Maven 本地仓库，远程仓库，中央仓库，Nexus私服，镜像 详解

标签： Maven 本地仓库 远程仓库 中央仓库 Nexus私服 镜像

2017-06-14 21:12 85人阅读 评论(0)

分类： Maven (3)

一. 本地仓库

本地仓库是远程仓库的一个缓冲和子集，当你构建Maven项目的时候，首先会从本地仓库查找资源，如果没有，那么Maven会从远程仓库下载到你本地仓库。这样在你下次使用的时候就不需要从远程下载了。如果你所需要的jar包版本在本地仓库没有，而且也不存在于远程仓库，Maven在构建的时候会报错，这种情况可能发生在有些jar包的新版本没有在Maven仓库中及时更新。

Maven缺省的本地仓库地址为${user.home}/.m2/repository 。也就是说，一个用户会对应的拥有一个本地仓库。当然你可以通过修改${user.home}/.m2/settings.xml 配置这个地址：

Xml代码

```
1. <settings>
2.   <localRepository> E:/repository/maven/repos</localRepository>
3. </settings>
```

如果你想让所有的用户使用统一的配置那么你可以修改Maven主目录下的setting.xml:

${M2_HOME}/conf/setting.xml

二. 远程仓库

除本地仓库以外的仓库都叫做远程仓库

本地仓库配置在： <localRepository> E:/repository/maven/repos</localRepository>

远程仓库配置在：

```
<profiles>
    <profile>
        <id></id>
        <repositories>
            <repository>远程仓库配置</repository>
        </repositories>
    </profile>
</profiles>
```

三. 中央仓库

中央仓库也属于远程仓库的一种，特征就是 <id>central</id> id名为 central，

即，告诉Maven从外网的哪个地方下载jar包

Maven的安装目录中，在lib目录下，maven-model-builder-3.1.0.jar中，有一个默认的pom.xml文件

其中就配置了Maven默认连接的中心仓库

Maven的中央仓库地址默认是：https://repo.maven.apache.org/maven2/,可以通过修改settings.xml文件来修改默认的中央仓库地址：

```
<profile>
  <id>repository-profile</id>
  <repositories>
    <repository>
      <id>central</id>
      <name>Central Repository</name>
      <layout>default</layout>
      <url>http://maven.aliyun.com/nexus/content/groups/public</url>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
      <releases>
        <enabled>true</enabled>
      </releases>
    </repository>
  </repositories>
</profile>
```

要注意的是如果修改的是中央仓库地址，那么repository下面的id标签值一定得是central，此外，还需要激活这个profile才能生效，这里的标签值就是profile标签下面的id标签值

```
<activeProfiles>
        <activeProfile>repository-profile</activeProfile>
</activeProfiles>
```

四. Nexus私服

私服也属于远程仓库的一种，只是这个远程仓库的地址是本地服务器而已

配置在局域网环境中，为局域网中所有开发人员提供jar包的统一管理

本地仓库（本机）--->私服（局域网）--->中心仓库（外部网络）

私服的安装

1.下载NEXUS，http://www.sonatype.org

2.解压

3.配置环境变量：

新建环境变量：NEXUS_HOME = E:\soft\nexus-2.5.1-01

加入到path中：%NEXUS_HOME%\bin;

4.打开CMD命令行

C:\Users\Administrator>**nexus install** **安装服务**

C:\Users\Administrator>**nexus start** **启动服务**

C:\Users\Administrator>**nexus uninstall** **卸载服务**

5.访问私服

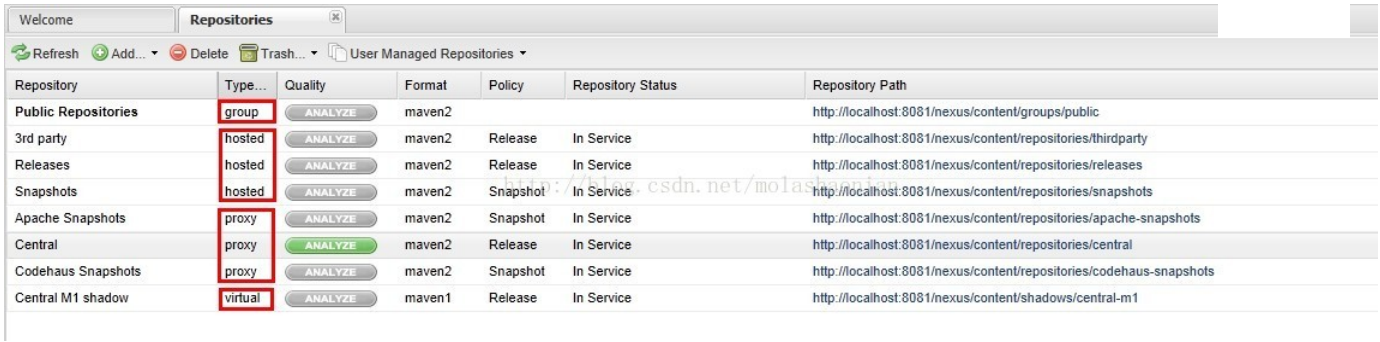
使用默认账户：admin 密码：admin123

NEXUS内部使用Jetty作为服务器

http://localhost:8081/nexus 【界面用extjs开发的】

仓库的分类

查看Repository



Repository	Type...	Quality	Format	Policy	Repository Status	Repository Path
Public Repositories	group	ANALYZE	maven2			http://localhost:8081/nexus/content/groups/public
3rd party	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/thirdparty
Releases	hosted	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/releases
Snapshots	hosted	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/snapshots
Apache Snapshots	proxy	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/apache-snapshots
Central	proxy	ANALYZE	maven2	Release	In Service	http://localhost:8081/nexus/content/repositories/central
Codehaus Snapshots	proxy	ANALYZE	maven2	Snapshot	In Service	http://localhost:8081/nexus/content/repositories/codehaus-snapshots
Central M1 shadow	virtual	ANALYZE	maven1	Release	In Service	http://localhost:8081/nexus/content/shadows/central-m1

host仓库--->内部项目的发布仓库

Snapshots 发布内部snapshots版本的仓库

Releases 发布内部release版本的仓库

3rd party 发布第3方jar包的仓库，如oracle数据库驱动，open-189.jar

proxy仓库--->从远程中心仓库查找jar包的仓库

Apache Snapshots 查找Apache项目的快照版本的仓库

Central 中心仓库http://repo1.maven.org/maven2/

Codehaus Snapshots 查找Codehaus 的快照版本的仓库

group仓库--->把仓库按组划分，以组为单位进行管理

virtual仓库

私服的配置/ Repository的配置

在parent模块的pom.xml中加入私服的配置，让Maven从私服下载jar包，而不直接去远程仓库下载。

默认情况下，Maven下载jar包将直接连接到外网http://repo1.maven.org/maven2/去下载；

安装私服之后，让Maven下载jar包先从私服查找，如果没有，再从外网下载并保存在私服上

在POM在加入下面的配置，其中url为NEXUS私服的Public Repository对外的地址

以后，Maven下载构建（jar包或插件）都将从这里开始下载

Xml代码

```
[html]
<span style="font-size:14px;"><project>

  ...

  <!-- 配置私服地址 -->
  <repositories>
    <repository>
      <id>nexus</id>
      <url>http://localhost:8081/nexus/content/groups/public/</url>
      <snapshots><enabled>true</enabled></snapshots>
      <releases><enabled>true</enabled></releases>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>nexus</id>
      <url>http://localhost:8081/nexus/content/groups/public/</url>
      <snapshots><enabled>true</enabled></snapshots>
      <releases><enabled>true</enabled></releases>
    </pluginRepository>
  </pluginRepositories>

  ...

<project>
 </span>
```

通过settings.xml来配置私服

由于所有的Maven项目都会用settings.xml中的配置进行解析，如果将Repository配置到这个文件中，那么对所有的Maven项目都将生效。

此时，Maven项目中的POM文件就不需要再配置私服地址了！

注意：修改settings.xml文件时，看IDE中关联的是哪个settings文件。

如C:\user\.m2目录下可能存在，Maven的解压目录下也存在，具体修改哪个根据实际情况而定。如，Eclipse下，查看Maven的User Settings选项即能看到关联。

我的IDE关联的是Maven\conf目录下的settings.xml:

E:\soft\apache-maven-3.1.0\conf\settings.xml

首先，通过<profile/>添加Repository和pluginRepository

Xml代码

```
[html]
<span style="font-size:14px;"><settings>

  ...

  <profiles>
     <profile>
      <id>profile-nexus</id>

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

```
        <releases><enabled>true</enabled></releases>
      </pluginRepository>
    </pluginRepositories>
  </profile>
 </profiles>

 ...

</settings>

</span>
```

然后，使用<activeProfiles>对上面的配置进行激活（通过配置的id标识进行激活）

Xml代码

```
[html]

<span style="font-size:14px;"><activeProfiles>
  <activeProfile>profile-nexus</activeProfile>
</activeProfiles>
</span>
```

现在，本地机器上创建Maven项目，都会使用settings中有关仓库的配置了

本地仓库：

<localRepository>E:/repository/maven/repos</localRepository>

本地Maven下载的依赖包和插件都将放到E:/repository/maven/repos目录中

私服：

本地所有Maven项目，下载构建都统一从http://localhost:8081/nexus/content/groups/public/ 下载！

【私服上不存在某个构建时，再从远程仓库下载】

远程仓库：

如果远程仓库连接不上，则通过nexus修改central的地址即可！

当前使用Maven的默认配置：http://repo1.maven.org/maven2/

五. 镜像

Maven的镜像是指如果仓库X可以提供仓库Y存储的所有内容，那么就可以认为X是Y的一个镜像，也就是说任何一个可以从仓库Y获得的依赖，都能够从它的镜像中获取。比如阿里的Maven仓库http://maven.aliyun.com/nexus/content/groups/public就可以理解为是中央仓库https://repo.maven.apache.org/maven2/在中国的镜像，由于地理位置的因素，该镜像往往能够提供比中央仓库更快的服务。要为一个仓库配置镜像只需要修改settings.xml文件，如下，其中的mirrorOf标签值就是仓库的id标签值，中央仓库就是默认就是central，表示这是为中央仓库配置的镜像，以后所有的依赖下载都会从这个镜像中进行下载。

```
<mirror>
  <id>central-repository-mirror</id>
  <name>Central Repository Mirror</name>
  <mirrorOf>central</mirrorOf>
  <url>http://maven.aliyun.com/nexus/content/groups/public</url>
</mirror>
```

repository是指在局域网内部搭建的repository，它跟central repository, jboss repository等的区别仅仅在于其URL是一个内部网址
mirror则相当于一个代理，它会拦截去指定的远程repository下载构件的请求，然后从自己这里找出构件回送给客户端。配置mirror的目的一般是出于网速考虑。
不过，很多internal repository搭建工具往往也提供mirror服务，比如Nexus就可以让同一个URL,既用作internal repository，又使它成为所有repository的mirror。
高级的镜像配置：
1.<mirrorOf>*</mirrorOf>
匹配所有远程仓库。这样所有pom中定义的仓库都不生效
2.<mirrorOf>external:*</mirrorOf>
匹配所有远程仓库，使用localhost的除外，使用file://协议的除外。也就是说，匹配所有不在本机上的远程仓库。
3.<mirrorOf>repo1,repo2</mirrorOf>
匹配仓库repo1和repo2，使用逗号分隔多个远程仓库。
4.<mirrorOf>*,!repo1</miiroOf>
匹配所有远程仓库，repo1除外，使用感叹号将仓库从匹配中排除。

mirrors可以配置多个mirror，每个mirror有id,name,url,mirrorOf属性，id是唯一标识一个mirror就不多说了，name貌似没多大用，是官方的库地址，mirrorOf代表了一个镜像的替代位置，例如central就表示代替官方的中央库。

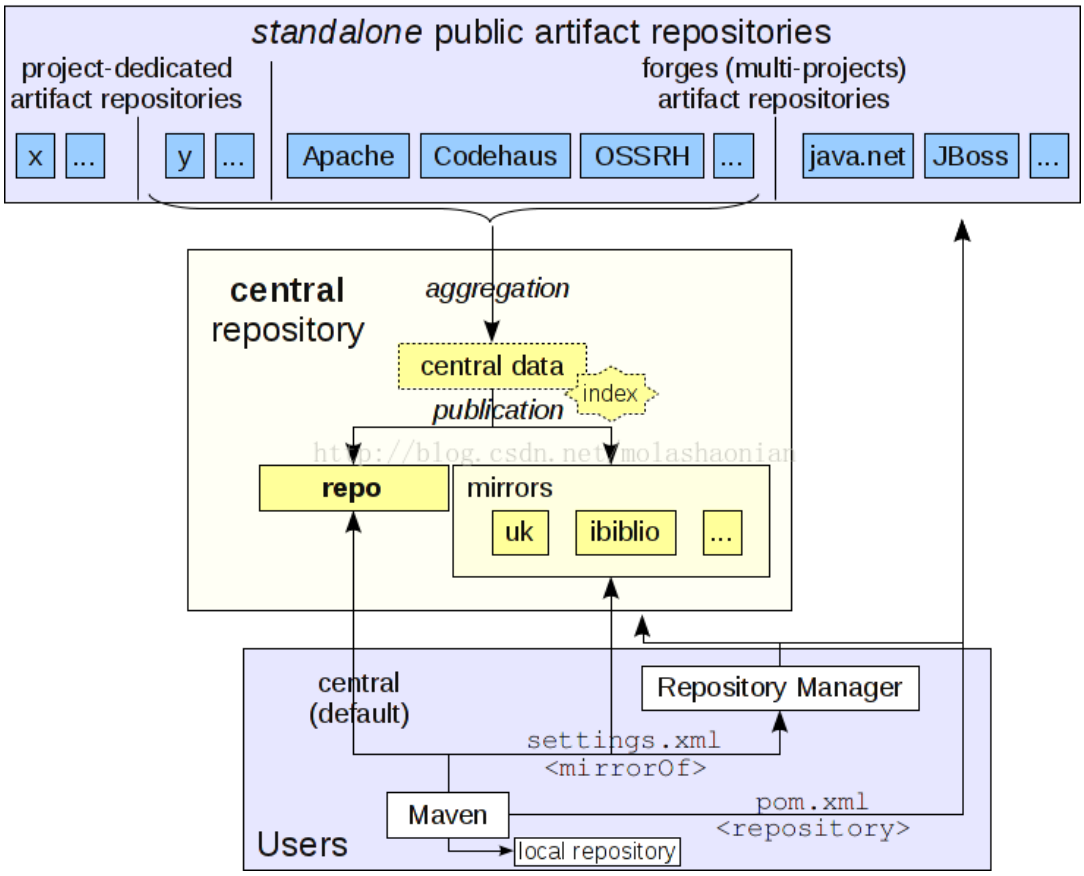
我本以为镜像库是一个分库的概念，就是说当a.jar在第一个mirror中不存在的时候，maven会去第二个mirror中查询下载。但事实却不个mirror中不存在a.jar的时候，并不会去第二个mirror中查找，甚至于，maven根本不会去其他的mirror地址查询。

后来终于知道，maven的mirror是镜像，而不是“分库”，只有当前一个mirror无法连接的时候，才会去找后一个，类似于备份和容灾

还有，mirror也不是按settings.xml中写的那样的顺序来查询的。

所谓的第一个并不一定是最上面的那个。

当有id为B,A,C的顺序的mirror在mirrors节点中，maven会根据字母排序来指定第一个，所以不管怎么排列，一定会找到A这个mirror来进行查找，当A无法连接，出现意外的情况下，才会去B查询。




```[html]
<span style="font-size:14px;"><?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

    <servers>
        <server>
```

```
            <id>repo-iss</id>
            <username>deployment</username>
            <password>deployment123</password>
        </server>
    </servers>

    <mirrors>
        <!-- osc镜像 -->
        <mirror>
            <!-- 镜像所有远程仓库，但不包括指定的仓库 -->
            <id>mirror-osc</id>
            <mirrorOf>external:*,!repo-osc-thirdparty,!repo-iss</mirrorOf>
            <url>http://maven.oschina.net/content/groups/public/</url>
        </mirror>
<!--
        <mirror>
            <id>mirror-iss</id>
            <mirrorOf>external:*</mirrorOf>
            <url>http://10.24.16.99:5555/nexus/content/groups/public/</url>
        </mirror>
-->
    </mirrors>

    <profiles>
        <profile>
            <id>profile-default</id>
            <repositories>
                <repository>
                    <id>central</id>
                    <url>http://central</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>false</enabled>
                    </snapshots>
                </repository>
                <repository>
                    <id>repo-osc-thirdparty</id>
                    <url>http://maven.oschina.net/content/repositories/thirdparty/</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>false</enabled>
                    </snapshots>
                </repository>
            </repositories>
            <pluginRepositories>
                <pluginRepository>
                    <id>central</id>
                    <url>http://central</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>false</enabled>
                    </snapshots>
                </pluginRepository>
            </pluginRepositories>
        </profile>
        <profile>
            <id>profile-iss</id>
            <repositories>
                <repository>
                    <id>repo-iss</id>
                    <url>http://10.24.16.99:5555/nexus/content/groups/public/</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </repository>
            </repositories>
            <pluginRepositories>
                <pluginRepository>
                    <id>repo-iss</id>
                    <url>http://10.24.16.99:5555/nexus/content/groups/public/</url>
```

```
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </pluginRepository>
            </pluginRepositories>
        </profile>
    </profiles>

    <activeProfiles>
        <activeProfile>profile-default</activeProfile>
        <!--<activeProfile>profile-iss</activeProfile>-->
    </activeProfiles>

<!--
    <proxies>
        <proxy>
            <active>true</active>
            <protocol>http</protocol>
            <host>10.10.204.160</host>
            <port>80</port>
        </proxy>
    </proxies>
-->
</settings>
</span>
```

Reference：

https://my.oschina.net/zhanghaiyang/blog/606130

顶 0　　踩 0

- 上一篇 CXF+JAXB处理复杂数据
- 下一篇 svn工具的使用问题总结

相关文章推荐

- 安卓搭建nexus私服-3.androidstudio中项目上传...
- Nexus搭建Maven私服(二) 分发构件至远程仓库
- Maven之搭建本地私服(nexus)仓库
- 一点一点学maven（03）——maven的坐标、构...
- 用nexus搭建自己的maven远程仓库。
- eclipse+maven+本地仓库+远程仓库+私服nexu...
- 使用Nexus2.x为Maven3.x搭建私服构件仓库
- maven学习笔记（四）仓库、nexus私服
- maven 添加jar到中央/远程仓库
- Maven总结【含配置文件】

猜你在找

C语言及程序设计（讲师：贺利坚）
Python全栈开发入门与实战课（讲师：李杰）
Python爬虫工程师培养课程全套（讲师:韦玮）
2017软考网络规划设计师视频套餐（讲师：任铄）