New project

Maven section is there it mean maven is already there in ur eclipse

If not please download latest eclipse

If still its not there go to help marketplace and maven

Maven is build automation tool

Set up :

Group Id – Project name

Artifact id – package

Maven Compile—

Whatever no if java class, it will compile and compilation will happen using maven plugin compiler

Compileer plugin

Maven Test

Test cases, unit test cases or integration test cases or functional test cases selenium junit or Test NG

Surefire plugin which is responsible for testing

Maven Resource jar

Once compilation and testing done jar files is created.

JARs/Wars/EARs can be created

Maven resources plugin

At last of project this is created ,

Its complete build tool compilation teting and resoyrc generation.

ANT GRADLE MAVEN are build automation tool

MAVEN Dependencty

Can be local or centralized

Maven commands

Setup maven project

New Maven project

Give artifect and version

Files will be created

POM is heart of maven

All interaction happens throught this file only

In POM

Packaging mean which form u want to create the resources – JAR files

Add dependency

And save in fw seconds all files will be downloaded

Change the version and show that version is not available and it will fail

Do the clean install if some time its not able to download for that right click on the project run as maven build and give goal as clean install

It will be success latest download is success

Note: TEST class name should end with Test

Create the test cases and for execution there are 2 way

1. Execute as testing by right clicking
2. Right click and do the maven test

Using command line

Go to the maven director

Mvn clean install

It will execute all life cycle

Installing C:\Users\assharma\eclipse-workspace\maven\target\maven-0.0.1-SNAPSHOT.jar to C:\Users\assharma\.m2\repository\MavenDemo\maven\0.0.1-SNAPSHOT\maven-0.0.1-SNAPSHOT.jar

Is the jar files which developer used to give to tester

IF we directly want to execute the test cases instead of build

It will execute only test cases

Mvn test

Mvn package -DskipTests: no test cases will be executed only build

Mvn package -Dmaven.test.skip=true

But for above we need to add property in pom file

<properties>

<maven.test.skip>true</maven.test.skip>

<properties>

Executing multiple class

Add the testing xml file in the POM file

Run the jenkin

Configugre Jenkin🡪 plugin 🡪 search for maven

New item—Maven project

Source code management – none

|  |  |  |  |
| --- | --- | --- | --- |
| **Build** | | | |
|  | Maven Version | **Jenkins needs to know where your Maven is installed. Please do so from**[**the tool configuration**](http://localhost:8080/configureTools)**.** |  |

Select tool configuration

Seletct Maven do no select any thing else

Give the Pom file location of your computer

Goal and option give “clean install”

|  |
| --- |
| **Post-build Actions** |
| |  |  |  |  | | --- | --- | --- | --- | | **Publish TestNG Results** | | |  | |  | TestNG XML report pattern |  | [Help for feature: TestNG XML report pattern](http://localhost:8080/me/my-views/view/all/job/Mavenproject/configure) | |  | |  |  | |  |  | |  | |  |  | Advanced... |  | | Delete | | | |   Add post-build action |

Plugin is required for test ng report

* **groupId** will identify your project uniquely across all projects, so we need to enforce a naming schema. It has to follow the package name rules, what means that has to be at least as a domain name you control, and you can create as many subgroups as you want. [Look at More information about package names](http://java.sun.com/docs/books/jls/third_edition/html/packages.html#7.7).

eg. org.apache.maven, org.apache.commons

A good way to determine the granularity of the groupId is to use the project structure. That is, if the current project is a multiple module project, it should append a new identifier to the parent's groupId.

eg. org.apache.maven, org.apache.maven.plugins, org.apache.maven.reporting

* **artifactId** is the name of the jar without version. If you created it then you can choose whatever name you want with lowercase letters and no strange symbols. If it's a third party jar you have to take the name of the jar as it's distributed.

eg. maven, commons-math

* **version** if you distribute it then you can choose any typical version with numbers and dots (1.0, 1.1, 1.0.1, ...). Don't use dates as they are usually associated with SNAPSHOT (nightly) builds. If it's a third party artifact, you have to use their version number whatever it is, and as strange as it can look.

eg. 2.0, 2.0.1, 1.3.1