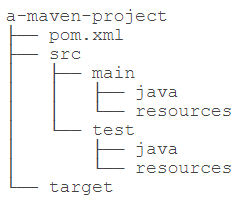
# Maven

## 理论介绍

1. Maven是一个Java项目管理和构建工具，它可以定义项目结构、项目依赖，并使用统一的方式进行自动化构建
2. Maven主要包括三部分
   1. 标准化的项目结构
   2. 标准化的构建流程
   3. 依赖管理机制
3. POM( Project Object Model，项目对象模型 ) 是 Maven 工程的基本工作单元，是一个XML文件，包含了项目的基本信息，用于描述项目如何构建，声明项目依赖，等等

## 标准项目结构



1. pom.xml：项目描述文件、第三方依赖信息
2. src/main/java：Java源码
3. src/main/resources：资源文件、配置文件
4. src/test/java：测试源码
5. src/test/resources：测试资源文件
6. target：编译、打包生成的文件

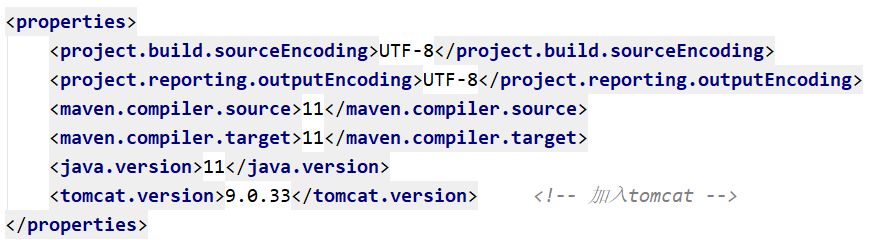
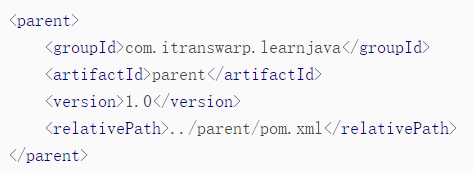
## pom.xml常见元素

### XML属性

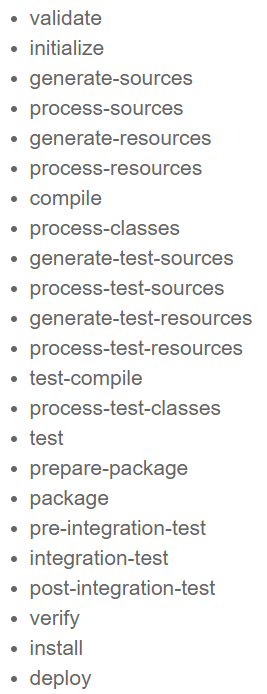


1. xmlns：定义该元素内子元素所属的默认命名空间
2. xmlns:xsi：定义以xsi为前缀的元素所属的命名空间
3. xsi:schemaLocation=”key value”
   1. 声明使用value对应的xsd文件中的定义去检测key对应的xml文件是否符合格式语法要求

### 项目基本属性

1. groupId：项目组织的唯一标识
2. artifactId：项目的唯一标识
3. version：项目版本
4. packaging：打包类型
   1. jar(一般java项目)、war(web项目) 、ear、pom
5. modelVersion：pom文件遵循的POM模型版本
6. properties：变量定义，可以在文件内其他位置使用 ${变量名} 来调用
   1. 
   2. 
7. dependencies：本项目所需要使用的一组第三方依赖库
   1. 
   2. 每一个依赖都是一个dependency，通过groupId、artifactId、version唯一指定
   3. scope 字段定义项目与依赖库之间的关系
      1. compile：编译时需要用到该jar包（默认）
      2. test：编译Test时需要用到该jar包
      3. runtime：编译时不需要，但运行时需要用到
      4. provided：编译时需要用到，但运行时由JDK或某个服务器提供
   4. 加载依赖库时，依赖库所需的其他依赖库也会自动加载
8. Parent：继承父类中定义的依赖
   1. 
   2. relativePath：指明父类pom.xml文件的相对路径
9. plugins：构建本项目所需插件，与依赖用法一致
   1. 依赖是项目编译运行都需要的代码，会打包进项目
   2. 插件则是辅助项目开发、编译、打包的工具，一般不会被打包进项目
10. exclusions：引用别的项目依赖时，可以指定该项目的某些jar包不加入本项目的依赖，多用于防止版本冲突问题

## 构建流程

1. 基本概念
   1. Lifecycle：生命周期，由一系列阶段(phase)组成，常见的生命周期有default和clean
   2. Phase：生命周期中的某一个阶段，包含多个绑定的goal
   3. Goal：绑定到phase上的方法，定义了需要具体执行的操作，是最小的任务单元
   4. 每一个phase对应一个插件，mvn通过调用相应插件完成phase的处理任务，phase对应的插件也可以自定义。
2. Default生命周期
   1. 
3. 命令行，mvn会自动查找phase对应的生命周期
   1. mvn package：执行default生命周期，它会从开始一直运行到package这个phase为止
   2. mvn clean package：执行clean生命周期到阶段clean，再执行default生命周期到阶段package
   3. mvn tomcat:run：执行tomcat这个阶段的goal run

## 常见命令行操作

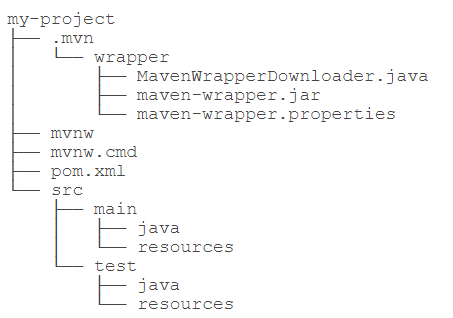
1. 执行mvn命令，需要进入到pom.xml文件所在路径

# 引入Maven管理

## IDEA

1. 选中现有项目，右键，选中Add Framwork Support
2. 可以选择多种框架支持，包括Maven

## Mvnw

1. 一般整个系统中的所有项目共用一个版本的Maven进行项目管理，如果有个别项目需要使用特殊版本的Maven进行单独管理，就需要安装Maven Warpper，负责该项目的管理，而对其他项目不造成影响。
2. 安装时进入该项目的pom.xml文件所在路径，执行命令行
   1. mvn -N io.takari:maven:**wrapper\_version**:wrapper -Dmaven=**maven\_vresion**
   2. 可以指定Maven Warpper的版本和特殊Maven的版本
3. 新的目录结构如下
   1. 
4. 然后就可以使用mvnw命令替代mvn命令调用特殊版本的Maven进行项目管理