**Maven**

Maven is a powerful build automation tool used primarily for Java projects. Understanding Maven and its functionality is crucial for any developer working in the Java ecosystem. To help prepare for Maven-related interviews, it's essential to have a strong grasp of common Maven interview questions and their corresponding answers.

**1. What is Maven?**

Maven is a Build automation tool. This tool is primarily used in Java projects.

* It is based on the concept of POM (Project Object Model).
* It manages project dependencies, compiling source code, reporting, and documentation from the Maven repository.
* It simplifies the build process by packaging the application into distributed formats like JAR or WAR files.
* Maven provides a standardized project structure and lifecycle management.

**2. What does Build Tool mean?**

A build tool is a software framework. It is used to automate the process of generating source code (if any auto-generated code is used), compiling source code, running the tests, and packaging the software components into executable formats like JAR or WAR files.

In short, it does the following tasks:

* It generates source code.
* It compiles the source code.
* It runs automated Tests.
* It packaging application into executable JAR or WAR files for deployment.
* Manages Dependencies.
* It can install the packaged code in various repositories like, Local Repository, Server Repository, and Central Repository.

**3. What is the Command Line to Check Maven Version?**

To check the Maven version from command line, type the below command on console:

mvn -version

or we can also use,

mvn -v

After executing this command, it will display the Maven Version installed in our system along with Java version and Maven home directory.

**4. Explain the Concept of POM in Maven.**

**POM** stands for **Project Object Model**.

* It is a fundamental concept in Maven.
* It is an **XML file**and it is named as **pom.xml.**
* To build the project, it contains project metadata and configuration details used by Maven.
* It defines build settings, dependencies, and plugins etc.

**5. What is a MOJO?**

MOJO stands for **Maven plain Old Java Object**.

* MOJO is a Java class and it implements executable goal within a Maven plugin.
* Plugin is a distribution of one or more than one related MOJOs.
* MOJOs define the process, that Maven can execute during the build process, that are generating and compiling source code, packaging artifacts, and running tests.

**6. What is the Difference Between Ant and Maven?**

| **Feature** | **Ant** | **Maven** |
| --- | --- | --- |
| **Tool Type** | It is a Build Tool. | It is a Project Management Tool. |
| **Build Structure** | It uses XML-based configuration to build scripts for tasks and explicit configuration. | It also uses XML-based scripts but it follows convention-over-configuration. |
| **Dependency Management** | It lacks built-in dependency management. | It automates dependency resolution from central repository. |
| **Plugin System** | It offers a basic plugin system. | It offers various plugins for different tasks. |
| **Convention** | It provides flexibility and naming convention. | It offers convention over configuration. |

**7. How to create a new Maven project using the command line?**

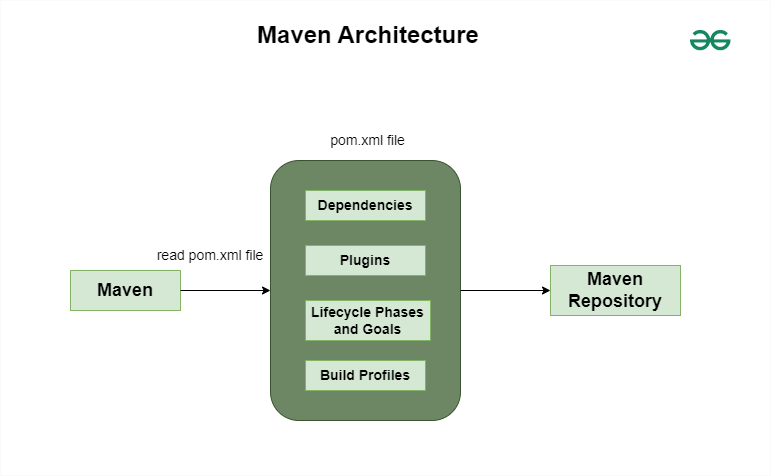
To create Maven Project using command line, we need to use the below command:

mvn archetype:generate -DgroupId=your\_group\_id -DartifactId=your\_artifact\_id -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

* After running the **mvn archetype:generate** command, we need to select an archetype for our project.
* For this, we need to provide details such as group ID, artifact ID, and version.
* Then Maven will generate the project structure based on the selections by downloading necessary dependencies.

**8. How does Maven Architecture work?**

Maven architecture works in three steps.



* **Step 1**: Read the pom.xml file.
* **Step 2**: Then it downloads the dependencies that are defined in pom.xml file into the local repository from the central repository.
* **Step 3**: After that, it creates and generates the report according to the requirement and executes the lifecycle phases, goals etc.

**9. What is Maven Repositories and define the types of Maven Repository?**

Maven repositories are the directories of packaged JAR files which contains all the metadata. These are mainly the locations where Maven retrieves dependencies for a project. The metadata refers to the POM files for each project. There are three types of Maven repository, that are:

* **Local Repository:**Local repository stores the downloaded dependencies and it is created by Maven in the local system when we run any maven command.
* **Remote Repository:**Remote repositories are accessed over the internet and it can be public or private.
* **Central Repository:**It is created by Maven Community and it is the default public repository. It contains collection of plugins and open-source libraries.

**10. What is Maven Plugins and Why Maven plugins are used?**

Maven plugins are important features of Maven. These are the extensions that provides additional functionality to the Maven build process.

* These are used to reuse the common build logic across various projects.
* The plugins allows customization and enhance the build lifecycle such as compiling code, testing, creating JAR files, packaging applications, and deploying artifacts.

We use Maven plugins to,

* Create JAR or WAR files.
* Compile Code
* Running Tests
* Packaging Application
* Create Project documentation and Reports

**11. What is Artifact in Maven?**

An Artifact refers to a deployable component of a project in Maven.

* It can be a JAR (Java Archive), WAR (Web Application Archive), or EAR (Enterprise Archive) file produced during build process along with its metadata.
* Artifact is identified by its groupId, artifactId, and version in Maven repository.
* It is used for dependency management and it is stored in Maven repositories.

**Example**:

<groupId>org.springframework.security</groupId>  
<artifactId>spring-security-web</artifactId>  
<version>6.2.4.RELEASE</version>

**12. Why is the use of the Profile Required in Maven?**

In Maven, use of the profile is required to provide portability to projects.

It defines sets of configurations or dependencies that are activated under different environments (e.g. development, testing, production).

**Example**: We can have a profile for testing dependencies which are only activated when running tests:

<profiles>  
 <profile>  
 <id>test</id>  
 <dependencies>  
 <!-- Test dependencies -->  
 </dependencies>  
 </profile>  
</profiles>

**13. What is the Command for Offline Maven Project Creation?**

The command create maven project in offline mode is:

mvn -o package

**14. What is the command to Package a Maven Project?**

To package a Maven project, the command is:

mvn package

**15. What is the Command to Install JAR Files in the Local Repository in Maven?**

To install JAR files in the local repository, the below command is used:

mvn install

To install JAR manually in the local repository, the below plugin is used:

mvn install:install-file-Dfile=<path to file>

**16. How to Run the Clean Plugin Automatically during the Build in a Maven project?**

To run the clean plugin automatically during the build process, we need to put the clean plugin inside the <**execution>** tag within the plugin configuration in the **pom.xml**file.

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-clean-plugin</artifactId>  
 <version>3.1.0</version>  
 <executions>  
 <execution>  
 <id>auto-clean</id>  
 <phase>clean</phase>  
 <goals>  
 <goal>clean</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
</build>

It binds the clean goal of the clean plugin to the clean phase.

**17. How to stop the propagation of plugins to child POMs in Maven?**

To stop the propagation of plugins to child POMs in Maven, we should set the **<inherited>** element to "false" within the **<plugin>** configuration in the parent POM file.

set <inherited> to false.

This defines that the plugin which is defined in the parent POM is not inherited by the child POMs.

**Maven Interview Questions for Experienced**

**18. What is the Maven Surefire plugin, and what is its role in the build process?**

Maven Surefire plugin is used for running unit tests written in Java at the time of build process.

* It executes test classes and also generates reports on test results.
* The Surefire plugin is useful for automating the testing phase and maintaining the code reliability in Maven projects.

For example, we have a Maven project with JUnit test cases written in Java. To execute these tests using Surefire plugin, we need to include the plugin in the pom.xml file of our project.

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>3.0.0-M5</version>  
 </plugin>  
 </plugins>  
</build>

After this we can run the below command:

mvn test

The above command will compile the project and execute all the unit tests using JUnit and the Surefire plugin will generate the reports accordingly.

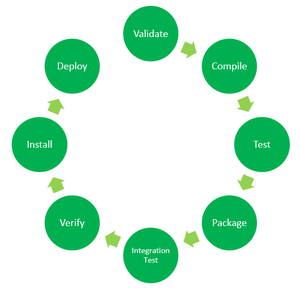
**19. What are the Three Build Life Cycles of Maven?**

The three built-in build life cycles of Maven are,

* **Clean:** It removes the files from the previous build and cleans the project.
* **Build (Default):** It handles the project deployment, compilation, packaging, and installation to the local repository.
* **Site:** It creates the project's site documentation and generates reports.

**20. What are the different Build Phases in Maven Build Lifecycle?**

The Maven Build Lifecycle has different build phases that are mentioned below:



* Validate
* Test-Compile
* Test
* Package
* Integration-test
* Verify
* Install
* Deploy

**21. What are the Maven Dependency Scopes?**

**Dependency Scope**in Maven defines the visibility and accessibility of a dependency which is already declared during different phases of the build lifecycle. Maven provides different dependency scopes to handle the use of dependencies.

* Compile Scope (default)
* Provided Scope
* Runtime Scope
* Test Scope
* System Scope
* Import Scope

**22. What are the Dependencies in Maven and how to specify them?**

In Maven, dependencies are the external libraries or modules.

* Maven project uses these dependencies to compile, build, and run successfully.
* To specify the dependencies in the project, we need to write them inside the **pom.xml**file within the **<dependencies>** tag section by providing the **group ID, artifact ID**, and **version** of each dependency.

**23. What is Dependency Exclusion in Maven?**

In Maven, Dependency Exclusion is a mechanism which is used to exclude specific transitive dependencies from the project's dependency.

* When we encounter any conflicts or any compatibility issues with dependencies, it would be useful.
* For example, if we have a dependency of older version of a library and that conflicts with the newer version required by another dependency, in this case, we can exclude the older version by using the **<exclusions>** tag in dependency declaration.

<dependency>  
 <groupId>example.group</groupId>  
 <artifactId>example-artifact</artifactId>  
 <version>1.0.0</version>  
 <exclusions>  
 <exclusion>  
 <groupId>conflicting.group</groupId>  
 <artifactId>conflicting-artifact</artifactId>  
 </exclusion>  
 </exclusions>  
</dependency>

**24. How to Exclude Dependency in Maven?**

In Maven, to exclude dependency, use the **<exclusions> tag** within the dependency declaration inside the pom.xml file.

**25. What is Archetype in Maven?**

Archetype refers to a Maven plugin.

* It creates the project structure as per its template.
* These archetypes are only the project templates that are generated by Maven at the time of any new project creation.