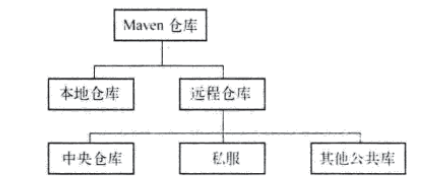
# Maven仓库

## 一 何为maven仓库

得益于maven的坐标机制,任何一个maven项目使用任何一个构建的方式是完全相同的。在此基础上maven可以在某个位置统一存储所有maven项目共享的构建。这个统一的位置就叫做仓库。

## 二 maven仓库分类



### 2.1 本地仓库

<localRepository>D:/Java/repo2</localRepository>

在setting文件中指定本地仓库的配置。Maven在执行编译测试时，如需要使用依赖的文件时，总是基于坐标使用本地依赖的文件。那么构建怎么从安装到本地仓库这里有三种方法

第一：配置pom.xml的索引文件，从远程仓库下载到本地。

第二: 通过执行maven的mvn install命令把构建安装到本地仓库。

第三: 一个纯粹的jar要放到本地仓库如odbc那么就要执行如下所示的mvn脚本命令安装到本地

mvn install:install-file

-Dfile=D:/taobao-sdk-java-auto-20160607-source.jar -DgroupId=com.ganshane.specs -DartifactId=taobao-sdk-java-auto-20160607-source

-Dversion=1.0.0

-Dpackaging=jar

如上所示的命令可以将一个特使的jar宝安装到本地。

### 2.2 远程仓库

本地仓库找不到时，才会访问远程仓库。

#### 2.2.1 远程仓库认证



Setting文件中配置认证字段。这里server的id要与pom文件中单个repository的id一致，id将认证信息与仓库配置联系在一起。认证信息必须配置在setting.xml文件中。

#### 2.2.2 远程仓库配置

<!-- 配置远程仓库 -->

<repositories>

<repository>

<id>central</id>

<name>JBoss Repository</name>

<url>http://repository.jboss.com/maven2/</url>

<releases>

<enabled>true</enabled>

<updatePolicy>daily</updatePolicy>

</releases>

<snapshots>

<enabled>false</enabled>

<checksumPolicy>warn</checksumPolicy>

</snapshots>

<layout>default</layout>

</repository>

<!-- 私有库地址 -->

<repository>

<id>nexus</id>

<url>http://192.168.232.130:8081/nexus/content/groups/public/</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

<pluginRepositories>

<!--插件库地址 -->

<pluginRepository>

<id>nexus</id>

<url>http://192.168.232.130:8081/nexus/content/groups/public/</url>

<releases>

<enabled>true</enabled>

</releases>

<snapshots>

<enabled>true</enabled>

</snapshots>

< /pluginRepository>

</pluginRepositories>

Maven自带的远程仓库的id为central。如果其他的仓库也使用该id会覆盖该仓库。这里可以配置一个或者多个远程仓库,Realease与snapshots都比较重要，他们用来控制maven对于发布构建与快照构建的下载。一般snapshots都为false为的就是防止编译的时候下载了还没有发布的构建，不稳定。updatePolicy配置是用来指定更新策略的,daily每天更新,never从不更新。always每次构建都检查。interval每格多长时间检查一次。

checksumPolicy 用来配置校验文件的，上传构建的时候会一起上传校验文件，下载的时候会执行校验和文件。如上所示片段可以配置在pom.xml文件中，也可以配置在setting.xml文件中，不同的是setting.xml文件中是全局的配置。Pom.xml中是只针对当前构建的。正如linux中的/etc/profile与.bash nano环境是一样的。

#### 2.2.3 发布构建到远程仓库

<distributionManagement>

<repository>

<id>nexus-releases</id>

<name>Nexus Release Repository</name>

<url>http://192.168.232.130:8081/nexus/content/repositories/releases/

</url>

</repository>

<snapshotRepository>

<id>nexus-snapshots</id>

<name>Nexus Snapshot Repository</name>

<url>http://192.168.232.130:8081/nexus/content/repositories/snapshots/

</url>

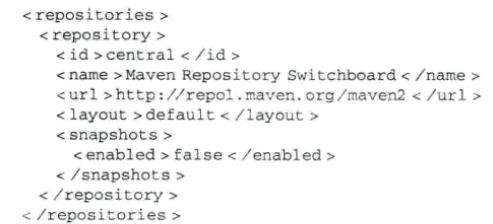
</snapshotRepository>

</distributionManagement>

如上所示repository表示发布版本的仓库，snapshotRepository表示快照版本的仓库。配置了如上所示的信息就可以把本地的构建部署到远程仓库了，然而这里要注意，如果需要认证的远程仓库也是，需要到setting.xml文件中配置认证信息的。

### 2.3 中央仓库

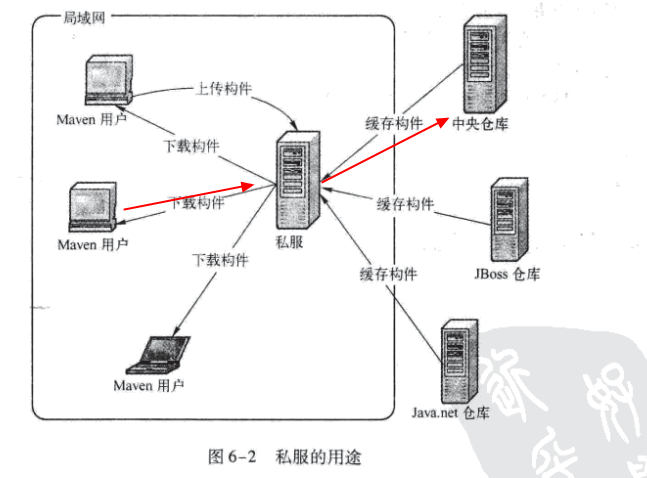
由于原始的本地仓库是空的，maven必须知道至少一个可用的远程仓库。才能执行maven命令的时候下载到需要的构建。中央仓库就是这样一个默认的远程仓库。Maven文件自带了中央仓库的配置。



中央仓库中包含了普通项目中用到的基本构建，这也解释了为什么maven能够开箱即用的原理。

### 2.4 私服

是一种特殊的远程仓库，是架设在局域网内的远程仓库服务。私服代理广域网上的远程仓库，供局域网内部的用户使用。用户下载构建的时候先走私服，私服没有的时候再走远程仓库，从远程仓库下载，缓存到私服以后再提供下载。



私服的作用:

1 节省自己的外网带宽

2 加速maven的构建

3 不熟第三方的构建

4 提高稳定性，增强控制

5 降低中央仓库的负荷

## 三 镜像

如果仓库X可以提供仓库Y存储的内容，那么我们就认为X是Y的一个镜像。换句话说任何一个可以从Y获得的内容，都可以从X获得。



如上图所示是镜像的配置，假设我们在国外的中央仓库地址是

**http://maven.net.cn/content/xxx** 然而在国内有一个仓库

<http://maven.net.cn/content/groups/public/可以完全替代国外的center仓库，我们就可以做如上所示的镜像配置> **<mirrorOf>的值为central表示任何代替中央仓库。当然这里也是需要认证的。这里常用来局域网内部maven的高效管理，列如**

<mirrors>

<mirror>

<id>mirrorId</id>

<name>localrepo</name>

<url>http://192.168.232.130:8081/nexus/content/groups/public/</url>

<mirrorOf>\*</mirrorOf>

</mirror>

</mirrors>

例如我们可以把，所有的远程仓库的映射都指向本地仓库镜像，这样他就会把所有的远程仓库的配置，指向本地私有服务器。

