Maven is a powerful project management tool that is based on POM (project object model). It is used for project build, dependency, and documentation

**Maven Repository**

***Local repository***- A local repository is a directory on the machine of developer. This repository contains all the dependencies Maven downloads. Maven only needs to download the dependencies once, even if multiple projects depends on them (e.g. ODBC). By default, maven local repository is user\_home/m2 directory. example – C:\Users\asingh\.m2

***Central repository*** *-* The central Maven repository is created Maven community. Maven looks in this central repository for any dependencies needed but not found in your local repository. Maven then downloads these dependencies into your local repository. You can view central repository by this link.

***Remote repository***- remote repository is a repository on a web server from which Maven can download dependencies.it often used for hosting projects internal to the organization. Maven then downloads these dependencies into your local repository.

1. Maven can add all the dependencies required for the project automatically by reading pom file.
2. One can easily build their project to jar, war etc. as per their requirements using Maven.
3. Maven makes easy to start project in different environments and one doesn’t needs to handle the dependencies injection, builds, processing, etc.
4. Adding a new dependency is very easy. just write the dependency code in pom file.
5. If the maven code for an existing dependency is not available, then one cannot add that dependency using maven.
6. When there are a lot of dependencies for the project. Then it is easy to handle those dependencies using maven.
7. When dependency version update frequently. Then one has to only update version ID in pom file to update dependencies.
8. Continuous builds, integration, and testing can be easily handled by using maven.
9. When one needs an easy way to Generating documentation from the source code, Compiling source code, Packaging compiled code into JAR files or ZIP files.

***Commands to create project***

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

**General Phrases used in Maven**:

**groupId:** Generally groupId refers to domain id

**artifactId:** It is basically the name of the Jar without version(Project name).

**modelVersion**- modelversion means what version of the POM model you are using. Use version 4.0.0 for maven 2 and maven 3.

**version**- version element contains the version number of the project

**Local repository:** Maven downloads all the required dependencies and stores in the local repository called m2.

***Other Elements of Pom.xml file***

**dependencies**- dependencies element is used to defines a list of dependency of project.

**dependency**- dependency defines a dependency and used inside dependencies tag. Each dependency is described by its groupId, artifactId and version.

**name**- this element is used to give name to our maven project.

**scope**- this element used to define scope for this maven project that can be compile, runtime, test, provided system etc.

**packaging**- packaging element is used to packaging our project to output types like JAR, WAR etc.

**Build Life Cycle:**

***Basic maven phases are used as below.***

***mvn clean***: deletes all artifacts and targets which are created already.

***mvn compile***: used to compile the source code of the project.

***mvn test***: test the compiled code and these tests do not require to be packaged or deployed.

***mvn package***: package is used to convert your project into a jar or war etc.

***mvn install***: install the package into the local repository, for use as a dependency in other projects locally

***mvn validate***— mvn validatevalidate, if the project is correct

***mvn verify***— run any checks to verify the package is valid and meets quality criteria

***mvn deploy***— done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects

***Skip running tests***

* Compiles the tests, but skips running them

mvn install -DskipTests=true

***Dependency Management***

* *Check dependencies for newer versions*: mvn versions:display-dependency-updates
* *Check plugins for newer versions*: mvn versions:display-plugin-updates
* *Check for newer versions defined as properties*: mvn versions:display-property-updates
* *Display project dependencies*: mvn dependency:tree
* *Analyze project dependencies*: mvn dependency:analyze

Maven will download all the dependency jars into the local repository called .m2.