**1.What is Maven and its explain its Use.**

Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. Maven is a tool that can be used

for building and managing any Java-based project. maven make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

**Understanding the problem without Maven**

There are many problems that we face during the project development. They are discussed below:

1.Adding set of Jars in each project: In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

2.Creating the right project structure: We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

3.Building and Deploying the project: We must have to build and deploy the project so that it may work.

**Maven simplifies the above mentioned problems. It does doe’s a lot of helpful task like**

1.We can easily build a project using maven.

2.We can add jars and other dependencies of the project easily using the help of maven.

3.Maven provides project information (log document, dependency list, unit test reports etc.)

4.Maven is very helpful for a project while updating central repository of JARs and other dependencies.

5.With the help of Maven we can build any number of projects into output types like the JAR, WAR etc without doing any scripting.

6.Using maven we can easily integrate our project with source control system (such as Subversion or Git).

**Installing Maven**

1.Check your system has java installed or not if not then download and install it.

2.Set the environmental variable for java.

3.Download Maven from website: https://maven.apache.org/download.cgi

4.Unpack the zip and place it at your choice in system.

5.Add the bin directory of the created directory apache-maven to the PATH environment variable and system variable 6. Open cmd and run mvn -v command

**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. elect 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.learningmaven”

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-exploler. 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.Command used in maven? 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.

Maven Commands

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