**1. What is Maven and explain its life cycle phases?**

**Answer:** Maven is a build / project management tool, based on the concept of a project object model (POM) contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

The default life cycle comprises of the following phases:

* **Validate**- validate the project is correct and all necessary information is available
* **Compile**- compile the source code of the project
* **Test**- test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* **Package**- take the compiled code and package it in its distributed format, such as a JAR.
* **Verify**- run any checks on results of integration tests to ensure quality criteria are met
* **Install**- install the package into the local repository, for use as a dependency in other projects locally
* **Deploy**- done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

Maven provides a common platform to perform these activities which makes programmer’s life easier while handling the huge project.

Maven Build Life Cycle:

Basic maven phases are used as below.

* **clean**: deletes all artifacts and targets which are created already.
* **compile**: used to compile the source code of the project.
* **test**: test the compiled code and these tests do not require to be packaged or deployed.
* **package**: package is used to convert your project into a jar or war etc.
* **install**: install the package into the local repository for use of another project.

**2. What is Maven and its advantages of using it in your Selenium Project?**

**Answer:**

**Advantages:**

The benefits of Maven

* Better dependency management
* More powerful builds
* Better debugging
* Better collaboration
* More componentized builds
* Reduced duplication
* More consistent project structure

**Uses in Selenium:**

We can create Maven project for writing script and create dependency-using POM.xml once dependency is set Maven will download all the dependent jar files automatically and in future if any update comes from Selenium or TestNG side it will simply update all the required changes.

**3. Why maven preferred on other?**

**Answer:** The benefits of Maven

* Better dependency management
* More powerful builds
* Better debugging
* Better collaboration
* More componentized builds
* Reduced duplication
* More consistent project structure

**4. What plug-in you used in maven?**

**What are Maven Plugins?**

Maven is actually a plugin execution framework where every task is actually done by plugins. Maven Plugins are generally used to −

* create jar file
* create war file
* compile code files
* unit testing of code
* create project documentation
* create project reports

A plugin generally provides a set of goals, which can be executed using the following syntax −

**mvn [plugin-name]:[goal-name]**

For example, a Java project can be compiled with the maven-compiler-plugin's compile-goal by running the following command.

mvn compiler:compile

**Plugin Types**

Maven provided the following two types of Plugins −

**Build plugins**- They execute during the build process and should be configured in the <build/> element of pom.xml.

**Reporting plugins**-They execute during the site generation process and they should be configured in the <reporting/> element of the pom.xml.

Plugin

* **clean**-clean up after build.
* **compiler**-compiles java source code.
* **deploy**-deploys the artifact to the remote repository.
* **failsafe**-runs the JUnit integration tests in an isolated classloader.
* **install**-installs the built artifact into the local repository.
* **resources-copies**the resources to the output directory for including in the JAR.
* **site**-generates a site for the current project.
* **surefire**-runs the JUnit unit tests in an isolated classloader.
* **verifier**-verifies the existence of certain conditions. It is useful for integration tests.

**5. How do you define dependencies in your Maven Project?**

**Answer:** We define all the dependencies inside <dependencies> tag

Please refer the below sample pom.xml file for reference:

<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/xsd/maven-4.0.0.xsd">

<modelVersion>**4.0**.**0**</modelVersion>

<groupId>MyStoreProject</groupId>

<artifactId>MyStoreProject</artifactId>

<version>**0.0**.**1**-SNAPSHOT</version>

<dependencies>

<dependency>

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

<artifactId>selenium-java</artifactId>

<version>**3.141**.**59**</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>**7.1**.**0**</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>**4.1**.**2**</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>**4.1**.**2**</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>**4.0**.**9**</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>**2.6**</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>**1.2**.**17**</version>

</dependency>

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>**4.0**.**0**</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>maven-compiler-plugin</artifactId>

<version>**3.8**.**1**</version>

<configuration>

<source>**1.8**</source>

<target>**1.8**</target>

</configuration>

</plugin>

<plugin>

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

<artifactId>maven-release-plugin</artifactId>

<version>**3.0**.**0**-M1</version>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>**1.6**.**0**</version>

</plugin>

<plugin>

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

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

<version>**3.0**.**0**-M4</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>${xmlFiles}</suiteXmlFile>

</suiteXmlFiles>

<systemPropertyVariables>

<appURL>${url}</appURL>

</systemPropertyVariables>

</configuration>

</plugin>

</plugins>

</build>

</project>

**6. What is the name of maven folder which contains all the libraries?**

**Answer:**  The local repository of Maven is a folder location on the developer's machine, where all the project artifacts are stored locally. When maven build is executed, Maven automatically downloads all the dependency jars into the local repository.

Usually this folder is named .m2.

Here's where the default path to this folder is – based on OS:

* Windows: C:\Users\<User\_Name>\.m2
* Linux: /home/<User\_Name>/.m2
* Mac: /Users/<user\_name>/.m2
* And of course, for both on Linux or Mac:
* Linux/Mac: ~/.m2

**7. Without internet can we work with maven repository?**

**Answer:** You need an internet connection. Maven isn't initially self-sufficient. It needs to download a bunch of plugins along with their dependencies and the dependencies of your own project.

**8. What is the difference between Maven and TestNG?**

**Answer:** TestNG is a testing framework. It also generates testing reports.

Maven is a software project management and comprehension tool.

It manages all dependencies and different flows for building a project.

**9. in Maven, from where the jar files will get downloaded?**

**Answer:** When you add a dependency ("Dependency" here means what your project depends on e.g., if you want to use Log4j in your project, then you depend on that library) to the pom.xml, your IDE (Eclipse) will use Maven to download the jars and store them in your local repository (%HOME%/.m2 folder) so that you can compile your project and run it. Maven also allows manages transitive dependencies for you (libraries that your dependencies rely on).

**10. In Maven, do we have to manually download and configure/update the required jar files?**

**Answer:**No

**11. What are the different plugins used for Maven? And it’s use?**

**Answer:** Covered on question#4

**12. What’s the difference between a Maven project and a Java project?**

**Answer:**In Normal Java Project, if you want to work on any third party / API applications then you have to associate those jar files and associate/configure those jar files to your project manually, whereas in Maven project provide the third party/API applications dependency in POM file and then click on Maven install then automatically those respective libraries automatically to your project.

**13. In Maven what are the two setting files called and what are their location?**

**Answer:** In Maven, the setting files are called settings.xml, and the two setting files are located at

Maven installation directory: $M2\_Home/conf/settings.xml

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

**14. What is POM?**

**Answer:** POM stands for Project Object Model. In Maven, it is a fundamental Unit of Work and it is an XML file. You can find it in the base directory of the project. It consists of information about the project and various configuration details used by Maven to build the project(s).

**15. What is the command to build your Maven site?**

**Answer:** Type the command − mvn site

**16. What would the command mvn clean do?**

**Answer:**This command deletes the target directory with all the build data before starting the build process.

**17. Tell me the command to install JAR file in local repository.**

**Answer:** mvn install

**18. What is Archetype?**

**Answer:** An archetype is a Maven plugin whose task is to create a project structure as per its template.

**19. What is the command to create a new project based on an archetype?**

**Answer:** Type the following command − mvn archetype:generate

**20. What is SNAPSHOT in Maven?**

**Answer:** SNAPSHOT can be defined as a special version that indicates a current development copy.