# 构建工具

* [构建工具](#构建工具)
  + [一、构建工具的作用](#一构建工具的作用)
  + [二、Java 主流构建工具](#二java-主流构建工具)
  + [三、Maven](#三maven)
  + [参考资料](#参考资料)

## 一、构建工具的作用

构建一个项目通常包含了依赖管理、测试、编译、打包、发布等流程，构建工具可以自动化进行这些操作，从而为我们减少这些繁琐的工作。

其中构建工具提供的依赖管理能够可以自动处理依赖关系。例如一个项目需要用到依赖 A，A 又依赖于 B，那么构建工具就能帮我们导入 B，而不需要我们手动去寻找并导入。

在 Java 项目中，打包流程通常是将项目打包成 Jar 包。在没有构建工具的情况下，我们需要使用命令行工具或者 IDE 手动打包。而发布流程通常是将 Jar 包上传到服务器上。

## 二、Java 主流构建工具

Ant 具有编译、测试和打包功能，其后出现的 Maven 在 Ant 的功能基础上又新增了依赖管理功能，而最新的 Gradle 又在 Maven 的功能基础上新增了对 Groovy 语言的支持。

Gradle 和 Maven 的区别是，它使用 Groovy 这种特定领域语言（DSL）来管理构建脚本，而不再使用 XML 这种标记性语言。因为项目如果庞大的话，XML 很容易就变得臃肿。

例如要在项目中引入 Junit，Maven 的代码如下：

<?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">  
 <modelVersion>4.0.0</modelVersion>  
   
 <groupId>jizg.study.maven.hello</groupId>  
 <artifactId>hello-first</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.10</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
</project>

而 Gradle 只需要几行代码：

dependencies {  
 testCompile "junit:junit:4.10"  
}

## 三、Maven

### 概述

提供了项目对象模型（POM）文件来管理项目的构建。

### 仓库

仓库的搜索顺序为：本地仓库、中央仓库、远程仓库。

* 本地仓库用来存储项目的依赖库；
* 中央仓库是下载依赖库的默认位置；
* 远程仓库，因为并非所有的依赖库都在中央仓库，或者中央仓库访问速度很慢，远程仓库是中央仓库的补充。

### POM

POM 代表项目对象模型，它是一个 XML 文件，保存在项目根目录的 pom.xml 文件中。

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 <scope>test</scope>  
</dependency>

[groupId, artifactId, version, packaging, classifier] 称为一个项目的坐标，其中 groupId、artifactId、version 必须定义，packaging 可选（默认为 Jar），classifier 不能直接定义的，需要结合插件使用。

* groupId：项目组 Id，必须全球唯一；
* artifactId：项目 Id，即项目名；
* version：项目版本；
* packaging：项目打包方式。

### 依赖原则

#### 1. 依赖路径最短优先原则

A -> B -> C -> X(1.0)  
A -> D -> X(2.0)

由于 X(2.0) 路径最短，所以使用 X(2.0)。

#### 2. 声明顺序优先原则

A -> B -> X(1.0)  
A -> C -> X(2.0)

在 POM 中最先声明的优先，上面的两个依赖如果先声明 B，那么最后使用 X(1.0)。

#### 3. 覆写优先原则

子 POM 内声明的依赖优先于父 POM 中声明的依赖。

### 解决依赖冲突

找到 Maven 加载的 Jar 包版本，使用 mvn dependency:tree 查看依赖树，根据依赖原则来调整依赖在 POM 文件的声明顺序。

## 参考资料

* [POM Reference](http://maven.apache.org/pom.html#Dependency_Version_Requirement_Specification)
* [What is a build tool?](https://stackoverflow.com/questions/7249871/what-is-a-build-tool)
* [Java Build Tools Comparisons: Ant vs Maven vs Gradle](https://programmingmitra.blogspot.com/2016/05/java-build-tools-comparisons-ant-vs.html)
* [maven 2 gradle](http://sagioto.github.io/maven2gradle/)
* [新一代构建工具 gradle](https://www.imooc.com/learn/833)