# MAVEN

by Apache Foundations

## MAVEN

It is an Automation Project management tool developed by Apache software foundation.

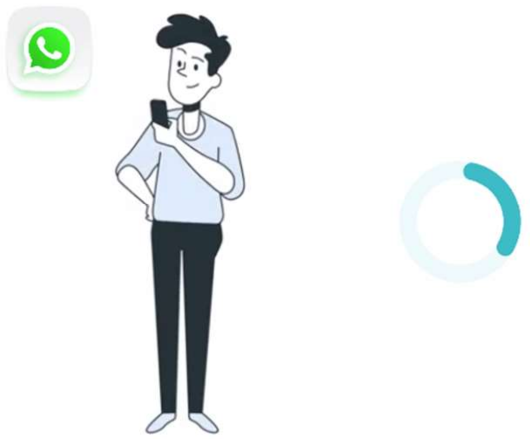
It is based on POM (Project Object Model). (POM.xml) xml: extensible Markup Language.

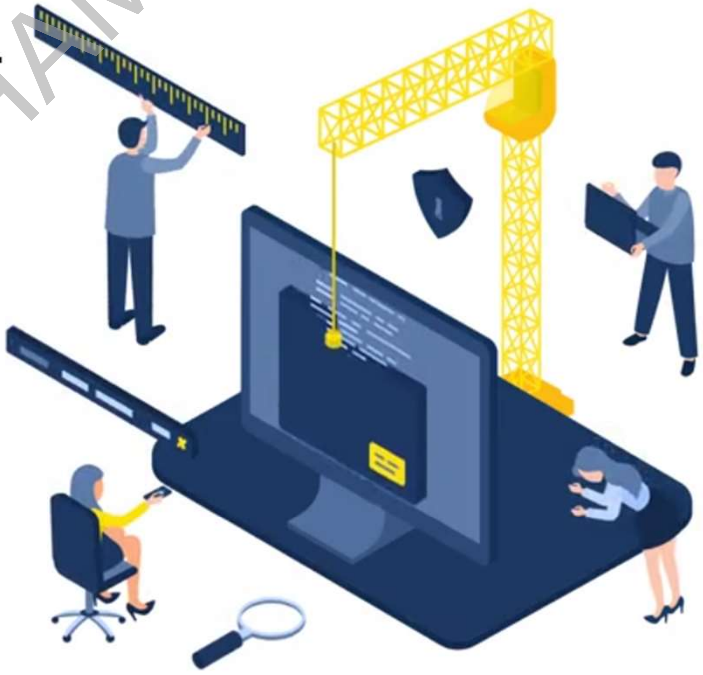
- POM is nothing but Project Object Model
- xml is extensible markup language.
- POM.xlm files consists METADATA, DEPENDENCIES, KIND OF OUTPUT, KIND OF PROJECT, DESCRIPTION.

**NOTE:** One project contains only one workspace, each workspace consists of one POM.xml files.

MAVEN is a build tool and manage dependencies.

**Manage Dependencies:**





## WHAT IS BUILD TOOL:

- it is used to set up everything which is required to run your java code This can be applied to your entire java project.
- It generates source code, compiling code, packaging code to a jar etc.
- POM refers the XML file that have all information regarding project and configuration details
- Main configuration file is in pom.xml.
- It has description of the Project details regarding version and configuration management.
- The XML file is in the Project home directory.

- Maven can build any number of projects into desired Output such as .jar, .war and metadata

  **.jar** = java archive file
  **.war** = web archive file
  **metadata** = data about data

- It is mostly used for java-based projects.
- It was initially released on 13 July 2004.
- Maven is written in java.

- Maven helps in getting the right jar file for each project as there may be different versions of separate packages.
- For downloading dependencies visit mvnrepository.com.

**Dependencies:** It refers to the java libraries that are needed for the project
**Repositories:** Refers to the directories of Packaged jar files.
**Build tools:**

| | | |
|---|---|---|
| ➢ C, C ++ | : | make file |
| ➢ .Net | : | Visual studio |
| ➢ Java | : | Ant, Maven, Gradle |

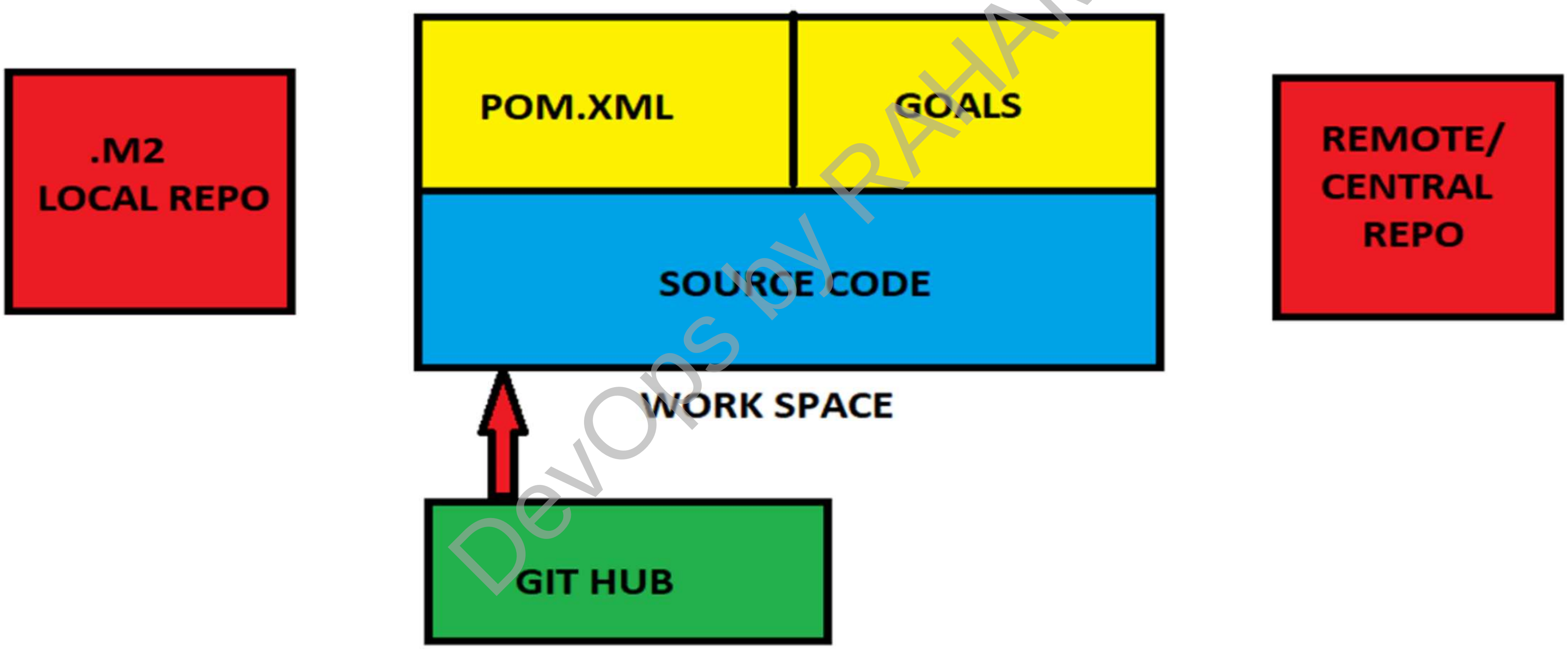
## JAVA PROJECT STRUCTURE:

- SOURCE CODE
- TEST CODE
- PROJECT STRUCTURE (ASSERTS, DIRECTORIES, RESOURCES)
- DEPENDENCIES/LIBRARY
- CONFIGURATION
- TASK RUNNER
- REPORTING

## PROBLEMS WITHOUT MAVEN:

- **Adding set of jars in each project:** In the case of Struts, Springs we need to add jar files in each project. It must include all the dependencies of jars also.
- **Creating the right project structure:** We must create the right project structure in Servlet, Struts etc. Otherwise it will not be executed.
- **Building and Deploying the Project:** We must build and deploy the Project, so it may work.

## MAVEN ARCHITECTURE:



## COMPONENTS:

- **Local Repo:** Refers to the machine of the developer where all the Project materials are saved.
- **Remote Repo**: Refers to the repository present on a web server which is used when maven needs to download dependencies.
- **Central Repo**: Refers to the maven community that comes into the action when there is a need of dependencies and those dependencies cannot be found in the Local repository.
- **GOALS:** It is nothing but a task

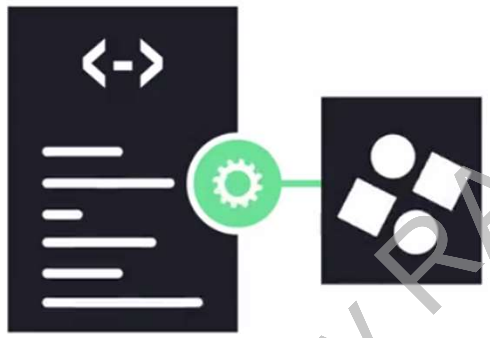
## HOW IT WORKS:

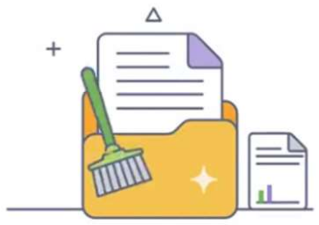
There are various components of Maven architecture – a local repository or the local machine that you work on (please refer to the diagram above). There is a central repository and then there is a remote repository or the remote web server.

So, whenever you specify any dependency in the POM.xml file, Maven will look for it in the central repository first. If the dependency is present in the central repository it will copy the same onto your local machine.

However, if that dependency is not present there, Maven will fetch it from the remote repository or remote web server using the internet. So, the internet is very much mandatory for using Maven.

## MAVEN BUILD LIFE CYCLE:





1 Generate Resource (Dependencies)

2. Compile Code = mvn compile

3. Unit Test = mvn test

4. Package (build) = mvn package

5. install (into Local repo or Artifactory) = mvn install

6. Deploy (to servers)

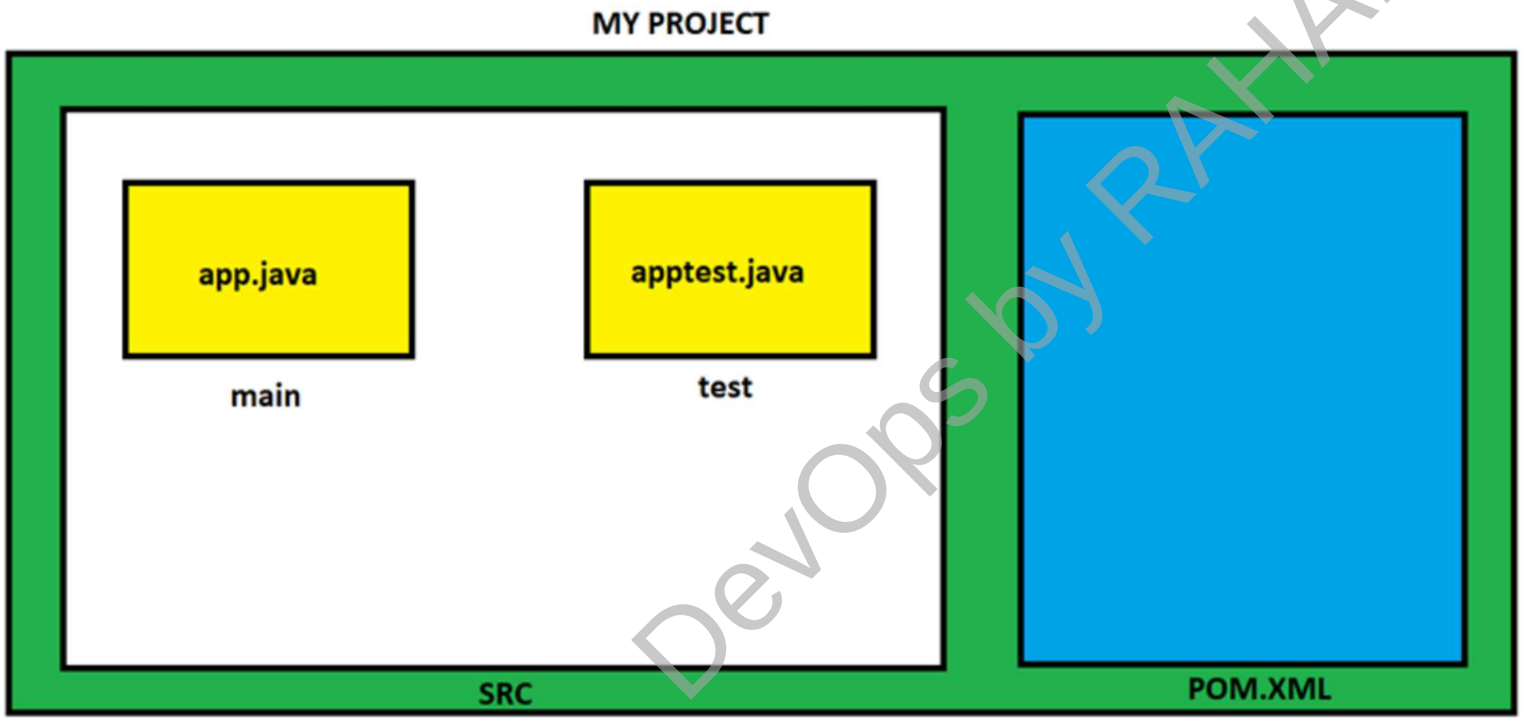
7. Clean (to delete all the runtime files) = mvn clean

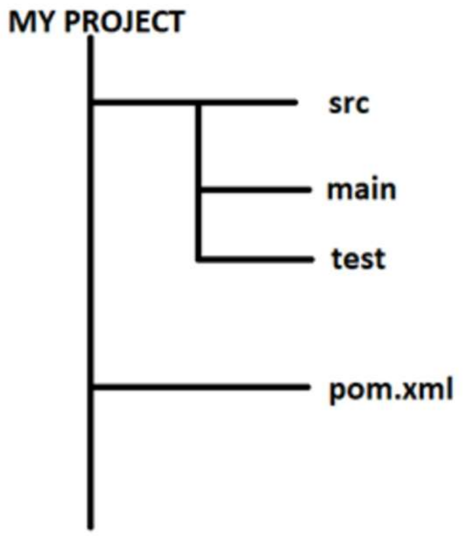
- Build lifecycle consists of a sequence of build phases & each build phase consists sequence of goals
- Each goal is responsible for a particular task. Goals are nothing but commands.
- When a phase is run, all the goals related to that phase and its plugins are also compiled.

## MAVEN VS ANT

| 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. |

## MAVEN DIRECTORY STRUCTURE:


MY PROJECT
app.java
main
apptest.java
test
SRC
POM.XML


MY PROJECT
src
main
test
pom.xml