**一.介绍**

---------

\*\*Maven\*\*是基于项目对象模型（POM），可以通过一小段描述信息来管理项目的构建、报告和文档的软件项目管理工具。

1. bin目录是包含mvn的运行脚本

2. boot目录包含一个类加载器的框架，maven使用它加载自己的类库

3. conf配置文件

4. lib包含maven运行时的依赖类库

**二.环境变量的配置**

---------

[点击下载](http://maven.apache.org/download.cgi "下载地址")<br>

\*\*maven\*\*环境配置，增加一个环境变量\*\*MAVEN\_HOME\*\*,值是maven的安装路径（`C:\Program Files\apache-maven-3.5.0-bin\apache-maven-3.5.0`）

修改path则是在path最后面添加`;%MAVEN\_HOME%\bin`。

### Maven的项目结构 ###

项目名

-src

-main

-java

-package

-test

-java

-package

-pom.xml

**三.常用命令**

------

mvn -v 查看maven版本

compile 编译

test 测试

package 打包

clean 删除target

install 安装jar包到本地仓库

### maven快速创建项目骨架目录 ###

#### 两种方式： ####

1. mvn archetype:generate 按照提示进行选择

2. mvn archetype:generate -DgroupId=com.imooc.maven -DartifactId=

maven-service -Dversion=1.0.0SNAPSHOT -Dpackage=com.imooc.maven.demo

1. -DgroupId=组织名，公司网址反写+项目名

2. -DartifactId=项目名+模块名

3. -Dversion=版本号

4. -Dpackage=代码所存在的包名

**四.Maven中的坐标和仓库**

--------------

\*\*构件坐标:\*\*

1. groupId:公司名字+项目名

2. artifactId：项目名+模块名

3. varsion:版本号

\*\*仓库：\*\*

1. 本地仓库

2. 远程仓库

3. 镜像仓库

五.更改仓库默认路径

------

已安装到本地仓库中的jar包位置：

C:\Users\用户\.m2\repository\com\tiakon\demo

安装路径conf文件夹下settings.xml文件

打开找到这段备注是的代码：

<!-- localRepository

| The path to the local repository maven will use to store artifacts.

| Default: ${user.home}/.m2/repository

<localRepository>/path/to/local/repo</localRepository>

-->

复制粘贴出来

<localRepository>/path/to/local/repo</localRepository>

\*\*将localRepository便签内的值替换成新路径即可。\*\*

**六.maven生命周期**

---------

完整的项目构建过程包括：

\*\*清理、编译、测试、打包、集成测试、验证、部署\*\*

\*\*maven三套独立的生命周期\*\*

clean 清理项目

1.pre-clean 执行清理前的工作

2.clean 清理上一次构建生成的所有文件

3.post-clean 执行清理后的文件

default 构建项目（最核心）

compile test package install

site 生成项目站点

1. pre-site 在生成项目站点前要完成的工作

2. site 生成项目的站点文档

3. post-site 在生成项目站点后要完成的工作

4. site-deploy 发布生成的站点到服务器上

**七.maven中pom.xml常见元素介绍**

---------------

<?xml version="1.0" encoding="UTF-8"?>

<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">

<!--指定了当前pom的版本-->

<modelVersion>4.0.0</modelVersion>

<groupId>com.tiakon.maven.demo</groupId>

<artifactId>HoictasStudio-MavenDemo01</artifactId>

<version>1.0-SNAPSHOT</version>

<!--

第一个0表示大版本号

第二个0表示分支版本号

第三个0表示小版本号

0.0.1

snapshot 快照

alpha 内部测试

beta 公测

Release 稳定

GA 正式发布

-->

<!--

打包方式:默认是jar,可选war、zip、pom

<packaging></packaging>

   -->

    <!--指定编码格式-->

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<!--

项目名

<name></name>

项目地址

<url></url>

项目描述

<description></description>

开发人员列表

<developers></developers>

许可证信息

<licenses></licenses>

组织信息

<organization></organization>

-->

<!--依赖列表-->

<dependencies>

<!--依赖项-->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>RELEASE</version>

<!--<type></type>-->

<!--依赖范围-->

<!--<scope></scope>-->

<!--设置依赖是否可选（默认）false-->

<!--<optional></optional>-->

<!--排斥依赖传递列表-->

<!--

<exclusions>

<exclusion>

</exclusion>

</exclusions>

-->

</dependency>

</dependencies>

<!--依赖的管理，作用主要定义在父模块中，对子模块进行管理-->

<!--

<dependencyManagement>

<dependencies>

</dependencies>

</dependencyManagement>

-->

<!--对构件的行为提供相应的支持-->

<build>

<!--插件列表-->

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-source-plugin</artifactId>

<version>3.0.1</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>

jar-no-fork

</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

<!--通常用于子模块对父模块pom的继承-->

<!--<parent></parent>-->

<!--用来聚合运行Maven项目，指定多个模块一起编译-->

<!--

<modules>

<module></module>

</modules>

-->

</project>

**八.Maven的依赖范围**

----------

三种classpath

1. 编译

2. 测试

3. 运行

\*\*maven提供了6种可选依赖范围:\*\*

1. compile: 默认范围，编译测试运行都有效。

2. provided: 在编译和测试时有效。（比如说在开发JavaEE时，你在本地运行的servlet，

是需要调用已添加到项目中的servlet-api.jar这个jar包的。这个过程就包含了编译【就是

把Java文件编译成class文件的过程中也要调用】和测试【测试就是在本地运行】，那么他说

的运行是指，整个项目已开发完成，编译、测试通过后，将class文件或包含有class文件的war包

发布到服务器上的Tomcat中运行，这时启动项目，就可以直接调Tomcat中的servlet-api.jar，

不必再将自己的jar包添加到项目中去。也就是说当你选择provided时，项目发布时Maven不会将

你添加的jar包，加入到项目中。）

3. runtime: 在测试和运行时有效。(典型例子：JDBC驱动的实现。)

4. test: 只在测试是有效。

5. system: 类似provided，与本机系统相关联，可移植性差。

6. import: 导入范围，他只是用在dependencyManagement中，表示从其他的pom中导

入dependecy的配置。（以下引用官网案例并不难理解。）

Importing Dependencies

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>maven</groupId>

<artifactId>B</artifactId>

<packaging>pom</packaging>

<name>B</name>

<version>1.0</version>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>maven</groupId>

<artifactId>A</artifactId>

<version>1.0</version>

<type>pom</type>

<scope>import</scope>

</dependency>

<dependency>

<groupId>test</groupId>

<artifactId>d</artifactId>

<version>1.0</version>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

<groupId>test</groupId>

<artifactId>a</artifactId>

<version>1.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>test</groupId>

<artifactId>c</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

</project>

Assuming A is the pom defined in the preceding example, the end result would be the same. \*\*All of A's managed dependencies would be incorporated into B except for d since it is defined in this pom.\*\*

假设A是前面示例中定义的pom，那么最终结果将是相同的。\*\*所有管理的依赖项都将被合并到B中，除了在这个pom中定义的d之外。\*\*

**九.依赖冲突**

------

1.短路优先:

C->B->A->X1(jar)

C->B->X2(jar)

【C依赖B,B依赖A,A和B都包含同一个不同版本的Jar,则取B的依赖版本。（c的pom.xml中不必注明jar坐标）】

2.先声明先优先

如果路径相同长度相同，则谁先声明，先解析谁。

【C依赖A和B,A和B都包含同一个不同版本的Jar,谁依赖在前取谁的依赖版本。】

十.聚合与继承

-------

### 聚合 ###

<packaging>pom</packaging>

<modules>

<module>../HoictasStudio-MavenDemo01</module>

<module>../HoictasStudio-MavenDemo02</module>

<module>../HoictasStudio-MavenDemo03</module>

</modules>

假设在\*\*HoictasStudio-MavenParent\*\*模块中添如以上代码，输入`clean install`命令后，即可同时安装多个jar到本地仓库中

[INFO] HoictasStudio-MavenDemo01 .......................... SUCCESS [ 4.618 s]

[INFO] HoictasStudio-MavenDemo02 .......................... SUCCESS [ 0.828 s]

[INFO] HoictasStudio-MavenDemo03 .......................... SUCCESS [ 0.923 s]

[INFO] HoictasStudio-MavenParent .......................... SUCCESS [ 0.021 s]

### 继承 ###

\*\*根据官方文档说明继承会根据父模块与子模块的包含与否，对pom.xml的写法则有两种。\*\*

#### 第一种写法 ####

假设我们有两个模块，前一个叫 `com.mycompany.app:my-app:1`，后一个叫`com.mycompany.app:my-module:1`。

my-app的pom文件为：

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany.app</groupId>

<artifactId>my-app</artifactId>

<version>1</version>

</project>

my-module的pom文件为：

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany.app</groupId>

<artifactId>my-module</artifactId>

<version>1</version>

</project>

我们指定如下项目结构：

.

|-- my-module

| `-- pom.xml

`-- pom.xml

那么，我们需要`my-module`去继承`my-app`，则需要在`my-module`的pom文件中添加以下代码：

<project>

<parent>

<groupId>com.mycompany.app</groupId>

<artifactId>my-app</artifactId>

<version>1</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<groupId>com.mycompany.app</groupId>

<artifactId>my-module</artifactId>

<version>1</version>

</project>

#### 第二种写法 ####

However, that would work if the parent project was already installed inour local repository or was in that specific

directory structure (parent pom.xml is one directory higher than that of the module's pom.xml). But what if the parent

is not yet installed and if the directory structure is

.

|-- my-module

| `-- pom.xml

`-- parent

`-- pom.xml

上一段话摘自官网对继承的介绍，就是说如果你的父模块已在本地安装或者父模块不包含子模块，目录级别甚至是

比子模块的还要高，就在第一种写法上添加`<relativePath>`标签。

<project>

<parent>

<groupId>com.mycompany.app</groupId>

<artifactId>my-app</artifactId>

<version>1</version>

<relativePath>../parent/pom.xml</relativePath>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>my-module</artifactId>

</project>

笔者在看视频时就发现，当父模块与子模块处于同一级别时，在按照视频中的写法（第一种写法）test时就会报错，

而此时的情况是不包含子模块，所以应该在`<parent>`标签中添加`<relativePath>`标签即可测试通过。