**Selenium Maven Interview Questions:**

**What is maven?**

Maven is a project management tool. Using Maven you can Validate the code of your project, compile the code, run tests, generates the jar/war/ear files and deploys them the server.

**Can you explain maven’s life cycle?**

Maven life cycle consists the following phases.

1. Validate
2. process-resources
3. Compile
4. Test
5. Package
6. Verify
7. Install
8. Deploy

Validate – Checks if a project is correct and all information is available.

Resources – Downloads all the needed repositories and keep them in “.m2” local repository.

Compile – Compile the code (.class files will be generated)

Test – Run the unit tests/TestNG tests

Package – Bundle the package (a jar/tar/war file will be created)

Verify – Run any checks on the results of the tests to ensure quality criteria is met

Install – Generate the artefact (The result of the package) and keep it in local repository (.m2 folder)

Deploy – Upload the artefact into remote repository (Nexus)

**Can you give some maven commands?**

mvn validate – runs the validate command to check if a project is correct and all information is available

mvn process-resources – runs the command to check if the required resources are there or not. If not it will download the resources from maven repository into the local repository which is .m2 folder.

Note: maven repository is a public repository where people upload their repositories and maven will download the required repositories from there. We will discuss the repositories (dependencies and plugins) required in the maven xml section.

Maven repository: <https://mvnrepository.com/>

mvn compile – This command will compile and generate class files.

mvn test – This command will run the tests like Junit tests or TestNG tests.

mvn package – This command will generate the jar files.

mvn verify – This command verify the tests that were run and checks if the required quality criteria is met.

mvn install – This keeps the generated artefact from package command and stores it in local repository (.m2 folder)

mvn deploy – It deploys the generated jar files/war files into remote repository or server.

**What happens when we run the command “mvn test”?**

Maven works in hierarchical model. That means when you run the command “mvn test”, it runs all the previous commands as well starting from validate.

**What is pom.xml and what is it’s importance?**

Maven needs some resources for building the project and running the tests. Pom.xml provides these resources that maven needs to build and run the tests.

When you run any command like mvn deploy it searches for the pom.xml file in the local directory and runs the command.

**What is main information that we need to specify in pom.xml ?**

We need to mention two main types of information.

1. Dependency information
2. Plugin information

**Can you mention some dependencies that we use in pom.xml?**

1. Selenium dependency
2. TestNG dependency

**What plug in we use in pom.xml for running TestNG tests?**

Maven Surefire Plugin

**What are the requirements for installing Maven?**

You should have java installed on your system.

**How can you create a maven project using command prompt?**

mvn archetype:generate -DgroupId=subbu.selenium.tutorials -DartifactId=MavenSelenium -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

Explanation of the above command:

mvn archetype:generate – This will actually create a project

groupId – Unique name to the project

artifactId – Project name

archetypeArtifactId – it’s a template of the project to start with

**How can you import a project into eclipse using command prompt?**

To import a mvn project into eclipse, Eclipse looks for classpath file. So we have to create this first.

Go to the project folder Mavenjava and execute the command

mvn eclipse:eclipse

Now go to eclipse and go to File -> Import -> Maven -> Existing Maven Projects. Find the project and import it. It should import the maven project created using cmd.

You can see that groupId we gave will always be the package name and artifactId will always be the project name.

**What command you use to run tests through maven?**

mvn test

**How can you execute only test cases to a particular test in TestNG xml through Maven?**

mvn -Dtest=CheckBoxes test

**What is profiling in maven?**

If you have to run different tests each time like regression, smoke etc., create different TestNG files. But when running these from mvn define profiles and run the following commands each time.

mvn test -PRegression

mvn test -PSmoke

mvn test -Pall

P is profile in the above.

POM.xml for the above with profiles.

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>qaclickacademy</groupId>

<artifactId>Mavenjava</artifactId>

<packaging>jar</packaging>

<version>1.0-SNAPSHOT</version>

<name>Mavenjava</name>

<url>http://maven.apache.org</url>

<profiles>

<profile>

<id>Regression</id>

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.0</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testngapiumrest.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

</profile>

<profile>

<id>Smoke</id>

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.0</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testngselenium.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

</profile>

<profile>

<id>All</id>

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.0</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</pluginManagement>

</build>

</profile>

</profiles>

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.13.0</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.rest-assured</groupId>

<artifactId>rest-assured</artifactId>

<version>3.1.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.appium</groupId>

<artifactId>java-client</artifactId>

<version>6.1.0</version>

</dependency>

</dependencies>

</project>