**Maven**

**About maven:**

Maven uses as build tool, which manage dependencies, convention over configuration.

Maven is a project management tool mostly used for java projects

jar-java archieve, war-web archieve.

POM: Project object model, xml file which contains information regarding project and configuration details, version,(dependencies, compiler).

Version:eg.20.3.5--(major change, minor change, patch).

pom contains--metadata, dependencies, kind of project(jar, war),description.

one project--one workspace--one pom.xml file.

**1.What is Maven? And usage?**

Maven is a powerful project management tool that is based on POM (project object model).

It is used for projects build, dependency and documentation.

It simplifies the build process like ANT. But it is too much advanced than ANT.

**Usage:**

Maven can add all the dependencies required for the project automatically by reading pom file.

One can easily build their project to jar, war etc. as per their requirements using Maven.

Maven makes easy to start project in different environments and one doesn’t needs to handle the dependencies injection, builds, processing, etc.

Adding a new dependency is very easy. One has to just write the dependency code in pom file.

**2.How to create a Maven project?**

After opening Eclipse, choose the workspace you want to use.

The Eclipse window opens on the screen. Since there aren’t any projects yet, complete the following steps:

1)Go to the File option In the drop-down menu, select New Select the Project option

If you want to create a Java project, you can select the “Java Project” option. Since we are not creating a Java project specifically,

we have chosen the “Project” option.

2)The dialog box that appears on the screen will display different types of projects.

Select the Maven Project option, Click on Next, new\_project

A dialog box will appear. Select the default workspace.

Click on “Next”, Several Group IDs, Artifact IDs, and Versions will then appear.

3)Select a plugin there and click on “Next”

new-maven-project

In the next dialog box that appears, you’ll complete the following steps:

Enter the Group ID

“com.simplilearn”

Enter the Artifact ID

“mavenproject”

4)The version will appear on the screen

These items can all be modified at a later time if needed.

Click on “Finish”

maven-project

5)The project is now created.

Open the pom.xml file

You can see all the basic information that you have entered on the screen, such as the Artifact ID, Group ID, etc.

You can see the junit dependencies have been added.

This process takes place by default in Eclipse. There will also be some by default test cases.

default-test

6)There you can find AppTest.java to be a default test case.

When you click on that, you can see the test cases written in JUnit on your Eclipse screen.

package-explorer

When it comes to adding more test cases, it will depend on the user, but these test cases and commands can easily be added in the workspace.

If we try to remove certain dependencies from our file, we will receive error messages.

7)To troubleshoot this, complete the following steps:

Go to another tab: mavenproject/pom.xml

Delete any dependencies

8)Save the file

Immediately, there will be several error messages in the AppTest.java.

Return to the previous screen and undo the deletion. The errors that occurred will disappear.

The demo shows the relationship between the dependencies and the Eclipse. When a Maven project is selected,

all such dependencies are automatically downloaded. If any dependencies are not present, Eclipse will show errors.

**3.What Commands are used in Maven? Mention with each command purpose?**

**Maven Commands:**

mvn clean: Cleans the project and removes all files generated by the previous build.

mvn compile: Compiles source code of the project.

mvn test-compile: Compiles the test source code.

mvn test: Runs tests for the project.

mvn package: Creates JAR or WAR file for the project to convert it into a distributable format.

mvn install: Deploys the packaged JAR/ WAR file to the local repository.

mvn deploy: Copies the packaged JAR/ WAR file to the remote repository after compiling, running tests and building the project.

mvn clean: Cleans the project and removes all files generated by the previous build.

mvn compile:

mvn test-compile:

mvn --version Prints out the version of Maven you are running.

mvn clean

mvn package -Dmaven.test.skip=true

mvn clean package

mvn clean package -Dmaven.test.skip=true

mvn verify

mvn clean verify

mvn install

mvn install -Dmaven.test.skip=true

mvn clean install

mvn clean install -Dmaven.test.skip=true

mvn dependency:copy-dependencies

mvn clean dependency:copy-dependencies

mvn dependency:tree

mvn dependency:tree -Dverbose

mvn dependency:tree -Dincludes=com.fasterxml.jackson.core

mvn dependency:tree -Dverbose -Dincludes=com.fasterxml.jackson.core

mvn dependency:build-classpath

$ mvn clean

$ mvn compiler:compile

$ mvn compiler:testCompile

$ mvn package

$ mvn install

**4.Maven Life cycle?**

When we build a Maven project, it executes a set of clearly defined tasks based on the project pom.xml configuration and the command-line options.

This standard set of tasks creates the maven build lifecycle.

The benefit of a clearly defined lifestyle is that we have to remember only a few sets of commands to compile, build, install, and deploy our projects.

1)There are three built-in build lifecycles.

default: handles project build and deployment

clean: handles project cleaning

site: handles the creation of project site documentation.

2)Maven build lifecycle goes through a set of stages, they are called build phases.

For example, the default lifecycle is made up of the following phases.

validate, compile, test, package, verify, install, deploy

The build phases are executed sequentially. When we run a maven build command, we specify the phase to be executed.

Any maven build phases that come before the specified phase is also executed.

For example, if we run mvn package then it will execute validate, compile, test, and package phases of the project.

3)A build phase is made up of a set of goals. Maven goals represent a specific task that contributes to the building and managing of a project.

Sometimes, a maven goal is not bound to a build phase. We can execute these goals through the command line.

The syntax to execute a goal is:$ mvn plugin-prefix:goal

$ mvn plugin-group-id:plugin-artifact-id[:plugin-version]:goal