Maven

简介

Apache Maven是一个软件管理和综合工具，基于项目对象模型（POM）的概念

安装配置

下载好压缩包解压即可。确定安装了jdk（并配好JAVA_HOME环境变量），再将maven的bin配到环境变量中。都配置好，用mvn -v指令测试是否成功安装

Maven仓库

Maven中央仓库 http://repo1.maven.org/maven2/

本地仓库

远程仓库 自定义远程仓库

私服和镜像可理解为远程仓库

> 一般国内网络从Maven中央仓库下载包比较慢，推荐配置阿里镜像仓库，需在maven settings.xml中配置
>
> alimaven
> aliyun maven
> http://maven.aliyun.com/nexus/content/groups/public/
> central

> maven的配置文件settings.xml，存在多个。
>
> MAVEN_HOME/conf/settings.xml，是maven的全局配置文件；
>
> HOME_DIR/.m2/settings.xml，家目录maven仓库下的配置文件，是用户级配置文件
>
> 优先级： 用户级 > 全局

settings.xml

proxies 代理配置

localRepository 本地仓库路径配置

Maven构建生命周期

官网详情：http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

validate 验证，校验项目是否正确并且所有的必要信息可以完成项目的构建过程

compile 编译

test 运行测试文件（可跳过）

package 打包项目

verify 对集成测试的结果进行检查，以确保项目的质量

install 将打包好的包放到本地仓库

deploy 将打包好的包发送到远程仓库

> 上述构建生命周期指令从上往下顺序执行，即运行指令会先把前面的指令先运行一遍。
>
> 比如：mvn package，会先 validate compile test完成在package

Maven插件

clean 清除构建项目生成的文件（java工程删除target目录）

site 针对项目生成文档站点，类似于项目介绍文档之类的

不常见

prerequisites 描述了项目构建的前提条件

  maven 构建项目需要的Maven最低版本

issueManagement 项目问题管理系统（Bugzilla，Jira，Scarab）

ciManagement 项目持续集成信息

release 自动化部署

> 插件是在pom.xml的plugins元素定义的

上传jar包至私服-客户端上传

首先在maven的settings.xml配置账号信息

```
<servers>
  <server>
     <id>snapshots</id>
     <username>username</username>
     <password>password</password>  /*jd私服，填写API key即可
  </server>
</servers>
```

在项目pom中配置

```
<distributionManagement>
    <snapshotRepository>
        <id>snapshots</id>
        <name>JD maven2 repository-snapshots</name>
        <url>http://artifactory.jd.com/libs-snapshots-local</url>
    </snapshotRepository>
</distributionManagement>
```

命令上传语法

```
mvn deploy:deploy-file -DgroupId=testArty -DartifactId=testArty-artyjava -
Dversion=1.0-SNAPSHOT -Dpackaging=jar -Dfile=D:\testArty-artyjava-1.0-
SNAPSHOT.jar -Durl=http://artifactory.jd.com/libs-snapshots-local/ -
DrepositoryId=snapshots

/**注意：DrepositoryId 必须与setting中server模块的ID一致，否则不能上传
```

项目pom文件

modelVersion 模型版本，一般使用默认的4.0.0

groupId 工程组标识，必需

artifactId 工程标识，必需，与groupId确定唯一一个项目

version 工程版本号，必需

relativePath 父项目的pom.xml文件相对路径，默认../pom.xml，可以手动指定。Maven构建时先在当前项目寻找父项目的pom，其次再去文件系统的这个位置（relativePath位置）找pom，再依次是本地仓库、远程仓库

packaging 项目产生的构建类型，例如jar、war、ear、pom。

build 构建项目需要的信息

sourceDirectory 设置项目源码目录，当构建是系统会编译目录里的源码，路径是相对于pom.xml的相对路径

resource/resources 项目相关所有资源路径列表

targetPath 资源的目标路径，相对于target/classes目录，

directory 描述存放资源的目录，相对于pom路径

filtering 是否使用参数值代替参数名。参数值取自properties元素或文件里的配置

incldes/excludes 包含/排除 的模式列表

testResources 单元测试存放的资源

filters 当filtering开关打开时，使用到的过滤器文件列表

pluginManagement 子项目可以引用的默认插件信息。该配置知道被引用时才会被解析或绑定到生命周期

plugins 使用到的插件列表

profile/profiles 构建配置，可通maven指定profile。dev、test、preprod、prod

activation 自动触发profile的条件逻辑。

activeByDefault profile默认被激活的标志

modules 模块/子模块，用于多模块项目。列出每个模块的路径

repositoriy/repositories 发现依赖和扩展的远程仓库

releases 正式发行版本包配置

enabled true/false，表示该仓库是否开启下载该类型（正式、快照）包

updatePolicy 包更新的频率。选项：always（一直）、daily（每日，默认）、interval：X（单位为分钟）、never

checksumPolicy 当验证失败时怎么做，ingore、fail、warn

snapshots 快照版本包配置

pluginRepositories 发现插件的远程插件列表，用于构建和报表

dependences/dependency 依赖

type 包类型，默认jar

classifer 依赖的分类器。可用于标识同一个包不同编译器类型编译，例如java7和java8编译器

scope 依赖范围，在项目发布过程中，决定哪些构件被包括起来

> scope依赖范围取值
>
> compile：默认，用于编译
>
> provided： 类似于编译，但支持jdk或容器提供，类似于classpath
>
> runtime：执行时使用
>
> test：测试任务时使用
>
> system：系统本地jar包，需要提供外在相应的元素，通过systemPath获得

> optional：当自身被依赖时，标注依赖是否被传递。用于连续时依赖

systemPath 与system对应

exclusions 排除依赖构件

optional 可选依赖，阻断依赖的传递性。

reporting 生成报表插的规范，用于 mvn site

***dependencyManagement** 所有子项目的默认依赖信息。这部分的依赖信息不会立即解析，而是当子项目中声明一个依赖没有声明除groupId和artifactId之外的信息，子项目就会到着来匹配其它信息，如版本号

dependencies 同上

distributionManagement 项目发布配置，在使用 mvn deploy 指定发布的配置

repository 项目产生的构件到远程仓库的配置信息

snapshotRepository 快照配置信息

relocation 如果构件有了新的groupId和artifactId，此处列出构件的重定位信息

properties 配置变量，用pom中

快照&正式版

快照 项目开发时每天都在更新，而开发中的项目别人又需要使用，所以制定一个快照版本，指不稳定版本，snapshot。快照版本maven每次构建都会去下载

正式版 与快照相对的是正式版，一个版本一个，稳定，release。正式版maven构建时只要仓库存在就不会去远程仓库下载

依赖管理

传递依赖：B依赖A，C依赖B，C就依赖。可以利用exclusion排除传递依赖，或optional设置可选传递依赖

就近原则，两个依赖版本相同深度下，第一个声明的依赖会被使用。不同深度下依赖深度较小的

参考

1. https://www.runoob.com/maven
2. https://maven.apache.org/