<Codingcompiler/>

Maven Interview Questions And Answers For Experienced 2018

Maven Interview Questions And Answers For Experienced 2018. Here Coding compiler sharing 45 *Real-Time Maven Interview Questions*. These **scenario-based Maven interview questions** were asked in various interviews conducted by top MNC companies across the globe. These Apache Maven **questions** are prepared by industry expert *Maven professionals*. We hope that this list of **interview questions on**

Maven will help you to crack your next *Maven job interview*. All the best for your future and happy learning.

Maven Interview Questions

1. **What is Maven Tool?**
2. **Why should we use Maven?**
3. **What is a Maven POM file?**
4. **What are the repositories in Maven?**
5. **What is an artifact in Maven?**
6. **What is a Maven dependency?**
7. **What does Maven dependency plugin do?**
8. **What is the use of plugins in Maven?**
9. **What is meant by** goal **in Maven?**
10. **Where are Maven dependencies downloaded to?**
11. **What is a maven assembly?**
12. **What is meant by Mojo in Maven?**
13. **What is the use of Maven clean?**
14. **Where are Maven dependencies stored?**
15. **What is a snapshot Maven?**

Maven Interview Questions And Answers

Maven Interview Questions	
Maven is an	Open source software
Maven is a	Build automation tool used primarily for Java projects.
Maven is	Based on the concept of a project object model (POM).
Maven can	Manage a project's build, reporting and

	documentation from a central piece of information.
Maven developed by	Apache Software Foundation
Maven License	Apache License 2.0
Maven has written in	Java

1) What is Maven Tool?

A) Maven is a build automation tool used primarily for Java projects. Maven is a build automation tool used primarily for Java projects.

2) Why should we use Maven?

A) Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. Maven repository is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories.

3) What is a Maven POM file?

A) A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

4) What are the repositories in Maven?

A) In Maven terminology, a repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

5) What is an artifact in Maven?

A) An artifact is a file, usually a JAR, that gets deployed to a Maven repository. A Maven build produces one or more artifacts, such as a compiled JAR and a “sources” JAR. Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string.

6) What is a Maven dependency?

A) When maven is run on project B version 1.0 of artifacts a, b, c, and d will be used regardless of the version specified in their pom. a and c both are declared as dependencies of the project so version 1.0 is used due to dependency mediation. Both will also have runtime scope since it is directly specified.

7) What does Maven dependency plugin do?

A) Apache Maven Dependency Plugin. The dependency plugin provides the capability to manipulate artifacts. It can copy and/or unpack artifacts from local or remote repositories to a specified location.

8) What is the use of plugins in Maven?

A) “Maven” is really just a core framework for a collection of Maven Plugins. In other words, plugins are where much of the real action is performed, plugins are used to: create jar files, create war files, compile code, unit test code, create project documentation, and on and on.

9) What is meant by a goal in Maven?

A) Executing a phase means executes all previous phases. Plugin is a collection of goals. Plugin is a class and goals are methods within the class. Maven is based around the central concept of a build lifecycle.

10) Where are Maven dependencies downloaded to?

A) The jars , dependency files and other files which are downloaded by Maven reside in the Maven local repository. By default the Maven local repository is the .m2 folder. You can copy the jar directly into where it is meant to go. Maven will find this file next time it is runs.

Top 45 Maven Interview Questions And Answers

11) What is a maven assembly?

A) The Assembly Plugin for Maven is primarily intended to allow users to aggregate the project output along with its dependencies, modules, site documentation, and other files into a single distributable archive.

12) What is meant by Mojo in Maven?

A) A mojo is a Maven plain Old Java Object. Each mojo is an executable goal in Maven, and a plugin is a distribution of one or more related mojos. In short, a mojo is a maven goal, to extend functionality not already found in maven.

13) What is the use of Maven clean?

A) The Maven Clean Plugin, as the name implies, attempts to clean the files and directories generated by Maven during its build. While there are plugins that generate additional files, the Clean Plugin assumes that these files are generated inside the target directory.

14) Where are Maven dependencies stored?

A) The maven local repository is a local folder that is used to store all your project's dependencies (plugin jars and other files which are downloaded by Maven). In simple, when you build a Maven project, all dependency files will be stored in your Maven local repository.

15) What is a snapshot Maven?

A) A snapshot version in Maven is one that has not been released. The idea is that before a 1.0 release (or any other release) is done, there exists a 1.0-SNAPSHOT . That version is what might become 1.0 . It's basically " 1.0 under development".

16) What is an archetype in Maven?

A) Archetype is a Maven project templating toolkit. An archetype is defined as an original pattern or model from which all other things of the same kind are made. The name fits as we are trying to provide a system that provides a consistent means of generating Maven projects.

17) What are the Maven advantages over Ant?

A) Convention over Configuration – Maven uses a distinctive approach for the project layout and startup, that makes easy to just jump in a project. Usually it only takes the checkount and the maven command to get the artifacts of the project.

Project Modularization – Project conventions suggest (or better, force) the developer to modularize the project. Instead of a monolithic project you are often forced to divide your project in smaller sub components, which make it easier debug and manage the overall project structure

Dependency Management and Project Lifecycle – Overall, with a good SCM configuration and an internal repository, the dependency management is quite easy, and you are again forced to think in terms of Project Lifecycle – component versions, release management and so on. A little more complex than the ant something, but again, an improvement in quality of the project.

What is wrong with maven?

Maven is not easy. The build cycle (what gets done and when) is not so clear within the POM. Also, some issue arise with the quality of components and missing dependencies in public repositories.

The best approach (to me) is to have an internal repository for caching (and keeping) dependencies around, and to apply to release management of components. For projects bigger than the sample projects in a book, you will thank maven before or after.

The Best Maven Interview Questions And Answers

Maven Interview Questions # 18) How do you tell Maven to use the latest version of a dependency?

In Maven, dependencies are usually set up like this:

```
<dependency>
<groupId>wonderful-inc</groupId>
<artifactId>dream-library</artifactId>
<version>1.2.3</version>
</dependency>
```

Now, if you are working with libraries that have frequent releases, constantly updating the <version> tag can be somewhat annoying. Is there any way to tell Maven to always use the latest available version (from the repository)?

A) In Maven 2, If you always want to use the newest version, Maven has two keywords you can use as an alternative to version ranges. You should use these options with care as you are no longer in control of the plugins/dependencies you are using.

A square bracket ([&]) means "closed" (inclusive).
A parenthesis ((&)) means "open" (exclusive).
Here's an example illustrating the various options. In the Maven repository, com.foo:my-foo has the following metadata:

```
<?xml version="1.0″ encoding="UTF-8″?><metadata>
<groupId>com.foo</groupId>
```

```
<artifactId>my-foo</artifactId>
<version>2.0.0</version>
<versioning>
<release>1.1.1</release>
<versions>
<version>1.0</version>
<version>1.0.1</version>
<version>1.1</version>
<version>1.1.1</version>
<version>2.0.0</version>
</versions>
<lastUpdated>20090722140000</lastUpdated>
</versioning>
</metadata>
```

If a dependency on that artifact is required, you have the following options (other version ranges can be specified of course, just showing the relevant ones here):

Declare an exact version (will always resolve to 1.0.1):

```
<version>[1.0.1]</version>
```

Declare an explicit version (will always resolve to 1.0.1 unless a collision occurs, when Maven will select a matching version):

```
<version>1.0.1</version>
```

Declare a version range for all 1.x (will currently resolve to 1.1.1):

```
<version>[1.0.0,2.0.0)</version>
```

Declare an open-ended version range (will resolve to 2.0.0):

```
<version>[1.0.0,)</version>
```

Declare the version as LATEST (will resolve to 2.0.0) (removed from maven 3.x)

```
<version>LATEST</version>
```

Declare the version as RELEASE (will resolve to 1.1.1) (removed from maven 3.x):

```
<version>RELEASE</version>
```
Note that by default your own deployments will update the “latest” entry in the Maven metadata, but to update the “release” entry, you need to activate the “release-profile” from the Maven super POM. You can do this with either “-Prelease-profile” or “-DperformRelease=true”

Maven Interview Questions # 19) How can you create an executable JAR with dependencies using Maven?

I want to package my project in a single executable JAR for distribution.

How can I make Maven package all dependency JARs into my JAR?

A) You can use the dependency-plugin to generate all dependencies

```
<build>
<plugins>
<plugin>
<artifactId>maven-assembly-plugin</artifactId>
<configuration>
<archive>
<manifest>
<mainClass>fully.qualified.MainClass</mainClass>
</manifest>
</archive>
<descriptorRefs>
<descriptorRef>jar-with-dependencies</descriptorRef>
</descriptorRefs>
</configuration>
</plugin>
</plugins>
</build>
```
and you run it with

mvn clean compile assembly:single
Compile goal should be added before assembly:single or otherwise the code on your own project is not included.

See more details in comments.

Commonly this goal is tied to a build phase to execute automatically. This ensures the JAR is built when executing mvn install or performing a deployment/release.

```
<plugin>
<artifactId>maven-assembly-plugin</artifactId>
<configuration>
<archive>
<manifest>
<mainClass>fully.qualified.MainClass</mainClass>
</manifest>
</archive>
<descriptorRefs>
<descriptorRef>jar-with-dependencies</descriptorRef>
</descriptorRefs>
</configuration>
<executions>
<execution>
<id>make-assembly</id> <!– this is used for inheritance merges –>
<phase>package</phase> <!– bind to the packaging phase –>
<goals>
<goal>single</goal>
</goals>
</execution>
</executions>
</plugin>
```

Scenario Based Maven Interview Questions

Maven Interview Questions # 20) How do you convert input stream into string?

If you have java.io.InputStream object, how should you process that object and produce a String?

Suppose I have an InputStream that contains text data, and I want to convert this to a String. For example, so I can write the contents of the stream to a log file.

What is the easiest way to take the InputStream and convert it to a String?

```
public String convertStreamToString(InputStream is) {
// ???
}
```

A) A nice way to do this is using Apache commons IOUtils to copy the InputStream into a StringWriter… something like

```
StringWriter writer = new StringWriter();
IOUtils.copy(inputStream, writer, encoding);
String theString = writer.toString();
```

or even

```
// NB: does not close inputStream, you can use IOUtils.closeQuietly for that
String theString = IOUtils.toString(inputStream, encoding);
```

Alternatively, you could use ByteArrayOutputStream if you don't want to mix your Streams and Writers

Maven Interview Questions # 21) What are the different ways to convert an InputStream to a Sting?

A) Ways to convert an InputStream to a String:

1) Using IOUtils.toString (Apache Utils)

```
String result = IOUtils.toString(inputStream, StandardCharsets.UTF_8);
```

2) Using CharStreams (guava)

```
String result = CharStreams.toString(new InputStreamReader(
inputStream, Charsets.UTF_8));
```

3) Using Scanner (JDK)

```
Scanner s = new Scanner(inputStream).useDelimiter("\\A");
String result = s.hasNext() ? s.next() : "";
```

4) Using Stream Api (Java 8). Warning: This solution convert different line breaks (like \r\n) to \n.

```
String result = new BufferedReader(new InputStreamReader(inputStream))
.lines().collect(Collectors.joining("\n"));
```

5) Using parallel Stream Api (Java 8). Warning: This solution convert different line breaks (like \r\n) to \n.

```
String result = new BufferedReader(new InputStreamReader(inputStream)).lines()
.parallel().collect(Collectors.joining("\n"));
```

6) Using InputStreamReader and StringBuilder (JDK)

```
final int bufferSize = 1024;
final char[] buffer = new char[bufferSize];
final StringBuilder out = new StringBuilder();
Reader in = new InputStreamReader(inputStream, "UTF-8");
for (; ; ) {
int rsz = in.read(buffer, 0, buffer.length);
if (rsz < 0)
break;
out.append(buffer, 0, rsz);
}
return out.toString();
```

7) Using StringWriter and IOUtils.copy (Apache Commons)

```
StringWriter writer = new StringWriter();
IOUtils.copy(inputStream, writer, “UTF-8”);
return writer.toString();
```

8) Using ByteArrayOutputStream and inputStream.read (JDK)

```
ByteArrayOutputStream result = new ByteArrayOutputStream();
byte[] buffer = new byte[1024];
int length;
while ((length = inputStream.read(buffer)) != -1) {
result.write(buffer, 0, length);
}
// StandardCharsets.UTF_8.name() > JDK 7
return result.toString(“UTF-8”);
```

9) Using BufferedReader (JDK). Warning: This solution convert different line breaks (like \n\r) to line.separator system property (for example, in Windows to “\r\n”).

```
String newLine = System.getProperty(“line.separator”);
BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
StringBuilder result = new StringBuilder();
String line; boolean flag = false;
while ((line = reader.readLine()) != null) {
result.append(flag? newLine: “”).append(line);
flag = true;
}
return result.toString();
```

10) Using BufferedInputStream and ByteArrayOutputStream (JDK)

```
BufferedInputStream bis = new BufferedInputStream(inputStream);
ByteArrayOutputStream buf = new ByteArrayOutputStream();
```

```
int result = bis.read();
while(result != -1) {
buf.write((byte) result);
result = bis.read();
}
// StandardCharsets.UTF_8.name() > JDK 7
return buf.toString("UTF-8");
```

11) Using inputStream.read() and StringBuilder (JDK). Warning: This solution has problem with Unicode, for example with Russian text (work correctly only with non-Unicode text)

```
int ch;
StringBuilder sb = new StringBuilder();
while((ch = inputStream.read()) != -1)
sb.append((char)ch);
reset();
return sb.toString();
```

Maven Technical Interview Questions And Answers

Maven Interview Questions # 22) How do you find Oracle JDBC driver in Maven repository?

I want to add the oracle jdbc driver to my project as dependency (runtime scope) – ojdbc14. In MVNrepository site the dependency to put in the POM is:

```
<dependency>
<groupId>com.oracle</groupId>
<artifactId>ojdbc14</artifactId>
<version>10.2.0.3.0</version>
</dependency>
```

of course this does't work as it is not in the central repository used by maven. 2 questions:

How do I find a repository (if any) that contains this artifact?

How do I add it so that Maven will use it?

A) Unfortunately due the binary license there is no public repository with the Oracle Driver JAR. This happens with many dependencies but is not Maven's fault. If you happen to find a public repository containing the JAR you can be sure that is illegal.

How do I add it so that Maven will use it?

Some JARs that can't be added due to license reasons have a pom entry in the Maven Central repo. Just check it out, it contains the vendor's preferred Maven info:

```
<groupId>com.oracle</groupId>
<artifactId>ojdbc14</artifactId>
<version>10.2.0.3.0</version>
```

...and the URL to download the file which in this case is
http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html.

Once you've downloaded the JAR just add it to your computer repository with (note I pulled the groupId, artifactId and version from the POM):

```
mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc14 \
-Dversion=10.2.0.3.0 -Dpackaging=jar -Dfile=ojdbc.jar -DgeneratePom=true
```

The last parameter for generating a POM will save you from pom.xml warnings

If your team has a local Maven repository this guide might be helpful to upload the JAR there.

Maven Interview Questions # 23) What is force maven update?

A) mvn clean install -U
-U means force update of snapshot dependencies. Release dependencies can't not be updated this way.

Maven Interview Questions # 24) How to add local jar files to a Maven project?

How do I add local jar files (not yet part of the Maven repository) directly in my project's library sources?

A) Install the JAR into your local Maven repository as follows:

```
mvn install:install-file
-Dfile=<path-to-file>
-DgroupId=<group-id>
-DartifactId=<artifact-id>
-Dversion=<version>
-Dpackaging=<packaging>
-DgeneratePom=true
```

Where: <path-to-file> the path to the file to load
<group-id> the group that the file should be registered under
<artifact-id> the artifact name for the file
<version> the version of the file
<packaging> the packaging of the file e.g. jar

or

2) You can add local dependencies directly (as mentioned in build maven project with propriatery libraries included) like this:

```
<dependency>
<groupId>com.sample</groupId>
<artifactId>sample</artifactId>
<version>1.0</version>
<scope>system</scope>
<systemPath>${project.basedir}/src/main/resources/yourJar.jar</systemPath>
</dependency>
```

25) How do you build fat jar using maven?

I have a code base which I want to distribute as jar. It also have dependency on external jars, which I want to bundle in the final jar.

I heard that this can be done using maven-assembly-plug-in, but I don’t understand how. Could someone point me to some examples.

Right now, I’m using fat jar to bundle the final jar. I want to achieve the same thing using maven.

A) Add following plugin to your pom.xml The latest version can be found at https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-assembly-plugin

```
...
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-assembly-plugin</artifactId>
<version>CHOOSE LATEST VERSION HERE</version>
<configuration>
<descriptorRefs>
<descriptorRef>jar-with-dependencies</descriptorRef>
</descriptorRefs>

</configuration>
<executions>
<execution>
<id>assemble-all</id>
<phase>package</phase>
<goals>
<goal>single</goal>
</goals>
</execution>
</executions>
```

```
</plugin>
</plugins>
</build>
```
After configuring this plug-in, running mvn package will produce two jars: one containing just the project classes, and a second fat jar with all dependencies with the suffix "-jar-with-dependencies".

if you want correct classpath setup at runtime then also add following plugin

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-jar-plugin</artifactId>
<configuration>
<archive>
<manifest>
<addClasspath>true</addClasspath>
<mainClass>fully.qualified.MainClass</mainClass>
</manifest>
</archive>
</configuration>
</plugin>
```

Maven Interview Questions # 26) What is pluginManagement?

This is a snippet of my pom file.

```
....
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-dependency-plugin</artifactId>
<version>2.4</version>
<executions>
<execution>
<phase>install</phase>
```

```
<goals>
<goal>copy-dependencies</goal>
</goals>
<configuration>
......
</configuration>
</execution>
</executions>
</plugin>
</plugins>
...
```

I use it successfully with the command

```
mvn install
```

But, when I try to enclose it into the “pluginManagement” tag, the maven-dependency-plugin stops working when I launch the install goal. Why does the “pluginManagement” tag change the build behavior? Or should I use another goal or option?

A) Plugin Management is an element that is seen along side plugins. Plugin Management contains plugin elements in much the same way, except that rather than configuring plugin information for this particular project build, it is intended to configure project builds that inherit from this one. However, this only configures plugins that are actually referenced within the plugins element in the children. The children have every right to override pluginManagement definitions.

You still need to add

```
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-dependency-plugin</artifactId>
</plugin>
</plugins>
```

in your build, because pluginManagement is only a way to share the same plugin configuration across all your project modules.

Advanced Maven Interview Questions And Answers

Maven Interview Questions # 27) How do you run JUnit tests in parallel in a Maven build?

I'm using JUnit 4.4 and Maven and I have a large number of long-running integration tests.

When it comes to parallelizing test suites there are a few solutions that allow me to run each test method in a single test-class in parallel. But all of these require that I change the tests in one way or another.

I really think it would be a much cleaner solution to run X different test classes in X threads in parallel. I have hundreds of tests so I don't really care about threading individual test-classes.

Is there any way to do this?

A) From junit 4.7 it's now possible to run tests in parallel without using TestNG. Actually it has been possible since 4.6, but there are a number of fixes being made in 4.7 that will make it a viable option. You may also run parallel tests with spring.

or

Use maven plugin:

```
<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-plugin</artifactId>
```

```
<version>2.7.1</version>
<configuration>
<parallel>classes</parallel>
<threadCount>5</threadCount>
</configuration>
</plugin>
</plugins>
</build>
```

Maven Interview Questions # 28) How do you host a Maven repository on github?

I have a fork of a small open sourced library that I'm working on on github. I'd like to make it available to other developers via maven, but I don't want to run my own Nexus server, and because it's a fork I can't easily deploy it to oss.sonatype.org.

What I'd like to do is to deploy it to github so that others can access it using maven. What's the best way to do this?

A) The best solution I've been able to find consists of these steps:

Create a branch called mvn-repo to host your maven artifacts.
Use the github site-maven-plugin to push your artifacts to github.
Configure maven to use your remote mvn-repo as a maven repository.
There are several benefits to using this approach:

Maven artifacts are kept separate from your source in a separate branch called mvn-repo, much like github pages are kept in a separate branch called gh-pages (if you use github pages)
Unlike some other proposed solutions, it doesn't conflict with your gh-pages if you're using them.
Ties in naturally with the deploy target so there are no new maven commands to learn. Just use mvn deploy as you normally would
The typical way you deploy artifacts to a remote maven repo is to use mvn deploy, so let's patch into that mechanism for this solution.

Maven Interview Questions # 29) What is an uber jar**? What does an uber-jar mean and what are its features/advantages?**

A) an uber-jar is an "over-jar", one level up from a simple "jar", defined as one that contains both your package and all its dependencies in one single JAR file. The name can be thought to come from the same stable as ultrageek, superman, hyperspace, and metadata, which all have similar meanings of "beyond the normal".

The advantage is that you can distribute your uber-jar and not care at all whether or not dependencies are installed at the destination, as your uber-jar actually has no dependencies.

All the dependencies of your own stuff within the uber-jar are also within that uber-jar. As are all dependencies of those dependencies.

Maven Interview Questions # 30) What are the methods for constructing an uber jar**?**

A) ubar jar is also known as fat jar i.e. jar with dependencies.

There are three common methods for constructing an uber jar:

Unshaded: Unpack all JAR files, then repack them into a single JAR. Works with Java's default class loader. Tools maven-assembly-plugin

Shaded: Same as unshaded, but rename (i.e., "shade") all packages of all dependencies. Works with Java's default class loader. Avoids some (not all) dependency version clashes. Tools maven-shade-plugin

JAR of JARs: The final JAR file contains the other JAR files embedded within. Avoids dependency version clashes. All resource files are preserved.

Apache Maven Interview Questions For Experienced

Maven Interview Questions # 31) How do you get source JARs from Maven repository?

A) Get sources and Javadocs – When you're using Maven in an IDE you often find the need for your IDE to resolve source code and Javadocs for your library dependencies. There's an easy way to accomplish that goal.

```
mvn dependency:sources
mvn dependency:resolve -Dclassifier=javadoc
```

The first command will attempt to download source code for each of the dependencies in your pom file.

The second command will attempt to download the Javadocs.

Maven is at the mercy of the library packagers here. So some of them won't have source code packaged and many of them won't have Javadocs.

In case you have a lot of dependencies it might also be a good idea to use inclusions/exclusions to get specific artifacts, the following command will for example only download the sources for the dependency with a specific artifactId:

```
mvn dependency:sources -DincludeArtifactIds=guava
```

Maven Interview Questions # 32) Difference between maven scope compile and provided for JAR packaging?

What it difference in using maven scope compile and provided when artifact is built as a JAR? If it was WAR, then I understand – artifact would be included or not in WEB-INF/lib. But in case of the JAR it doesn't matter – dependencies aren't included. They have to be on classpath when their scope is compile or provided. I know that provided dependencies aren't transitive – but is it only one difference?

A) Compile means that you need the JAR for compiling and running the app. For a web application, as an example, the JAR will be placed in the WEB-INF/lib directory.

Provided means that you need the JAR for compiling, but at run time there is already a JAR provided by the environment so you don't need it packaged with your app. For a web app, this means that the JAR file will not be placed into the WEB-INF/lib directory.

compile – This is the default scope, used if none is specified. Compile dependencies are available in all classpaths of a project. Furthermore, those dependencies are propagated to dependent projects.

provided – This is much like compile, but indicates you expect the JDK or a container to provide the dependency at runtime. For example, when building a web application for the Java Enterprise Edition, you would set the dependency on the Servlet API and related Java EE APIs to scope provided because the web container provides those classes. This scope is only available on the compilation and test classpath, and is not transitive.

Recap:

dependencies are not transitive (as you mentioned)
provided scope is only available on the compilation and test classpath, whereas compile scope is available in all classpaths.
provided dependencies are not packaged

Maven Interview Questions # 33) What is Maven error "Failure to transfer…"?

I am trying to set up a project using Maven (m2eclipse), but I get this error in Eclipse:

Description Resource Path Location Type Could not calculate build plan: Failure to transfer org.apache.maven.plugins:maven-compiler-plugin:pom:2.0.2 from http://repo1.maven.org/maven2 was cached in the local repository, resolution will not be reattempted until the update interval of central has elapsed or updates are forced.

Original error: Could not transfer artifact org.apache.maven.plugins:maven-compiler-plugin:pom:2.0.2 from/to central (http://repo1.maven.org/maven2): No response received after 60000 ExampleProject Unknown Maven Problem

Any ideas? It would be helpful if you could show me how to check if everything is configured fine…

A) Remove all your failed downloads:

```
find ~/.m2 -name “*.lastUpdated” -exec grep -q “Could not transfer” {} \; -print -exec rm {} \;
```

For windows:

```
cd %userprofile%\.m2\repository
for /r %i in (*.lastUpdated) do del %i
```

Then rightclick on your project in eclipse and choose Maven->”Update Project …”, make sure “Update Dependencies” is checked in the resulting dialog and click OK.

Maven Interview Questions # 34) How do you specify the Java compiler version in a pom.xml file?

A) Here is the method:

```
<project>
[…]
<build>
[…]
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>(whatever version is current)</version>
<configuration>
```

```
<!– or whatever version you use –>
<source>1.7</source>
<target>1.7</target>
</configuration>
</plugin>
</plugins>
[…]
</build>
[…]
</project>
```

Maven Interview Questions # 35) Maven Run Project

Is there a Maven "phase" or "goal" to simply execute the main method of a Java class? I have a project that I'd like to test manually by simply doing something like "mvn run".

A) You can run Java classes using:

```
mvn exec:java -Dexec.mainClass="com.example.Main" [-
Dexec.args="argument1″] …
```

The invocation can be a simple as mvn exec:java if the plugin configuration is in your pom.xml. The plugin site on Mojohaus has a more detailed example.

```
<project>
<build>
<plugins>
<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>exec-maven-plugin</artifactId>
<version>1.2.1</version>
<configuration>
<mainClass>com.example.Main</mainClass>
<arguments>
<argument>argument1</argument>
```

```
</arguments>
</configuration>
</plugin>
</plugins>
</build>
</project>
```

Maven Interview Questions For DevOps

Maven Interview Questions # 36) How do you force Maven to copy dependencies into target/lib?

How do I get my project's runtime dependencies copied into the target/lib folder?

As it is right now, after mvn clean install the target folder contains only my project's jar, but none of the runtime dependencies.

A) The best approach depends on what you want to do:

If you want to bundle your dependencies into a WAR or EAR file, then simply set the packaging type of your project to EAR or WAR. Maven will bundle the dependencies into the right location.

If you want to create a JAR file that includes your code along with all your dependencies, then use the assembly plugin with the jar-with-dependencies descriptor. Maven will generate a complete JAR file with all your classes plus the classes from any dependencies.

If you want to simply pull your dependencies into the target directory interactively, then use the dependency plugin to copy your files in.

If you want to pull in the dependencies for some other type of processing, then you will probably need to generate your own plugin. There are APIs to get the list of dependencies, and their location on disk. You will have to take it from there.

or use this:

```
<project>
...
<profiles>
<profile>
<id>qa</id>
<build>
<plugins>
<plugin>
<artifactId>maven-dependency-plugin</artifactId>
<executions>
<execution>
<phase>install</phase>
<goals>
<goal>copy-dependencies</goal>
</goals>
<configuration>
<outputDirectory>${project.build.directory}/lib</outputDirectory>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</profile>
</profiles>
</project>
```

Maven Interview Questions # 37) Is there a way I can configure maven to always download sources and javadocs?

Specifying -DdownloadSources=true -DdownloadJavadocs=true everytime (which usually goes along with running mvn compile twice because I forgot the first time) becomes rather tedious.

A) Open your settings.xml file ~/.m2/settings.xml (create it if it doesn't exist). Add a section with the properties added. Then make sure the activeProfiles includes the new profile.

```
<settings>

<!– … other settings here … –>

<profiles>
<profile>
<id>downloadSources</id>
<properties>
<downloadSources>true</downloadSources>
<downloadJavadocs>true</downloadJavadocs>
</properties>
</profile>
</profiles>

<activeProfiles>
<activeProfile>downloadSources</activeProfile>
</activeProfiles>
</settings>
```

Maven Interview Questions # 38) Missing artifact com.microsoft.sqlserver:sqljdbc4:jar:4.0

I am trying to add MS SQL driver dependency in my POM.xml file and the following is the dependency.

```
<dependency>
<groupId>com.microsoft.sqlserver</groupId>
<artifactId>sqljdbc4</artifactId>
<version>4.0</version>
</dependency>
```
but I get this exception

Missing artifact com.microsoft.sqlserver:sqljdbc4:jar:4.0

A) Microsoft recently open sourced their jdbc driver.

You can now find the driver on maven central:

```
<!– https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc –>
<dependency>
<groupId>com.microsoft.sqlserver</groupId>
<artifactId>mssql-jdbc</artifactId>
<version>6.1.0.jre8</version>
</dependency>
```

or for java 7:

```
<!– https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc –>
<dependency>
<groupId>com.microsoft.sqlserver</groupId>
<artifactId>mssql-jdbc</artifactId>
<version>6.1.0.jre7</version>
</dependency>
```

Maven Interview Questions # 39) Maven: Lifecycle vs. Phase vs. Plugin vs. Goal

Relatively new developer here, even though I've been using it for a little while, I'm hoping to solidify my Maven fundamentals. Part of my problem is that I have no experience with Ant, which seems to be from where many explanations stem. I've been reading and watching tutorials, and I keep hearing the same terms:

Lifecycle
Phase
Plugin
Goal

From what I've learned, it seems that lifecycle is the broadest of the bunch, and is composed of (or completed by) phases, plugins, and/or goals.

Question: Could you provide any info on how these terms are related and the most common examples?

A) A Maven lifecycle is an (abstract) concept that covers all steps (or better: all the steps the Maven designers decided to support) that are expected to occur in a project's development lifetime. These steps (or stages) are called phases in Maven terminology.

A Maven plugin is a container for/supplier of goals. Code implemented in goals is the real workhorse. (Maven in its core itself is just managing plugins and executing goals). Each of a plugin's goals can be assigned/bound to any of the lifecycle phases.

When invoking mvn <phase> Maven passes all phases (every time) and executes all goals (supplied by plugins) that have been bound to any of the phases prior and up to (and including) the given phase. If there is a phase with no goal bound to it nothing is done. But the phase is passed nevertheless.

I.e. you can't "'insert' additional phases" into one of Maven's built-in lifecycles. They are already there, always! You could develop your own lifecycle with its own phases but that's far beyond of simply using Maven as it is.

Phases called "pre-install" or "post-package" do not exist.

Maven Interview Questions # 40) How to force the intellij idea to reread/update all dependencies specified in the pom file?

A) Press Ctrl+Shift+A to find actions, and input "reimport", you will find the "Reimport All Maven Projects".

Apache Maven Build Automation Interview Questions

41) What is the difference between Ant and Maven?

A) Ant is mainly a build tool. Maven is a project and dependencies management tool (which of course builds your project as well). Ant+Ivy is a pretty good combination if you want to avoid Maven.

42) What is the difference between Jenkins and Maven?

A) Maven is a build tool, in short a successor of ant. It helps in build and version control. However Jenkins is continuous integration system, where in maven is used for build. Jenkins can be used to automate the deployment process.

43) What are Gradle and Maven?

A) Gradle is an open-source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven for declaring the project configuration.

44) What is the use of Maven in Selenium?

A) Maven is a build automation tool which is distributed under Apache Software Foundation. It is mainly used for Java projects. It makes build consistent with another project. Maven is also used to manage the dependencies.

45) What is Maven JXR?

A) Maven JXR project (formally Java Cross Reference) is a library to analyze a set of Java source files and produces documentation in HTML format.

RELATED INTERVIEW QUESTIONS

1. VirtualBox Interview Questions
2. Laravel Interview Questions
3. Logstash Interview Questions
4. Elasticsearch Interview Questions

5. Kibana Interview Questions
6. JBehave Interview Questions
7. Openshift Interview Questions
8. Kubernetes Interview Questions
9. Nagios Interview Questions
10. Jenkins Interview Questions
11. Chef Interview Questions
12. Puppet Interview Questions
13. RPA Interview Questions And Answers
14. Demandware Interview Questions
15. Visual Studio Interview Questions
16. Vagrant Interview Questions
17. 60 Java Multiple Choice Questions
18. 40 Core Java MCQ Questions
19. Anaplan Interview Questions And Answers
20. Tableau Multiple Choice Questions
21. Python Coding Interview Questions
22. CSS3 Interview Questions
23. Linux Administrator Interview Questions
24. SQL Interview Questions
25. Hibernate Interview Questions
26. Android Interview Questions
27. Mulesoft Interview Questions
28. JSON Interview Questions
29. PeopleSoft HRMS Interview Questions
30. PeopleSoft Functional Interview Questions

Coding Compiler / March 2, 2018 / Interview Questions, Maven Interview Questions / Interview Questions, Maven, Maven Interview Questions