**1. What is maven**

Maven is a project management and comprehension tool that provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

**2. Why maven is required**

Maven is chiefly used for Java-based projects, **helping to download dependencies**, which refers to the libraries or JAR files. The tool helps get the right JAR files for each project as there may be different versions of separate packages.

**3. What is compiler and surefire plugins in maven**

maven-compiler-plugin, as its name suggests, handles compiling your code. maven-surefire-plugin handles [unit] test execution and failing the build process if there are test failures.

**4. What is dependency in maven**

In Maven, a dependency is just **another archive—JAR, ZIP, and so on—which our current project needs in order to compile, build, test, and/or run**. These project dependencies are collectively specified in the pom. xml file, inside of a <dependencies> tag.

**5. What is benefits of maven usage?**

Maven **provides information present in pom**. It provides Unit test reports, list of mailing lists, dependency lists, cross-referenced sources,etc. It will manage your Selenium test project's build compilation, documentation and other related project tasks itself.

**6. What is pom.xml file in maven**

What is a POM? A Project Object Model or POM is the fundamental unit of work in Maven. It is **an XML file that contains information about the project and configuration details used by Maven to build the project**. It contains default values for most projects.

**7. What is maven life cycle?**

The default Maven lifecycle consists of 8 major steps or phases for compiling, testing, building and installing a given Java project as specified below:

1. **Validate:** This step validates if the project structure is correct. For example – It checks if all the dependencies have been downloaded and are available in the local repository.
2. **Compile:** It compiles the source code, converts the .java files to .class and stores the classes in target/classes folder.
3. **Test:** It runs unit tests for the project.
4. **Package:** This step packages the compiled code in distributable format like JAR or WAR.
5. **Integration test:** It runs the integration tests for the project.
6. **Verify:** This step runs checks to verify that the project is valid and meets the quality standards.
7. **Install:** This step installs the packaged code to the local Maven repository.
8. **Deploy:** It copies the packaged code to the remote repository for sharing it with other developers.