Prasad +91-8802399967 devopstraining2015@gmail.com

MAVEN

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.

What it does?

Maven simplifies the above mentioned problems. It does mainly following tasks.

It makes a project easy to build

It provides uniform build process (maven project can be shared by all the maven projects)

It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.)

It is easy to migrate for new features of Maven

Apache Maven helps to manage

Builds

Documentation

Reporing

SCMs

Releases

Distribution

What is Build Tool

A build tool takes care of everything for building a process. It does following:

Generates source code (if auto-generated code is used)

Generates documentation from source code

Compiles source code

Packages compiled code into JAR of ZIP file

Installs the packaged code in local repository, server repository, or central repository

Difference between Ant and Maven

Ant and Maven both are build tools provided by Apache. The main purpose of these technologies is to ease the build process of a project.

There are many differences between ant and maven that are given below:

Ant

Ant doesn't has formal conventions, so we need to provide information of the project structure in build.xml file.

Ant is procedural, you need to provide information about what to do and when to do through code. You need to provide order.

There is no life cycle in Ant.

It is a tool box.

It is mainly a build tool.

It is less preferred than Maven.

The ant scripts are not reusable

Maven

Maven has a convention to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.

Maven is declarative, everything you define in the pom.xml file.

There is life cycle in Maven.

It is mainly a project management tool.

It is a framework.

The maven plugins are reusable.

It is more preferred than Ant.

Maven Repository

A maven repository is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories. There are 3 types of maven repository:

Local Repository

Central Repository

Remote Repository

Maven searches for the dependencies in the following order:

Local repository then Central repository then Remote repository.

maven repositories

1) Maven Local Repository

Maven local repository is located in your local system. It is created by the maven when you run any maven command.

By default, maven local repository is %USER_HOME%/.m2 directory. For example: C:\Users\.m2.

2) Maven Central Repository

Maven central repository is located on the web. It has been created by the apache maven community itself.

The path of central repository is: http://repo1.maven.org/maven2/.

The central repository contains a lot of common libraries that can be viewed by this url http://search.maven.org/#browse.

3) Maven Remote Repository

Maven remote repository is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

Maven pom.xml file

POM is an acronym for Project Object Model. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.Maven reads the pom.xml file, then executes the goal.

project:- It is the root element of pom.xml file.

modelVersion :-It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0.

groupId :-It is the sub element of project. It specifies the id for the project group.

artifactId :-It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.

version :-It is the sub element of project. It specifies the version of the artifact under given group.

System Requirement

Maven is Java based tool, so the very first requirement is to have JDK installed on your machine.

JDK

1.5 or above.

Memory

no minimum requirement.

Disk Space

no minimum requirement.

Operating System

no minimum requirement.

Step 1 - verify Java installation on your machine

Now open console and execute the following java command

Linux : $ java -version

Step 2: Set JAVA environment

Set the JAVA_HOME environment variable to point to the base directory location where Java is installed on your machine.

Linux: export JAVA_HOME=/usr/local/java-current

Append Java compiler location to System Path.

Linux: export PATH=$PATH:$JAVA_HOME/bin/

Step 3: Download Maven archive

Download Maven 3.3.3 from http://maven.apache.org/download.cgi

Step 4: Extract the Maven archive

Extract the archive, to the directory you wish to install Maven 3.3.3. The subdirectory apache-maven-3.3.3 will be created from the archive.

Linux: /usr/local/apache-maven

Step 5: Set Maven environment variables

Add M2_HOME, M2, MAVEN_OPTS to environment variables.

export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.3

Note: M2_HOME copy the directory of your maven folder

export M2=$M2_HOME/bin

Step 6: Add Maven bin directory location to system path

Now append M2 variable to System Path

```
export PATH=$M2:$PATH
```

Step 7: Verify Maven installation

Now open console, execute the following mvn command.

```
$ mvn --version
```

Maven Examples:

We can create a simple maven example by executing the archetype:generate command of mvn tool.

To create a simple java project using maven, you need to open command prompt and run the archetype:generate command of mvn tool.

Syntax

The syntax to generate the project architecture is given below:

```
mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=booleanValue
```

Example:1

The example to generate the project architecture is given below:

```
mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGenerator

-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Maven Web Application

We can create a simple maven web application example by executing the archetype:generate command of mvn tool.

To create a simple java project using maven, you need to open command prompt and

run the archetype:generate command of mvn tool.

Syntax

The syntax to generate the project architecture is given below:

```
mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=booleanValue
```

Example:2

The example to generate the project architecture is given below:

```
mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWeb
-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

Default life cycle Phases:

- **validate - validate the project is correct and all necessary information is available**
- **compile - compile the source code of the project**
- **test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed**
- **package - take the compiled code and package it in its distributable format, such as a JAR.**
- **integration-test - process and deploy the package if necessary into an environment where integration tests can be run**
- **verify - run any checks to verify the package is valid and meets quality criteria**
- **install - install the package into the local repository, for use as a dependency in other projects locally**
- **Deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.**