Maven3 实战笔记 03Maven 仓库

刘岩

Email:suhuanzheng7784877@163.com

1. 仓库的概念

大家可能注意到了,在基于 Maven 管理的项目开发中,这个项目自身是不引进第三方 jar 包的,使用的时候通过 pom.xml 的依赖机制,从本地仓库或者远程仓库去获取第三方 jar 包。这个其实是打破了以往的开发习惯,一般我们是在开发项目的时候需要哪个 jar 包了,立刻 google 一下,找到相关网址,之后下载,放到我们项目的 classpath 中。现在是不必强制引用 jar,只要通过 pom.xml 配置,到一定的时刻,比如编译、测试、打包、部署,自然会将依赖的 jar 放进您预先的位置。Maven 仓库是基于简单文件系统存储的,根据咱们之前提到的坐标,可以找到该组件在仓库的位置。

2. 仓库的分类

一般说 Maven 的仓库就是指 2 个类型,一个就是我们自己的 PC 机器本地仓库,另一个就是指远程的 Maven 中心仓库(天啊,如果某天 Maven 组织宣布不对中国开放中心仓库那是怎样的局面啊?我的妈呀,太刺激啦~)。当本地仓库没有您需要的 jar 包的时候,它会从远程核心中心仓库中下载,所以说美国人想制约咱们中国的软件发展是很容易的,现在搞得连仓库都云计算、云存储了,真要是中美打起仗来……。不过我们有私服,私服就是另一种特殊的远程仓库,为了节省 Maven 核心仓库的带宽和时间,很多企业都在公司的局域网内搭建了私有的仓库服务器,内部项目先从私服下载东东,没有的时候私服从外网中心仓库下载,内部开发的项目还能上传到私服上供其他项目组使用。

3. 本地仓库明细

本地仓库大家已经不陌生了,默认是在用户临时文件夹的~/.m2/repository 下。

settings.xml 中

```
<!-- localRepository
 | The path to the local repository maven will use to store artifacts.
 |
 | Default: ~/.m2/repository
<localRepository>/path/to/local/repo</localRepository>
-->
```

是对本地仓库的说明。若想换地方，简单编辑<localRepository>标签内容即可。组件必须通过安装到本地仓库的过程才能提供给其他项目用。

```
mvn clean install
```

是安装组件的命令，通过此命令成功后，项目会打包注册到本地仓库中，之后其他项目就能把他当成第三方组件使用了。

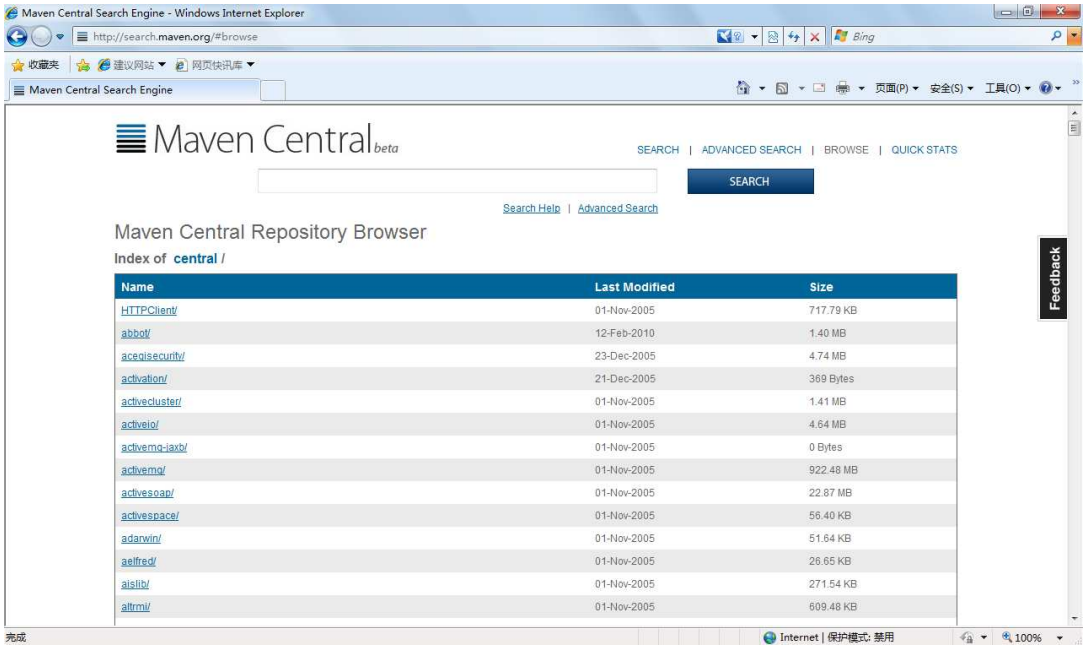
4. 中心仓库明细

中心仓库储存着目前最主流的开源组件（大概 2000 多个），随着项目需要要从中心仓库下载组件。Maven 自身配置了中心仓库的 url，若有一天，不幸 2012 来了，幸运的是仅仅美国淹没了，apache 的 maven 中心仓库服务器玩完了。那怎么办？没关系，我们仅仅配置一下 Maven 自己的源码 xml 即可修改默认中心仓库的位置，打开 ${Maven_HOME}/lib/maven-model-builder-3.0.3.jar 文件，打开里面的 org/apache/maven/model/pom-4.0.0.xml 文件。里面的

```
<pluginRepositories>
  <pluginRepository>
    <id>central</id>
```

```
        <name>Maven Plugin Repository</name>
        <url>http://repo1.maven.org/maven2</url>
        <layout>default</layout>
        <snapshots>
          <enabled>false</enabled>
        </snapshots>
        <releases>
          <updatePolicy>never</updatePolicy>
        </releases>
      </pluginRepository>
    </pluginRepositories>
```

里面的 url 就是中心仓库域名地址。Snapshots 代表不下载组件的快照版本。此配置还有一个作用就是，可以指定其位置是本局域网的私有服务器 url。



中央仓库是默认的远程仓库。私服需要其他软件辅助才能建立，这个咱们以后再说。

5. 远程仓库

其实远程仓库和中心仓库功能差不多，那区别呢就是当中心仓库里面的东西找遍了，还是没有我们项目需要的东西，那就证明此项目需要的东西比较特殊，中心仓库无法满足项目的需求。需要从特定的地方 download 下来，远程仓库，比如 JBoss Maven 仓库。

```
<project>
<repositories>
       <repository>
           <id>jboss</id>
           <url>http://repository.jboss.com/maven2/</url>
           <releases>
               <enabled>true</enabled>
           </releases>
           <snapshots>
               <enabled>false</enabled>
           </snapshots>
           <layout>default</layout>
       </repository>
</repositories>
</project>
```

项目还可以发布 deploy 到局域网服务器或者私服，关键是我们目前暂时还没介绍私服，先留着，到了那个时候再讨论。

6. 快照 SNAPSHOT

任何项目或者组件都有自己的版本，Maven 区分发布版本和快照版本，快照版本是为了解决如下问题的：在一个大项目组中分为模块 1、模块 2，假象他们都是十分复杂的模块啊。模块 2 的开发依赖于模块 1 的部分功能。如果不采用快照版本的概念，有可能模块 1 开发出一个版本 build 一次给模块 2 用，开发出一个版本就给 2 用，这样模块 1 的版本号有可能更新十分频繁，由“模块 1-201106010953.jar”一直到“模块 1-201106011953.jar”。那么还有一种保守的做法就是模块 2 先等等，等我们模块 1 都稳定了，做完了再 build 一个版本出来给你使用。这两种方案好像都不是很好，前者好像 build 太多，版本滥用；后者貌似效率太低。所以 Maven 引入了快照版本的概念。当模块 1build 出来一个快照版

本后上传到本地的私服中后，即模块 2 下载的时候，Maven 会自动去寻找快照的最新版本，下载之。这样模块 2 就不必关心模块 1 的版本 build 了没有，有没有什么更新，模块 2 的 Maven 自然会去仓库下载、更新快照的新 build 版本。

当然了，项目经过了完整的测试后，就应该让它从试用期转正成为正式员工了。SNAPSHOT 版本对应了很多细小的时间戳版本。打个比方说就是项目好比是一个职位，而每一个快照版本都相当于这个职位不同时期的试用期员工，稳定了，久经考验了，没问题了，留下当正式员工，有问题的，就留在了历史的长河中，成为了神马~~~~。