**MAVEN**

**Created by**

**Saurav Kumar Jha(**[**321sauravkumar@gmail.com**](mailto:321sauravkumar@gmail.com)**)**

**Note** : **You can't sell or redistribute the document without the consent of the author**

* Maven is project management build automation tool
* It will automate the life cycle phases of the project development

For example 🡪

Build, testing , packages ,verifying, installing , deployment etc

.**Advantages**

1. Automation tool for project build or management
2. Portable (Standard project structure)
3. Maven will pull the repository from central repository and keeps the local maven repository on our system , so next time when we run the application or it needs library it will take from the local repository

Central maven repository

Finance

Telephonic

Crm

chat

Wipro group

Verizon group

Finance

chat

Suppose if need chat library and in central repository we have two group wipro group and Verizon group both have chat library now which group chat library you want to use so you need co-ordinate that you can use named groupid

Groupid domain mainly written in reverse order like for wipro.com there will be com.wipro

Now this group might have many libraries like finance,telephonic, crm and chat etc so groupid is not enough now we mention another co-ordinate Artifactid so for chat library artifactid will be chat

* Artifactid is unique name of the library within the group but in another group can have same name like in org.verizon artifactid is chat
* Now this chat library can have multiple version 1,2,3 so which version you want so we will have to also mention version for example version 3
* So with these 3 co-ordinates we can locate any library

groupid – com.trainingapps

Artifactid –empms

Version --1.0

* So with these three co-ordinates we can uniquely locate our library in your local repository also and central maven repository also
* Now since your project itself is maven project so your project itself will also have to follow the same i.e the same convention (groupid,Artifactid,version)
* Project itself can be used as a library to another project
* In maven it is convention or strictly mentioned your project should have these 3 co-ordinates

## Installation

*Maven is a Java tool, so you must have*[*Java*](https://www.oracle.com/technetwork/java/javase/downloads/index.html)*installed in order to proceed.*

First, [download Maven](https://maven.apache.org/download.html) and follow the [installation instructions](https://maven.apache.org/install.html). After that, type the following

in a terminal or in a command prompt:

mvn --version

Creating a Project

You need somewhere for your project to reside. Create a directory somewhere and start a shell in that directory. On your command line, execute the following Maven goal:

[mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false](file:///C:\Users\SAURAV\AppData\Roaming\Microsoft\Word\mvn%20archetype:generate%20-DgroupId=com.mycompany.app%20-DartifactId=my-app%20-DarchetypeArtifactId=maven-archetype-quickstart%20-DarchetypeVersion=1.4%20-DinteractiveMode=false)

there are hundreds of archetype, archetype creates project skeleton project structure that is empty project in which you can work

Scope inside dependency

There are many scope inside pom.xml file(inside dependency) mainly

Dev(compile)

This is the default scope, used if none is specified. Compile dependencies are available in all classpaths of a project. Furthermore, those dependencies are propagated to dependent projects.

Production(provided)

This scope indicates that the dependency is not required for compilation, but is for execution. It is in the runtime and test classpaths, but not the compile classpath.

test

This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases.

* Inside the <properties> </properties> we keep variables and constants

|  |  |
| --- | --- |
| **Phase life cycle**  validation  Compile  Test  Packaging(creating jars)  Verify  Install  deploy | **Goals**  Compile  Test  package  install |

Phases are run by itself we don’t need to run it we never run the phases , when we run the goal then phases run automatically

If we run the compile goal then first validation phase will run then compile phase will run

In valid phase first it will check whether the pom.xml file is valid or not

* Jars contains all the compiled classes that resides in target directory
* When we will run install goal then the jars that has been created will be installed in the local maven repository
* Goals are not fixed in maven today you have 100 goals tomorrow you can have 1000 goals coz when we add plugin to the project then it also contains goals, every plugin comes with the goals and features

Actual syntax to run

mvn plugin:goal

but for simple goal where phases and goals are simple so we can use mvn goal

* Clean goal is used to delete the target directory (command 🡪 mvn clean)
* One more thing that maven does is dependency management and dependency management is very very intelligent thing

Suppose

Suppose we install software 1 and it needs software 2 with version 5 but we have already software 2 with version 6 but we need to install with version 5 and so on…

Maven does this automatically this is called transitive dependency

So our project requires X and X requires Y so X will be there in project class path and Y will be also there in project class path if Y depends on Z library then we will get Z automatically along with the versions so maven handles this transitive dependency automatically

* Project directory is that which contains pom.xml file

**END**