# Maven仓库管理

## 什么是Maven仓库？

用来统一存储所有Maven共享构建的位置就是仓库。根据Maven坐标定义每个构建在仓库中唯一存储路径大致为：groupId/artifactId/version/artifactId-version.packaging

## 仓库的分类

1、本地仓库

~/.m2/repository

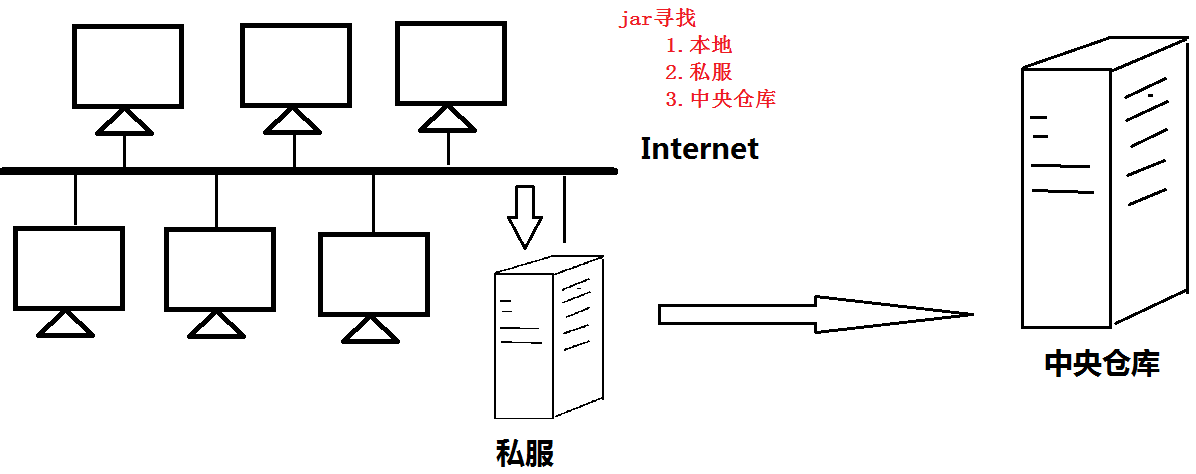
每个用户只有一个本地仓库

2、远程仓库

* 中央仓库：Maven默认的远程仓库，不包含版权资源

<http://repo1.maven.org/maven2>

* 私服：是一种特殊的远程仓库，它是架设在局域网内的仓库



# Maven私服Nexus

## 安装Nexus

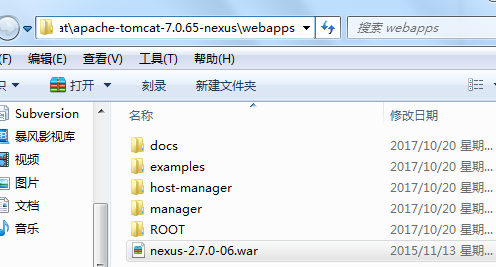
为所有来自中央仓库的构建安装提供本地缓存。

下载网站：<http://nexus.sonatype.org/>

安装版本：nexus-2.7.0-06.war

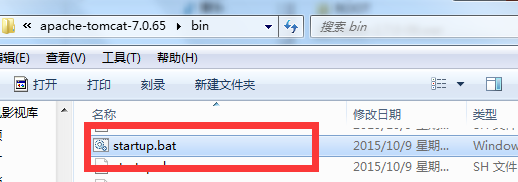
### 第一步：

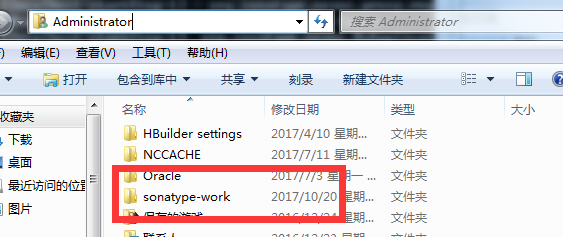
将下载的nexus的war包复制到tomcat下的webapps目录。



### 第二步：启动tomcat。

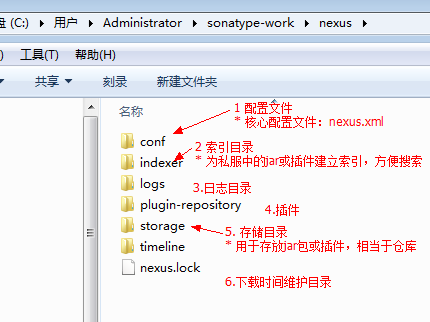
nexus将在c盘创建sonatype-work目录【C:\Users\当前用户\sonatype-work\nexus】。



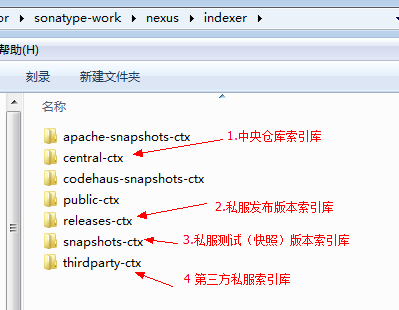


## Nexus的目录结构

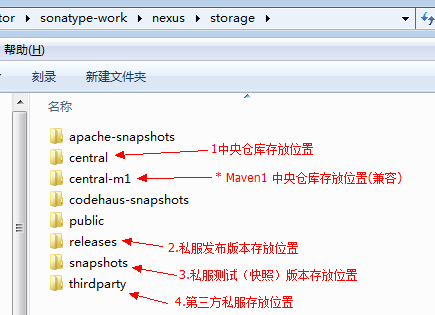
* 目录结构如下：



* Indexer 索引目录结构：



* Storage存储目录结构：



## 访问Nexus

### 登录

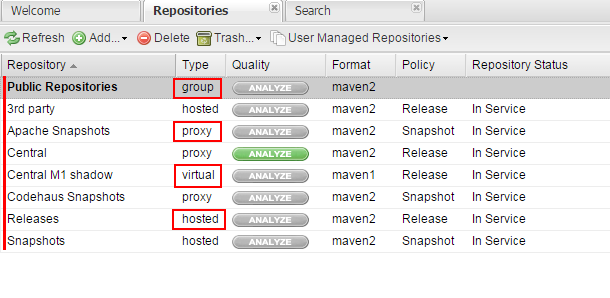
访问URL: <http://localhost:8080/nexus-2.7.0-06/>

默认账号:

用户名： admin

密码： admin123

### Nexus的仓库和仓库组



**仓库有4种类型 :**

* group(仓库组)：一组仓库的集合
* hosted(宿主)：配置第三方仓库 （包括公司内部私服 ）
* proxy(代理)：私服会对中央仓库进行代理，用户连接私服，私服自动去中央仓库下载jar包或者插件
* virtual(虚拟)：兼容Maven1 版本的jar或者插件

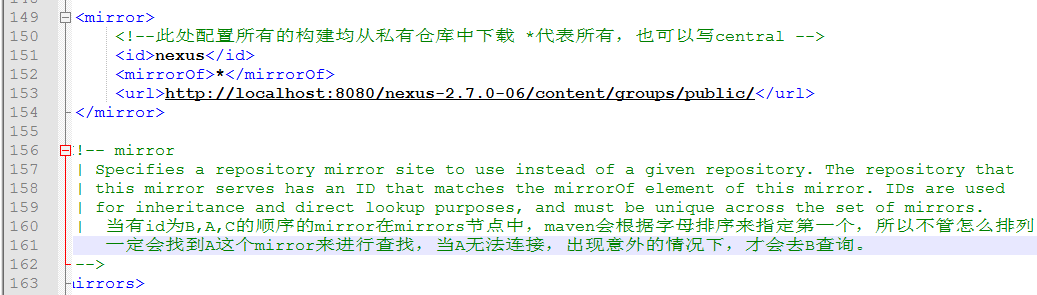
**Nexus的仓库和仓库组介绍:**

* 3rd party: 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建
* Apache Snapshots: 一个策略为Snapshot的代理仓库，用来代理Apache Maven仓库的快照版本构建
* Central: 代理Maven中央仓库
* Central M1 shadow: 代理Maven1 版本 中央仓库
* Codehaus Snapshots: 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
* Releases: 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: 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
* **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**

## 配置所有构建均从私服下载

在本地仓库的setting.xml中配置如下：

|  |
| --- |
| <mirrors>  <mirror>  <!--此处配置所有的构建均从私有仓库中下载 \*代表所有，也可以写central -->  <id>nexus</id>  <mirrorOf>\*</mirrorOf>  <url>http://localhost:8080/nexus-2.7.0-06/content/groups/public/</url>  </mirror>  </mirrors> |

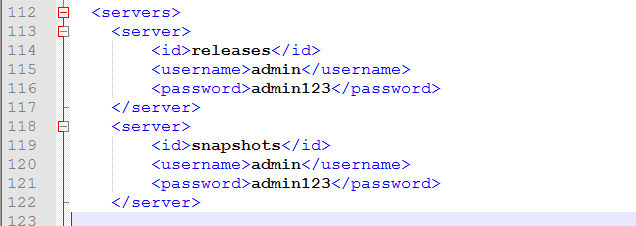


## 部署构建到Nexus

### 第一步：Nexus的访问权限控制

在本地仓库的setting.xml中配置如下：

|  |
| --- |
| <server>  <id>releases</id>  <username>admin</username>  <password>admin123</password>  </server>  <server>  <id>snapshots</id>  <username>admin</username>  <password>admin123</password>  </server> |

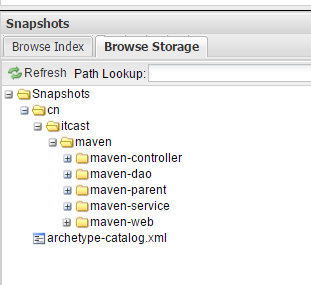


### 第二步：配置pom文件

在需要构建的项目中修改pom文件

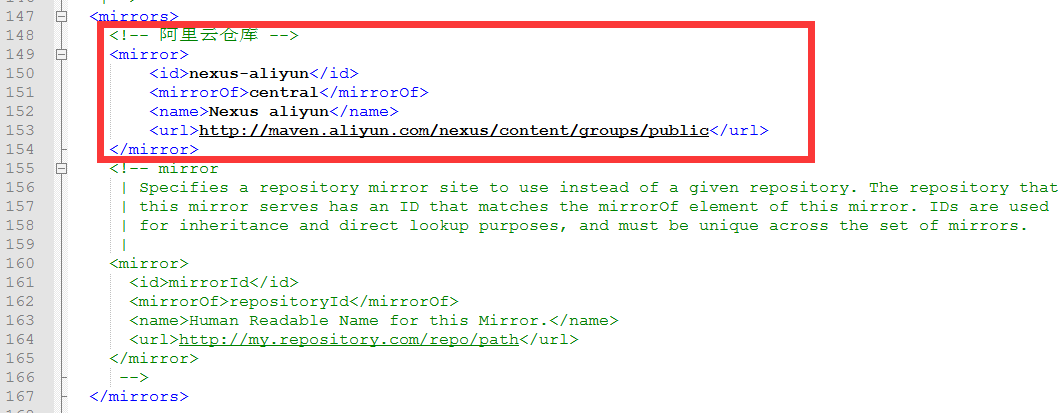
|  |
| --- |
| <distributionManagement>  <repository>  <id>releases</id>  <name>Internal Releases</name>  <url>http://localhost:8080/nexus-2.7.0-06/content/repositories/releases/</url>  </repository>  <snapshotRepository>  <id>snapshots</id>  <name>Internal Snapshots</name>  <url>http://localhost:8080/nexus-2.7.0-06/content/repositories/snapshots/</url>  </snapshotRepository>  </distributionManagement> |

### 第三步：执行maven的deploy命令



# 中央仓库

## 阿里云仓库



<!-- 阿里云仓库 -->

<mirror>

<id>nexus-aliyun</id>

<mirrorOf>central</mirrorOf>

<name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

## 私服和阿里云搭配

http://www.manongjc.com/detail/8-argxgrzgisoomqw.html

### Pom文件使用

<repository>

<id>alimaven</id>

<name>Maven Aliyun Mirror</name>

<url>http://maven.aliyun.com/nexus/content/repositories/central/</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

https://www.cnblogs.com/cralor/p/9092971.html