@thetestingacademy
PRAMOD DUTTA

Maven

1. **What is Maven?**

   Maven is a build automation tool used primarily for Java projects. It provides a standard way to manage project dependencies, build processes, and project lifecycle tasks.

2. **What are the advantages of using Maven?**

   Maven provides several benefits, including:

   Dependency management: Maven simplifies the management of project dependencies by automatically downloading and including required libraries.

   Consistent build process: Maven standardizes the build process across different projects and environments, making it easier to maintain and share code.

   Convention over configuration: Maven follows conventions and defaults, reducing the need for manual configuration and allowing developers to focus on writing code.

   Centralized repository: Maven Central Repository serves as a central repository for sharing and distributing libraries, making it easy to find and use third-party dependencies.

3. **What is a POM file?**

   POM stands for Project Object Model. It is an XML file that contains project configuration information such as project dependencies, build settings, plugin configurations, and project metadata. The POM file serves as the centrepiece of a Maven project.

   So, the pom.xml file in a Maven project is used to manage project dependencies and build settings, such as specifying external libraries (dependencies) needed by your application, configuring plugins, defining project properties, and specifying the version of Java to use for compilation.

4. **What are dependencies in Maven?**

   Dependencies are external libraries or modules required by a project to compile, build, or run. They are declared in the POM file using `<dependency>` elements, along with coordinates such as group id, artifact id, and version.

5. What are transitive dependencies in Maven?

Transitive dependencies are dependencies that are not explicitly declared in a project's `pom.xml` file but are required by other dependencies. Maven automatically resolves transitive dependencies and includes them in the project's build path. This simplifies dependency management by allowing developers to specify only direct dependencies, while Maven takes care of resolving and including transitive dependencies.

6. What is the difference between clean and install goals in Maven?

The `clean` goal removes all build artifacts and temporary files from the project, preparing it for a clean build. The `install` goal builds the project and installs the resulting artifact (JAR, WAR, etc.) into the local Maven repository for use by other projects.

7. What is a Maven archetype?

A Maven archetype is a project template or blueprint used to generate new Maven projects with predefined directory structures, configurations, and build settings. Archetypes provide a starting point for creating new projects and help maintain consistency across different projects within an organization. Maven provides several built-in archetypes for common project types, such as web applications, Java libraries, and more.

8. What is a Maven repository?

A Maven repository is a directory or server containing packaged JAR files, along with metadata such as POM files. There are two types of repositories: local repositories (stored on the local file system) and remote repositories (accessed over the network).

9. Explain the Maven lifecycle phases.

Maven follows a predefined set of lifecycle phases, which include `validate`, `compile`, `test`, `package`, `install`, and `deploy`. Each phase represents a specific stage in the build process, and plugins are bound to these phases to execute tasks.

Maven organizes the build process into lifecycles, each of which consists of a series of phases. Phases represent a stage in the lifecycle, and goals are tasks that are executed within each phase. Understanding the default lifecycles (clean, validate, compile, test, package, install, deploy)

10. How to exclude a specific dependency in pom.xml?

To exclude a specific dependency from being included in your project's build, you can use the `<exclusions>` element within the `<dependency>` declaration in your `pom.xml` file. Here's how you can do it:

```
<dependencies>
        <dependency>
                <groupId>group-id</groupId>
                <artifactId>artifact-id</artifactId>
                 <version>version</version>
        <exclusions>
        <exclusion>
                <groupId>excluded-group-id</groupId>
                <artifactId>excluded-artifact-id</artifactId>
        </exclusion>
        </exclusions>
        </dependency>
</dependencies>
```

Replace `group-id`, `artifact-id`, and `version` with the coordinates of the dependency you want to include. Then, within the `<exclusions>` element, specify the coordinates of the dependency you want to exclude by providing its `groupId` and `artifactId`.

For example, if you want to exclude a specific version of a transitive dependency that is pulled in by another dependency, you will include an `<exclusions>` block like this:

```
<dependency>
        <groupId>com.example</groupId>
         <artifactId>my-project</artifactId>
        <version>1.0.0</version>
<exclusions>
        <exclusion>
        <groupId>org.unwanted</groupId>
        <artifactId>unwanted-library</artifactId>
        </exclusion>
</exclusions>
</dependency>
```

This tells Maven to exclude the `unwanted-library` artifact from being included as a dependency of `my-project`.

11. How do you execute Maven goals from the command line?

Maven goals can be executed from the command line using the `mvn` command followed by the desired goal. For example: mvn clean install

This command executes the `clean` and `install` goals, which clean the project and install artifacts into the local Maven repository, respectively.

12. What is the Maven local repository?

The Maven local repository is a local cache where Maven stores project dependencies and artifacts downloaded from remote repositories. It is typically located in the `.m2/repository` directory in the user's home directory. Knowing how to manage and troubleshoot issues related to the local repository is important for Maven users.

13. What is a Maven profile?

A Maven profile is a set of configuration settings that can be activated or deactivated based on certain conditions or criteria. Profiles are defined in the `pom.xml` file and can be used to customize the build process, such as defining environment-specific configurations or activating specific build profiles based on properties or command-line arguments.

14. What are Maven plugins?

Maven plugins are extensions that provide additional functionality to Maven's build process. Plugins can be used to perform various tasks such as compiling code, running tests, generating documentation, and deploying artifacts. Understanding how to configure and use Maven plugins is important for customizing and extending the build process in Maven projects.

15. What is the purpose of the `<scope>` element in Maven?

The `<scope>` element in Maven is used to specify the scope of a dependency. Common values for the `<scope>` element include `compile`, `provided`, `runtime`, `test`, `system`, and `import`. Understanding the different scopes and their implications is important for managing dependencies effectively in a Maven project.

16. **What is the difference between dependencyManagement and dependencies in Maven?**

The ><dependencies section in pom.xml directly adds dependencies to the project.
The <dependencyManagement> section is used in parent POMs to specify dependency versions without including them in the child module unless explicitly referenced.

This helps maintain a consistent dependency version across multiple modules in a project.
Parent POM:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.3.9</version>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Child POM:

```
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
  </dependency>
</dependencies>
```

Even though the child module does not specify the version, it will use 5.3.9 from the parent POM.

17. **What is the difference between parent and modules in Maven?**

The parent POM is used to define shared configurations, dependency versions, and plugin settings for multiple projects.
The modules section in the parent POM is used to define multi-module projects where multiple sub-projects (modules) are built together.
Parent POM:

```
<modules>
<module>moduleA</module>
<module>moduleB</module>
</modules>
```

Each module (moduleA, moduleB) has its own pom.xml, but they share the parent's configuration.

18. How can you run a specific Maven phase without executing previous phases?

By default, Maven executes all earlier phases before executing a specific phase. However, you can use:

1. Skipping earlier phases using plugins
   To directly execute test phase (without running compile and package):
   mvn surefire:test
   To directly run package without running test:
   mvn package -DskipTests

2. Using mvn-rf: module-name (Resume from a module)
   If you have multiple modules and one module fails, you can resume from that point:
   mvn install -rf :moduleB

19. What is a Maven coordinate?

A Maven coordinate uniquely identifies a project or dependency in a repository. It consists of:

<groupId>com.example</groupId>
<artifactId>myapp</artifactId>
<version>1.0.0</version>

groupId – Organization or company name (com.example)
artifactId – Project name (myapp)
version – Version of the dependency (1.0.0)

20. How do you create a new Maven project from the command line?

mvn archetype:generate-DgroupId=com.example-DartifactId=myapp-DarchetypeArtifactId=maven-archetype-quickstart-DinteractiveMode=false

This generates a basic project structure with:
src/main/java → Source code
src/test/java → Test cases
pom.xml → Project configuration