**Maven’s Objectives**

Maven’s primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, there are several areas of concern that Maven attempts to deal with:

* Making the build process easy
* Providing a uniform build system
* Providing quality project information
* Providing guidelines for best practices development
* Allowing transparent migration to new features

<https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>

mvn --version

Output:**Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-05T00:30:29+05:30)**

Maven home: /Users/viveksharma/Documents/oracle/software/maven-3.6.1

Java version: 1.8.0\_211, vendor: Oracle Corporation, runtime: /Library/Java/JavaVirtualMachines/jdk1.8.0\_211.jdk/Contents/Home/jre

Default locale: en\_GB, platform encoding: UTF-8

OS name: "mac os x", version: "10.14.6", arch: "x86\_64", family: "mac"

**Creating a Project:**

**Mvn archetype:generate -DgroupId=com.myapp -DartifactId=my-app -DarchtypeArtifactId=maven-archtype-quickstart -DarchtypeVersion=1.4**

**-DinteractiveMode=false**

**Archtype:generate. : it is a goal (like task in Ant) archetype is plugin that provide the goal.**

**Build the Project:**

**Mvn package**

**Mvn package is a build phase ( you can say a step of build life cycle) .**When a phase is given, Maven will execute every phase in the sequence up to and including the one defined. For example, if we execute the *compile* phase, the phases that actually get executed are:

Validate

Generate-sources

Process-sources

Generate-resource

Process-resources

Compile

**Running Maven Tools**

**Maven Phases**

Although hardly a comprehensive list, these are the most common *default* lifecycle phases executed.

* **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.

There are two other Maven lifecycles of note beyond the *default* list above. They are

* **clean**: cleans up artifacts created by prior builds
* **site**: generates site documentation for this project

Phases are actually mapped to underlying goals. The specific goals executed per phase is dependant upon the packaging type of the project. For example, *package* executes *jar:jar* if the project type is a JAR, and *war:war* if the project type is - you guessed it - a WAR.

An interesting thing to note is that phases and goals may be executed in sequence.

1. mvn clean dependency:copy-dependencies package

This command will clean the project, copy dependencies, and package the project (executing all phases up to *package*, of course).

**Generating the Site**

1. mvn site

This phase generates a site based upon information on the project's pom. You can look at the documentation generated under target/site.

<https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>

**Introduction to the Build Lifecycle**

**Table Of Contents**

* [Build Lifecycle Basics](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Build_Lifecycle_Basics)
* [Setting Up Your Project to Use the Build Lifecycle](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Setting_Up_Your_Project_to_Use_the_Build_Lifecycle)
  + [Packaging](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Packaging)
  + [Plugins](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Plugins)
* [Lifecycle Reference](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference)
* [Built-in Lifecycle Bindings](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Built-in_Lifecycle_Bindings)

**Build Lifecycle Basics :**

There are three built-in build lifecycles: default, clean and site.

**Default**: default lifecycle handle project deployment

**Clean :** Clean lifecycle handle cleaning project artefacts

**Site**: create project site documentation.

**A Build Lifecycle is Made Up of Phases :**

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference)):

**Usual Command Line Calls**

In a development environment, use the following call to build and install artifacts into the local repository.

**mvn install**

This command executes each default life cycle phase in order (validate, compile, package, etc.), before executing install. You only need to call the last build phase to be executed, in this case, install:

In a build environment, use the following call to cleanly build and deploy artifacts into the shared repository.

**mvn clean deploy**

It will first execute clean then call deploy