Maven学习

# Maven是什么

1. Apache基金会出品
2. 开源
3. Java写的

通常把Maven理解成一个项目构建和依赖管理工具

# 为什么是Maven

1. 约定（惯例）优先原则——默认限定了目录结构
2. 三方依赖管理——解决依赖维护
3. 提供了一直的项目构建管理方式——减少了构建工作量
4. 插件式的架构，大量可重用插件
5. 很方便地集成IDE
6. 开源项目都使用Maven

# Maven安装

1. 依赖JDK，保证Java已经安装
2. 配置环境变量
   1. M2\_HOME=/path/to/maven/root
   2. PATH=$PATH:$M2\_HOME/bin

# pom.xml

1. groupId，组织
2. artifactId，项目标识符
3. version，版本
4. packaging，类型，war
5. dependencies，依赖的项目

# 基本命令

1. mvn archetype:generate——使用模板生成项目
2. mvn compile——编译源代码
3. mvn test——单元测试
4. mvn package——打包war
5. mvn deploy——部署
6. mvn site——项目相关站点，在线文档
7. mvn clean——清理
8. mvn install——安装到本地仓库

# Tomcat插件

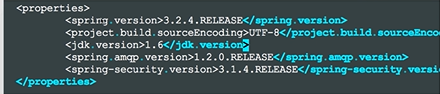
1. Tomcat官方提供的一个Maven插件——方便开发调试
2. 详细帮助
   1. mvn help:describe -Dplugin=tomcat7
3. 常用命令
   1. mvn tomcat7:run
   2. mvn tomcat:deploy
   3. mvn tomcat:undeploy

# Maven核心概念

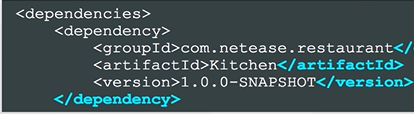
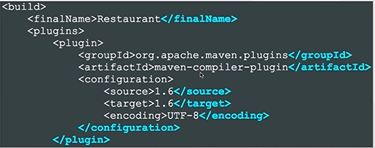
约定（惯例）优先于配置

1. 标准的目录结构
   1. \*.java放到src/main/java目录下
   2. \*.class在target目录下
   3. 配置及静态文件
   4. ……
2. 项目描述符，pom.xml

## POM

1. Xml文件，pom.xml
2. Project⬄artifact
3. 项目类型
   1. Pom，jar，war
4. 坐标
   1. groupId，artifactId，version
5. 属性，${property}

这里写好，其他地方直接引用

1. 依赖
2. 构建配置



1. 多项目，继承
2. 项目总体信息

## 问题

1. 多个子项目重复
2. 修改繁琐
3. 依赖版本不一致
4. 配置管理混乱

借助面向对象的思想

## 继承

1. 继承或者覆盖，override
2. 可继承项
   1. 坐标属性，如groupId
   2. 依赖配置
   3. 插件配置
   4. 一般性信息，如开发者信息

### Super POM

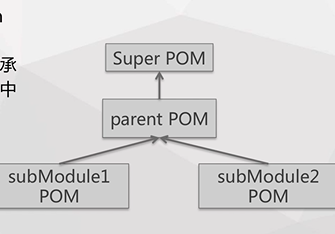
1. 所有的Maven项目的POM都继承Super POM
2. 是Maven的组成部分
3. Super POM定义了一组被所有项目共享的默认设置
   1. 默认文件位置
   2. 统一的插件配置
   3. 默认的中央仓库配置
   4. ……
4. mvn help:effective-pom 可以看到继承下来的最后的pom是什么样的
5. 继承配置
   1. 使用parent元素

## 多模块构建

1. Parent项目，类型是pom
   1. 定义modules



1. 子模块中配置parent，继承



1. 统一配置提到parent项目中即使用继承又使用组合

## 依赖配置

1. 依赖坐标
2. scope配置
   1. compile，默认，在编译时使用该依赖，在打包时也需要
   2. provided，仅在编译源代码时使用，打包时不包含进去
3. 用于继承的pom配置
   1. dependencyManagement，说明子pom可继承

## 仓库

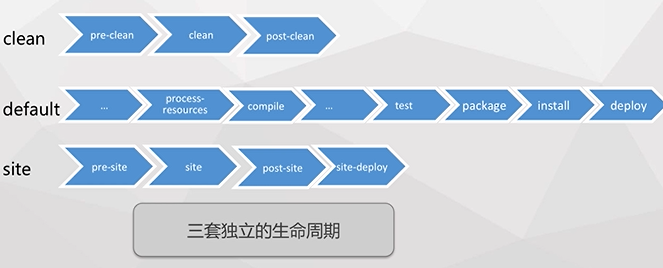
1. 用于统一存储所有Maven项目，共享的构件的空间
2. 仓库分类
   1. 本地仓库——Maven本地仓库的目录为~/.m2/repository
   2. 远程仓库

### 远程仓库

1. 本地仓库查找优先
2. 提供Maven项目访问并下载构件至本地仓库
3. 路径/<groupId>/<artifactId>/<version>/<artifactId>-<version>.<packaging>
4. 分类
   1. 中央仓库（默认）
   2. 其他公共仓库（镜像库）
   3. 私服（公司内部）
5. 配置
   1. 使用配置文件setting.xml
   2. 在pom.xml为依赖指定repositories

## 构建生命周期

1. 一个构建生命周期时一组精心组织的有序阶段
2. 每一个阶段执行预先定义的“动作”
   1. 编译
   2. 打包
   3. 部署
   4. ……
3. 这些“动作”会根据项目进行选择





### 默认生命周期

1. Validate
2. Process-resource
3. Compile
4. Test
5. Package——jar,war
6. Install——安装到本地仓库
7. Deploy——上传到远程仓库

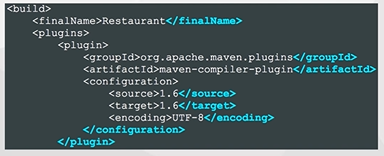
## 插件和目标

1. 插件（Plugin）式架构
2. 所有的插件本身也是一个Maven构件，又Maven仓库管理
3. 每一个插件提供多个目标（Goal）
4. 调用目标的格式
   1. mvn <Plugin>:<Goal>

### 插件目标绑定构建生命周期

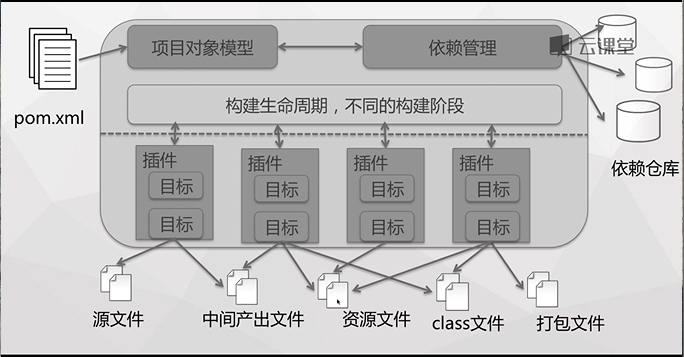
1. 生命周期阶段与目标绑定
2. 用户通过指定生命周期阶段就能够隐式的插件执行任务
3. 对于package阶段，会根据项目类型不同，绑定到不同的目标

### 插件配置

1. 插件配置元素
2. Configuration配置
   1. mvn help:describe -Dplugin=<plugin\_name> -Dgoal=<goal> -Ddetail=true 输出配置参数
3. 对于继承的父pom
   1. PluginManagement

### Plugins

1. mvn 内建插件
2. help
   1. mvn help:help -Ddetail=true
3. tomcat7
4. exec
5. 更多插件
6. 自定义插件



## 配置文件

配置mvn的配置文件是${M2\_HOME}/conf/settings.xml

# 总结

1. 默认标准的目录结构
2. Pom继承，Super POM，多项目管理
3. 依赖及依赖配置
4. 仓库的概念
5. 构建生命周期
6. 插件及目标