# Maven

## Maven简介

### 1. 构建构建&依赖管理工具&项目信息管理工具

作为Apache组织中的一个颇为成功的开源项目，Maven主要服务于基于Java平台的项目构建、依赖管理和信息管理。

Maven是一个异常强大的构建工具，能够帮助门额自动化构建过程，从清理、编译、测试到生成报告，再到打包和部署。

Maven是跨平台的，在Windows、Linux、Mac上都可以使用同样的命令。

Maven最大化地消除了构建的重复，抽象了构建声明周期。Maven不仅是构建工具，还是一个依赖管理工具和项目信息管理工具。

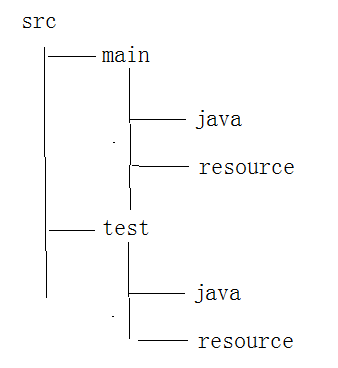
### pom.xml

POM(Project Object Model 项目对象模型)是maven的核心，定义了项目的基本信息，用于描述项目如何构建，声明项目依赖等。

* **modelVersion**：指定了当前POM模型的版本，对于Maven2和Maven3来说，它只能是4.0.0
* **groupId**:项目所属的组
* **artifactId**:当前Maven项目在组中唯一的ID
* **version**：项目当前的版本
* **scope**: 依赖范围，默认值compile，对主代码和测试代码都有效
* **name**：声明一个对用户更加友好的项目名称
* **packaging**: 默认打包类型为jar

### Maven约定--约定优于配置

* 在项目的根目录放置pom.xml
* 在src/main/java放置项目主代码
* 在src/test/java中放置项目测试代码



## 坐标和依赖

世界上任何一个构件都可以使用Maven坐标唯一标识，Maven坐标的元素包括groupId、artifactId、version、packaging、classifier。

### 坐标详解

* groupId：当前Maven项目隶属的实际项目，通常域名反向。
* artifactId：推荐使用实际项目名称作为artifactId
* version：Maven项目当前所处的版本
* packaging：Maven项目的打包方式，默认值jar.
* classifier：用来帮助定义构建输出的一些附属构件，与主构件对应。附属构件不是项目直接默认生成的，而是由附加的插件帮助生成，因此不能直接定义项目的classifier。

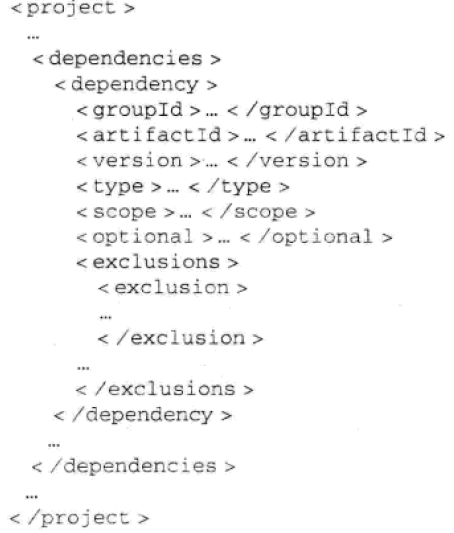
groupId、artifactId、version是必须定义 的，packaging是可选的(默认值jar)，而classifier是不能直接定义的.

项目构件的文件名一般的规则为：artifactId-version [-classifier].packaging

### 依赖配置

完整的依赖声明包含以下元素：

* groupId、artifactId、version：依赖的基本坐标
* type: 依赖的类型，对应于项目坐标定义的packaging,不必声明，默认值为jar
* scope：依赖的范围
* optional: 标记依赖是否可选
* exclusions: 用来排除传递性依赖。



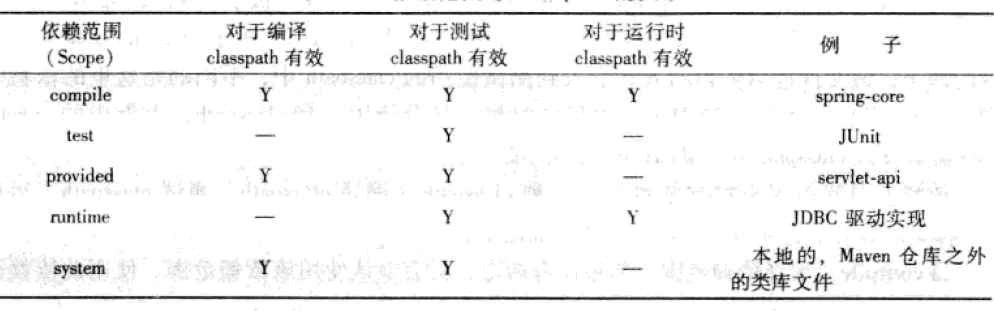
### 依赖范围

测试范围用元素scope表示。依赖范围是用来控制依赖于编译classpath、测试classpath、运行classpath的关系。

* compile: 编译依赖范围，默认的依赖范围，对编译、测试、运行三种classpath都有效。
* test: 测试依赖范围。只对测试classpath有效，在编译主代码或者运行项目的时候将无法使用此类依赖。
* provided: 已提供依赖范围，对于编译和测试有效，不会打包进发布包中。
* runtime: 对测试、运行classpath有效，但编译主代码时无效。
* system: 和provided一致。不从maven仓库获取该jar,而是通过systemPath指定该jar的路径

|  |
| --- |
| <dependency>  <groupId>javax.sql</groupId>  <artifactId>jdbc.stdext</artifactId>  <version>2.0</version>  <scope>system</scope>  <systemPath>${java.home}/lib/rt.jar</systemPath>  </dependency> |

* Import：导入依赖范围。



### 传递性依赖

Maven会解析各个直接依赖的POM，将那些必要的间接依赖，以传递性依赖的形式引入到当前的项目中。

①当第二直接依赖的范围是compile和runtime的时候，传递性依赖的范围与第一直接依赖的范围一致；

②当第二直接依赖的范围是test的时候，依赖不会得以传递；

③当第二直接依赖的范围是provided，且第一直接依赖范围也是provided的时候，传递性依赖的范围同样为provided。

### 依赖冲突&依赖调解

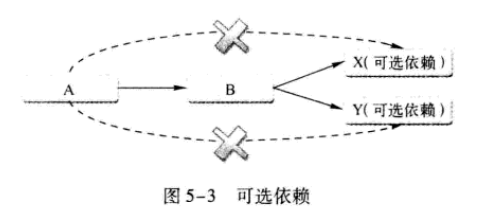
两条依赖路径上有某个依赖的两个版本，造成依赖冲突。

调解原则：

* 第一原则：路径最近者优先。
* 第二原则：第一声明者优先。顺序最靠前的那个依赖优胜。

### 可选依赖(optional)

可选依赖在传递性依赖中不被传递。Optional表示某个依赖为可选依赖。

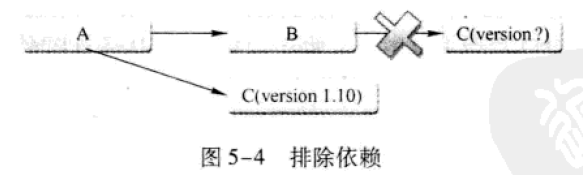


|  |
| --- |
| A->B、B->X(可选)、B->Y(可选)，则X、Y将不会对A有任何影响 |

### 最佳实践

排除依赖

使用exclusions元素声明排除依赖，声明exclusion的时候只需要groupId和artifactId，而不需要version元素(Maven解析后的依赖中，不可能出现groupId和artifactId相同两个依赖)



归类依赖

|  |
| --- |
| <properties>  <xxx.version>2.5.6</xxx.version>  </properties> |

首先使用properties定义Maven属性，Maven运行的时候会将所有${xx.version}替换为properties定义的属性值。

#### 优化依赖

查看已解析依赖：

1. mvn dependency:tree 树形形式
2. mvn dependency:list 列表形式

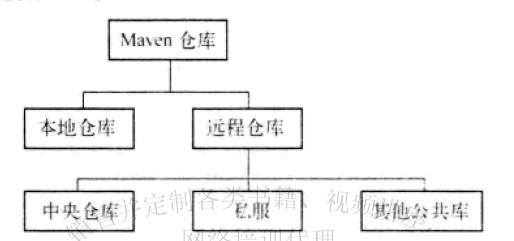
## 仓库

### 定义

仓库分为两类：本地仓库和远程仓库。

中央仓库是Maven核心自带的远程仓库，在默认配置下，当本地仓库没有Maven需要的构件的时候，就会尝试从中央仓库下载。

私服是另一种特殊的远程仓库，为了节省带宽和时间，应该在局域网内架设一个私有的仓库服务器，用其代理所有外部的远程仓库。



### 本地仓库

修改本地仓库地址：

|  |
| --- |
| <settings>  <localRepository>本地仓库地址</localRepository>  </settings> |

安装完Maven后，如果不执行Maven命令，本地仓库目录是不存在的，但用户输入第一条Maven命令后，Maven才会创建本地仓库。

### 远程仓库

#### 3.1 中央仓库

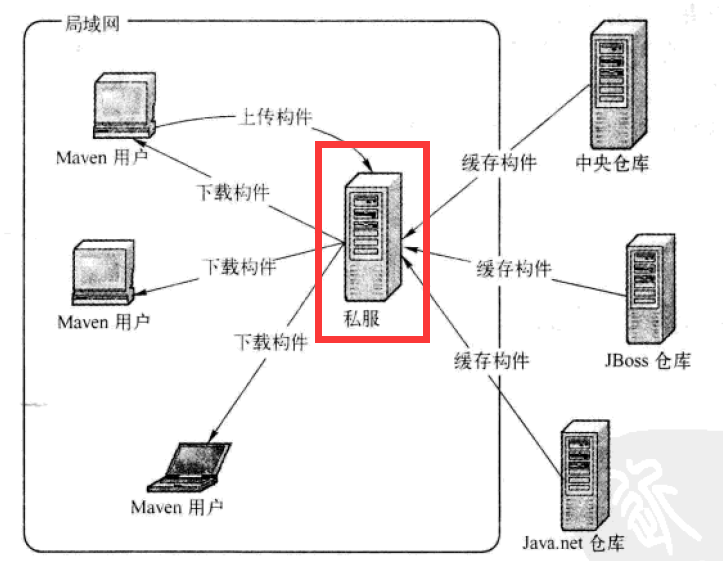
中央仓库是默认的远程仓库，Maven的安装文件中的超级POM自带了中央仓库的配置。

#### 3.2 私服

私服是一种特殊的远程仓库，它是架设在局域网内的仓库服务，私服代理广域网上的远程仓库，供局域网内的Maven用户使用。

当Maven需要下载构件的时候，它从私服请求，如果私服上不存在构件，则从外部的远程仓库下载，缓存在私服上之后，再为Maven的下载请求提供服务。

一些无法从远程仓库中下载到的构件也能从本地上传到私服上供Maven用户使用。



#### 3.3 远程仓库配置



(Maven只会从JBoss仓库下载发布版的构件，而不会下载快照版的构件)

任何一个仓库声明的id必须是唯一的。

**Layout仓库布局**

* default表示仓库的布局是Maven2及Maven3的默认布局
* Legacy表示仓库的布局是Maven1的布局

**updatePolicy配置从远程仓库检查更新的频率**

* daily：默认值，表示Maven每天检查一次。
* never: 从不检查更新。
* always: 每次构建都检查更新
* interval: X 每隔X分钟检查一次更新

通过maven-metadata-local.xml的时间戳与远程仓库进行比较，判断是否需要更新。

**checksumPolicy配置Maven检查校验和文件的策略**

Maven检查校验和文件失败时：

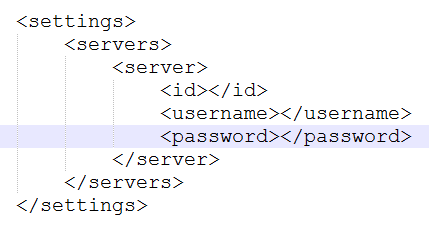
* warn：在构建时输出警告信息
* fail: 构建失败
* ignore: 使Maven完全忽略校验和错误

|  |
| --- |
| 安装nexus私服，修改sha1值，设置checksumPolicy，验证checksumPolicy工作原理。 |

#### 3.4 远程仓库认证

配置远程仓库信息是在POM文件中，而配置远程仓库认证信息必须配置在settings.xml文件中。

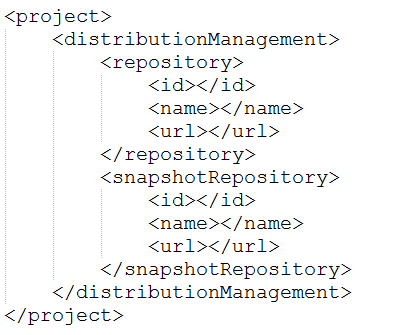
在settings.xml中配置仓库认证信息：



settings.xml中server元素的id必须与POM中需要认证的repository元素的id完全一致。

#### 3.5 部署远程仓库

在pom.xml文件中配置distributionManagement元素：



distributionManagement包含repository和snapshotRepository子元素，分别表示发布版本构建和快照版本构件的部署仓库。