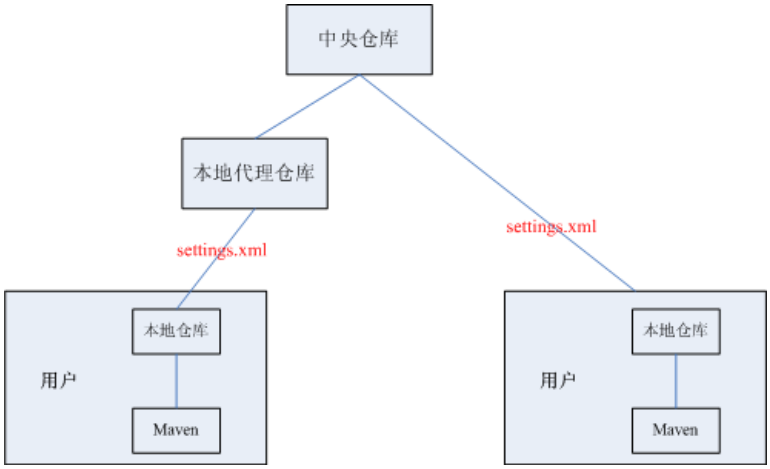
使用nexus搭建代理仓库

nexus搭建代理仓库

使用Maven构建和管理项目是非常享受的一件事，我们可以从Maven中央仓库下载所需要的构件(artifact)，但实际开发中由于种种原因我们需要在架设一个Maven本地代理仓库，如：不方便访问公网、节省带宽和时间、管理自家的共用artifact等等。本地地理仓库是我自己取的名字，为了不与下文的本地仓库想混淆。



获取构建的流程如下：用户使用Maven构建项目时，首先是要直接从本地仓库获取的，如果本地仓库没有，它会根据setting.xml的设置去首先尝试从远程仓库下载构件至本地仓库，然后再使用本地仓库的构件。如果setting.xml设置的远程仓库是本地代理仓库，则本地代理仓库先尝试从自己的库中获取，如果没有再从远程仓库(比如中央仓库)下载构件至本地仓库。

Nexus 是一个优秀的Maven仓库管理器，还提供了强大的仓库管理功能，构件搜索功能，它基于REST，友好的UI是一个extjs的REST客户端，它占用较少的内存，基于简单文件系统而非数据库。这些优点使其日趋成为最流行的Maven仓库管理器。本文就使用Nexus搭建一个本地代理仓库。

下载和安装

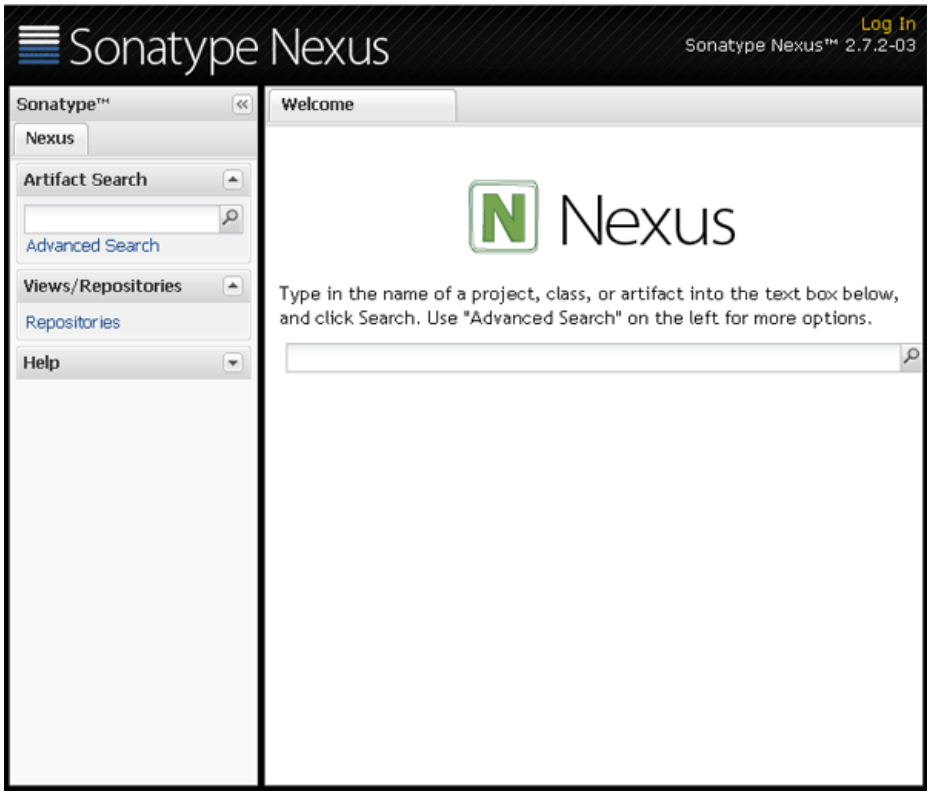
Nexus官方下载地址：http://www.sonatype.org/nexus/go，目前最新的版本是2.7.2。

Nexu安装非常容易，因为它内嵌了Jetty，只要有JRE救能直接运行。解压Nexu包会得到两个目录nexus-2.7.2-03和sonatype-work，sonatype-work是默认仓库目录。运行、安装都是使用nexus-2.7.2-03/bin/nexus.bat文件，它的使用方式：

1. Usage:nexus.bat { console : start : stop : restart : install : uninstall }

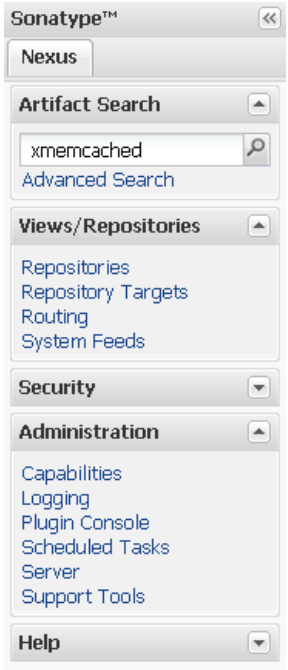
其中console是控制台方式运行，install是以windows service寄存，uninstall是下载windows service，start是运行windows service，stop是停止windows service，restart是重启windows service，。

Nexus默认端口是8081，可以在nexus-2.7.1-01/conf/nexus.properties中修改，启动后就可以通过地址：http://localhost/:8081/nexus 来访问了。界面如下：



管理仓库

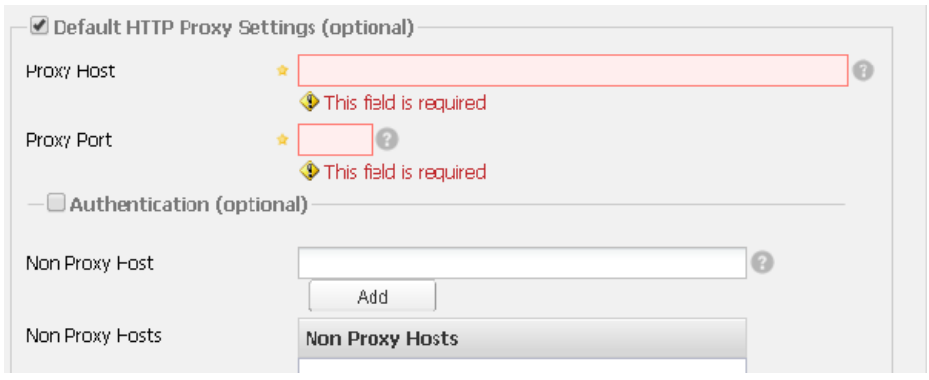
管理仓库需要先登录，默认登录用户名/密码是admin/admin123。登录后就可以看到左栏菜单的管理项。



这里，可以管理仓库，配置Nexus系统，管理任务，管理用户，角色，权限，查看系统的RSS源，管理及查看系统日志，等等。

1. 设置Nexus代理上公网

如果您的本地代理仓库服务器没有公网权限，Nexus为你留了使用代理的路子，点击右边菜单的“Server”，在右边找到：



添加你的代理服务器即可。

2. 仓库管理

点击左边导航栏的Repositories，界面的主面板会显示所有一个所有仓库及仓库组的列表，你会看到它们的Type字段的值有group，hosted，proxy，virtual。这里我们不关心virtual，只介绍下另外三种类型：

- hosted，本地代理仓库，通常我们会部署自己的构件到这一类型的仓库。
- proxy，代理的远程仓库，它们被用来代理远程的公共仓库，如maven中央仓库。
- group，仓库组，用来合并多个hosted/proxy仓库，通常我们配置maven依赖仓库组。

如何管理、添加等操作，Nexus都写的很清楚了，我就不一一赘述了。

修改setting.xml配置Maven的仓库

Maven安装后默认的是使用中央仓库，这是为了能让Maven开箱即用。而Maven缺省的本地仓库地址为${user.home}/.m2/repository。也就是说，一个用户会对应的拥有一个本地仓库。你也可以自定义本地仓库的位置，修改${user.home}/.m2/settings.xml。

首先需要修改Mirrors

```
<mirrors>
  <!-- mirror | Specifies a repository mirror site to use instead of a given
   repository. The repository that | this mirror serves has an ID that matches
   the mirrorOf element of this mirror. IDs are used | for inheritance and direct
   lookup purposes, and must be unique across the set of mirrors. | -->
 <mirror>
  <id>nexusc</id>
  <mirrorOf>*</mirrorOf>
  <name>Nexus</name>
   <url>http://localhost:8081/nexus/content/groups/public/</url>
  </mirror>
</mirrors>

```

在执行 Maven 命令的时候， Maven 还需要安装一些插件包，这些插件包的下载地址也让其指向本地代理仓库的地址，修改如下:

```
<profile>
 <id>jdk-1.4</id>
 <activation>
  <jdk>1.4</jdk>
 </activation>
 <repositories>
  <repository>
   <id>nexus</id>
   <name>local private nexus</name>
    <url>http://localhost:8081/nexus/content/groups/public/</url>
    <releases>
     <enabled>true</enabled>
    </releases>
    <snapshots>
     <enabled>false</enabled>
    </snapshots>
  </repository>
 </repositories>
 <pluginRepositories>
  <pluginRepository>
   <id>nexus</id>
   <name>local private nexus</name>
   <url>http://localhost:8081/nexus/content/groups/public/</url>
   <releases>
    <enabled>true</enabled>
   </releases>
   <snapshots>
    <enabled>false</enabled>
   </snapshots>
```

```
30. </pluginRepository>
31. </pluginRepositories>
32. </profile>
```

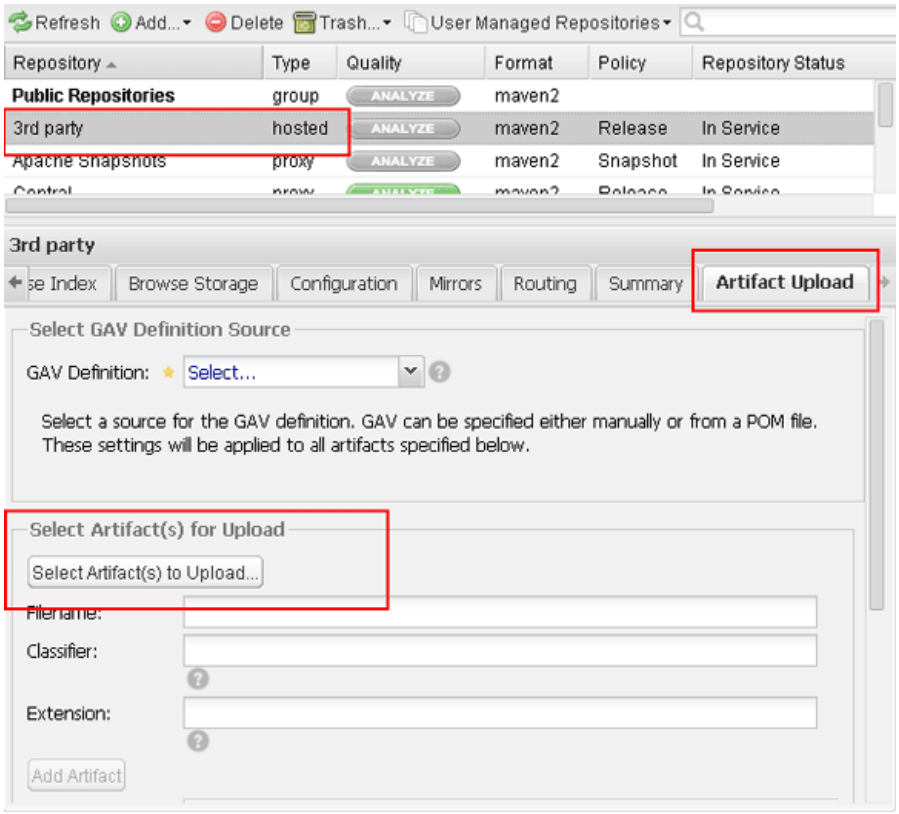
关于更多setting.xml字段解释请看这里。
现在你就可以使用本地仓库构建Maven项目了，踏上愉快的构建之旅！！

部署构件至Nexus

1. 通过Nexus UI部署

有时候有个jar文件你无法从公共Maven仓库找到，但是你能从其它得到这个jar文件（甚至是POM），那么你完全可以将这个文件部署到Nexus中，使其成为标准流程的一部分。步骤如下：

> 点击左边导航栏的"Repository"，在右边的仓库列表中选择一个仓库，如"3rd Party"，然后在页面下方的tab选择"Artifact Upload Artifact(s)"，你会看到构件上传界面。选择你要上传的构件，并指定POM.（或者手工编写GAV等信息），最后点击Upload，该构件就直接被部署到了Nexus的"3rd Party"仓库中。



2. 通过Maven部署

更常见的用例是：团队在开发一个项目的各个模块，为了让自己开发的模块能够快速让其他人使用，你会想要将snapshot版本的构件部署到Maven仓库中，其他人只需要在POM添加一个对于你开发模块的依赖，就能随时拿到最新的snapshot。
以下的pom.xml配置和settings.xml能让你通过Maven自动化部署构件：
pom.xml

```
a. <project>
b. ...
c. <distributionManagement>
d.  <repository>
e.   <id>nexus-releases</id>
f.    <name>Nexus Release Repository</name>
g.     <url>http://localhost:8081/nexus/content/repositories/releases/</url>
h.  </repository>
i.  <snapshotRepository>
j.   <id>nexus-snapshots</id>
k.    <name>Nexus Snapshot Repository</name>
l.   <url>http://localhost:8081/nexus/content/repositories/snapshots/</url>
m.  </snapshotRepository>
n. </distributionManagement>
o. ...
p. </project>
```

settings.xml

```
<settings>
...
<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>
...
</settings>
```

这里我们配置所有的snapshot版本构件部署到Nexus的Snapshots仓库中， 所有的release构件部署到Nexus的Releases仓库中。由于部署需要登陆，因为我们在settings.xml中配置对应Repository id的用户名和密码。

然后，在项目目录中执行mvn deploy ，你会看到maven将项目构件部署到Nexus中，浏览Nexus对应的仓库，就可以看到刚才部署的构件。当其他人构建其项目时，Maven就会从Nexus寻找依赖并下载。