**Maven**

1. Maven is a build tool.
2. Gradle is another tool which can be use for a same purpose.
3. Maven is use to create a project structure. There are multiple project structure available inside maven, this structure is also known as **archetype** (Project template).
4. Maven also help you to compile the code. (mvn compile)
5. Maven also help us to execute the test cases. (mvn test)
6. You can package your application in .jar or .war using maven package command (mvn package)
7. Can deploy the maven application which can be run further using deploy command(mvn deploy)
8. Along with this maven will also help us to manage the jar file in the project which is also known as dependencies.

**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 code 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: 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**

****

**POM.xml file**

1. This is the maven configuration file



1. POM sands for **P**roject **O**bject **M**odel



1. Using this file, you can perform following maven configuration
   1. You can set the Artifact, Group, version and packaging of the project.



* 1. Can set the Java Compilation and Execution version for this use tags

<maven.compiler.source>11</maven.compiler.source>



<maven.compiler.target>11</maven.compiler.target>

* 1. Set the dependencies
     1. In Maven dependencies are the jar file which can be manage by Maven.



* + 1. To Manage a dependencies you can use following tags

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

* + 1. You can get all the available dependencies from the maven remote/centralized repository using following URL

<https://mvnrepository.com/>

* 1. You can also add plugins, to carry out extra functionalities like generating test report, Identifying the code coverage of the application or some analysis tools etc.

**To Update Project Forcefully**

1. Right click on the project
2. Select **Maven** Option -> click on **Update Project…** option
3. Select a checkbox in new window “Force Update of Snapshot/Releases”
4. Click on **Ok** Button



**Maven Life Cycle**

1. **Clean**: in this stage the cleanup activities on target folder will be happened.
2. **Validate**: In this stage the project correctness, syntax and dependencies will be verified.
3. **Compile**: Project classes will be compiled to .class.
4. **Test**: The unit test cases will be executed.
5. **Package**: In this stage the project bundle will be created like .jar or .war.
6. **Verify**: here the bundled will be verified after packaging stage.
7. **Install**: Prepare the package for the execution.
8. **Deploy**: Executing of application.

**Maven Scope**

This is the Dependencies scope. Using scope you can specify till which maven lifecycle you need the specific dependency(jar).

There are 5 types of maven scope

1. **Compile:** the dependency added using this scope is available only till the compilation stage
2. **Runtime**: is a default scope if it is not explicitly set. These dependencies available in all the stage like code compilation and execution.
3. **Provided**: the dependencies with this scope will be search internally inside the project in jre or inside server.
4. **Test**: the dependencies available till the test stage of the application
5. **System**: the dependencies are present inside a system but not inside the project, so you can specify a path of your dependency manually.