**中软国际考题-笔试**

1. 单选题

1、下列不属于Maven的缺点的选项是（B）

A. 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。

B. 简单易学，容易上手。

C. 当依赖很多时，会导致开发工具运行效率降低

D. 中国的网络环境差，很多repository无法访问

2、Maven哪个目录下有运行依赖jar包（B）

A.bin目录 B.lib目录 C.conf目录 D.boot目录

3、下列不属于maven的生命周期的是（ D ）

A . clean B . compile C . package D . done

4、通常我们会把java项目打包成jar包或war包，maven中打包的命令为？（A）

A.mvn clean package B.mvn package clean

C.mvn package D.package clean

5、关于Apache Maven的说法不正确的是（D）

A． Maven 是一个依赖管理工具

B． Maven 是一个构建工具

C. 通过Maven 命令可以直接生成一个网站或构建一个非java语言的项目

D. Maven目前只支持Window和Linux平台下运行

6、初始化一个Gradle项目用的命令是（A）

A． init B. start C. build D. create

7、下列对maven环境设置说明错误的是（B）

A.maven 3.3 要求使用JDK1.7及以上

B.maven目前支持的最高版本是3.5.4

C.配置maven环境变量需要新建MAVEN\_HOME并配置path变量

D.maven 3.2 要求使用JDK1.6及以上

8、下列关于pom.xml文件的配置项描述错误的是（D）

A.项目依赖 B.插件 C.项目版本 D.项目运行的时间

9、下列关于maven插件的作用描述错误的是（C）

A.创建jar文件 B.编译代码文件 C.表示工程名称 D.创建工程文档

10、下面那个不是maven的优点？（D）

A.依赖管理，节省空间 B.一键构建 编码 编译 测试(junit) 运行 打包 部署

C.可以提高开发效率 D.不能应用于大型项目

11、以下哪个不能用于确定Maven的坐标？（C）

A.groupId B.artifactId C.class D.version

12、常见的Maven私服的仓库类型不包括（A）

A.多态仓库 B.宿主仓库 C.代理仓库 D.仓库组

13、maven项目的配置文件是（B）

A..java B..pom C..properties D.xml

1. 多选题
2. 下面哪些项属于Maven的生命周期？（A、D、E）
3. clean周期 B. 阶段周期 C. 初始化周期
4. site周期 E. 默认周期
5. 下面哪些项属于Gradle的基本组成成分？（A、B、C、D）
6. Project与Task B. 插件 C. 配置文件

D. 构建脚本 E. 平台下载机制

3、Maven仓库有？（A、B、C）

A. 本地 B.中央 C. 远程 D.数据仓库

4、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？（ ABC ）

A . clean B . compile C . package D . install

5、Maven 能够帮助开发者完成哪些工作？（A、BCDEFGH）

A. 构建 B. 文档生成 C. 报告 D. 依赖 E. SCMs

F. 发布 G. 分发 H. 邮件列表

6、以下哪项属于maven默认生命周期的阶段（B C）

A. clean B. compile C. package D. site

7、关于Gradle的说法不正确的是（BCD）

A． Gradle 是Android开发的首选构建工具

B． Gradle不能使用Maven仓库

C. Gradle不能编译除java外的其他语言

D. Gradle的DSL是XML

8、在build.gradle文件中都会包含对application插件的应用，以下对该插件的作用描述正确的是（A、B、C、D）

A.assemble: 构建项目的输出（apk）

B.check: 进行校验工作

C.build: 执行assemble任务与check任务

D.clean: 清除项目的输出

9、Maven的生命周期包括：（A、B、C）

A.clean生命周期 B.default生命周期 C.site生命周期 D.init生命周期

10、下面哪些项属于maven的依赖范围？（A、B、C、D、E）

A. compile B. test C . provided D. runtime

E. import F. package

1. 填空题（每空1分，共10分）
2. 配置一个打包类型为pom的聚合模块，然后在该pom中使用\_\_\_\_\_\_\_\_\_\_元素声明要聚合的模块。(<module>)
3. 构建工具，ant、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、结束。(Maven、Gradle)
4. Maven的三个生命周期为：默认（default），清洁（clean）和站点（site）。
5. Maven 是一个\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_工具。(项目管理、项目整合)
6. 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。
7. Gradle脚本的执行分为三个过程：­­­­初始化、配置、执行
8. 通过gradle构建的工程在添加依赖是需要在build.gradle文件的\_\_\_\_\_\_\_\_\_\_\_模块中添加compile。(dependencies)
9. Mave多模块聚合：配置一个打包类型为pom的聚合模块，然后在该pom中使用<\_\_\_\_\_\_\_\_\_\_ >元素声明要聚合的模块。（module）
10. Maven的依赖范围：\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_结束。(compile、test、provided、runtime、system、import)
11. Maven项目的坐标组成：\_\_\_\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_\_\_ 、\_\_\_\_\_\_\_\_\_。（groupId、artifactId、version）
12. Maven项目的配置文件pom的全称：\_\_\_\_\_\_\_\_\_。（Project Object Model）
13. 验证maven安装成功的命令（ ）.cmd中mvn -v
14. Maven的本地仓库设置在( )文件.settings.xml文件
15. Maven的仓库设置是（ ）标签.localRepository标签
16. Maven测试仓库是否成功的命令( ).mvn help:system
17. pom的全称为( ).project object model
18. Gradle是继 Ant 、Maven 之后的现代( ).构建工具
19. cmd中窗口中，查看path的命令是( )echo %path%
20. Gradle 中非常重要的两个对象( )Project和Task。
21. 常用的构建工具有()和(). Maven和Gradle
22. 判断题
23. Gradle插件的核心工作有两个：一是定义Task,二是执行Task。（ T ）
24. 一般maven使用[groupID,graupName,source]来表示一个项目的某个版本（ F ）
25. 编译源代码mvn compile。（ T ）
26. 配置本地仓库和远程仓库是在setting.xml文件中。（ T ）
27. Gradle是一个基于Apache Ant和Apache Maven概念的项目自动化构建开源工具。（ T ）
28. 能用Maven的项目不能使用Gradle。（ F ）
29. Gradle是一种可切换的，像 maven 一样的基于约定约定优于配置的构建框架。 （ T ）
30. Gradle支持已有的 maven 和 ivy 仓库。（ T ）
31. pom.xml文件是maven项目的核心文件。( T )
32. Maven能够将任意数量的项目构建到预定义的输出类型中，如 JAR，WAR 或基于项目元数据的分发，而不需要在大多数情况下执行任何脚本。 （ T ）
33. 使用与构建过程相同的元数据，Maven 能够生成一个网站或PDF，包括您要添加的任何文档，并添加到关于项目开发状态的标准报告中。 （ T ）
34. 子项目使用父项目依赖时，正常情况子项目应该继承父项目依赖，无需使用版本号。（ T ）
35. Maven 为开发者提供了一套完整的构建生命周期框架。。（ T ）
36. Maven的功能集比后出现的Ant的功能集小（ F）
37. Gradle作为编译工具，可以自动完成Java的从编译到运行的全部过程（T）
38. Groovy 的插件继承自 Java 插件并添加了对 Groovy 项目的支持。它可以处理 Groovy 代码，以及混合的 Groovy 和 Java 代码，但是不能处理纯 Java 代码。（ F ）
39. 通过gradle构建工程时，将工程打包为war文件，需要使用apply plugin: ‘war’。（ T ）
40. 通过gradle构建的项目，在启动Web项目时，使用Jetty plugin插件，那么在build.gradle中需要添加apply plugin：‘jetty:running’。（ F ）
41. maven采用约定优于配置的策略，虽然上手容易，但是一旦出了问题，难于调试。（T）
42. static不能修饰方法中定义的变量。（ T ）

五、简答题

1、Maven有哪些优点和缺点？

答案：

简化了项目依赖管理

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

便于与持续集成工具（jenkins）整合

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。

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。

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.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常见的依赖范围有哪些?

答案：

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。

3、描述maven依赖的5个作用范围，即<scope>标签的值

答案：

Compile：默认的范围，编译、测试、运行的classpath都有效

Provided：编译、测试有效，最后运行的时候不会被加入

Runtime：测试、运行有效

Test：仅测试有效

System：与本机系统相关联，编译测试有效，但是可移植性差

Import：导入的依赖范围，它只是用在dependencyManagemet中，表示从其他的pom中导入dependency的配置

4、compile 和 provided区别

答案：

对于scope=compile的情况（默认scope)，也就是说这个项目在编译，测试，运行阶段都需要这个artifact对应的jar包在classpath中。

而对于scope=provided的情况，则可以认为这个provided是目标容器已经provide这个artifact。换句话说，它只影响到编译，测试阶段。在编译测试阶段，需要这个artifact对应的jar包在classpath中，而在运行阶段，假定目标的容器（如tomcat、jboss）已经提供这个jar包，所以无需这个artifact对应的jar包。

实际插件的行为：

用maven install生成最终的构件包demo.war后，在其下的WEB-INF/lib中，会包含被标注为scope=compile的构件的jar包，而不会包含被标注为scope=provided的构件的jar包。这也避免此类构件当部署到目标容器后产生包依赖冲突。

5、Maven常见的依赖范围有哪些?

答案：

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。

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。

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。

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。

system: 不从maven仓库获取该jar,而是通过systemPath指定该jar的路径。

import: 用于一个dependencyManagement对另一个dependencyManagement的继承。

6、Maven的生命周期：

答案：

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:

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,

compile 编译src/main/java下的java文件，产生对应的class,

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,

test-compile 编译src/test/java下的java文件，产生对应的class,

test 运行测试用例,

package 打包构件，即生成对应的jar, war等,

install将构件部署到本地仓库,

deploy 部署构件到远程仓库

3、site周期

主要阶段包含

site 产生项目的站点文档

site-deploy 将项目的站点文档部署到服务器

7、Maven 的主要目的是为开发者提供什么？

答案：一个可复用、可维护、更易理解的工程综合模型

与这个模型交互的插件或者工具

Maven 工程结构和内容被定义在一个 xml 文件中 － pom.xml，是 Project Object Model (POM) 的简称，此文件是整个 Maven 系统的基础组件。详细内容请参考 [Maven POM](https://www.w3cschool.cn/maven/varq1ht4.html) 部分。

8、Maven查找依赖的过程（5分）

答案：本地仓库->镜像仓库->中央仓库

9、Gradle的优势在哪里（5分）

使用Gradle的原因：

(1)、使用领域专用语言(Domain Specific Language)来描述和处理构建逻辑。

(2)、基于Groovy。DSL可以混合各种声明元素，用代码操控这些DSL元素达到逻辑自定义。

(3)、支持已有的Maven或者Ivy仓库基础建设。

(4)、非常灵活，允许使用best practices，并不强制让你遵照它的原则来。

(5)、其它插件时可以暴露自己的DSL和API来让Gradle构建文件使用。

(6)、允许IDE集成，是很好的API工具。

10、请描述Maven依赖搜索的顺序。

答案：

步骤 1 － 在本地仓库中搜索，如果找不到，执行步骤 2，如果找到了则执行其他操作。

步骤 2 － 在中央仓库中搜索，如果找不到，并且有一个或多个远程仓库已经设置，则执行步骤 4，如果找到了则下载到本地仓库中以备将来引用。

步骤 3 － 如果远程仓库没有被设置，Maven 将简单的停滞处理并抛出错误（无法找到依赖的文件）。

步骤 4 － 在一个或多个远程仓库中搜索依赖的文件，如果找到则下载到本地仓库以备将来引用，否则 Maven 将停止处理并抛出错误（无法找到依赖的文件）。

11、maven的镜像服务怎么配置.

找到maven安装目录下的settings.xml文件,找到mirror标签设置如下:

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

12. eclipse中怎么关联安装的maven.

首先:window–>preference–>maven

add maven的安装路径 如:D:\maven\apache-maven-3.6.0

其次:更改用户设置(如D:\maven\apache-maven-3.6.0\conf\settings.xml)和本地仓库(如d:/m2/repos)

13.Maven中的常用目录结构有哪些？

src/main/java目录用于存放java文件，例如A.java

src/main/resources目录用于存放应用相关的配置文件，例如struts.xml

src/main/webapp目录用于存放页面相关的文件，例如.jsp或者.jpg、.css、.js等

src/test/java目录用于存放单元测试的java文件

src/test/resources目录用于存放单元测试相关的配置文件

14.当我们的项目中用到了一些第三方库时，我们就需要进行一些配置，以保证能正确导入相关依赖。设置方法如下( )

dependencies {

...

compile 'com.facebook.fresco:fresco:0.11.0'

}

15.导入本地jar包怎么设置?

Dependencies{

compile files('libs/xxx.jar')

}

或

Dependencies{

compile fileTree(dir: 'libs', include: ['\*.jar'])

}

16.依赖其它模块怎么设置?

假设我们的项目包含了多个模块，并且app模块依赖other模块，那么我们只需app\build.gradle的denpendencies块下添加如下语句：

Dependencies{

compile project(':other')

}

六、程序实现题

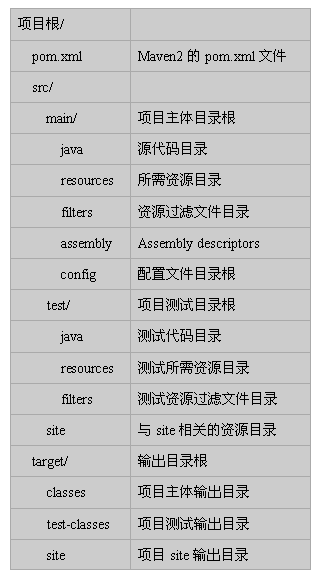
1、maven 修改maven本地仓库的位置

（10分）

|  |
| --- |
| 打开Maven目录下的文件：conf/settings.xml  <localRepository>d:\repository</localRepository> |

2、请写出maven的标准目录结构

答案：



3、编写一个Maven插件，挂载在package生命阶段，统计当前工程里面有多少个java文件：

@Mojo(name="countFile",defaultPhase= LifecyclePhase.PACKAGE)

public class CountFileMojo extends AbstractMojo {

    @Parameter(property = "path")

    private String path;

    int num1 = 0;   //总文件数量

    int num2 = 0;   //目录数量

    int num3 = 0;   //java文件数量

    public void execute() throws MojoExecutionException,MojoFailureException{

        System.out.println(path);

        System.out.println(countFile(path));

    }

    public String  countFile(String dir){

        File f = new File(dir);

        File fs[] = f.listFiles();

        if(fs!=null){

            for(int i=0;i<fs.length;i++){

                File  currenFile = fs[i];

                if(currenFile.isFile()){

                    num1+=1;   //如果是文件就加1

                    String fileName = currenFile.getName();

                    String suffix = fileName.substring(fileName.lastIndexOf(".") + 1);

                    if(suffix.equals("java")){

                        num3+=1;

                    }

                }else{

                    //否则就是目录

                    num2+=1;

                    countFile(currenFile.getAbsolutePath());

                }

            }

        }

        return "Total number of File:"+num1+"||||The number of dir is:"+num2+"||||Total number of Java File:"+num3;

    }

}

4、在一个JavaEE工程中需要操作数据库，选择使用mysql6.0.6版本，请分别写出通过maven以及gradle引入依赖的代码。

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>6.0.6</version>  </dependency>  // https://mvnrepository.com/artifact/mysql/mysql-connector-java  compile group: 'mysql', name: 'mysql-connector-java', version: '6.0.6' |

5、写出Maven中配置mysql数据库驱动的代码（版本8.0.15）

（10分）

|  |
| --- |
| <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.15</version>  </dependency> |