**Apache Maven** is a **software project management** **tool**.

Based on the concept of a **project object model (POM)(**Describes a project - Name and Version - Artifact Type - Dependencies - Plugins – Profiles),

Maven can manage a **project's build, reporting and documentation** from a central piece of information.

**Features**

* **Simple** project setup
* **Consistent** usage across all projects
* **Dependency management**
* Able to easily **work with multiple projects** at the **same time**
* Instant access to new features with little or **no extra configuration**
* Web site or PDF generation including any **documentation**
* **Release management** and **distribution publication**

**Similar apps**: Apache ANT | Buildr | Gradle | Ivy

*JAR Java Archive | EAR Enterprise Archive | WAR Web Archive*

|  |  |
| --- | --- |
| **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 toolbox**. | 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. |

Maven directory structure

my-app

|-- pom.xml

`-- src

|-- main

| `-- java

| `-- com

| `-- mycompany

| `-- app

| `-- App.java

`-- test

`-- java

`-- com

`-- mycompany

`-- app

`-- AppTest.java

The src/main/java directory contains the project source code.

The src/test/java directory contains the test source

The pom.xml file is the project's Project Object Model, or POM.

**mvn archetype:generate**

to generate command prompt based interactive maven project.

**mvn archetype:generate -DgroupId=com.demo -DartifactId=demo -DarchetypeArtiFactId=maven-archetype-quickstart -DinteractiveMode=false**

to generate non interactive maven quick start project.

**mvn compile** = to compile the project

**mvn clean** = removes earlier compiled class, packed code, jar file etc. and remove a file i.e., target folder

**mvn test** = to run test cases (compile + test)

**mvn package** = make code ready for deployment (compile + test + package)

**mvn site** = to create documentation for project

Execute the jar file

Lifecycle

**cd target**

Phase

**java -cp demo-1.0-SNAPSHOT.jar com.demo.App**

Goal

**Lifecycle** - broadest unit

**default** - handles your project deployment, some examples are

• **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.

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

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

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

**clean** - handles project cleaning

**pre-clean** = execute processes needed prior to the actual project cleaning

**clean** = remove all files generated by the previous build

**post-clean** = execute processes needed to finalize the project cleaning

**site** - handles the creation of your project's site documentation

**pre-site** = execute processes needed prior to the actual project site generation

**site** = generate the project's site documentation

**post-site** = execute processes needed to finalize the site generation, and to prepare for site deployment

**site-deploy** = deploy the generated site documentation to the specified web server

**Phase** - A stage in the lifecycle.

**Goal** - A plugin goal represents a specific task

Maven Coordinates

• **groupId**

- The group, company, team, organization, project, or other group

- The convention for group identifiers is that they begin with the reverse domain name of the organization that creates the project

- Project groupid created by apache would be org.apache.maven

• **artifactId**

- A unique identifier under groupId that represents a single project

• **version**

- A specific release of a project.

- Projects that have been released have a fixed version identifier that refers to a specific version of the project.

- Projects undergoing active development can use a special identifier that marks a version as a SNAPSHOT

• **Packaging**

- Jar/war/ear/pom

Project Inheritance

• There are times when you want a project to inherit values from a parent POM.

• Using a tag in the project POM, it inherits from the information from the parent project’s POM.

• By default a POM inherits from Super POM

• The elements in the parent POM that are inherited by its children are: - dependencies - developers and contributors - plugin lists - report lists - plugin executions with matching ids - plugin configuration

POM Aggregation

• Aggregation (or Multi-Module) - A project with modules is known as multi-module, or aggregator project - Modules are projects that the POM lists and are executed as a group.

Maven Profiles

• Profiles in Maven supports portability between different built environment.

• Profiles are specified in pom.xml files

• A "profiles" section in the pom.xml file can contain one or more profile definitions.

• To define two profiles called "development" and "production" simply enter the following fragment:

Graphical user interface, text, application, email

Description automatically generated

To build for the production/development environment you can run:

mvn -Pproduction package | mvn -Pdevelopment package

Plugins

Maven is - at its heart - a plugin execution framework; all work is done by plugins.

**Build plugins** will be executed during the build and they should be configured in the <build/> element from the POM.

**Reporting plugins** will be executed during the site generation and they should be configured in the <reporting/> element from the POM.