

Maven 仓库

在 Maven 的术语中，仓库是一个位置（place）。

Maven 仓库是项目中依赖的第三方库，这个库所在的位置叫做仓库。

在 Maven 中，任何一个依赖、插件或者项目构建的输出，都可以称之为构件。

Maven 仓库能帮助我们管理构件（主要是JAR），它就是放置所有JAR文件（WAR，ZIP，POM等等）的地方。

Maven 仓库有三种类型：

- 本地（local）
- 中央（central）
- 远程（remote）

本地仓库

Maven 的本地仓库，在安装 Maven 后并不会创建，它是在第一次执行 maven 命令的时候才被创建。

运行 Maven 的时候，Maven 所需要的任何构件都是直接从本地仓库获取的。如果本地仓库没有，它会首先尝试从远程仓库下载构件至本地仓库，然后再使用本地仓库的构件。

默认情况下，不管Linux还是 Windows，每个用户在自己的用户目录下都有一个路径名为 .m2/respository/ 的仓库目录。

Maven 本地仓库默认被创建在 %USER_HOME% 目录下。要修改默认位置，在 %M2_HOME%\conf 目录中的 Maven 的 settings.xml 文件中定义另一个路径。

```
<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">
<localRepository>C:/MyLocalRepository</localRepository>
</settings>
```

当你运行 Maven 命令，Maven 将下载依赖的文件到你指定的路径中。

中央仓库

Maven 中央仓库是由 Maven 社区提供的仓库，其中包含了大量常用的库。

中央仓库包含了绝大多数流行的开源Java构件，以及源码、作者信息、SCM、信息、许可证信息等。一般来说，简单的Java项目依赖的构件都可以在这里下载到。

中央仓库的关键概念：

- 这个仓库由 Maven 社区管理。
- 不需要配置。
- 需要通过网络才能访问。

要浏览中央仓库的内容，maven 社区提供了一个 URL：<http://search.maven.org/#browse>。使用这个仓库，开发人员可以搜索所有可以获取的代码库。

远程仓库

如果 Maven 在中央仓库中也找不到依赖的文件，它会停止构建过程并输出错误信息到控制台。为避免这种情况，Maven 提供了远程仓库的概念，它是开发人员自己定制仓库，包含了所需要的代码库或者其他工程中用到的 jar 文件。

举例说明，使用下面的 pom.xml，Maven 将从远程仓库中下载该 pom.xml 中声明的所依赖的（在中央仓库中获取不到的）文件。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.companyname.projectgroup</groupId>
  <artifactId>project</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
      <groupId>com.companyname.common-lib</groupId>
      <artifactId>common-lib</artifactId>
      <version>1.0.0</version>
    </dependency>
  </dependencies>
  <repositories>
    <repository>
      <id>companyname.lib1</id>
      <url>http://download.companyname.org/maven2/lib1</url>
    </repository>
    <repository>
      <id>companyname.lib2</id>
      <url>http://download.companyname.org/maven2/lib2</url>
    </repository>
  </repositories>
</project>
```

Maven 依赖搜索顺序

当我们执行 Maven 构建命令时，Maven 开始按照以下顺序查找依赖的库：

- **步骤 1** - 在本地仓库中搜索，如果找不到，执行步骤 2，如果找到了则执行其他操作。
- **步骤 2** - 在中央仓库中搜索，如果找不到，并且有一个或多个远程仓库已经设置，则执行步骤 4，如果找到了则下载到本地仓库中以备将来引用。
- **步骤 3** - 如果远程仓库没有被设置，Maven 将简单的停滞处理并抛出错误（无法找到依赖的文件）。
- **步骤 4** - 在一个或多个远程仓库中搜索依赖的文件，如果找到则下载到本地仓库以备将来引用，否则 Maven 将停止处理并抛出错误（无法找到依赖的文件）。

Maven 阿里云(Aliyun)仓库

Maven 仓库默认在国外，国内使用难免很慢，我们可以更换为阿里云的仓库。

第一步:修改 maven 根目录下的 conf 文件夹中的 setting.xml 文件，在 mirrors 节点上，添加内容如下：

```
<mirrors>
<mirror>
<id>alimaven</id>
<name>aliyun maven</name>
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>
<mirrorOf>central</mirrorOf>
</mirror>
</mirrors>
```

```
<!-- mirror
| Specifies a repository mirror site to use instead of a given repository. The
| this mirror serves has an ID that matches the mirrorOf element of this mirror
| for inheritance and direct lookup purposes, and must be unique across the
|
<mirror>
  <id>mirrorId</id>
  <mirrorOf>repositoryId</mirrorOf>
  <name>Human Readable Name for this Mirror.</name>
  <url>http://my.repository.com/repo/path</url>
</mirror>
-->
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
</mirrors>
```

第二步: pom.xml文件里添加：

```
<repositories>
<repository>
<id>alimaven</id>
<name>aliyun maven</name>
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>
<releases>
<enabled>true</enabled>
</releases>
<snapshots>
<enabled>false</enabled>
</snapshots>
</repository>
</repositories>
```

✎ 点我分享笔记

