Maven

1. Maven 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

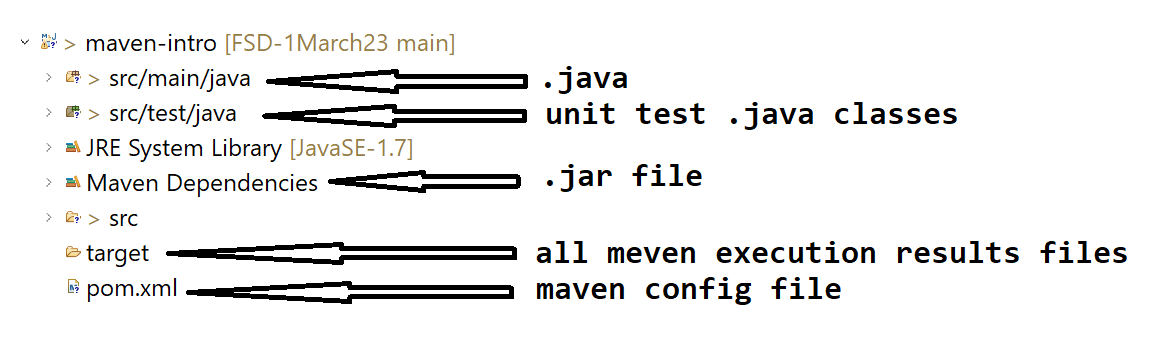
Open a command prompt and use following command **mvn -version**



**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 web application java project template
4. select an archetype and click on “Next”
5. Provide the following
   1. Group Id: Project package structure
   2. Artifact Id: Is 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**



**Apply the maven changes forcefully into Project**

* + - 1. Right Click on the project -> go to “Maven” option -> Select “Update Project….”
      2. On the new window select the checkbox for “Force Update on Snapshot/Release”
      3. Click on “Ok”

**Maven Dependency management**

Dependency is also considered as a jar file for which are required in an application.

These dependencies will be configured inside maven pom.xml file. The jars will be provided by maven.

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. Command **mvn clean**
2. Validate
   1. In this stage the project correctness will be check like project structure, dependencies etc. Command **mvn validate**
3. Compile
   1. In this stage all the java classes will be compiled and generate a .class file. Command **mvn compile**
4. Test
   1. In this stage the test cases will be executed if it is available. Command **mvn test**
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 ways to define the jar file availability in the project (Maven Lifecycle).
2. There are different scope in Maven
   1. runtime
      1. This is the default scope for the jar file if you do not specify explicitly.
      2. In this scope the jar file will be present in the application through the maven life cycle.
      3. The jar files will be available inside project for compile time and runtime as well.
   2. test
      1. In this scope the jar files will be available inside the project till the test life cycle step.
      2. Mostly in this scope the testing related jar files will be added.
   3. provided
      1. the jar file will be provided internal to project and no needs to search it inside a maven repository.
      2. These jar files are available in all the stage of the life cycle.
   4. system
      1. in this you can provide a custom path of the jar file which needs to add inside project. Here maven will not add the jar file from the maven repository.
      2. In this scope the files will be refer to given path and if it is not available then it will be a compile time error.
   5. compile
      1. In this scope the files will be available till the compilation step of the maven life cycle.

**Version in Maven**

1. Version is use to maintain the application changes.
2. Mostly the version are divided into 3 parts
   1. Major Version: Use for a new functionality.
   2. Mid Version: user for a changes into existing feature
   3. Lower/minor Version: Use for a defect fix, security fix, performance fixes

Example

3.1.2

3- Major Version

* + - 1. Mid Version
      2. Lower/minor version

**Different Environment in the development**

Developer

Local system

Dev Environment (External to your local system)

Tester

Functional tester (Stage, ITF Environment)

Performance test (MTF Environment)

Business User (Pre-Prod Environment)

Client (Prod Environment)

