**MAVEN**

* Maven is a project management and comprehension tool, which is based on project object model(**POM**). It can manage project build, reporting and documentation from a central piece of information. We can build and manage java projects using maven.
* Maven follow few things called **defaults** and **convention** **over** **configuration**.
* The defaults are
* Your **source** code should be in **src/main/java**.
* Your **test** code should be in **src/test/java**.
* **Pom.xml** should be in the root folder.
* You have to run maven commands from where the pom.xml file is located . When you run maven commands, it first looks for pom.xml file. Based on that file it will build the project. If pom.xml is no there in that dir, build will be failed.
* **Pom.xml** = It is an xml file which contains information about the project and configurations used by maven to build the project. It also contains goals and plugins. While executing, maven looks for pom file in current directory and reads the file and gets needed configuration information and executes the goals as defined in the pom file.
* In pom.xml file, you will see all your configurations like **packaging**(jar,war), **group** **ID**, **artifact** **ID**, **name**, **version**, **plugins**, **dependencies etc.**
* You can add any new **dependencies**, **plugins** in pom file if you want for the project.
* **Build life cycle** :
* Maven is based around the central concept of build lifecycle. It is the sequence of the phases(stages), which define the order to execute the goals.
* The phases are:
* **Validate** = validating the information.
* **Compile** = Compiling the source code.
* **Test** = Testing the compiled source code.
* **Packaging** = Create jar/war files as mentioned in pom.xml.
* **Integrated test =** Process and deploy the package into an environment if necessary, where integrate test can run. Test results are stored in **surefire-reports** dir.
* **Verify =** Run any checkups to verify the package is valid and meets quality criteria.
* **Install** = Install the package in local/maven repo.
* **Deploy** = Copy the final package to maven repo.
* Maven follow this life cycle to build a project. If you do maven install directly, it won’t skip the above stages, it will perform all the stages above install stage and finally installs it.
* To install maven, we need java. So, Install java first, set home path for java and download maven from official website, extract and set home path.
* Test wether maven is installed (or) Not…type…**mvn –version.**
* You can use maven commands along with the stage names…**mvn** **install**, **mvn** **package** etc.
* **mvn** **clean** = Cleans(removes) target directory.
* **mvn clean install =** removes target dir and install once again newly.
* You can also deploy artifacts to containers(tomcat) directly with tomcat plugin.
* By configuring the plugin in pom.xml file. After building an artifact, maven will deployed to tomcat.

**MAVEN REPOSITORIES**

* Maven ha 3 types of repositories.
* **Local** :
* By default, maven stores all your dependencies(plugins, jars and other downloaded by maven) in a local folder. In simple words, when you build a maven project, all the dependency files will be stored in **maven** **local** **repository**.
* The default local folder in linux is **~/.m2**. In this directory, you can see all your dependencies installed by maven.
* You can change this default repo dir by adding a line in setting.xml file.
* Go to settings.xml file, add a line below local repository arrow line.

**<localRepository>/path/to/repo</localRepository>**

* **Central :**
* It is a repository which is provided by **maven** **community**. It contains large number of commonly used libraries.
* When maven doesn’t find any dependency in local repository , it searches in central repo using this url **-** [**https://repo1.maven.org/maven2/**](https://repo1.maven.org/maven2/)**.**
* **Remote:**
* If maven doesn’t find dependency in both local and central repo, it gives you an error by stoping the build. To prevent this, maven provides concept of remote repository, which is our own custom repository containing dependencies and other jars.
* We have to mention remote repo url in pom file and it will search the remote repo for dependencies while building the project.
* Maven will search for dependencies from local repo to central, if its there in central, it will download the dependency. If its not there, it will search in remote only if we configured remote repo in pom.xml file.
* Once we configured the remote repo, it will search and download the dependencies to local.

**MAVEN COMPILER PLUGIN**

* It is used to compile the sources of your project. We don’t need to define it in pom file. Maven downloads it whenever it needed.
* We can configure the plugin in pom to define the way to compile our classes.
* Compiler-plugin has two goals.
* **Compile** = compile the class under /src/main/java
* **Test**-**compile** = compile the class under /src/test/java.
* As we said before, there is no need to specify these src directories in pom, maven will take care of it. All we need to do is put the source in these directories.