**What is Maven used for?**

Apache Maven is a project management tool. Maven provides a way to help with managing build life cycle of project. Maven uses standard directory layout and default build life cycle.

The most powerful feature is able to download the project dependency libraries automatically. Other features are also powerful like creating reports, checks, build and testing automation setups

Maven Repository is simply directory or folder where all jars, plugins or any projects related artifacts are available and stored for future preference. Maven searches for dependencies in the repositories.

1. Local repositories
2. Central Repositories
3. Remote Repositories

**Maven** is a build tool that manages dependencies and the application life cycle. It also had a plugins design that allows you to add other tasks to the standard compile/test/package/install/deploy tasks. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

**What is a POM.xml?**

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. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on.

When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

**Maven build life cycle**

1. **default**: the main life cycle as it’s responsible for project deployment.
2. **clean**: to clean the project and remove all files generated by the previous build.
3. **site**: to create the project’s site documentation.

A **build lifecycle** is made up of **Phases**. For example, **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 compute source code using suitable unit testing framework.

**package**- take the compiled code and package it in its distributable format such as JAR.

**verify**- run any checks on results of integration test to ensure quality criteria are met.

**install**- install the package into local repository, for use as a dependency in other project locally.

**deploy**- done in the build environment, copies final package to the remote repository for sharing with other developers