架构师面试题 - Maven 专题篇

目录

1. Maven 有哪些优点和缺点

优点如下：
1.简化了项目依赖管理：
2.易于上手，对于新手可能一个"mvn clean package"命令就可能满足他的工作
3.便于与持续集成工具（jenkins）整合
4.便于项目升级，无论是项目本身升级还是项目使用的依赖升级。
5.有助于多模块项目的开发，一个模块开发好后，发布到仓库，依赖该模块时可以直接从仓库更新，而不用自己去编译。
6.maven 有很多插件，便于功能扩展，比如生产站点，自动发布版本等

缺点如下：
1.maven 是一个庞大的构建系统，学习难度大
2.maven 采用约定优于配置的策略（convention over configuration），虽然上手容易，但是一旦出了问题，难于调试。
3.当依赖很多时，m2eclipse 老是搞得 Eclipse 很卡。

4.中国的网络环境差，很多 repository 无法访问，比如 google code， jboss 仓库无法访问等。

2. Maven 坐标

一般 maven 使用[groupID,artifactId,version，packaging]来表示一个项目的某个版本，有时还会使用 classifier 来表示项目的附属构建，常见的附属构建有 javadoc 和 sources 包。

3. Maven 常见的依赖范围有哪些?

1）compile:编译依赖，默认的依赖方式，在编译（编译项目和编译测试用例），运行测试用例，运行（项目实际运行）三个阶段都有效，典型地有 spring-core 等 jar。
2）test:测试依赖，只在编译测试用例和运行测试用例有效，典型地有 JUnit。
3）provided:对于编译和测试有效，不会打包进发布包中，典型的例子为 servlet-api,一般的 web 工程运行时都使用容器的 servlet-api。
4）runtime:只在运行测试用例和实际运行时有效，典型地是 jdbc 驱动 jar 包。
5）system: 不从 maven 仓库获取该 jar,而是通过 systemPath 指定该 jar 的路径。
6）import: 用于一个 dependencyManagement 对另一个 dependencyManagement 的继承。

4. Maven 的生命周期

maven 有三套生命周期，分别为：
1、clean 周期：主要用于清理上一次构建产生的文件，可以理解为删除 target 目录
2、默认周期，
主要阶段包含:
1）process-resources 默认处理 src/test/resources/下的文件，将其输出到测试的 classpath 目录中,
2）compile 编译 src/main/java 下的 java 文件，产生对应的 class,
3）process-test-resources 默认处理 src/test/resources/下的文件，将其输出到测试的 classpath 目录中,
4）test-compile 编译 src/test/java 下的 java 文件，产生对应的 class,
5）test 运行测试用例,
6）package 打包构件，即生成对应的 jar, war 等,
7）install 将构件部署到本地仓库,
8）deploy 部署构件到远程仓库
3、site 周期
主要阶段包含：
λsite 产生项目的站点文档
λsite-deploy 将项目的站点文档部署到服务器

5. 我们经常使用“Mvn Clean Package”命令进行项目打包，请问该命令执行了哪些动作来完成该任务？

在这个命令中我们调用了 maven 的 clean 周期的 clean 阶段绑定的插件任务，以及 default 周期的 package 阶段绑定的插件任务
默认执行的任务有（maven 的术语叫 goal，也有人翻译成目标，我这里用任务啦）：

```
maven-clean-plugin:clean->
maven-resources-plugin:resources->
maven-compile-plugin:compile->
mavne-resources-plugin:testResources->
maven-compile-plugin:testCompile->
maven-jar-plugin:jar
```

6. 依赖的解析机制

解析发布版本：如果本地有，直接使用本地的，没有就向远程仓库请求。
解析快照版本：合并本地和远程仓库的元数据文件-groupId/artifactId/version/maven-metadata.xml，这个文件存的版本都是带时间戳的，将最新的一个改名为不带时间戳的格式供本次编译使用。
解析版本为 LATEST,RELEASE，过于复杂，且解析的结果不稳定， 不推荐在项目中使用，感兴趣的同学自己去研究，简而言之就是合并 groupId/artifactId/maven-metadata.xml 找到对应的最新版本和包含快照的最新版本。

7. 插件的解析机制

当我们输入"mvn dependency:tree"这样的指令，解析的步骤为：
1）解析 groupID:maven 使用默认的 groupID:"org.apache.maven.plugins"或者 "org.codehaus.mojo"
2）解析 artifactId(maven 的官方叫做插件前缀解析策略)
3）合并该 groupId 在所有仓库中的元数据库文件（maven-metadata-repository.xml），比如 maven 官方插件的元数据文件所在的目录为 orgapachemavenplugins，该文件下有如下的条目

```
<plugin>
<name>MavenDependencyPlugin</name>
<prefix>dependency</prefix>
<artifactId>maven-dependency-plugin</artifactId>
</plugin>
```

通过比较这样的条目，我们就将该命令的 artifactId 解析为 maven-dependency-plugin
4）解析 version：如果你在项目的 pom 中声明了该插件的版本，那么直接使用该版本

的插件，否则合并所有仓库中 groupId/artifactId/maven-metadata-repository.xml，找到最新的发布版本。

对于非官方的插件，有如下两个方法可以选择：
1）使用 groupId:artifactId:version:goal 来运行
2）在 Settings.xml 中添加 pluginGroup 项，这样 maven 不能在官方的插件库中解析到某个插件，那么就可以去你配置的 group 下查找啦。

8. 多模块如何聚合

配置一个打包类型为 pom 的聚合模块，然后在该 pom 中使用<module>元素声明要聚合的模块

9. 对于一个多模块项目，如果管理项目依赖的版本

通过在父模块中声明 dependencyManagement 和 pluginManagement， 然后让子模块通过<parent>元素指定父模块，这样子模块在定义依赖是就可以只定义 groupId 和 artifactId，自动使用父模块的 version,这样统一整个项目的依赖的版本。

10. 一个项目的依赖来源于不同的组织，可能这些依赖还会依赖别的 Jar 包，如何保证这些传递依赖不会引起版本冲突。

使用<dependency>的<exclusion>元素将会引起冲突的元素排除。

11. 常见的 Maven 私服的仓库类型。

（宿主仓库）hosted repository, （代理仓库）proxy repository, （仓库组）group repository

12. 如何查询一个插件有哪些目标（Goal）

mvn help:describe -Dplugin=groupId:artifactId

13. 什么是 Maven？

Maven 使用项目对象模型(POM)的概念，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。
Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发布时使用

Maven，而且公司项目采用 Maven 的比例在持续增长。
Maven 的出现，解决了开发过程中的 jar 包升级及依赖的难题。它可以对项目依赖的 jar 包进行管理，可以让你的项目保持基本的依赖，排除冗余 jar 包，并且可以让你非常轻松的对依赖的 jar 包进行版本升级。而这些仅仅是 Maven 最基本的功能，它可以在这基础上对项目进行清理、编译、测试、打包、发布等等构建项目的工作。
可以说，Maven 是现在 Java 社区中最强大的项目管理和项目构建工具，而更加值得庆幸的是，这样一个强大的工具，它的使用也是非常简单的。
现在，JavaEE 项目使用的开源软件都可以通过 Maven 来获取，并且，越来越多的公司也开始使用 Maven 来管理构建项目了。

14. Maven 和 ANT 的区别

1.maven&ant 同属 apach 是流行的构建工具。
都是为了简化软件开发而存在的。但是 maven 因为自身管理一个项目对象模型（project object model），这个模型其实就是抽象了一个项目的开发流程，它包含了一个项目的生命周期的各个阶段，并将这个周期固定下来，这也就是约定大于配置。约定大于配置的意思就是，我 maven 将项目开发的各个阶段固定起来了，每个文件的存放位置，每个阶段要生成什么文件、保存为什么格式并且要把它放在什么位置，我都固定好了。我知道一个软件是怎么开发出来，如果一个项目要使用 maven，可以，但你要遵循我的规则，文件目录不要乱建乱放，只有这样 maven 才会将源码用起来。这就是约定大于配置，因为 maven 已经将流程固定下来了，只要遵守约定，就不需要自己手动去配置了，这将大大地提高开发效率。就像是开车一样，只要知道点火、油门、方向、刹车，就可以将车子开动起来（当然出于安全和法律考虑，还是要考驾照的。），关于车子内部的传动原理，电气原理，工程原理，普通人并不需要了解多少，日常够用就好了。这也是约定大于配置的一个例子。配置就是自己造一辆车去开，有必要，有能力，有时间吗？
2.maven 的中央仓库和 pom.xml 文件。中央仓库统一存放了开发用到的各种 jar 包，要用时只需要添加依赖到 pom 文件中，maven 就会自动下载，当然为了方便一般会在本地建一个仓库，减少下载时间。pom 文件是 maven 的配置文件，maven 就是通过管理 pom 文件和一些核心插件来管理项目。当然我前面将 maven 拟人化了，其实 maven 是没有智力的，一切都是封装好的流程，只是 maven 将很多操作隐藏起来了。
3.ant 的 build.xml 文件。build 文件是 ant 的配置文件，ant 依靠它来执行操作，与 maven 不同的是 ant 没有固定一条程序链。你想要执行什么操作以及操作之间的顺序和依赖关系，都需要手动添加到 build 文件中，一点一滴都要写清楚，否则 ant 就不会执行。
4.maven 和 ant 区别
Maven 拥有约定，只要遵守约定，它就知道你的源代码在哪里。Maven 是声明式的。你需要做的只是创建一个 pom.xml 文件然后将源代码放到默认的目录。Maven 会帮你处理其它的事情。Maven 有一个生命周期，当你运行 mvn install 的时候被调用。这条命令告诉 Maven 执行一系列的有序的步骤，直到到达你指定的生命周期。缺点是运行许多默认目标。
而 ant 没有约定，项目生命周期，它是命令式的。所有操作都要手动去创建、布置。甚至连 build.xml 文件都需要手动创建。

15. Maven 仓库是什么

Maven 仓库是基于简单文件系统存储的，集中化管理 Java API 资源（构件）的一个服务。仓库中的任何一个构件都有其唯一的坐标，根据这个坐标可以定义其在仓库中的唯一存储路径。得益于 Maven 的坐标机制，任何 Maven 项目使用任何一个构件的方式都是完全相同的，Maven 可以在某个位置统一存储所有的 Maven 项目共享的构件，这个统一的位置就是仓库，项目构建完毕后生成的构件也可以安装或者部署到仓库中，供其它项目使用。
对于 Maven 来说，仓库分为两类：本地仓库和远程仓库。

16. Maven 的工程类型有哪些？

1）POM 工程
POM 工程是逻辑工程。用在父级工程或聚合工程中。用来做 jar 包的版本控制。
2）JAR 工程
将会打包成 jar 用作 jar 包使用。即常见的本地工程 - Java Project。
3）WAR 工程
将会打包成 war，发布在服务器上的工程。如网站或服务。即常见的网络工程 - Dynamic Web Project。war 工程默认没有 WEB-INF 目录及 web.xml 配置文件，IDE 通常会显示工程错误，提供完整工程结构可以解决。

17. Maven 常用命令有哪些？

1）install
本地安装， 包含编译，打包，安装到本地仓库
编译 - javac
打包 - jar， 将 java 代码打包为 jar 文件
安装到本地仓库 - 将打包的 jar 文件，保存到本地仓库目录中。
2）clean
清除已编译信息。
删除工程中的 target 目录。
3）compile
只编译。javac 命令
4）deploy
部署。常见于结合私服使用的命令。
相当于是 install+上传 jar 到私服。
包含编译，打包，安装到本地仓库，上传到私服仓库。
5）package
打包。包含编译，打包两个功能。

18. 你们项目为什么选用 maven 进行构建？

①首先，maven 是一个优秀的项目构建工具。使用 maven，可以很方便的对项目进行分模块构建，这样在开发和测试打包部署时，效率会提高很多。
②其次，maven 可以进行依赖的管理。使用 maven，可以将不同系统的依赖进行统一管理，并且可以进行依赖之间的传递和继承。