 链接，在右边的面板中，选择 Add，接着在下拉菜单中选择 Repository Group，就会看到图 9-8 所示的配置界面。

配置中的 ID、Name 等信息这里不再赘述。需要注意的是，仓库组没有 Release 和 Snapshot 的区别，这不同于宿主仓库和代理仓库。在配置界面中，用户可以非常直观地选择 Nexus 中仓库，将其聚合成一个虚拟的仓库组。注意，仓库组所包含的仓库的顺序决定了仓库组遍历其所含仓库的次序，因此最好将常用的仓库放在前面，当用户从仓库组下载构件

的时候，就能够尽快地访问到包含构件的仓库。

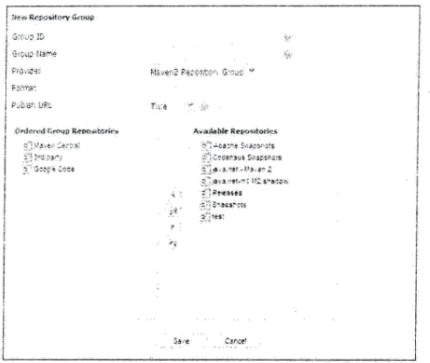

图 9-8　创建 Nexus 仓库组

9.4　Nexus 的索引与构件搜索

既然 Nexus 能够维护宿主仓库并代理缓存远程仓库（如 Maven 中央库），那么一个简单的需求就自然浮现出来了，这就是搜索。Maven 中央库有几十万构件供用户使用，但有时我们往往仅仅知道某个关键字，如 Ehcache，而不知道其确切的 Maven 坐标。Nexus 通过维护仓库的索引来提供搜索功能，能在很大程度上方便 Maven 用户定位构件坐标。

6.8.1 节介绍了 Sonatype 提供的在线免费搜索服务，其实用户可以很方便地自己维护一个 Nexus 实例，并提供搜索服务。

为了能够搜索 Maven 中央库，首先需要设置 Nexus 中的 Maven Central 代理仓库下载远程索引，如图 9-9 所示。需要注意的是，默认这个配置的值是关闭的。此外，由于中央库的内容比较多，因此其索引文件比较大，Nexus 下载该文件也需要比较长的时间，读者还需要耐心等待。

可以想象到，Nexus 在后台运行了一个任务来下载中央仓库的索引，幸运的是，用户可以通过界面直接观察这一任务的状态。单击界面左边导航栏中的 Scheduled Tasks 链接后，用户就能在界面的右边看到系统的调度任务，如果 Nexus 正在下载中央仓库的索引，用户就能看到图 9-10 所示的一个任务，其状态为 RUNNING。在索引下载完毕之后，该任务就会消失。

有了索引，用户即可搜索 Maven 构件了。Nexus 界面左边导航栏有一个快捷搜索框，在其中输入关键字后，单击搜索按钮就能快速得到搜索结果，如图 9-11 所示。

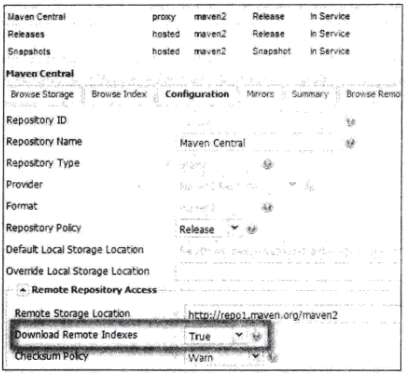

图 9-9 为 Maven Central 仓库开启远程索引下载

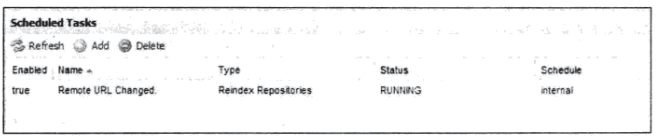

图 9-10 下载 Maven 中央仓库索引的后台任务

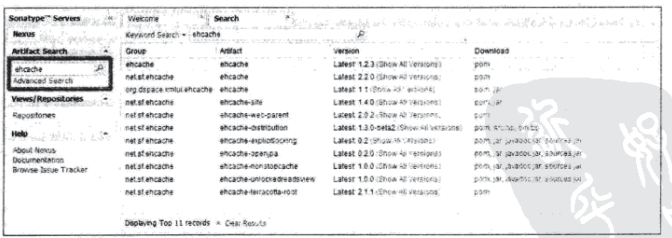

图 9-11 在 Nexus 中快速搜索构件

该例使用了 ehcache 关键字进行搜索，因此得到了大量与 ehcache 相关的结果，结果中的每一行都表示了一类构件，信息包括 GroupId、ArtifactId、最新版本以及最新版本的相关文件下载等。单击其中的某一行，界面的下端会浮出一个更具体的构件信息面板，如图 9-12 所示。

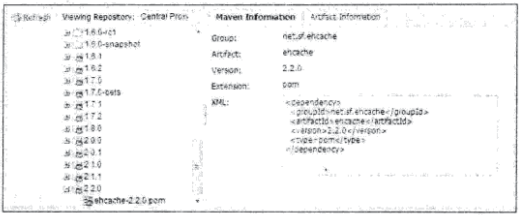

图 9-12　Nexus 的构件信息面板

该面板除了显示构件的坐标，还包含了一段 XML 依赖声明，用户可以直接复制粘贴到项目的 POM 中。此外，用户还能从该面板获知构件在仓库中的相对位置。单击 Artifact Information 还能看到文件具体的大小、更新时间、SHA1 和 MD5 校验和以及下载链接。除了简单的关键字搜索，Nexus 还提供了 GAV 搜索、类名搜索和校验和搜索等功能，用户可以单击搜索页面左上角的下拉菜单选择高级搜索功能：

- **GAV 搜索**（GAV Search）　允许用户通过设置 GroupId、ArtifactId 和 Version 等信息来进行更有针对性的搜索。
- **类名搜索**（Classname Search）　允许用户搜索包含某个 Java 类的构件。
- **校验和搜索**（Checksum Search）　允许用户直接使用构件的校验和来搜索该构件。

图 9-11 所示的结果中包含了各种坐标的结果。基于该结果的信息，笔者进一步确定了自己需要的构件的 GroupId 和 ArtifactId，它们分别为 net. sf. ehcache 和 ehcache。这时就可以单击对应的 Show All Versions 转到 GAV 搜索功能来缩小搜索范围，如图 9-13 所示。

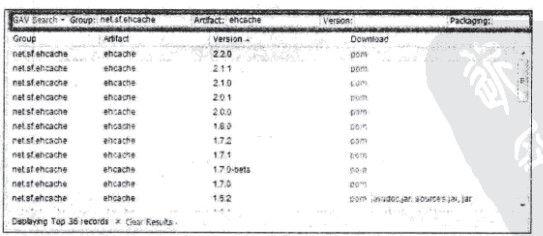

图 9-13　在 Nexus 中使用 GAV 搜索构件

当然，用户也可以自己手动输入 GroupId、ArtifactId 等信息来进行 GAV 搜索。

有了中央仓库的索引，用户不仅能够搜索构件，还能够直接浏览中央仓库的内容。这便是 Nexus 的索引浏览功能。在 Repositories 页面中，选择 Browse Index 选项卡，就能看到中央仓库内容的树形结构，如图 9-14 所示。

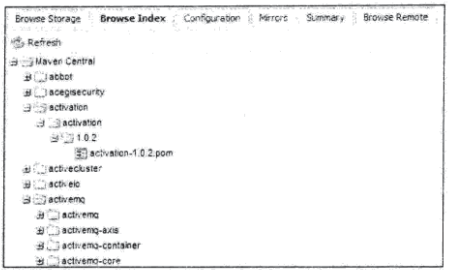

图 9-14 Nexus 的索引浏览

以上的搜索及浏览功能都是基于 Nexus 索引而实现的，确切地应该称之为 nexus-indexer。Nexus 能够遍历一个 Maven 仓库所有的内容，搜集它们的坐标、校验和及所含的 Java 类信息，然后以 nexus-indexer 的形式保存起来。中央仓库维护了这样的一个 nexus-indexer，因此本地的 Nexus 下载到这个索引之后，就能在此基础上提供搜索和浏览等服务。需要注意的是，不是任何一个公共仓库都提供 nexus-indexer，对于那些不提供索引的仓库来说，我们就无法对其进行搜索。

除了下载使用远程仓库的索引，我们也能为宿主仓库和代理仓库建立索引。只需要在仓库上右击，从弹出的快捷菜单中选择 ReIndex 即可，如图 9-15 所示。待索引编纂任务完成之后，就能搜索该仓库所包含的构件。

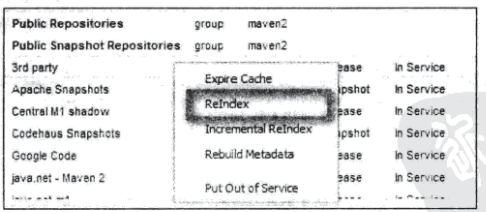

图 9-15 为 Nexus 仓库编纂索引

对于宿主仓库来说，ReIndex 任务会扫描该仓库包含的所有构件建立索引。对于代理仓库来说，ReIndex 任务会扫描所有缓存的构件建立索引，如果远程仓库也有索引，则下载后与本地的索引合并。对于仓库组来说，ReIndex 任务会合并其包含的所有仓库的索引。

9.5 配置 Maven 从 Nexus 下载构件

6.4 节与 7.5.1 节已经详细介绍了如何在 POM 中为 Maven 配置仓库和插件仓库。例如，

当需要为项目添加 Nexus 私服上的 public 仓库时，可以按代码清单 9-1 所示配置。

代码清单 9-1　在 POM 中配置 Nexus 仓库

```
<project>
  ...
  <repositories>
    <repository>
        <id>nexus</id>
        <name>Nexus</name>
        <url>http://localhost:8081/nexus/content/groups/public/</url>
        <releases><enabled>true</enabled></releases>
        <snapshots><enabled>true</enabled></snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
        <id>nexus</id>
        <name>Nexus</name>
        <url>http://localhost:8081/nexus/content/groups/public/</url>
        <releases><enabled>true</enabled></releases>
        <snapshots><enabled>true</enabled></snapshots>
    </pluginRepository>
  </pluginRepositories>
  ...
</project>
```

这样的配置只对当前 Maven 项目有效，在实际应用中，我们往往想要通过一次配置就能让本机所有的 Maven 项目都使用自己的 Maven 私服。这个时候读者可能会想到 settings. xml 文件，该文件中的配置对所有本机 Maven 项目有效，但是 settings. xml 并不支持直接配置 repositories 和 pluginRepositories。所幸 Maven 还提供了 Profile 机制，能让用户将仓库配置放到 setting. xml 中的 Profile 中，如代码清单 9-2 所示。

代码清单 9-2　在 settings. xml 中配置 Nexus 仓库

```
<settings>
      ...
  <profiles>
    <profile>
      <id>nexus</id>
      <repositories>
          <repository>
              <id>nexus</id>
              <name>Nexus</name>
              <url>http://localhost:8081/nexus/content/groups/public/
</url>
              <releases><enabled>true</enabled></releases>
              <snapshots><enabled>true</enabled></snapshots>
          </repository>
      </repositories>
      <pluginRepositories>
```

```
        <pluginRepository>
            <id>nexus</id>
            <name>Nexus</name>
            <url>http://localhost:8081/nexus/content/groups/public/</url>
            <releases><enabled>true</enabled></releases>
            <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
  ...
</settings>
```

该配置中使用了一个 id 为 nexus 的 profile，这个 profile 包含了相关的仓库配置，同时配置中又使用 activeProfile 元素将 nexus 这个 profile 激活，这样当执行 Maven 构建的时候，激活的 profile 会将仓库配置应用到项目中去。关于 Maven Profile，本书后面还会有专门的章节进一步介绍。

代码清单 9-2 中的配置已经能让本机所有的 Maven 项目从 Nexus 私服下载构件。细心的读者可能会注意到，Maven 除了从 Nexus 下载构件之外，还会不时地访问中央仓库 central，我们希望的是所有 Maven 下载请求都仅仅通过 Nexus，以全面发挥私服的作用。这个时候就需要借助于 6.7 节提到的 Maven 镜像配置了。可以创建一个匹配任何仓库的镜像，镜像的地址为私服，这样，Maven 对任何仓库的构件下载请求都会转到私服中。具体配置见代码清单 9-3。

代码清单 9-3　配置镜像让 Maven 只使用私服

```
<settings>
  ...
  <mirrors>
    <mirror>
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/nexus/content/groups/public</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <repositories>
        <repository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </repository>
      </repositories>
```

```
    <pluginRepositories>
      <pluginRepository>
        <id>central</id>
        <url>http://central</url>
        <releases><enabled>true</enabled></releases>
        <snapshots><enabled>true</enabled></snapshots>
      </pluginRepository>
    </pluginRepositories>
  </profile>
</profiles>
<activeProfiles>
  <activeProfile>nexus</activeProfile>
</activeProfiles>
...
</settings>
```

关于镜像、profile 及 profile 激活的配置不再赘述，这里需要解释的是仓库及插件仓库配置，它们的 id 都为 central，也就是说，覆盖了超级 POM 中央仓库的配置，它们的 url 已无关紧要，因为所有请求都会通过镜像访问私服地址。配置仓库及插件仓库的主要目的是开启对快照版本下载的支持，当 Maven 需要下载发布版或快照版构件的时候，它首先检查 central，看该类型的构件是否支持，得到正面的回答之后，再根据镜像匹配规则转而访问私服仓库地址。

9.6　部署构件至 Nexus

如果只为代理外部公共仓库，那么 Nexus 的代理仓库就已经能够完全满足需要了。对于另一类 Nexus 仓库——宿主仓库来说，它们的主要作用是储存组织内部的，或者一些无法从公共仓库中获得的第三方构件，供大家下载使用。用户可以配置 Maven 自动部署构件至 Nexus 的宿主仓库，也可以通过界面手动上传构件。

9.6.1　使用 Maven 部署构件至 Nexus

日常开发生成的快照版本构件可以直接部署到 Nexus 中策略为 Snapshot 的宿主仓库中，项目正式发布的构件则应该部署到 Nexus 中策略为 Release 的宿主仓库中。POM 的配置方式具体见 5.4 节，代码清单 9-4 列出了一段典型的配置。

代码清单 9-4　配置 Maven 部署构件至 Nexus

```
<project>
  ...
  <distributionManagement>
    <repository>
      <id>nexus-releases</id>
      <name>Nexus Releases Repository</name>
      <url>http://localhost:8081/nexus/content/repositories/releases/</url>
    </repository>
    <snapshotRepository>
```

```
        <id>nexus-snapshots</id>
        <name>Nexus Snapshots Repository</name>
        <url>http://localhost:8081/nexus/content/repositories/snapshots/</url>
      </snapshotRepository>
    </distributionManagement>
    ...
</project>
```

Nexus 的仓库对于匿名用户是只读的。为了能够部署构件，还需要在 settings.xml 中配置认证信息，如代码清单 9-5 所示。

代码清单 9-5　为部署构件至 Nexus 配置认证信息

```
<settings>
  ...
  <servers>
    <server>
      <id>nexus-releases</id>
      <username>admin</username>
      <password>******</password>
    </server>
    <server>
      <id>nexus-snapshots</id>
      <username>admin</username>
      <password>******</password>
    </server>
  </servers>
  ...
</settings>
```

9.6.2　手动部署第三方构件至 Nexus

某些 Java Jar 文件（如 Oracle）的 JDBC 驱动，由于许可证的因素，它们无法公开地放在公共仓库中。此外，还有大量的小型开源项目，它们没有把自己的构件分发到中央仓库中，也没有维护自己的仓库，因此也无法从公共仓库获得。这个时候用户就需要将这类构件手动下载到本地，然后通过 Nexus 的界面上传到私服中。

要上传第三方构件，首先选择一个宿主仓库如 3rd party，然后在页面的下方选择 Artifact Upload 选项卡。在上传构件的时候，Nexus 要求用户确定其 Maven 坐标，如果该构件是通过 Maven 构建的，那么可以在 GAV Definition 下拉列表中选择 From POM，否则就选 GAV Parameters。用户需要为该构件定义一个 Maven 坐标，例如上传一个 Oracle 11g 的 JDBC 驱动，则可以按图 9-16 所示输入坐标。

定义好坐标之后，单击 Select Artifact(s) to Upload 按扭从本机选择要上传的构件，然后单击 Add Artifact 按钮将其加入到上传列表中。Nexus 允许用户一次上传一个主构件和多个附属构件（即 Classifier）。最后，单击页面最下方的 Upload Artifact(s)按钮将构件上传到仓库中。

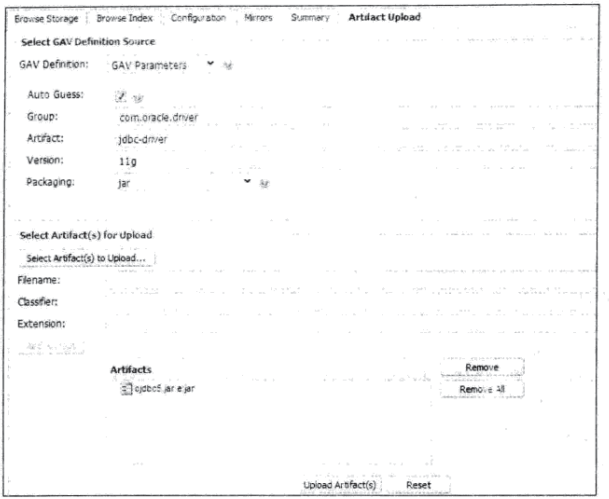

图 9-16　手动上传构件至 Nexus

9.7　Nexus 的权限管理

在组织中使用 Nexus 的时候往往会有一些安全性需求，例如希望只有管理员才能配置 Nexus，只有某些团队成员才能部署构件，或者更细一些的要求，例如每个项目都有自己的 Nexus 宿主仓库，且只能部署项目构件至该仓库中。Nexus 提供了全面的权限控制特性，能让用户自由地根据需要配置 Nexus 用户、角色、权限等。

9.7.1　Nexus 的访问控制模型

Nexus 是基于权限（Privilege）做访问控制的，服务器的每一个资源都有相应的权限来控制，因此用户执行特定的操作时就必须拥有必要的权限。管理员必须以角色（Role）的方式将权限赋予 Nexus 用户。例如要访问 Nexus 界面，就必须拥有 Status -（read）这个权限，而 Nexus 默认配置的角色 UI：Basic UI Privileges 就包含了这个权限，再将这个角色分配给某个用户，这个用户就能访问 Nexus 界面了。

用户可以被赋予一个或者多个角色，角色可以包含一个或者多个权限，角色还可以包含一个或者多个其他角色。

Nexus 预定义了三个用户，以 admin 登录后，单击页面左边导航栏中的 User 链接，就能看到所有已定义用户的列表，如图 9-17 所示。

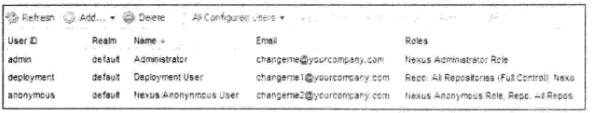

图 9-17 Nexus 的预定义用户

这三个用户对应了三个权限级别：

- **admin**：该用户拥有对 Nexus 服务的完全控制，默认密码为 admin123。
- **deployment**：该用户能够访问 Nexus，浏览仓库内容，搜索，并且上传部署构件，但是无法对 Nexus 进行任何配置，默认密码为 deployment123。
- **anonymous**：该用户对应了所有未登录的匿名用户，它们可以浏览仓库并进行搜索。

在 Users 页面中，管理员还可以添加用户。单击上方的 Add 按钮，选择 Nexus User，然后在用户配置面板中配置要添加用户的 ID、名称、Email、状态、密码以及包含的角色，最后单击 Save 按钮即可。

可以单击任何一个用户，然后选择页面下方的 Role Tree 选项卡，以树形结构详细地查看该用户所包含的角色以及进一步的权限。图 9-18 所示是 anonymous 用户的角色树。

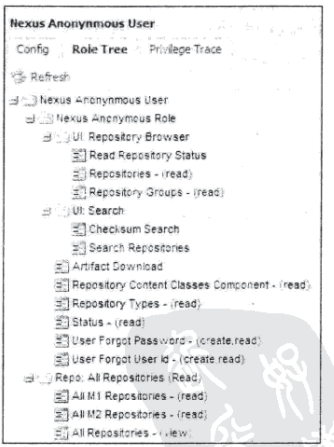

图 9-18 anonymous 用户的角色树

理解各个角色的意义对于权限管理至关重要。Nexus 预定义的一些常用且重要的角色包括：

- **UI：Basic UI Privileges**：包含了访问 Nexus 界面必须的最基本的权限。
- **UI：Repository Browser**：包含了浏览仓库页面所需要的权限。
- **UI：Search**：包含了访问快速搜索栏及搜索页面所需要的权限。
- **Repo：All Repositories（Read）**：给予用户读取所有仓库内容的权限，没有仓库的读权限，用户将无法在仓库页面上看到实际的仓库内容，也无法使用 Maven 从仓库下载构件。
- **Repo：All Repositories（Full Control）**：给予用户完全控制所有仓库内容的权限。用户不仅可以浏览、下载构件，还可以部署构件及删除仓库内容。

Nexus包含了一个特殊的匿名用户角色（Nexus Anonymous Role），默认配置下没有登录的用户都会拥有该匿名角色的权限。这个匿名用户角色实际包含了上述所列角色中，除Repo：All Repositories（Full Control）之外的所有角色所包含的权限。也就是说，匿名用户可以访问基本的Nexus界面、浏览仓库内容及搜索构件。

除上述角色之外，Nexus还预定义了很多其他角色，它们往往都对应了一个Nexus的功能。例如，UI：Logs and Config Files包含了访问系统日志文件及配置文件所需要的权限。

9.7.2 为项目分配独立的仓库

在组织内部，如果所有项目都部署快照及发布版构件至同样的仓库，就会存在潜在的冲突及安全问题，我们不想让项目A的部署影响到项目B，反之亦然。解决的方法就是为每个项目分配独立的仓库，并且只将仓库的部署、修改和删除权限赋予该项目的成员，其他用户只能读取、下载和搜索该仓库的内容。

假设项目名称为foo，首先为该项目建立两个宿主仓库Foo Snapshots和Foo Releases，分别用来部署快照构件和发布构件。具体步骤参见9.3.3节，这里不再赘述。

有了仓库之后，就需要创建基于仓库的增、删、改、查权限。在Nexus中，这样的权限是基于Repository Target建立的，Repository Target实际上是一系列正则表达式，在访问仓库某路径下内容的时候，Nexus会将仓库路径与Repository Target的正则表达式一一匹配，以检查权限是否正确。

单击左边导航栏中的Repository Targets链接，就能看到图9-19所示的页面。图中选中了All（Maven2）这一Repository Target，在下方可以看到它包含了一个值为“.*”的正则表达式，表示该Repository Target能够匹配仓库下的任何路径。

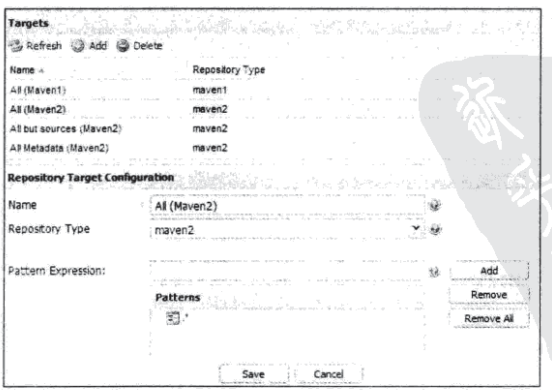

图9-19 Nexus的Repository Target

下一步就是基于该 Repository Target 和 Foo Releases、Foo Snapshots 两个仓库建立权限。单击页面左边导航栏中的 Privileges 链接进入权限页面，然后单击 Add 按钮，选择 Repository Target Privilege。图 9-20 所示为创建对应于 Foo Releases 的权限。

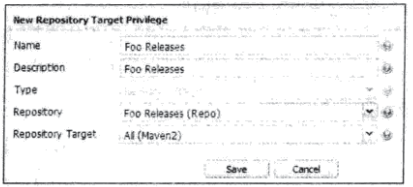

图 9-20 为 Foo Releases 创建仓库权限

图 9-20 中选择了 Foo Releases 仓库和 All（Maven2），表示创建匹配 Foo Releases 仓库任何路径的权限。单击 Save 按钮之后，就能在权限列表中看到相应的增、删、改、查权限，如图 9-21 所示。

Foo Releases - (create)	true	Repository Target	All (Maven2)	Foo Releases
Foo Releases - (delete)	true	Repository Target	All (Maven2)	Foo Releases
Foo Releases - (read)	true	Repository Target	All (Maven2)	Foo Releases
Foo Releases - (update)	true	Repository Target	All (Maven2)	Foo Releases

图 9-21 Foo Releases 仓库的增、删、改、查权限

然后，遵循同样的步骤，为 Foo Snapshots 建立增、删、改、查权限。

下一步是创建一个包含上述权限的角色。单击导航栏中的 Roles 进入角色页面，再单击页面上方的 Add 按钮并选择 Nexus Role。图 9-22 所示为将之前建立的权限加入到该角色中。

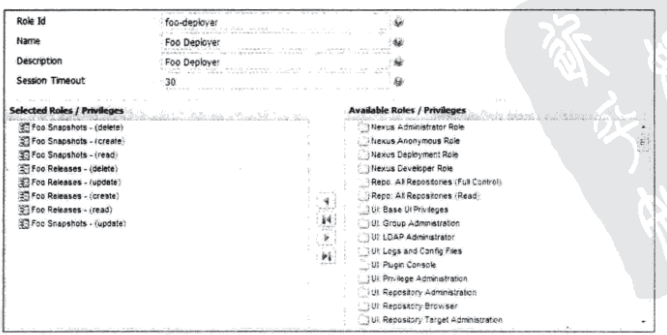

图 9-22 创建 Foo Deployer 角色

角色创建完成之后，根据需要将其分配给 Foo 项目的团队成员。这样，其他团队的成员默认只能读取 Foo Releases 和 Foo Snapshots 的内容，而拥有 Foo Deployer 角色的用户就可以执行部署构件等操作。

9.8　Nexus 的调度任务

Nexus 提供了一系列可配置的调度任务来方便用户管理系统。用户可以设定这些任务运行的方式，例如每天、每周、手动等。调度任务会在适当的时候在后台运行。当然，用户还是能够在界面观察它们的状态的。

要建立一个调度任务，单击左边导航栏中的 Scheduled Tasks 链接，然后在右边的界面上方单击 Add 按钮，接着就能看到图 9-23 所示的界面。用户可以根据自己的需要，选择任务类型，并配置其运行方式。

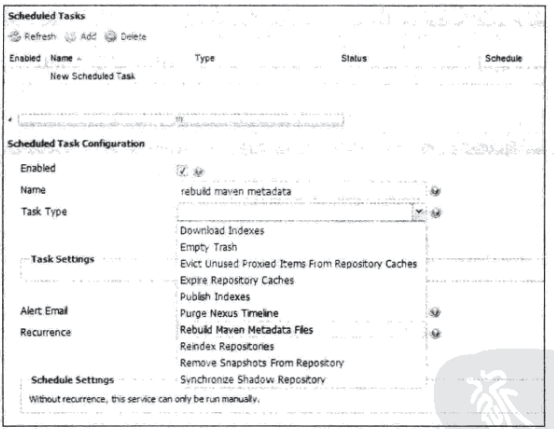

图 9-23　创建 Nexus 调度任务

Nexus 包含了以下几种类型的调度任务：

- **Download Indexes**：为代理仓库下载远程索引。
- **Empty Trash**：清空 Nexus 的回收站，一些操作（如删除仓库文件）实际是将文件移到了回收站中。
- **Evict Unused Proxied Items From Repository Caches**：删除代理仓库中长期未被使用的构件缓存。

- **Expire Repository Caches**：Nexus 为代理仓库维护了远程仓库的信息以避免不必要的网络流量，该任务清空这些信息以强制 Nexus 去重新获取远程仓库的信息。
- **Publish Indexes**：将仓库索引发布成可供 m2eclipse 和其他 Nexus 使用的格式。
- **Purge Nexus Timeline**：删除 Nexus 的时间线文件，该文件用于建立系统的 RSS 源。
- **Rebuild Maven Metadata Files**：基于仓库内容重新创建仓库元数据文件 maven-metadata. xml，同时重新创建每个文件的校验和 md5 和 sha1。
- **Reindex Repositories**：为仓库编纂索引。
- **Remove Snapshots From Repository**：以可配置的方式删除仓库的快照构件。
- **Synchronize Shadow Repository**：同步虚拟仓库的内容（服务于 Maven 1）。

9.9 其他私服软件

Nexus 不是唯一的 Maven 私服软件，正如本章一开始所提到的，用户还有另外两个选择，它们分别为 Apache 的 Archiva 与 JFrog 的 Artifactory。

Archiva 可能是历史最长的 Maven 私服软件，它早在 2005 年就作为 Apache Maven 的一个子项目存在，到 2008 年 3 月成为了 Apache 软件基金会的顶级项目。到本书编写的时候，Archiva 的最新版本为 1. 3. 1。

读者可以访问 http://archiva. apache. org 以具体了解 Archiva，其站点提供了一些入门指南及邮件列表等信息。Archiva 的下载地址为 http://archiva. apache. org/download. html。图 9-24 显示了 Archiva 的一个仓库管理界面。

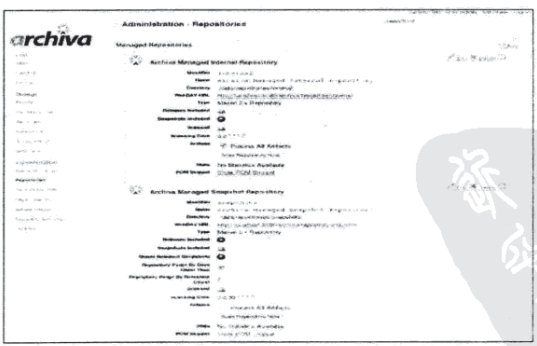

图 9-24 Archiva 的仓库管理界面

在 Nexus 发布之前，笔者曾一度是 Artifactory 的忠实用户，当时它是唯一的支持从用户界面配置仓库的私服。Artifactory 的一大特点是使用数据库来存储仓库内容。读者可以自行

访问 JFrog 站点以了解更多信息：http://www.jfrog.org/products.php。Artifactory 目前的最新版本为 2.2.5，其下载地址为 http://www.jfrog.org/download.php。图 9-25 所示是 Artifactory 的仓库浏览界面。

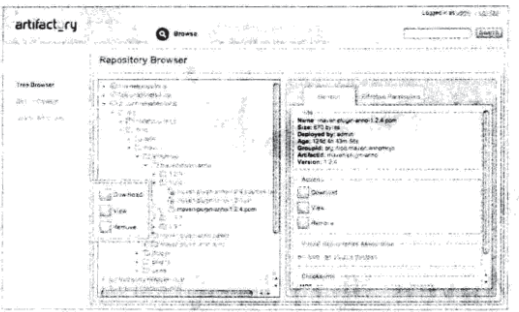

图 9-25　Artifactory 的仓库浏览界面

细心的读者会发现，Nexus 的主色调为蓝色，Archiva 的主色调为橙色，而 Artifactory 的主色调为绿色，这或许是各个团队自我风格的一种体现吧。

9.10　小结

建立并维护自己的私服是使用 Maven 必不可少的一步，Maven 私服软件有 Nexus、Archiva 和 Artifactory，它们都提供了开源的版本供用户下载。本章详细介绍了 Nexus 的安装和使用，包括如何分辨各种类型的仓库、如何建立仓库索引和搜索构件、如何使用权限管理功能、如何使用调度任务功能等。除了这些功能之外，Nexus 还有很多有趣的特性，如 RSS 源、日志浏览及配置等，用户可以从友好的界面中学习使用。

除了 Nexus 本身，本章还详述了如何配置 Maven 从私服下载构件，以及如何发布构件至私服供他人使用。结合了 Nexus 的帮助之后，再使用 Maven 时就会如虎添翼。