Maven

1. Maven/gradle is a Build tool.
2. Maven will help in the development activity. Starting from project template creation till the project execution in every activity Maven can be used.
3. The project template is also known as **Archetype**.
4. There are different stage in maven which helps in the development activity
5. Maven can compile you code in the **mvn compile** stage
6. Maven can help us to execute the test cases written inside project using **mvn test** stage
7. Maven can help us to create a package of the project using **mvn package** stage
8. Can execute the project using **mvn deploy** stage.
9. Maven will also help to manage the dependencies (jar files) inside project.

**Maven Setup**

1. Download Maven Zip file. (<https://maven.apache.org/download.cgi>)



1. Extract Zip file into any location (Prefer C drive)
2. Setting the environment variable for maven.
   1. **MAVEN\_HOME** : Create a New Variable and set Path of Maven extracted folder



* 1. **M2\_HOME**: Create a New Variable and set Path of Maven extracted folder



* 1. **Path**: Use Existing path variable and create new value inside Path variable

Value Must be: **%M2\_HOME%\bin**

1. Verify Maven Setup

Open a command prompt and use following command **mvn -version**



**Create Maven Project using IDE (Eclipse)**

1. “File” menu -> “New” Option -> select “Maven Project” option.
2. Keep default option as it is on the first page and click on “Next”
3. Search for the Archetype “org.apache.maven”
   1. **maven-archetype-quickstart**: This option is use to get the core java project template
   2. **maven-archetype-webapp**: This option is used to get the dynamic web application template
4. select an archetype and click on “Next”
5. Provide the following details
   1. Group Id: Project package structure
   2. Artifact Id: In the name of the project
   3. Version: Keep the default version as it is.
   4. Package: keep the group id and package name same.
6. Click on “Finish” button

**Maven Project Structure**

****

**Project Object Model file (pom.xml)**

* + - 1. This file is use to perform the maven configuration.
      2. In this file you can setup the jar details which is also known as dependencies.
      3. You can also set the properties such as Jdk Compilation and execution version and the jar version inside this.

**Update the Maven Project forcefully**

Select the project

Right click on the project -> go to “Maven” Option -> Click on “Update Project…”

Select the project from the list.

Checked the check box “Force Update of Snapshots/Release”

Click on “Ok”

**Maven Dependency management**

1. Dependency is also considered as a jar files, which are required in an application.
2. This dependencies will be configure inside maven pom.xml file. The jars will be provided by maven.
3. First Maven check for the jar file is present inside the local repository or not, if it is present then it will be directly added inside the project. But if it is not present inside local repository then it will be downloaded from the central/cloud repository downloaded inside local repo and then it will be added inside project.



**Maven Life Cycle(stages/goals)**

1. Clean
   1. In this stage the maven will clean the previously execution result and the target folder will be deleted in this stage.
2. Validate
   1. In this stage the project correctness will be check like project structure, dependencies etc.
3. Compile
   1. In this stage all the java classes will be compiled and generate a .class file.
4. Test
   1. In this stage the test cases will be executed if it is available.
5. Package
   1. In this stage the project will be bundled and converted into executable format like .jar and .war
6. Verify
   1. Maven will verify the bundle created from the previous step.
7. Install
   1. In this stage the bundle will be make ready for the execution.
8. Deploy
   1. The application can be start execution.

**Scope of Dependencies**

1. Scope is a way using which you can define the jar file availability in the maven stage execution.
2. Type of Scope
   1. Runtime
      1. This is the default scope for the jar file.
      2. If you do not provide any scope for the dependency then this will be applicable.
      3. In this scope the jar files will be available inside application through one the maven life cycle.
   2. Test
      1. In this scope the dependency will be available till the execution of Test stage/goal of the maven.
   3. Provided
      1. In this scope the jar files will be look up inside a project.
      2. Maven will not add this jar file from the remote or local repository.
   4. System
      1. This scope is use to add a jar file from the custom location.
      2. If jar file is available on any other location then it will be added using this scope.
   5. Compile
      1. In this scope the dependency will be available till the compilation stage of the maven.

**Versioning of application**

1. It is use to maintain the track of application changes.
2. Versions are mostly divided into 3 parts
   1. Major Version: Denotes the new Features
   2. Mid Version: Denotes a changes into existing feature
   3. Minor Version: Use for a defect fixes, security fix or performance fixes.

**Environment for the Development**

Developer

Local System

Dev Environment (External location where code will be deployed)

Tester

Functional Tester (STAGE, ITF environment)

Performance Tester (MTF environment)

Business Users (Pre-prod environment)

Client/User

Prod Environment