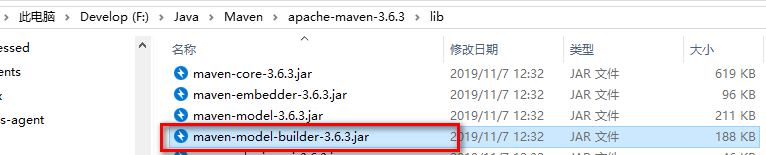
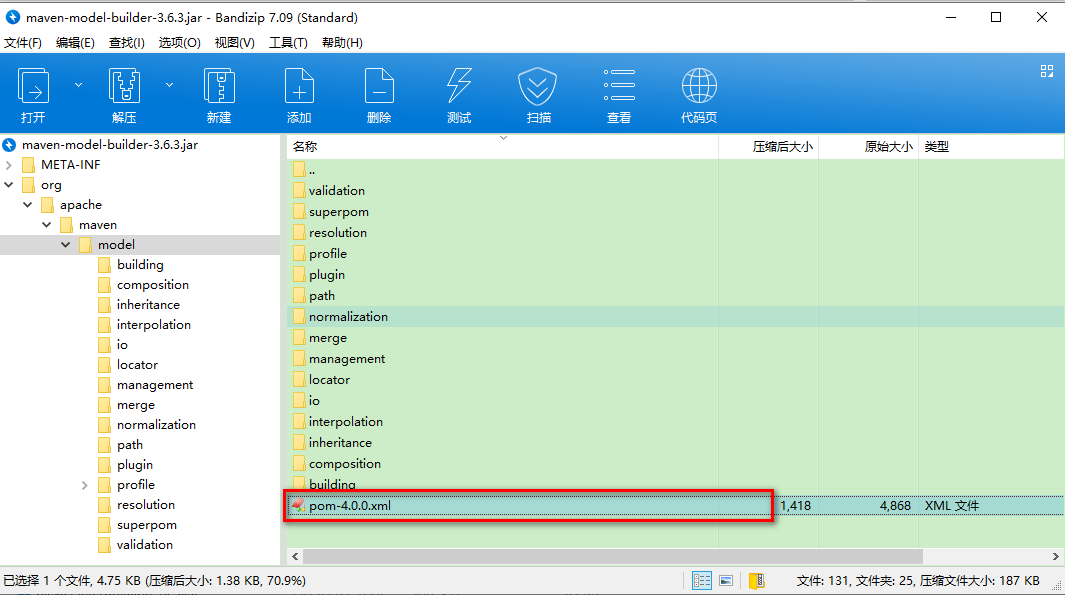
# 远程仓库配置

## 一、默认远程仓库配置：

之前介绍过Maven有一个Super POM，里面有一个Maven默认的配置，其中Maven默认有一个远程仓库，该仓库默认为Maven中央仓库，该配置在Jar包中：



打开此Jar包，可以看到一个pom-4.0.0.xml文件



打开此pom-4.0.0.xml文件，可以看到如下配置：

|  |
| --- |
|  |

配置的中央仓库路径为：**https://repo.maven.apache.org/maven2**

## 二、多仓库配置：

我们可以配置多个远程仓库，可以在settings.xml文件中配置，也可以在pom.xml文件中配置，而settings.xml又分为全局配置和用户配置，即有三个地方可以配置远程仓库，但归根节点还是分为settings.xml中配置和pom.xml中配置。

### 1、pom.xml中配置：

在pom.xml中配置属于局部配置，即每一个项目中都要有该配置。

我们可以在pom.xml文件中的<repositories>节点中配置<repository>节点，每一个<repository>节点代表一个远程仓库。

|  |
| --- |
| <**repositories**>  *<!-- 配置值Aliyun仓库 -->* <**repository**>  <**id**>nexus-aliyun</**id**>  <**name**>nexus-aliyun</**name**>  <**url**>http://maven.aliyun.com/nexus/content/groups/public/</**url**>  <**releases**>  <**enabled**>true</**enabled**>  </**releases**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**repository**>  *<!-- 配置JBoss仓库 -->* <**repository**>  <**id**>jboss-repo</**id**>  <**name**>jboss-rep</**name**>  <**url**>  <http://repository.jboss.org/nexus/content/groups/public-jboss/>  </**url**>  <**releases**>  <**enabled**>true</**enabled**>  </**releases**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**repository**> </**repositories**> |

其中我们在这里配置了两个远程仓库，Aliyun远程仓库和JBoss远程仓库。

但实际上，Maven在构建的时候，会从三个远程库中下载Jar包：Aliyun、JBoss、中央仓库。

### 2、settings.xml文件中配置：

settings.xml文件中配置，需要配置在<profiles>下的<profile>节点中，每一个<profile>节点代表一个项目环境。

|  |
| --- |
| <profile>  <!-- 生产环境配置,id必须唯一 -->  <id>env\_prod</id>  <repositories>  <repository>  <!-- 自己命名的仓库id,id必须唯一 -->  <id>myRepository1\_1</id>  <!-- 仓库的url地址 -->  <url>  <http://maven.aliyun.com/nexus/content/groups/public/>  </url>  <releases>  <enabled>true</enabled>  </releases>  <snapshots>  <enabled>true</enabled>  <updatePolicy>always</updatePolicy>  </snapshots>  </repository>  </repositories>  </profile> |

然后还要设置激活哪一个<profile>，需要在settings.xml文件中的<activeProfiles>中配置

|  |
| --- |
| <activeProfiles>  <!-- 激活env\_prod -->  <activeProfile>env\_prod </activeProfile>  </activeProfiles> |

## 三、多仓库配置，Jar包从哪里下载呢？

我们可以在pom.xml或者settings.xml文件中配置多个远程仓库，那么构件从哪个仓库下载的呢？

其实，Maven会在按照我们配置的远程仓库顺序，依次在远程仓库里面下载我们需要的构件，如果在当前远程仓库中下载不到，则会在下一个远程仓库中下载，如果在所有配置的远程仓库中都没有下载到，那么Maven就会报错。

如下，有一个远程仓库配置，我们使用在pom.xml文件中配置的方式：

|  |
| --- |
| <**repositories**>  *<!-- 配置值Aliyun仓库 -->* <**repository**>  <**id**>nexus-aliyun</**id**>  <**name**>nexus-aliyun</**name**>  <**url**>http://maven.aliyun.com/nexus/content/groups/public/</**url**>  <**releases**>  <**enabled**>true</**enabled**>  </**releases**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**repository**>  *<!-- 配置JBoss仓库 -->* <**repository**>  <**id**>jboss-repo</**id**>  <**name**>jboss-rep</**name**>  <**url**>  <http://repository.jboss.org/nexus/content/groups/public-jboss/>  </**url**>  <**releases**>  <**enabled**>true</**enabled**>  </**releases**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**repository**> </**repositories**> |

我们虽然显示的只声明了两个远程仓库，但是Maven在构建的时候，会从三个远程仓库中下载构件：Aliyun仓库、JBoss仓库、中央仓库。

下载顺序：

1. 首先在Aliyun仓库中查找，如果能够查找到则下载该构件，不在下一个仓库中查找；如果下载不到，则在下一个仓库中查找；
2. 在JBoss仓库中查找，如果能够查找到则下载该构件，不在下一个仓库中查找；如果下载不到，则在下一个仓库中查找；
3. 在中央仓库中查找，如果能够查找到则下载该构件，不在下一个仓库中查找；如果下载不到，则报错。

怎么证明我们的说法呢？我们可以故意将我们配置的Aliyun仓库和JBoss仓库配置一

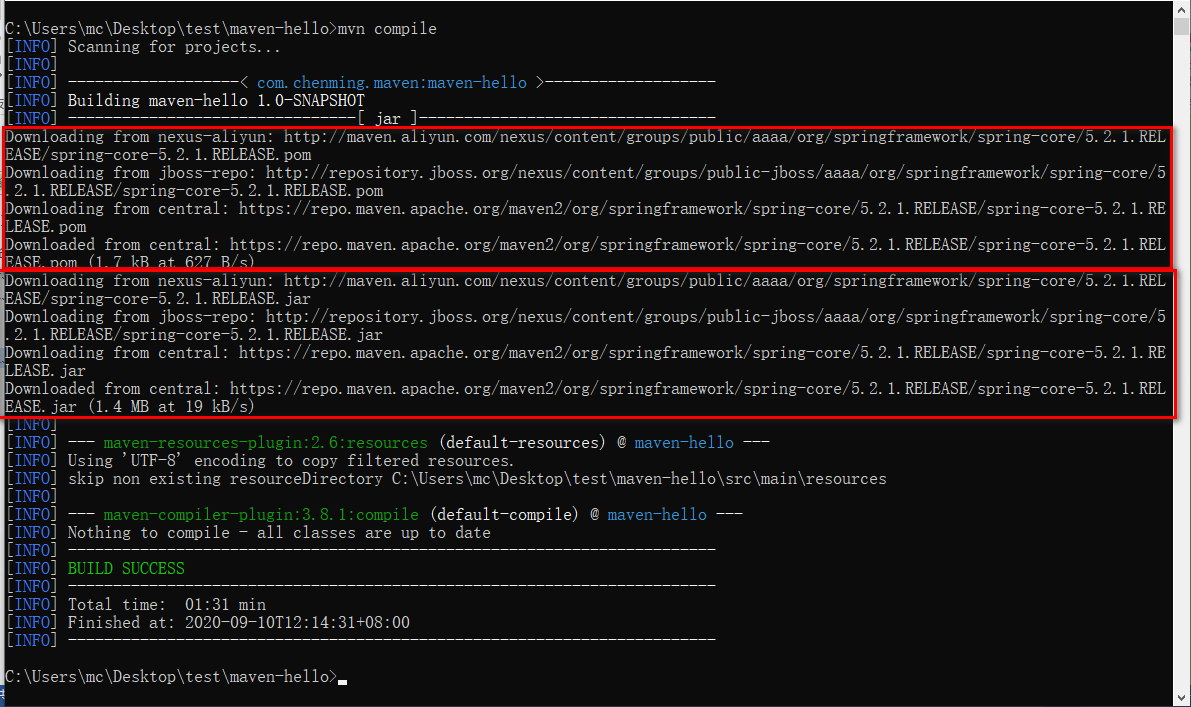
个错误的路径，重新让Maven下载构件，就可以看到整个下载过程了。

这里我们使用spring-core的5.2.1.RELEASE的Jar作为实验（任何一个Jar包都可以）：

【远程仓库错误配置】【注：仓库配置错误的时候，才更容易出现现象】

|  |
| --- |
| <**repositories**>  *<!-- 配置值Aliyun仓库 -->* <**repository**>  <**id**>nexus-aliyun</**id**>  <**name**>nexus-aliyun</**name**>  <**url**>**http://maven.aliyun.com/nexus/content/groups/public/aaaa**</**url**>  <**releases**>  <**enabled**>true</**enabled**>  </**releases**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**repository**>  *<!-- 配置JBoss仓库 -->* <**repository**>  <**id**>jboss-repo</**id**>  <**name**>jboss-rep</**name**>  <**url**>  [**http://repository.jboss.org/nexus/content/groups/public-jboss/**](http://repository.jboss.org/nexus/content/groups/public-jboss/)**aaaa**  </**url**>  <**releases**>  <**enabled**>true</**enabled**>  </**releases**>  <**snapshots**>  <**enabled**>false</**enabled**>  </**snapshots**>  </**repository**> </**repositories**> |

【下载Jar包】



从图中，我们可以看到Maven下载了spring-core-5.2.1.RELEASE.pom和spring-core-5.2.1.RELEASE两个构件，而这每一个构件都是先从Aliyun查找，查找不到再在JBoss中查找，最后才在中央仓库中找到并下载的。

至此，Maven下载构件的顺序已经清楚了。

【注】在settings.xml文件中配置的远程仓库下载构件的原理与pom.xml中配置的一样，这里就不重复了。