POM 简介和使用

# 1.dependencies

依赖关系，依赖关系列表

<dependencies>

   <dependency>

     <groupId>junit</groupId>

     <artifactId>junit</artifactId>

     <version>4.0</version>

     <scope>test</scope>

   </dependency>

   …

 </dependencies>

* groupid,artifactid,version，引用包的组织（公司），项目构建标识符及版本
* scope，该依赖所使用的范围

compile：适用于所有的阶段，随着项目一起发布

provider：compile，期望容器或使用者会提供这个依赖

runtime：只在运行时使用如JDBC驱动，适用运行和测试阶段，编译时不会使用该包

test：只在测试时使用，用于编译和运行测试代码，不会随项目发布

system：类似provided，需要显示提供包含依赖的Jar，maven不会在repository中查找， 将会在systemPath中查找，并会随项目发布

import：仅使用在<dependencyManagement>模块，标识一些特定的POM中的依赖被 dependencyManagment中的模块替换掉。

Transitive Dependencies（传递依赖）

项目中依赖的jar包，可能在项目中依赖其他jar包，使用maven传递机制可以很好的解决依赖的jar包，不需要手动下载相关依赖。

为了解决，cyclic依赖及层次过多的问题，有以下规则：

* Dependency mediation，当遇到多版本的依赖时，决定使用哪个版本。目前仅支持最近原则。例如，依赖树：A->B->C->D 2.0及A->E->D 1.0，那么会使用D 1.0，因此E的依赖树比较短（可以强制使用D 2.0）
* Dependency managemen，当遇到传递依赖时，允许用户定义使用的版本
* Excluded dependencies，如果project x依赖project y，y依赖z，x可以显式的exclude z（取出z的依赖关系，格式是添加元素exclustion）
* Optional dependencies，如果Y依赖Z，项目Y的所有者可以将Z标记为可选依赖，使用optional element。当X仅依赖Y，而不依赖Y的可选Z，那么X可以显式的添加对Z的依赖。

# 2.build，构建设置

在Pom中,build element用于声明项目目录结构和管理插件，所处的位置如下：

<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

https://maven.apache.org/xsd/maven-4.0.0.xsd">

...

<!-- "Project Build" contains more elements than just the BaseBuild set -->

<build>...</build>

<profiles>

<profile>

<!-- "Profile Build" contains a subset of "Project Build"s elements -->

<build>...</build>

</profile>

</profiles>

</project>

build element基本集合如下所示：

<build>

<defaultGoal>install</defaultGoal>

<directory>${basedir}/target</directory>

<finalName>${artifactId}-${version}</finalName>

<filters>

<filter>filters/filter1.properties</filter>

</filters>

...

</build>

* defaultGoal，

模块1，resources module

<resources>

<resource>

<directory>src/main/resources</directory>

<filtering>true</filtering>

</resource>

</resources>

# 3.Plugin

Maven是一个插件框架，它的核心并不执行任何具体的构建任务，这些任务都交给插件来完成，例如编译源代码是由maven-compiler-plugin完成，进一步说，每个任务对应一个插件目标（goal）。下面介绍几种常用的插件：

3.1

# 4.Profiles

参考链接：

Introduction to the Dependency Mechanism:

https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html

build setting

https://maven.apache.org/pom.html#Build\_Settings