## What is Build tool ?

It automates the creation of executable applications from the source code. Building incorporates compiling, linking and packaging the code into a usable or executable form. Build automation involves:

1. Downloading dependencies
2. Compiling source code into binary code
3. Packaging that binary code & Distributing
4. Running tests case & publish
5. Maven releases

## What is maven?

* It is build automation tool based pom.xml used primarily for java based project. It provides build lifecycle, dependencies , Release & distribution managements as well.
* To summarize, Maven simplifies and standardizes the project build process. It handles compilation, distribution, documentation, team collaboration and other tasks seamlessly. Maven increases reusability and takes care of most of the build related tasks.
* **Release management and distribution, publication**
  + Without much additional configuration, Maven will integrate with your source control system such as CVS and manage the release of a project based on a certain tag. It can also publish this to a distribution location for use by other projects. Maven is able to publish individual outputs such as a JAR, an archive including other dependencies and documentation, or as a source distribution.
* **Dependency management tool**
  + It resolves and manages the dependencies required by the application. Dependency management Maven encourages the use of a central repository of JARs and other dependencies. Maven comes with a mechanism that your project's clients can use to download any JARs required for building your project from a central JAR repository much like Perl's CPAN. This allows users of Maven to reuse JARs across projects and encourages communication between projects to ensure backward compatibility issues are dealt with.

Maven provides developers ways to manage the following

* Builds
* Documentation
* Reporting
* Dependencies
* SCMs
* Releases
* Distribution
* Mailing list

## Features of Maven

* Dependency management including automatic updating.
* Consistent usage across all projects.
* A large and growing repository of libraries.
* Extensible, with the ability to easily write plugins in Java or scripting languages.
* Instant access to new features with little or no extra configuration.
* Model-based builds − Maven is able to build any number of projects into predefined output types such as jar, war, metadata.
* Coherent site of project information − Using the same metadata as per the build process, maven is able to generate a website and a PDF including complete documentation.
* Release management and distribution publication − Without additional configuration, maven will integrate with your source control system such as CVS and manages the release of a project.
* Backward Compatibility − You can easily port the multiple modules of a project into Maven 3 from older versions of Maven. It can support the older versions also.
* Automatic parent versioning − No need to specify the parent in the sub module for maintenance.
* Parallel builds − It analyzes the project dependency graph and enables you to build schedule modules in parallel. Using this, you can achieve the performance improvements of 20-50%.
* Better Error and Integrity Reporting − Maven improved error reporting, and it provides you with a link to the Maven wiki page where you will get full description of the error.

## MAVEN VS ANT

* Maven uses Convention over Configuration, which means developers are not required to create build process themselves.
* Developers do not have to mention each and every configuration detail. Maven provides sensible default behavior for projects. When a Maven project is created, Maven creates default project structure. Developer is only required to place files accordingly and he/she need not to define any configuration in pom.xml.
* Dependency management
* In order to build the project, Maven provides developers with options to mention life-cycle goals and project dependencies (that rely on Maven plugin capabilities and on its default conventions). Much of the project management and build related tasks are maintained by Maven plugins.
* While Ant gives flexibility and requires everything to be written from scratch, Maven relies on conventions and provides predefined commands (goals).

## JAVA Application JDK VS JRE VS JVM

**JVM**

Java Virtual machine(JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for executing the java program line by line hence it is also known as interpreter.

**JRE**

Java Runtime Environment (to say JRE) is an installation package which provides environment to only run(not develop) the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

**JDK**

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development

Java Development Kit (in short JDK) is Kit which provides the environment to develop and execute(run) the Java program. JDK is a kit(or package) which includes two things

* 1. Development Tools(to provide an environment to develop your java programs)
  2. JRE (to execute your java program).

**JDK is the SDK for Java**

****

## Maven installation on WINDOWS

**Installation Steps**

Prerequesites

**Step 1: Java Installation**

Download & Install the Java JDK. [jdk-16\_windows-x64\_bin.exe](https://download.oracle.com/otn-pub/java/jdk/16+36/7863447f0ab643c585b9bdebf67c69db/jdk-16_windows-x64_bin.exe)

**Set JAVA\_HOME env variable**

Go to setings >>Edit sys Env variable>>

Sys variable : update **PATH** : C:\Program Files\Java\jdk-16\bin

User vaiable : Create JAVA\_HOME: C:\Program Files\Java\jdk-16

**Verify**

**$ java -verison**

**$ javac -verison**

**$ javac Helloworld .java >>> it compiles & give HelloWorld class files**

**Step 2: MAVEN Installation**

Download [apache-maven-3.6.3-bin.zip](https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip)

Extract & rename the folder if required & copy the folder & place it in C Drive.

* Go to setings >>Edit sys Env variable>>

Sys variable : update **PATH** : C:\Program Files\Maven-3.6.3\bin

User vaiable : Create MAVEN\_HOME: C:\Program Files\Maven-3.6.3

M2\_HOME: C:\Program Files\Maven-3.6.3

**Verify**

**$ mvn -version**

## Maven installation on Linux

**JAVA INSTALATION**

**Step 1 : Install EPEL repo & update repo**

* $ sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
* $ sudo dnf update -y

**Step 2 : Install Open JDK [JAVA]**

* $ sudo dnf install java-11-openjdk-devel -y
* $ java –version
* $ javac –version
* $ whereis java
* $ readlink -f $(which java) /opt/maven/apache-maven-3.8.1

**SET ENV Variable [JAVA\_HOME]**

* $ sudo vim /etc/profile

#JAVA PATH

export JAVA\_HOME="/usr/lib/jvm/java-11-openjdk-11.0.10.0.9-4.el8\_3.x86\_64"

export PATH=$JAVA\_HOME/bin:$PATH

* Verification >> source /etc/profile

echo $JAVA\_HOME

java –version

javac -verison

**MAVEN INSTALLATION**

**Step 1 : Download maven**

$ cd /opt/ $ mkdir maven $ cd maven

$ wget <https://mirrors.estointernet.in/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.tar.gz>

$ tar -xvf filename

**Step 2 : SET ENV Variable [MAVEN\_HOME]**

* $ sudo vim /etc/profile

#MAVEN PATH

export MAVEN\_HOME="/opt/maven/apache-maven-3.8.1"

export PATH=$MAVEN\_HOME/bin:$PATH

#MAVEN PATH

export M2\_HOME="/opt/maven/apache-maven-3.8.1"

export PATH=$M2\_HOME/bin:$PATH

* Verification >> source /etc/profile

echo $ MAVEN\_HOME

echo $ M2\_HOME

mvn –version

**MAVEN installation details**

$ ls /opt/maven -p1

LICENSE.txt contains the software license for Apache Maven

NOTICE.txt contains some notices and attributions required by libraries that Maven depends on.

README.txt contains some installation instructions

bin/ contains the mvn script that executes Maven

boot/ contains a JAR file (classwords-1.1.jar) that is responsible for creating the Class Loader in which Maven executes

conf/ contains a global settings.xml that can be used to customize the behavior of your Maven.

lib/ contains a single JAR file (maven-core-3.0.3-uber.jar) that contains the core of Maven.

**Note:**Unless you are working in a shared Unix environment, you should avoid customizing the settings.xml in M2\_HOME/conf. Altering the global settings.xml file in the Maven installation itself is usually unnecessary and it tends to complicate the upgrade procedure for Maven as you’ll have to remember to copy the customized settings.xml from the old Maven installation to the new installation. If you need to customize settings.xml, you should be editing your own settings.xml in ~/.m2/settings.xml.

**User specific configuration && repository**

Once you start using Maven extensively, you’ll notice that Maven has created some local user-specific configuration files and a local repository in your home directory. In ~/.m2 there will be **~/.m2/settings.xml.**

A file containing user-specific configuration for authentication, repositories, and other information to customize the behavior of Maven. **~/.m2/repository/** This directory contains your local Maven repository. When you download a dependency from a remote Maven repository, Maven stores a copy of the dependency in your local repository.