Top 50 Maven Interview Questions & Answers

Knowledge Powerhouse

Top 50
Maven
Interview Questions & Answers

Knowledge Powerhouse

www.KnowledgePowerhouse.com

DEDICATION

To our readers!

CONTENTS

1. **What is Maven?**
2. **What are the main features of Maven?**
3. **What areas of a Project can you manage by using Maven?**
4. **What are the main advantages of Maven?**
5. **Why do we say “Maven uses convention over configuration”?**
6. **What are the responsibilities of a Build tool like Maven?**
7. **What are the differences between Ant and Maven?**
8. **What is MOJO in Maven?**
9. **What is a Repository in Maven?**
10. **What are the different types of repositories in Maven?**
11. **What is a local repository in Maven?**
12. **What is a central repository in Maven?**
13. **What is a Remote repository in Maven?**
14. **Why we should not store jars in CVS or any other version control system instead of Maven repository?**
15. **Can anyone upload JARS or artifacts to Central Repository?**
16. **What is a POM?**
17. **What is Super POM?**
18. **What are the main required elements in POM file?**
19. **What are the phases in Build lifecycle in Maven?**
20. **What command will you use to package your Maven project?**
21. **What is the format of fully qualified artifact name of a Maven**

project?

22. What is an Archetype in Maven?

23. What is the command in Maven to generate an Archetype?

24. What are the three main build lifecycles of Maven?

25. What are the main uses of a Maven plugin?

26. How will you find the version of a plugin being used?

27. What are the different types of profile in Maven? Where will you define these profiles?

28. What are the different setting files in Maven? Where will you find these files?

29. What are the main elements we can find in settings.xml?

30. How will you check the version of Maven in your system?

31. How will you verify if Maven is installed on Windows?

32. What is a Maven artifact?

33. What are the different dependency scopes in Maven?

34. How can we exclude a dependency in Maven?

35. How Maven searches for JAR corresponding to a dependency?

36. What is a transitive dependency in Maven?

37. What are Excluded dependencies in Maven?

38. What are Optional dependencies in Maven?

39. Where will you find the class files after compiling a Maven project successfully?

40. What are the default locations for source, test and build directories in Maven?

41. What is the result of jar:jar goal in Maven?

42. How can we get the debug or error messages from the execution of

Maven?

43. What is the difference between a Release version and SNAPSHOT version in Maven?

44. How will you run test classes in Maven?

45. Sometimes Maven compiles the test classes but doesn't run them? What could be the reason for it?

46. How can we skip the running of tests in Maven?

47. Can we create our own directory structure for a project in Maven?

48. What are the differences between Gradle and Maven?

49. What is the difference between Inheritance and Multi-module in Maven?

50. What is Build portability in Maven?

ACKNOWLEDGMENTS

We thank our readers who constantly send feedback and reviews to motivate us in creating these useful books with the latest information!

INTRODUCTION

This book contains basic to expert level Maven interview questions that an interviewer asks. Each question is accompanied with an answer so that you can prepare for job interview in short time. We have compiled this list after attending dozens of technical interviews in top-notch companies like- Google, Facebook, Ebay, Amazon etc.

Often, these Maven questions and concepts are used in our daily programming work. But these are most helpful when an Interviewer is trying to test your deep knowledge of Maven.

The difficulty rating on these Questions varies from a Fresher level software programmer to a Senior software programmer.

Once you go through them in the first pass, mark the questions that you could not answer by yourself. Then, in second pass go through only the difficult questions.

After going through this book 2-3 times, you will be very well prepared to face a technical interview on Maven for an experienced programmer.

All the best!!

Maven Interview Questions

1. What is Maven?

Maven is a software project management tool. It is open source software from Apache software foundation.

It is used for building, reporting and documenting a Software project. It is mainly based on POM (Project Object Model).

2. What are the main features of Maven?

Some of the main features of Maven are:

I. **Simple**: Maven provides simple project setup that is based on best practices.

II. **Fast**: You can get a new project or module started in a few seconds in Maven.

III. **Easy to learn**: Maven usage and commands are easy to learn across all projects. Therefore ramp up time for new developers coming onto a project is very less.

IV. **Dependency management**: Maven provides superior dependency management including automatic updates and transitive dependencies.

V. **Multiple Projects**: You can easily work with multiple projects at the same time by using Maven.

VI. **Large Library**: Maven has a large and growing repository of libraries and metadata to use out of the box.

VII. **Extensible**: Maven supports the ability to easily write plugins in Java or scripting languages for extending its core functionality.

VIII. **Instant**: Maven is online and it provides instant access to new features with very less configuration.

3. What areas of a Project can you manage by using Maven?

Maven can help us manage following areas of a project:

I. Build
II. Testing
III. Release
IV. Reporting
V. Software Change Management (SCM)
VI. Documentation
VII. Distribution

4. What are the main advantages of Maven?

Maven has a long list of advantages for Software development. Some of the main advantages are:

I. **Common Project Structure**: By using Maven, every developer has a common project structure that helps in understanding the code as well as developing new features in a new project.

II. **Modular Design**: Maven promotes modular design that divides a complex project into multiple modules that are easier to manage. By using Maven, it is easier to manage multiple modules for build, test, release etc.

III. **Centralized Dependency Management**: With Maven, each developer does not have to include the jars separately in each project or module. Maven provides a centralized dependency management that can help improve efficiency of software development.

IV. **Fewer Decisions**: With Maven a developer has to make fewer decisions about things unrelated to software development work. The project structure comes ready with Maven, dependency management is a uniform approach and build/release are handled by Maven. So a developer can focus on core work of developing software.

5. Why do we say “Maven uses convention over configuration”?

Convention over configuration is a Software Design Paradigm that decreases the number of decisions made by a software developer, without losing flexibility.

In Maven, there are many conventions for setting up the project, building the artifacts, running unit tests and releasing the code. These conventions lead to common process for Software development.

In case of other tools, there are a lot of configuration options are present. But most of the time, a developer uses same set of configuration options. So it is better to make these as a default options. Maven uses default options from best practices and provides right conventions for Software development.

6. What are the responsibilities of a Build tool like Maven?

A Build tool like Maven helps us with following tasks:

I. **Source Code**: A Build tool can generate source code based on templates.

II. **Documentation**: We can get documentation files from source code by using a build tool. E.g. Javadoc

III. **Compilation**: Primary responsibility of a Build tool is to compile source code into executable code.

IV. **Packaging**: A Build tool packages compiled code into a deployable file like- jar, zip war etc.

V. **Deployment**: We can deploy the packaged code on server by using a Build tool.

7. What are the differences between Ant and Maven?

Key differences between Ant and Maven are:

I. Ant is a Java library and command line toolbox for build process. Maven is a framework for many aspects of software development like- project setup, compile, build, documentation etc.

II. Ant does not have any conventions for project structure or build processes. Maven has conventions for setting up project structure as well as for build processes.

III. Ant is based on procedural programming. We have to write code for compilation build, copy etc. Maven is based on declarative programming. We have to just configure it for our project setup and programming.

IV. Ant does not impose any lifecycle. We need to create the sequence of tasks manually. Maven has a lifecycle for software build processes. There are well-defined phases that we can use in Maven.

V. Ant scripts are not reusable in multiple projects. Maven has plugins that are reusable across multiple projects.

8. What is MOJO in Maven?

MOJO stands for Maven plain Old Java Object.

Every MOJO is an executable goal in Maven. It is like an annotated Java class. It specifies metadata about a goal like- goal name, phase of lifecycle for goal and parameters required by goal.

A Maven plugin can contain multiple MOJOs.

9. What is a Repository in Maven?

A repository is a location on file system where build artifacts, jars, dependencies and pom.xml files are stored.

10. What are the different types of repositories in Maven?

There are mainly two types of repositories in Maven:

I. **Local Repository**: This is your local folder in which a copy of your installation and dependencies is stored.

II. **Remote Repository**: This is a remote folder in which jars and other build artifacts are stored. These can be located on servers within your organization.

III. **Central Remote Repository**: This is the central Maven repository that is located on repo.maven.apache.org or uk.maven.org or any other third party location. This where we can find artifacts from different providers that are available for download and use. Like- Hibernate, Spring libraries etc.

11. What is a local repository in Maven?

Maven local repository is a folder in your local files system in which your project’s installation, dependency jars, plugins etc. are stored.

Default location of Maven local repository is .m2 folder. It can be located under following location on file system:

Windows – C:\Documents and Settings\{ username}\.m2
Unix/Linux/Mac – ~/.m2

12. What is a central repository in Maven?

Maven central repository is a truly remote repository that is located on repo.maven.apache.org or uk.maven.org or any other third party location.

This contains the jars and artifacts provided by various software providers.

Central repository contains a large amount of data. Therefore it is not allowed to scrape the whole site. But you can use the relevant jars that you want for download and use in your Maven project.

13. What is a Remote repository in Maven?

A Remote repository is a remote location on the internet where the jars and dependencies from different vendors are stored.

These files can be accessed by protocols like- file:// or http:// etc.

These can be truly remote repositories set up by third party vendors or locations inside your organization that contains the relevant jars required by your project.

14. Why we should not store jars in CVS or any other version control system instead of Maven repository?

Maven recommends storing jars in local repository instead of CVS or any other version control system. There are following advantages of storing it in Maven repo vs. CVS:

Less Storage: A repository is very large, but it takes less space because each JAR is stored only in one place. E.g. If we have 10 modules dependent on Spring jar, then they all refer to same Spring jar stored in local repository.

Quicker Checkout: Project checkout is quicker from local repository, since there is not need to checkout jars if they are already present in repo.

No need for versioning: There is no need to version JARS since external dependencies do not change so often.

15. Can anyone upload JARS or artifacts to Central Repository?

No, we need special permissions to upload JARS and artifacts to Central Maven Repository?

16. What is a POM?

POM is an abbreviation for Project Object Model. This is the basic unit of work in Maven. It is an XML file with name pom.xml.

It contains details of project and project configuration that are used by Maven to build the project.

It also contains default values for many projects. E.g. target is the name of build directory for Java Maven project.

17. What is Super POM?

Super POM is Maven’s default POM. All the POM files extend from Super POM.

18. What are the main required elements in POM file?

Every POM file should have following required elements:

I. project root
II. modelVersion
III. groupID: the id of the project's group.
IV. artifactID: the id of the artifact (project)
V. version: the version of the artifact under the specified group

19. What are the phases in Build lifecycle in Maven?

In Maven, each build lifecycle consists of many phases. Default build lifecycle has following phases:

I. **validate**: In this phase, Maven validates that the project is correct and all necessary information is available to run next phase.

II. **compile**: Maven compiles the source code of the project in this phase.

III. **test**: This is the phase to run unit tests on the compiled source. There should not be any need to package or deploy the code to run these tests.

IV. **package**: In this phase, Maven takes the compiled code and packages it in its distributable format, such as a JAR.

V. **verify**: Maven runs any checks on results of integration tests to ensure that quality criteria are met.

VI. **install**: In this phase, Maven installs the package into local repository. After this it can be used as a dependency in other projects locally.

VII. **deploy**: In the build environment, Maven copies the final package to the remote repository for sharing with other developers and projects.

20. What command will you use to package your Maven project?

To package a project into a distributable format we use following command:

mvn -package

21. What is the format of fully qualified artifact name of a Maven project?

A Maven project has artifact name with following format:

<groupId>:<artifactId>:<version>

Following is the convention used by some organizations:

Parent pom

groupId: org.Orgname.Projectname
artifactId: org.Orgname.Projectname
version: x.x.x

E.g. org.Orgname.Projectname:org.Orgname.Projectname-1.0.0.pom

Modules

groupId: org.Orgname.Projectname
artifactId: org.Orgname.Projectname.Modulename
version: x.x.x

E.g. org.Orgname.Projectname:org.Orgname.Projectname.Modulename 1.0.0.jar

22. What is an Archetype in Maven?

As per official definition, an Archetype is a Maven project templating toolkit.

By using an Archetype, an author of Archetype can create a Project template. Users of this project template (archetype) can pass different parameters to this template and start using it.

Archetype promotes consistency in the process of creating and working on a project. It also helps in reducing the ramp up time for new developers to come on board on a project.

23. What is the command in Maven to generate an Archetype?

In Maven, we can use following command to generate an Archetype:

mvn archetype:generate

24. What are the three main build lifecycles of Maven?

Maven has following three build lifecycles that further contain multiple phases:

I. **clean**: In this lifecycle any files generated by previous builds are removed.

II. **default**: This lifecycle is used for validating, compiling and creating the application. It has multiple phases like-compile, test, package inside it.

III. **site**: Maven generates and deploys the documentation of a site in this phase.

25. What are the main uses of a Maven plugin?

Maven is mainly a plugin execution framework. At the code of Maven all the work is done by plugins. A Maven plugin can be used for following purposes:

I. Cleaning up the code
II. Compiling the code
III. Creating a JAR file
IV. Deploying the artifacts
V. Running the unit tests
VI. Documenting the project
VII. Generating the site of a project
VIII. Generating a WAR file
IX. Generate a checkstyle report

26. How will you find the version of a plugin being used?

Maven Help Plugin has a describe goal. This can be used for listing the version of a plugin. Sample command for this is:

mvn -Dplugin=install help:describe

Note: In the above command replace Dplugin with the plugin prefix as the argument. Do not use the artifact ID of plugin here.

27. What are the different types of profile in Maven? Where will you define these profiles?

In Maven, we can have following types of Profile:

Per Project
It is defined in the POM itself (pom.xml).

Per User
We can define it in the Maven-settings (%USER_HOME%/.m2/settings.xml).

Global
It is defined in the global Maven-settings (${maven.home}/conf/settings.xml).

Profile descriptor
Descriptor is located in project basedir (profiles.xml) (It is not supported in Maven 3.0)

28. What are the different setting files in Maven? Where will you find these files?

Maven is very simple to use. At the core it has a setting file names settings.xml. This file contains the setting element that is used to configure the Maven with different options.

The main locations where this file can be found are:

Maven Installation directory: ${maven.home}/conf/settings.xml

User Home directory: ${user.home}/ .m2 / settings.xml

29. What are the main elements we can find in settings.xml?

In settings.xml we can have all the configuration information for Maven. Some of the important elements are:

localRepository: The value of this element is the path of this build system's local repository. The default value is ${user.home}/.m2/repository.

It is used for a main build server to allow all logged-in users to build from a common local repository.

interactiveMode: If it is true then Maven should attempt to interact with the user for input. If it is false then Maven does not interact with the user. Default setting is true.

usePluginRegistry: If it is true Maven uses the ${user.home}/.m2/plugin-registry.xml file to manage plugin versions. By defaults it is false.

offline: If it is true this build system should be able to operate in offline mode. By default it is false. This element is used for build servers that cannot connect to a remote repository due to network setup or security reasons.

30. How will you check the version of Maven in your system?

We can use following command in console to check the version of Maven in our system.

mvn -version

31. How will you verify if Maven is installed on Windows?

To check this, type mvn –version in cmd prompt of Windows. This will give you the version of Maven installed on Windows.

32. What is a Maven artifact?

A Maven artifact is a file that gets deployed to a Maven repository. In most cases it is a JAR file.

When Maven build runs, it creates one or more artifacts. In case of Java projects, it produces a compiled jar and a sources jar.

Every artifact in Maven has a group ID, an artifact ID and a version string. These three attributes uniquely identify an artifact.

In Maven, we specify a project's dependencies as artifacts.

33. What are the different dependency scopes in Maven?

Maven supports following dependency scopes:

compile: This is the default dependency scope in Maven. The compile level dependencies are available in all classpaths of a project. These dependencies are also propagated to dependent projects.

provided: This scope is similar to compile. But in this scope we expect the JDK or a container to provide the dependency at runtime. E.g. While building a web application for the Java Enterprise Edition, we can set the dependency on the Servlet API and related Java EE APIs to scope provided. The web container will provide these classes at runtime to our application.

This scope is only available on the compilation and test classpath, and is not transitive.
runtime: The dependency in this scope is not required for compilation. It is required for execution. It is available in the runtime and test classpaths. It is not present in the compile classpath.

test: This scope is used for dependencies that are required for test compilation and execution phases. This scope is not transitive.

system: This scope is same as provided scope, except that you have to provide the JAR that contains it explicitly. In this case, he artifact is always available. There is no need to look it up in a repository.

import: This scope is only used on a dependency of type pom in the <dependencyManagement> section. In this case, the specified

POM has to be replaced with the dependencies in that POM's <dependencyManagement> section. This scope is only available in Maven 2.0.9 or later.

34. How can we exclude a dependency in Maven?

To exclude a dependency we can add the <exclusions> tag under the <dependency> section of the pom.

E.g.

```
<dependencies>
   <dependency>
     <groupId>test.ProjectX</groupId>
     <artifactId>ProjectX</artifactId>
     <version>1.0</version>
     <scope>compile</scope>
     <exclusions>
       <exclusion>  <!-- exclusion is mentioned here -->
         <groupId>test.ProjectY</groupId>
         <artifactId>ProjectY</artifactId>
       </exclusion>
     </exclusions>
   </dependency>
</dependencies>
```

35. How Maven searches for JAR corresponding to a dependency?

Maven first looks for a JAR related to a dependency in the local repository. If it finds it there then it stops.

If it does not find it in local repo, it looks for the JAR in the remote repository and downloads the corresponding version of JAR file. From remote repository it stores the JAR into local repository.

36. What is a transitive dependency in Maven?

Let say you have a Project A that depends on dependency B. The dependency B further depends on dependency C. So your dependency C is a Transitive Dependency of your project A.

In Maven, starting from 2.0, you do not have to specify transitive dependencies. You just mention your immediate dependencies in pom.xml.

Maven takes care of resolving the Transitive dependencies and includes them automatically.

37. What are Excluded dependencies in Maven?

Let say a project A depends on project B, and project B depends on project C. The developers of project A can explicitly exclude project C as a dependency. We can use the "exclusion" element to exclude it.

Such dependencies are called Excluded dependencies in Maven.

38. What are Optional dependencies in Maven?

Let say a project B depends on project C. The developers of project B can mark project C as an optional dependency by using the "optional" element.

In case project A depends on project B, A will depend only on B and not on B's optional dependency C.

The developers of project A may then explicitly add a dependency on C. The dependency of B on C is known as Optional dependency in Maven.

39. Where will you find the class files after compiling a Maven project successfully?

Once Maven completes the compilation successfully, it stores the files in target folder. The default location for class files is:

${basedir}/target/classes/

40. What are the default locations for source, test and build directories in Maven?

The default locations are as follows:

Source: src/main/java
Test: src/main/test
Build: Target

41. What is the result of jar:jar goal in Maven?

In Maven, jar:jar goal creates a jar file in the Maven build directory. Jar file is create with the name format ${project.id}-${project.currentVersion}.jar.

The id and currentVersion are mentioned in the project.xml of the project being built.

jar:jar does not recompile sources. It just creates a jar from already compiled classes.

42. How can we get the debug or error messages from the execution of Maven?

At times, project build or compile fails in Maven. At this time it is very helpful to see the debug or error messages from Maven execution.

To get the debug messages we can call Maven with -X option.

To get the error/exception messages we can call Maven with -e option.

43. What is the difference between a Release version and SNAPSHOT version in Maven?

A SNAPSHOT version in Maven is the one that has not been released.

Before every release version there is a SNAPSHOT version. Before 1.0 release there will be 1.0-SNAPSHOT.

If we download 1.0-SNAPSHOT today then we may get different set of files than the one we get on downloading it yesterday. SNAPSHOT version can keep getting changes in it since it is under development.

But release version always gives exactly same set files with each download.

44. How will you run test classes in Maven?

We need Surefire plugin to run the test classes in Maven.

To run a single test we can call following command:

mvn -Dtest=TestCaseA test

We can also use patterns to run multiple test cases:

mvn -Dtest=TestCase* test

or

mvn -Dtest=TestCaseA,TestCaseB,TestImportant* test

45. Sometimes Maven compiles the test classes but doesn't run them? What could be the reason for it?

In Maven, Surefire plugin is used for running the Tests.

We can configure it to run certain test classes. Sometimes we you may have unintentionally specified an incorrect value to ${test} in settings.xml or pom.xml.

We need to look for following in pom.xml/settings.xml and fix it:

```
<properties>
   <property>
     <name>test</name>
     <value>some-value</value>
    </property>
 </properties>
```

46. How can we skip the running of tests in Maven?

We can use the parameter -Dmaven.test.skip=true or -DskipTests=true in the command line for skipping the tests.

The parameter -Dmaven.test.skip=true skips the compilation of tests.

The parameter -DskipTests=true skips the execution of tests

Surefire plugin of Maven honors these parameters.

47. Can we create our own directory structure for a project in Maven?

Yes, Maven gives us the flexibility of creating our own directory structure. We just need to configure the elements like <sourceDirectory>, <resources> etc. in the <build> section of pom.xml.

48. What are the differences between Gradle and Maven?

Gradle is nowadays getting more popular. Google uses it for Android development and release. Companies like LinkedIn also use Gradle.

Gradle is based on Domain Specific Language (DSL). Maven is based on XML.

Gradle gives more flexibility to do custom tasks similar to ANT. Maven scripts have predefined structure. So it is less flexible.

Maven is mainly used for Java based systems. Gradle is used for a variety of languages. It is a Polyglot build tool.

49. What is the difference between Inheritance and Multi-module in Maven?

In Maven, we can create a parent project that will pass its values to its children projects.

A multi-module project is created to manage a group of other sub-projects or modules. The multi-module relationship is like a tree that starts from the topmost level to the bottom level. In a multi-module project, we specify that a project should include the specific modules for build. Multi-module builds are used to group modules together in a single build.

Whereas in Inheritance, the parent-child project relationship starts from the leaf node and goes upwards. It deals more with the definition of a specific project. In this case a child’s pom is derived from its parent’s pom.

50. What is Build portability in Maven?

In Maven, the portability of a build is the measure of how easy it is to take a particular project and build it in different environments.

A build that does not require any custom configuration or customization of properties files is more portable than a build that requires a lot of custom work to build it from scratch.

Open source projects from Apache Commons are one of the most portable projects. These build can work just out of the box.

Thanks

If you enjoyed this book and gained knowledge from it in any way, then I'd like to ask you for a favor. Would you be kind enough to leave a review for this book on Amazon.com?

It'd be greatly appreciated!

REFERENCES

https://maven.apache.org/

https://gradle.org/

http://ant.apache.org/