# Maven

## maven 创建项⽬

mvn archetype:generate

//使用generate生成maven骨架

-DgroupId=cn.codingxiaxw.helloword

//groupId，项目组织名称

-DartifactId=helloworld

//artifactId，具体项目名

-Dpackage=cn.codingxiaxw.helloword

//项目包名，可选填，与项目组名称保持一致

-Dversion=1.0-SNAPSHOT

//版本号，

-DarchetypeArtifactId=maven-archetype-quickstart

//项目结构，quickstart/webapp

## pom结构

不可缺少的四部分：

modelVersion：4.0.0 pom模版文件的版本

groupId 项目组织名

artifactId 具体项目名称

version 项目的版本号

## 配置maven私服

## plugin

maven本身不执行任务，它是一个插件框架，所有的任务都由插件执行。一个插件可能有多个插件目标，例如complie:complile指的是complie插件的complie目标。

插件工作有两种方式：

第一种是绑定到生命周期的phase上，例如执行clean生命周期执行clean插件。

第二种是通过命令调用插件。例如:clean:clean

除了默认的插件，其他的插件需要在pom中引入

<build>

<plugins>

<plugin>

<!-- groupId artifactId version 是maven的坐标 -->

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

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

<version>2.1.1</version>

<executions>

<execution>

<id>attach-sources</id>

<phase>verify</phase>

<goals>

<goal>jar-no-fork</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

## build 构建处理

build分为两种，project build全局配置 和porfile build

build 基本元素

<defaultGoal>install 默认mvn时执行的任务，

即mvn = mvn install

<directory> ${basedir}/target 生成结果的目录

<filename> 生成结果的名称

## package 打包

项目编译、测试、打包，包放到本地target中

## install 安装

项目编译、测试、打包，包放到本地target，再部署到本地仓库

## clean 清理

清理项目中target文件夹

## deploy 部署

编译、测试、打包、包放到本地target、部署到本地仓库和远程仓库

## dependencies tree 解决jar包冲突问题

引入大量依赖可能产生的问题：

1. 引入同一个jar包的不同版本，造成冲突、
2. 有些版本的jar包因为重名未被加载，但是却需要

使用dependency:tree可以查询jar的加载情况

compile 正常加载

omitted for duplicate 同一个版本的jar重复加载了

omitted for conflict with 1.8.3 和已经加载的1.8.3冲突了

解决方法：

<dependency>标签内有<exclusions>标签，可以排除本依赖中的部分jar包。