**What is Maven used for?**

Apache Maven is a project management tool. Maven provides a way to help with managing build life cycle of project. Maven uses standard directory layout and default build life cycle.

The most powerful feature is able to download the project dependency libraries automatically. Other features are also powerful like creating reports, checks, build and testing automation setups

Maven Repository is simply directory or folder where all jars, plugins or any projects related artifacts are available and stored for future preference. Maven searches for dependencies in the repositories.

1. Local repositories
2. Central Repositories
3. Remote Repositories

**Maven** is a build tool that manages dependencies and the application life cycle. It also had a plugins design that allows you to add other tasks to the standard compile/test/package/install/deploy tasks. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.

**What is a POM.xml?**

A **Project Object Model** or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project.

It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on.

When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal.

<project> it is the root element of pom.xml file.

<modelVersion> it is sub element of project. Specifies the model version.

<groupId> The group or organization that the dependency belongs to’

<artifactId> is produced or used by a project. artifacts produced by maven for project include jars, source and binary distribution and WARSs.

<version> The specific version of the library that is required’

<packaging> defines packaging type such as jar, war etc.

<dependencies> defines dependencies for the project.

<scope> defines scope for the maven project. It can be compile, provided, runtime, test and system.

[By default, all the dependencies are scoped as compile dependencies.

**Maven Plugins:** are central part of maven framework, it is used to perform specific goal.

1. **Build plugins:** these plugins are executed at the time of build. Should be declared in <build>.
2. **Reporting plugins:**  these plugins are executed at the time of site generation. Decaled in <reporting>

**[ Maven Core Plugins:**  verifier, compiler, failsafe, install, clean, site, deploy, site, surefire ]

**Maven build life cycle**

1. **default**: the main life cycle as it’s responsible for project deployment.
2. **clean**: to clean the project and remove all files generated by the previous build.
3. **site**: to create the project’s site documentation.

A **build lifecycle** is made up of **Phases**. For example, **default life cycle comprises** of the following phases.

**validate**- validate the project is correct and all necessary information is available.

**compile**- compile the source code of the project.

**test**- test the compute source code using suitable unit testing framework.

**package**- take the compiled code and package it in its distributable format such as JAR.

**verify**- run any checks on results of integration test to ensure quality criteria are met.

**install**- install the package into local repository, for use as a dependency in other project locally.

**deploy**- done in the build environment, copies final package to the remote repository for sharing with other developers

**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** | **Maven** |
| Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file. | 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. |
| Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order. | Maven is **declarative**, everything you define in the pom.xml file. |
| There is **no life cycle** in Ant. | There is **life cycle** in Maven. |
| It is **a tool** box. | It is **a framework**. |
| It is **mainly a build tool**. | It is **mainly a project management tool**. |
| The ant scripts are **not reusable**. | The maven plugins are **reusable**. |
| It is **less preferred** than Maven. | It is **more preferred** than Ant. |