<https://www.cnblogs.com/helong/articles/2254446.html>

# [使用Nexus创建私服](https://www.cnblogs.com/helong/articles/2254446.html)

**部署环境是windowsxp**

首先从 http://nexus.sonatype.rog/downloads 下载最新版的Nexus

我们可以根据个人的需求下载。里面有nexus-webapp-1.9.2.3-bundle.zip等其他系统运行的压缩文件 或者war包nexus-webapp-1.9.2.3.war。

nexus-webapp-1.9.2.3-bundle.zip 包里面自带了一个Jetty容器，这样我们不用另找容器，可以直接启动运行。

**Bundle 安装方式：**

首页解压Bundle文件，这时回得到两个子目录：

nexus-oss-webapp-1.9.2.3 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、日志文件、仓库文件等。其中，第一个目录是运行Nexus所必须的，而且所有相同版本Nexus实力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，不过他的内容对于各个Nexus实例不一样的，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。当用户需要本分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-webapp1.9.2.3目录下的内容是可以从安装包直接获取的。

如果要启动nexus需要在nexus-oss-webapp-1.9.2.3\bin\jsw\ 下面选择对应的系统版本，我的环境是：windows-x86-32，所以选择这个目录这个目录里面有一些脚本如下：  
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

Stopnexus.bat 停止Nexus windos 服务

Pausenexus.bat 暂停Nexus windows 服务

Resumenexus.bat 恢复暂停的 Nexus Windos服务

如果不想每次启动都使用这些脚本，我可以可以配置成widnwos服务，可以开机自动启动

在Linux 系统上启动Nexus也非常方便，例如Unbuntu 32 位系统，那么只需要进入到nexus-oss-webapp-1.9.2.3\bin\jsw\linux-x86-32 然后运行如下的命令：

$./nexus console

这时可以看到启动的命令行输出，可以使用Ctril+c停止Nexus

除了console之外，nexus的linux脚本还提供了如下命令：

./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服务状态

./nexus restart: 重新启动后台的Nexus服务

Nexus Bundle 默认使用的端口号是：8081，如果启动服务可以访问：

http://localhost:8081/nexus

启动Bundle时可能会出现端口冲突，或者想跟换端口时可以的打开:nexus-oss-webapp-1.9.2.3\conf\plexus.properties 文件里面找到属性:application-port 按自己的需要改成相应的端口 我的改成4040

这时在访问的时候地址应该是这样的：http://localhost:4040/nexus

**WAR的方式安装Nexus**

该war包支持主流的web容器，如tomcat 、Classfish、Jetty和Resin

比如tomcat ，把下载好的war放在webapps目录下 然后运行tomcat就行了，具体看看tomcat资料

**登录Nexus**

Nexus默认管理员用户名和密码为：admin/admin123

**Nexus 内置仓库介绍**

Maven Gentral：该残酷代理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。

Release: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仓库发布版本构件。

java.net Maven2：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

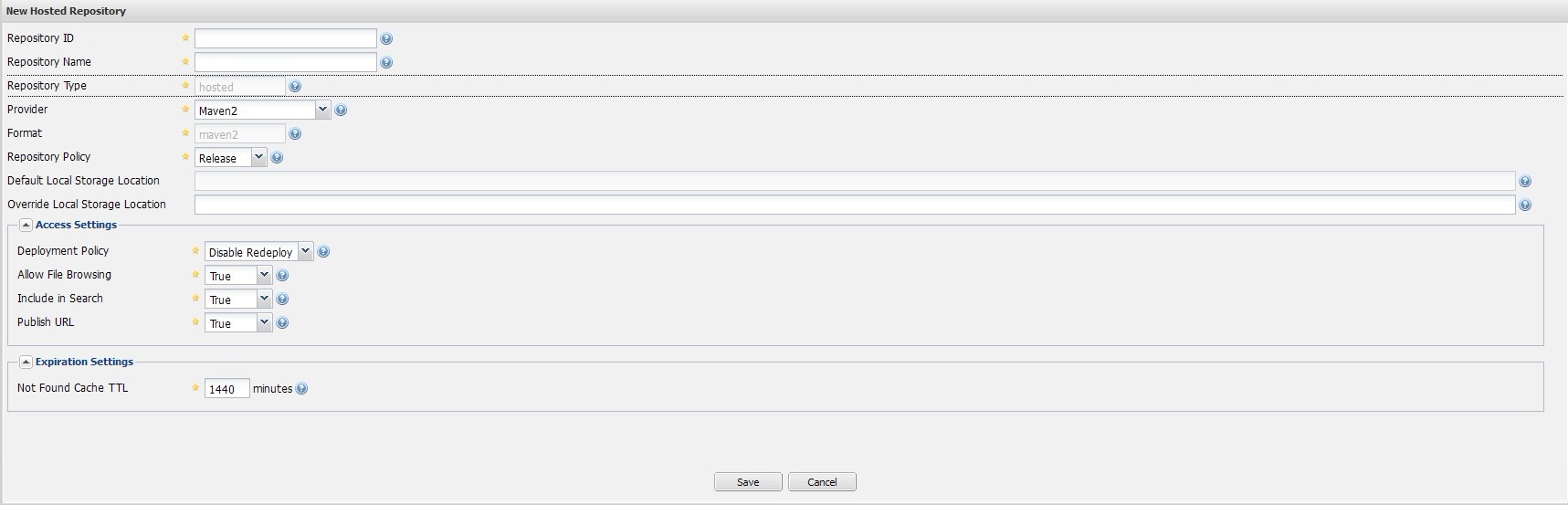
Public Snapshot Repositories： 该仓库组将上述所有的策略为Snapshot的仓库聚合并通过一致的地址提供服务。（在nexus-oss-webapp-1.9.2.3 这版本中没有此项）

**Nexus仓库分类概念**

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，它会转向其包含的宿主仓库或者代理仓库获得市级构件的内容。

**创建Nexus宿主仓库**

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，就会看到如下图的配置界面：



填写仓库ID Repostory ID  和仓库名称 Repository Name 以及仓库类型Respository Type  仓库的格式 Provider 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本构件仓库还是快照版本构件仓库。Defualt Local Storage Location 表示该仓库的没哦人存储目录，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**:E:\nexus-oss-webapp-1.9.2.3-bundle\sonatype-work\nexus\storage**

默认的地址仓库存储目录地址。

Access Setting 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

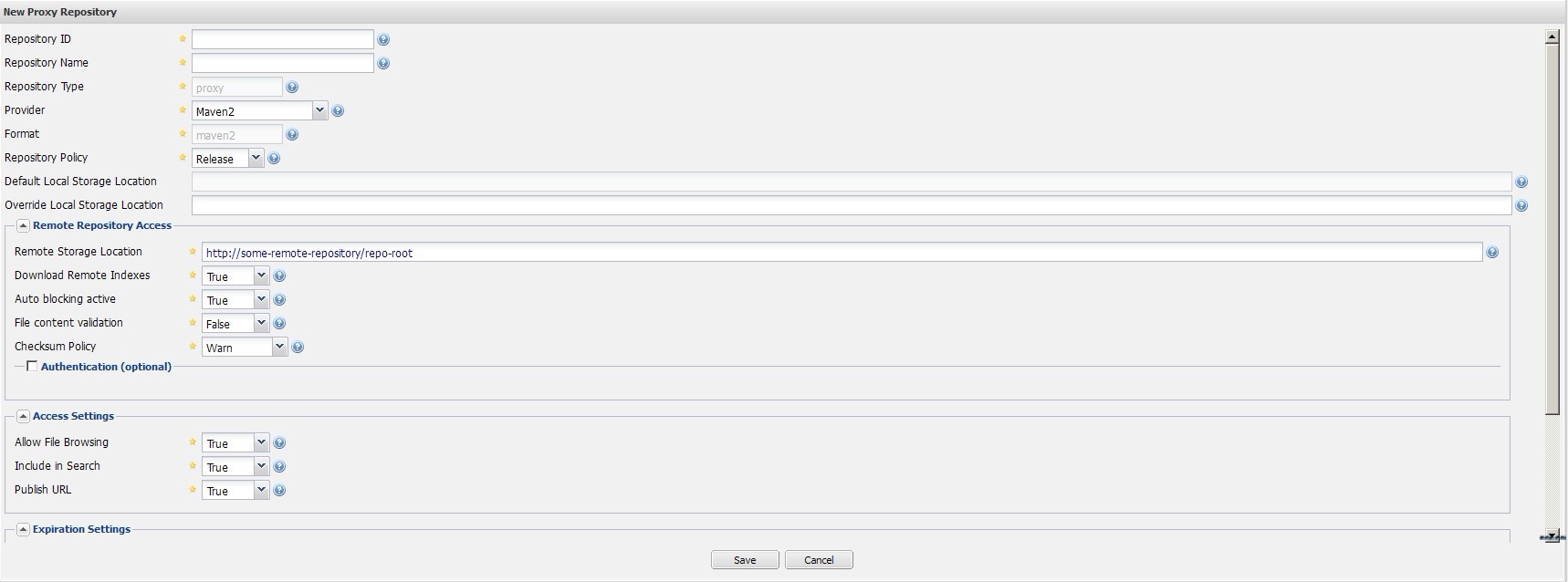
　　Include in Search 表示是否对该仓库进行索引并提供搜索

　　Publish URL 用来控制是否通过URL提供服务，如果选择false当访问仓库的地址时，会得到HTTP404 Not Found 错误

　　Not Found Cache TTL 表示当一个文件没有找到后，缓存这一不存在的信息的时间。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的信息，而不会查找位呢间系统。

**创建Nexus代理仓库**

操作和创建宿主仓库类似，主要Repository Type 的值改为proxy 这时看到如下图：



对于代理仓库来说，最重要的是远程仓库地址即

Remote Storage Location，用户必须输入有效的值

Download Remote Indexes 表示是否下载远程仓库的索引

Checksum Policy 配置校验出错时的策略，用户可以选择忽略、记录警告信息或者拒绝下载。

Authentication 当远程仓库需要认证的时候这里的时候，这里的Authentication 配置就能派上用处。

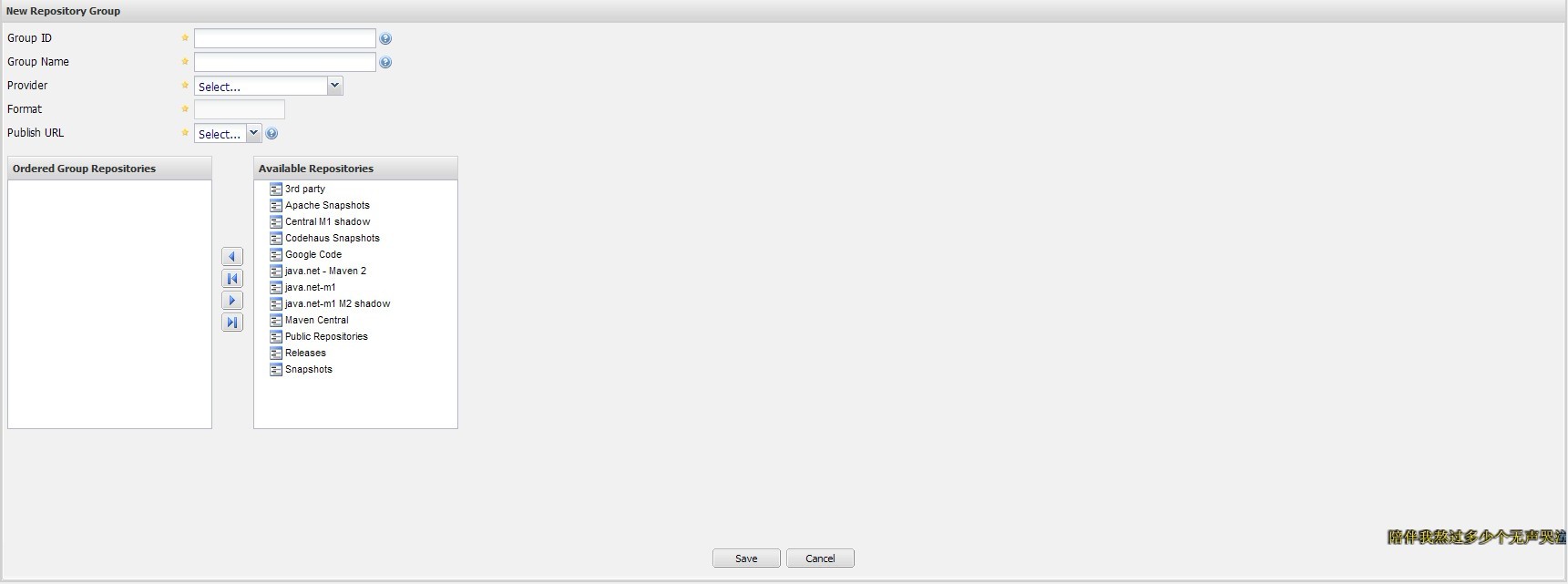
Artifact Max Age 构件缓存最长的时间，对于快照版本来说 Artifact Max Age 默认值为 -1，表示构件混存后就一直保存着，不在重新下载，对于快照版来说默认值为1440分钟表示每隔

Metadata Max Age 仓库元数据文件缓存的最长时间

Http Request Setting 和 Override HTTP proxy Setting 其中前者用来配置Nexus访问远程仓库时HTTP请求参数，后者用来配置HTTP代理

**创建Nexus仓库组**

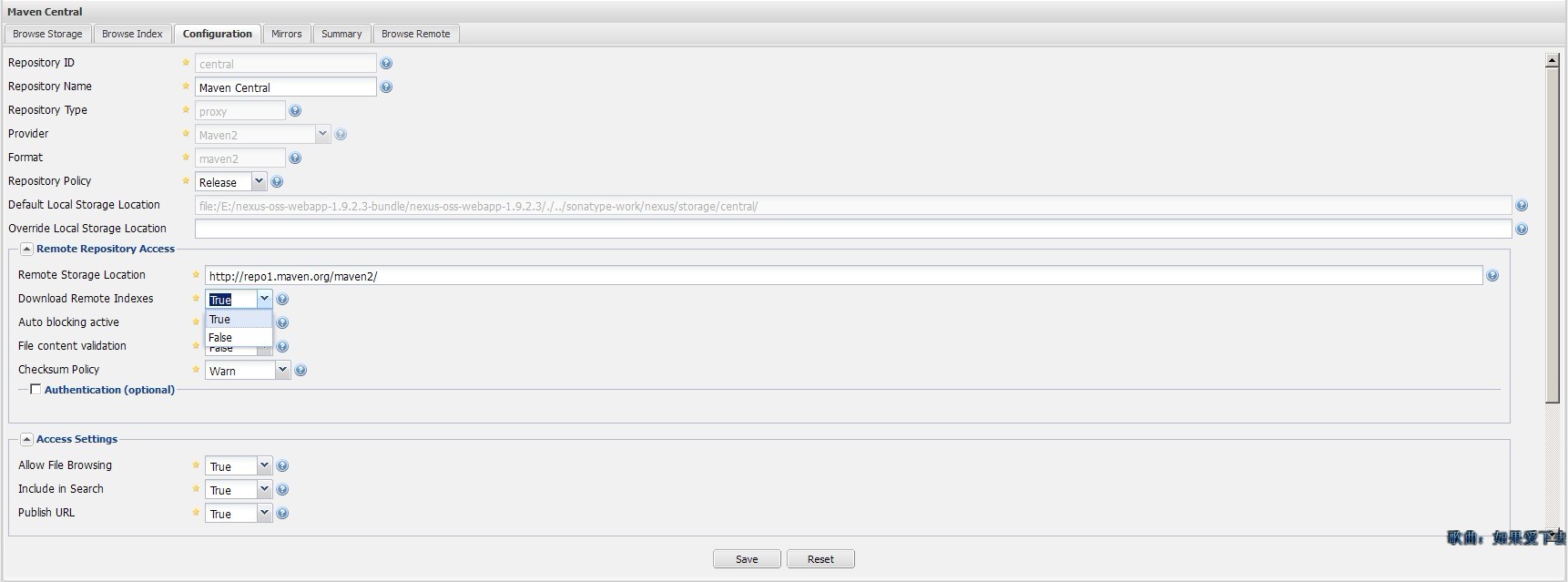
创建仓库组同其他的一样步骤是在选择add时选择下拉框中的Repository Group 就会看到如下的：



配置中的信息同其他的一样，仓库组中没有Release 和Snapshot ，这不同于宿主仓库和代理仓库。在配置界面中可以选择Nexus中的仓库，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便利其所含仓库的次序，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，就能经快的访问到包含构件的仓库。

**Nexus的索引与构件搜索**

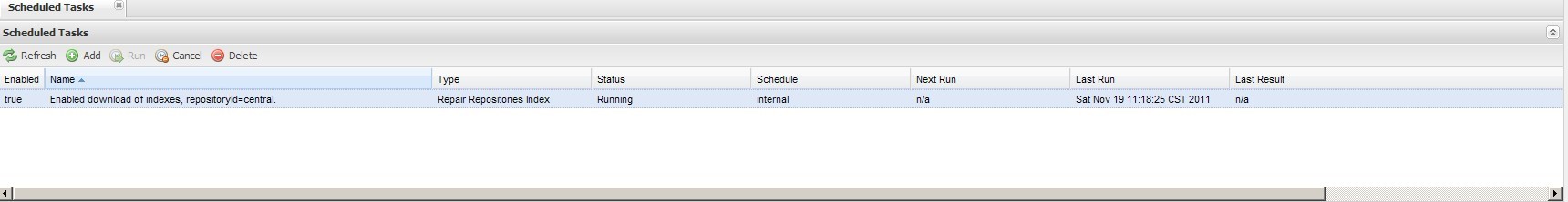
需要搜索Maven 中央库，首先需要设置Nexus中的Maven Central 代理仓库下载远程索引 如下图：



Download Remote Indexes 属性设置为true 默认为false

true是开启，false是关闭 由于中央仓库内容比较多，因此其索引文件比较大，需要查看下载如何了，我们可以单击界面左边导航栏中的 Scheduled Tasks 链接后，就可以看到系统调度的任务其状态为 runing,在说哦因下载完毕之后，该任务就会消失。

Scheduled Tasks 界面：



**配置Maven 从Nexus下载构件**

当需要为项目添加Nexus私服上的public仓库时，可以在项目pom.xml文件配置  代码如下：

[复制代码](javascript:void(0);)

<project>  
   
 <repositories>  
 <repository>  
 <id>nexus</id>  
 <url>http://http://localhost:4040/nexus/content/groups/public/</url>  
 <releases><enabled>true</enabled></releases>  
 <snapshots><enabled>true</enabled></snapshots>  
 </repository>  
 </repositories>  
 <pluginRepositories>  
 <pluginRepository>  
 <id>nexus</id>  
 <url>http://localhost:4040/nexus/content/groups/public/</url>  
 <releases><enabled>true</enabled></releases>  
 <snapshots><enabled>true</enabled></snapshots>  
 </pluginRepository>  
 </pluginRepositories>  
 </profile>  
   
 </project>

[复制代码](javascript:void(0);)

这样的配置只对当前的Maven项目有效，实际应用中，我们往往想要通过一次配置就完成能让本机所有的Maven项目都使用自己的Maven私服。这时配置本地仓库setting.xml 代码如下：

[复制代码](javascript:void(0);)

<settings>  
　　  
　　 <mirrors>  
   
 <mirror>  
 <id>central</id>  
 <mirrorOf>\*</mirrorOf>  
 <url>http://localhost:4040/content/groups/public/</url>  
 </mirror>  
 </mirrors>　　  
  
 <profiles>  
　　　　 <profile>  
 <id>JDK1.6</id>  
 <activation>  
 <activeByDefault>true</activeByDefault>  
 <jdk>1.6</jdk>  
 </activation>  
 <properties>  
 <maven.compiler.source>1.6</maven.compiler.source>  
 <maven.compiler.target>1.6</maven.compiler.target>  
 <maven.compiler.compilerVersion>1.6</maven.compiler.compilerVersion>  
 </properties>  
 </profile>   
 <profile>  
 <id>central</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>central</activeProfile>  
 <activeProfile>JDK1.6</activeProfile>  
 </activeProfiles>  
</settings>

[复制代码](javascript:void(0);)

**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.XML配置如下：

[复制代码](javascript:void(0);)

<project>  
 <distributionManagement>  
 <repository>  
 <id>nexus-releases</id>  
 <name>Nexus Releases Repository</name>  
 <url>http://localhost:4040/nexus/content/repositories/releases</url>   
 </repository>  
 <snapshotRepository>  
 <id>nexus-snapshots</id>  
 <name>Nexus Snapshots Repository</name>  
 <url>http://localhost:4040/nexus/content/repositories/snapshots</url>  
 </snapshotRepository>  
 </distributionManagement>  
</project>

[复制代码](javascript:void(0);)

Nexus 的仓库对于黎明用户是只读的为了能够部署构件，还需要在setting.xml 中配置认证信息代码如下：

[复制代码](javascript:void(0);)

<servers>  
   
 <server>  
 <id>releases</id>  
 <username>admin</username>  
 <password>iapppay</password>  
 </server>  
 <server>  
 <id>snapshots</id>  
 <username>admin</username>  
 <password>iapppay</password>  
 </server>  
 </servers>

[复制代码](javascript:void(0);)