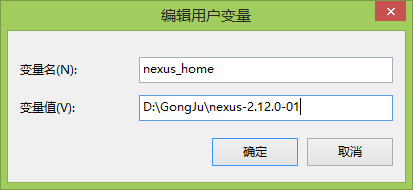
下载地址：<http://www.sonatype.org/nexus/>

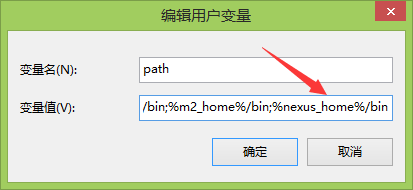
# 安装nexus

## 1.1解压zip包



## 1.2配置环境变量





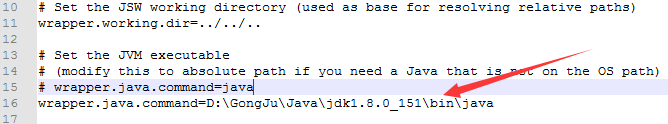
## 1.3安装nexus服务

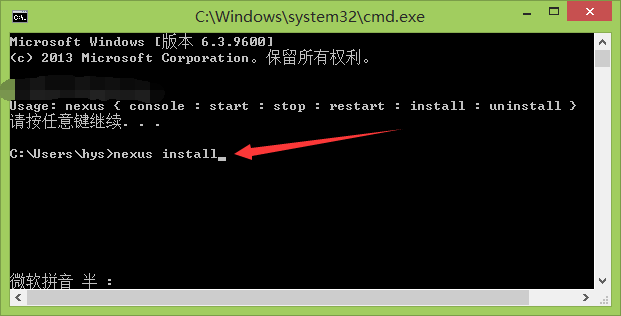
**注意：**

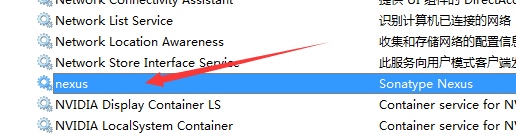
1. 用管理员打开命令窗，否侧无法安装。
2. 安装之前需要配置jdk路劲：

配置文件：wrapper.conf

文件路径：D:\GongJu\nexus-2.12.0-01\bin\jsw\conf



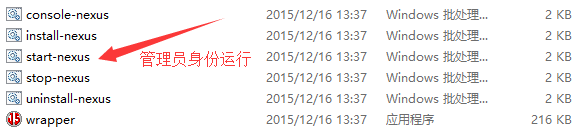




## 1.4访问nexus私服

1）启动nexus服务

也可以通过批处理启动服务：D:\GongJu\nexus-2.12.0-01\bin\jsw\windows-x86-64

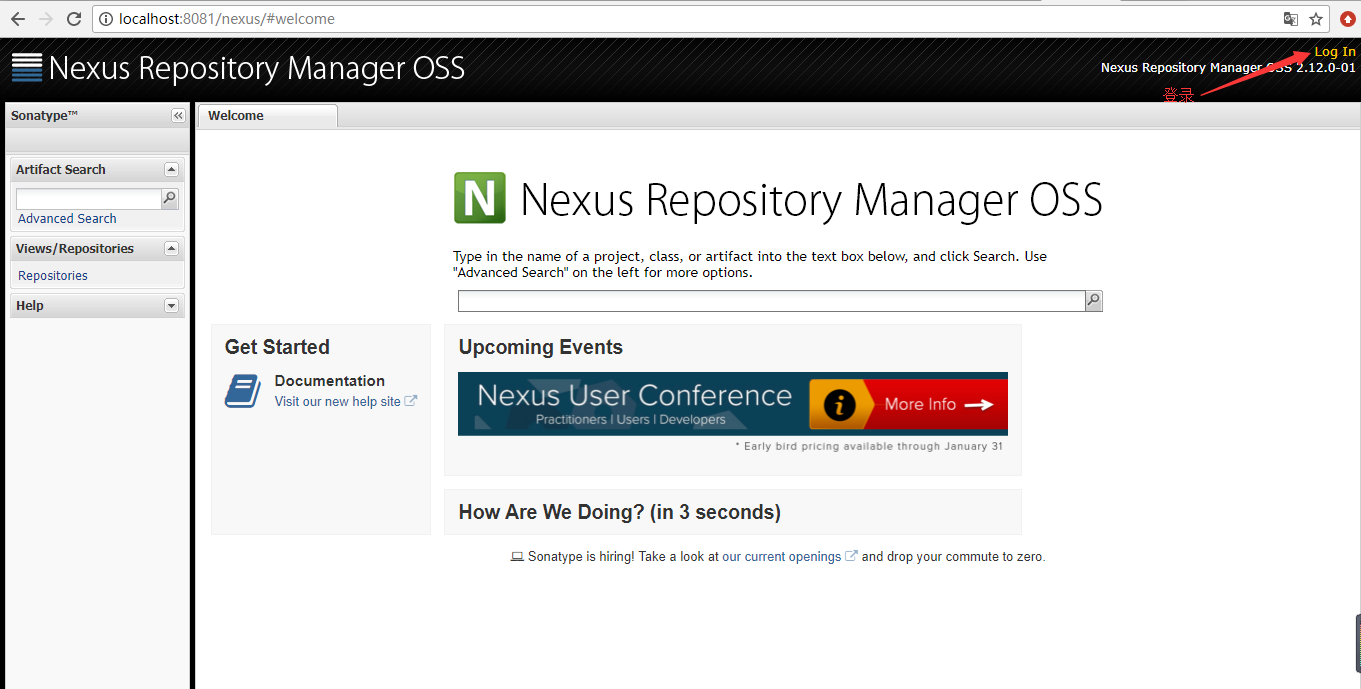


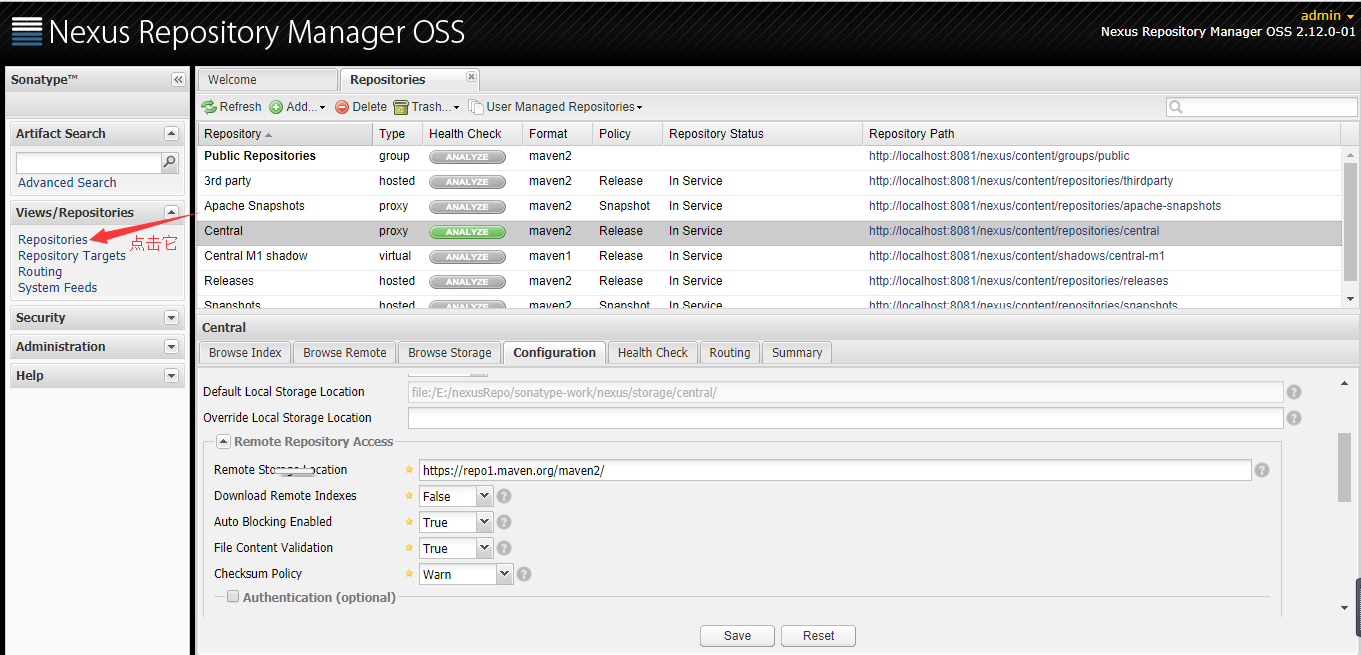
2）访问nexus

地址：http://localhost:8081/nexus

用户名：admin

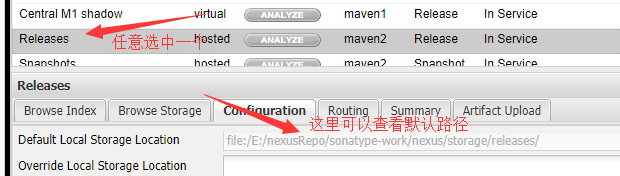
密码：admin123





## 1.5修改nexus默认jar包存放地址

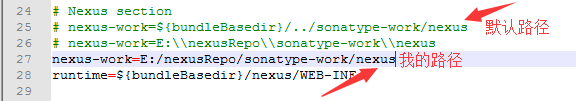




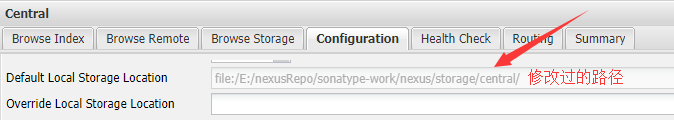
以上是已经修改过的路径（非默认路径）

1. 修改nexus.properties文件

文件路径：D:\GongJu\nexus-2.12.0-01\conf



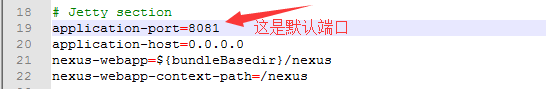
1. 重启服务



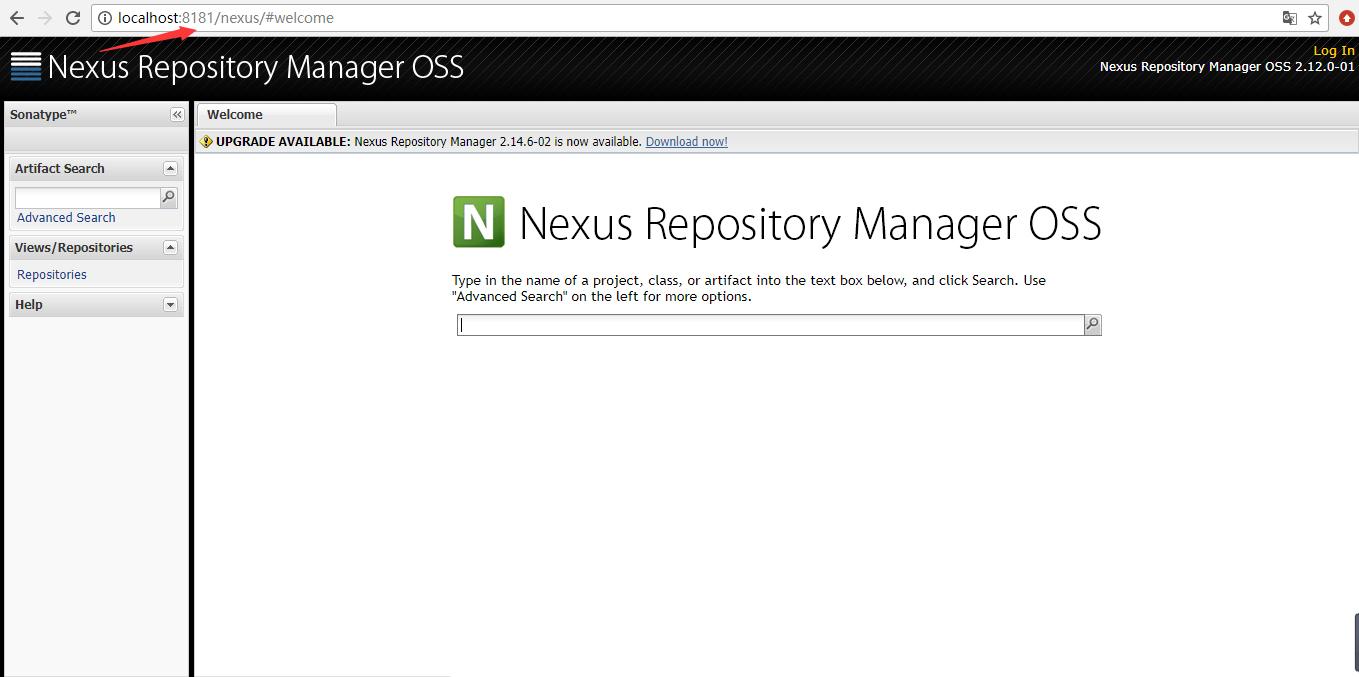
## 1.6修改nexus默认端口

1. 修改nexus.properties文件

文件路径：D:\GongJu\nexus-2.12.0-01\conf



1. 修改为8181（重启服务）

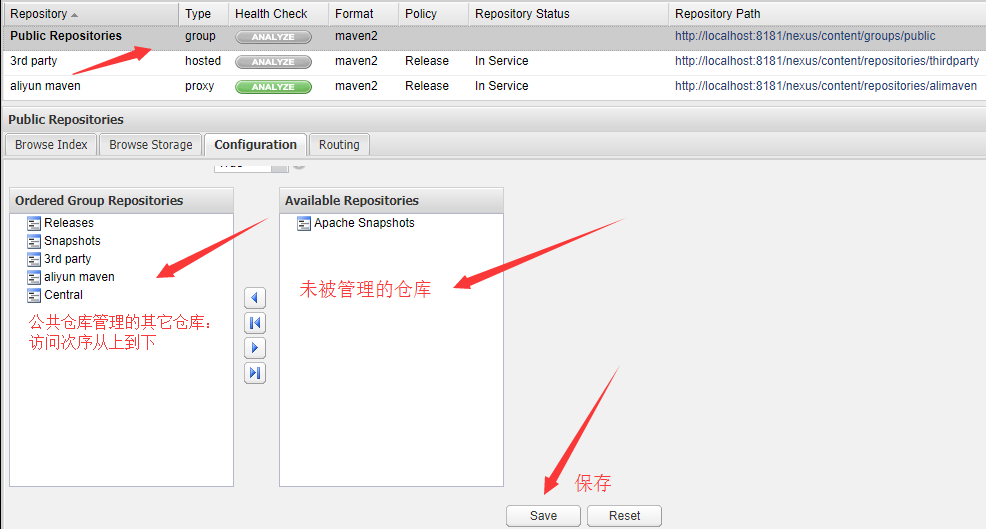


# nexus仓库介绍



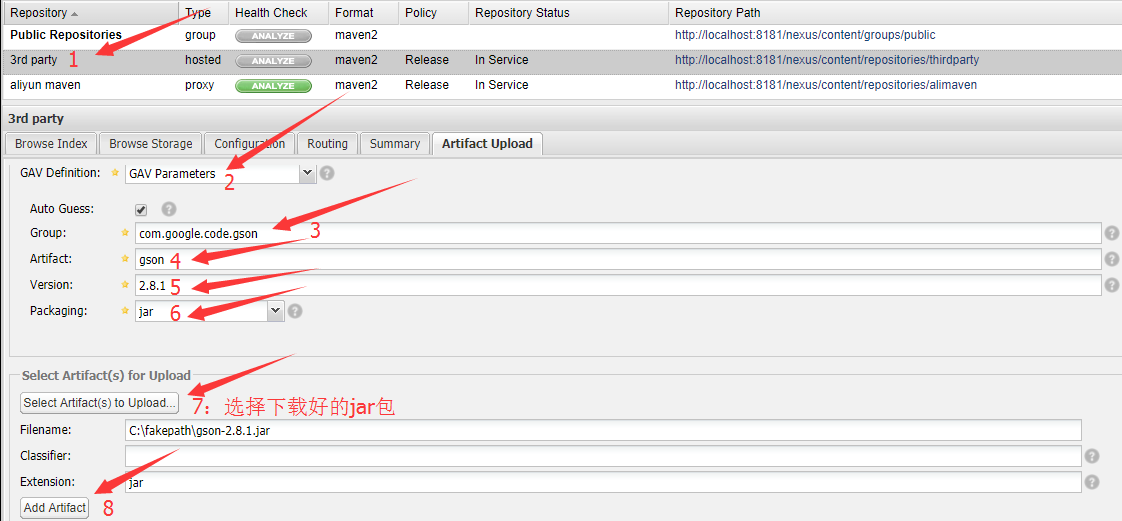
## 2.1 Public Repositories公共仓库

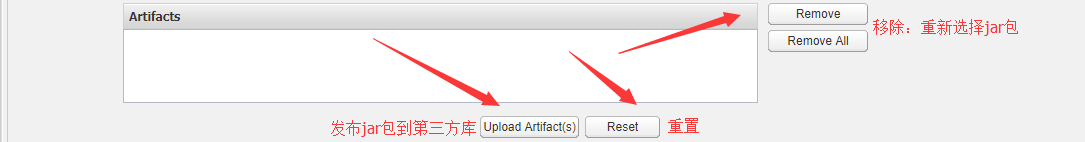
管理其它仓库，在maven中直接引用这个仓库就可以访问公共仓库里的其它仓库；公共仓库里的其它仓库是可配的。



## 2.2 3rd party第三方仓库

远程中央仓库没有需要的依赖包，可以下载下来再发布到这个仓库里。







## 2.3 Apache Snapshots代理仓库

Apache Snapshots的远程中央仓库。

## 2.4 Central代理仓库

Maven2的远程中央仓库。

## 2.5 Central M1 shadow虚拟仓库

Maven1仓库：现在基本上用不到。

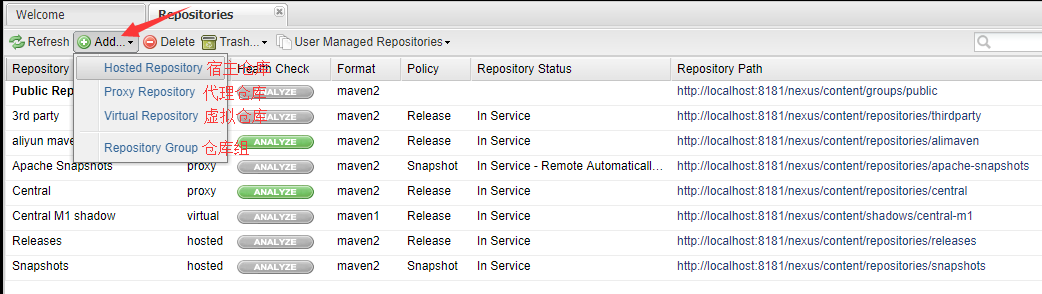
## 2.6 Releases宿主仓库

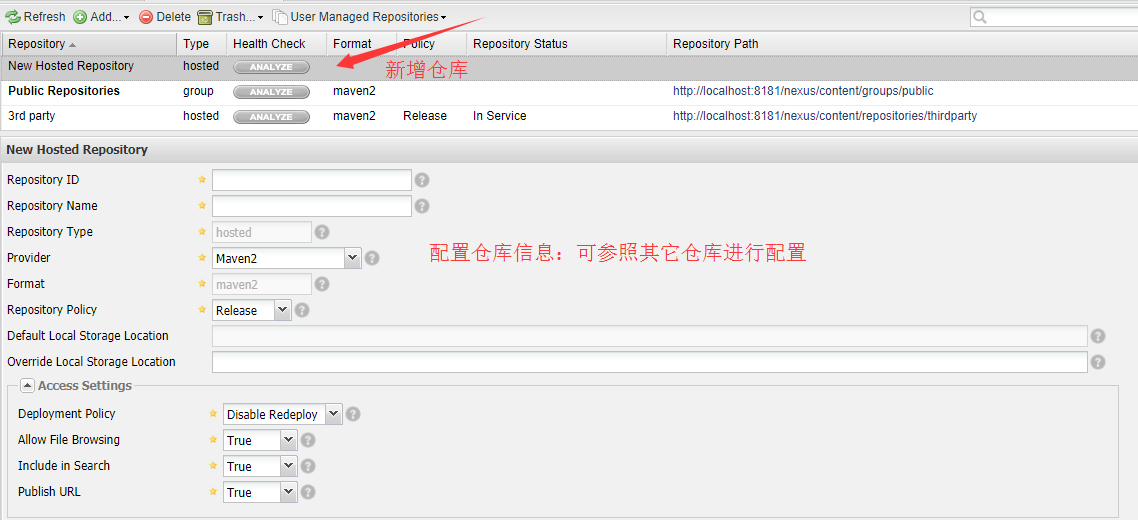
正式发布的版本仓库：项目组已经开发好的jar包一般发布到这个仓库里。

## 2.7 Snapshots宿主仓库

正在开发中的版本仓库：项目组还没开发完善的jar包一般发布到这个仓库里。

# 3、新增仓库





# maven使用nexus私服

## 3.1 maven中使用nexus私服

**依赖包搜索顺序：（前三种方式）**

1）在 Maven 本地资源库中搜索，如果没有找到，进入第 2 步，否则退出。

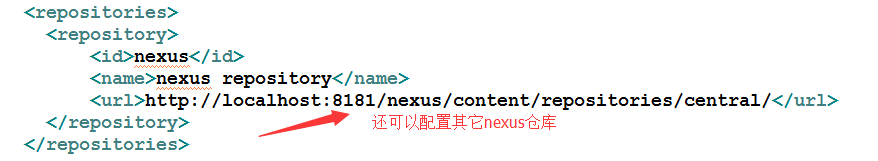
2）在nexus私服中搜索（需要配置），如果找不到，进入第3步，否侧退出。

3）在 Maven 中央存储库搜索，如果没有找到，提示错误，否则退出。

## 3.2 方式一

直接引用nexus里的仓库：引用nexus的中央仓库，只会到nexus的中央仓库里搜索依赖包。

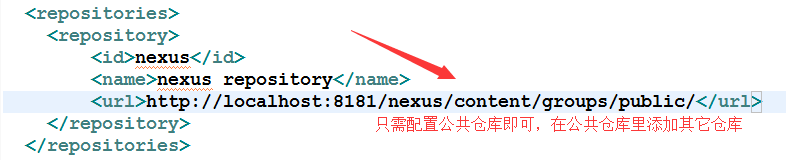
pom.xml



## 3.3方式二

引用nexus的公共仓库：会到nexus的公共仓库下的所有仓库里搜索依赖包。

pom.xml



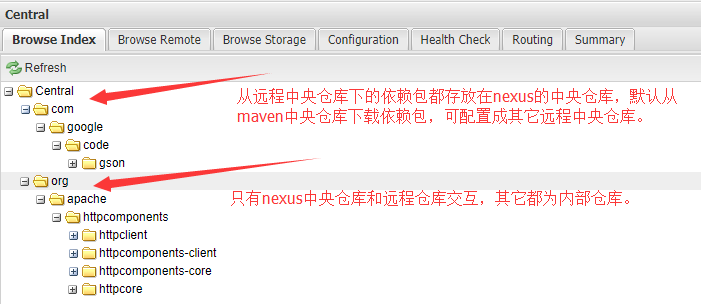
## 3.4方式三

在maven的settings.xml文件里配置nexus公共仓库，这样不用每次创建新项目时再配

置nexus公共仓库。

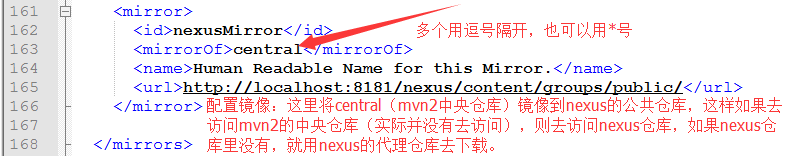


可以在nexus中央仓库查看从远程中央仓库下载的依赖包：如下图



## 3.5方式四：配置镜像（推荐）

在maven的settings.xml文件里配置镜像；说明：所有其它仓库都可以镜像到nexus的公共仓库（公共仓库管理nexus的其它的仓库），这样如果本地仓库没有需要的依赖包，都会到nexus仓库寻找，如果nexus仓库也没有，就用nexus的代理仓库去下载，先下载到nexus的对应仓库里，再从nexus仓库里下载到本地仓库。







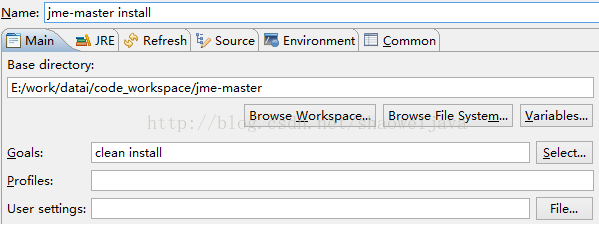


**注意：**这种方式如果本地仓库有需要的依赖包，但是nexus仓库里却没有，代码会报错；解决方式：删掉本地仓库里对应的依赖包，让nexus重新下载即可；或者更新nexus代理仓库的索引，但是下载索引很慢，离线更新又不一定能解决问题，还是直接删除，重新下载吧。

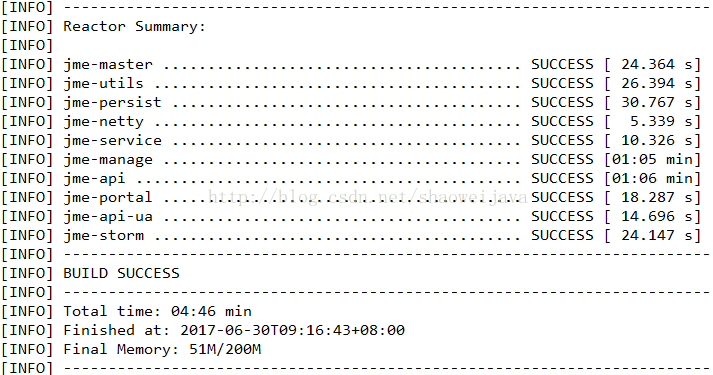
# 5、发布项目到本地仓库和私服仓库

**一、    将项目发布到本地maven仓库**

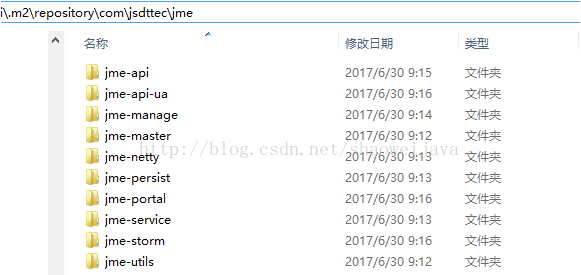
输入clean install



运行完成后，控制台可以看到成功的信息



然后去~\.m2\repository目录可以看到已经发布到本地仓库的文件



**二、    将项目发布到远程私服库**

在公司项目中，不同项目组之间，可能需要公用代码，互相依赖包的情况，这时候需要把自己的应用包发布到公司的私服仓库中，以便其他同事使用，下面介绍发布大nexus私服仓库的步骤：

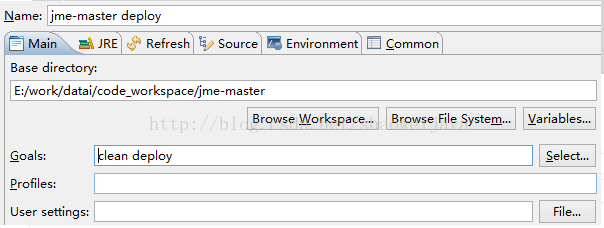
发布一个项目可以在nexus网页上upload，也可以通过mvn deploy命令方式发布，如果是后者则在pom.xml中添加distributionManagement标签

1. <distributionManagement>
2. <repository>
3. <id>releases</id>
4. <name>Nexus ReleaseRepository</name>
5. <url>http://192.168.10.6:8081/nexus/content/repositories/releases/</url>
6. </repository>
7. <snapshotRepository>
8. <id>snapshots</id>
9. <name>Nexus SnapshotRepository</name>
10. <url>http://192.168.10.6:8081/nexus/content/repositories/snapshots/</url>
11. </snapshotRepository>
12. </distributionManagement>

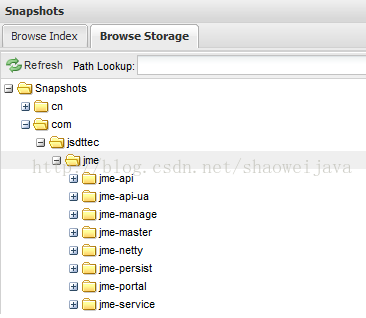
在maven的settings.xml文件中添加服务器配置，仓库id、用户名和密码，id需要与上面pom.xml中的仓库id一致，如果不一致会报Error code 401, Unauthorized错误，id可以与私服库上的id不一致，url正确即可

1. <servers>
2. <server>
3. <id>releases</id>
4. <username>admin</username>
5. <password>xxx</password>
6. </server>
7. <server>
8. <id>snapshots</id>
9. <username>admin</username>
10. <password>xxx</password>
11. </server>
12. </servers>

输入clean deploy



执行完成之后，查看nexus服务器，点开部署的snapshots仓库，则可以看到刚才新安装的项目



到此发布完成，可能有人会注意到，上面在settings.xml文件中保存的服务器密码是明文方式保存的，存在一定的安全隐患，下面介绍下如何加密保存

很简单，通过maven提供的命令就可以完成，mvn -emp or mvn --encrypt-master-password

首先创建个住安全密码：

E:\maven\apache-maven-3.2.5\bin>mvn-emp mypwd

{jHorajwXpMMH8urL7y9Xeh3vCOaW9OniO8la9ooPMOE=}

然后将生成的密码，copy到*~/.m2/settings-security.xml*中，没有这个文件则自己创建个

1. <settingsSecurity>
2. <master>{jHorajwXpMMH8urL7y9Xeh3vCOaW9OniO8la9ooPMOE=}</master>
3. </settingsSecurity>

然后将服务器密码进行加密

E:\maven\apache-maven-3.2.5\bin>mvn-ep xxx

{M80C6eJf+6IHYBPHzJr/mTOl12vl2aAOMPLMQAer6wg=}

然后将加密之后的密文替换之前的服务器明文密码

1. <servers>
2. <server>
3. <id>releases</id>
4. <username>admin</username>
5. <password>{M80C6eJf+6IHYBPHzJr/mTOl12vl2aAOMPLMQAer6wg=}</password>
6. </server>
7. <server>
8. <id>snapshots</id>
9. <username>admin</username>
10. <password>{M80C6eJf+6IHYBPHzJr/mTOl12vl2aAOMPLMQAer6wg=}</password>
11. </server>
12. </servers>

**三、**[**snapshot快照库和release发布库的应用**](http://www.mzone.cc/article/279.html)

首先说明下snapshop快照库和release发布库的区别：

snapshop快照仓库：用于保存还处于开发过程中不稳定的版本

release发布仓库：用于保存稳定的发行版本

定义一个模块为快照版本，需要在版本号后面加上-SNAPSHOT(必须是大写)

maven的依赖是基于版本管理的，对于发布状态的artifact，如果版本号相同，即使远程仓库的包比本地新，maven不会更新下载，还是会使用本地的旧包。如果在开发过程中，我们需要更改功能、修改bug，则需要频繁的发布包，这个时候每次发布都需要升级版本号，是不合理的，所以maven提供了快照版本，在不更改版本号的情况下，会自动下载最新的快照版本。

在mvn deploy的时候，根据snapshot和release的规则，判断版本号后面是否带有-SNAPSHOT，带有则认为是快照版本，会发布到snapshot库，不带则认为是发布版本，会发布到release库中。如过pom.xml没有配置快照库，只有发布库的话，而需要发布-SNAPSHOT版本默认是发布不了的，会报错：Error code 400, Bad Request，nexus的策略禁止了这样的操作。