Maven

What is Maven?
What Maven does?
Why Maven ?
Maven Advantages
Maven Disadvantages
Core concept of Maven
    POM Files : Project object Model
    Dependencies and Repositories
Installation process of Maven
Compiling Java code from command line
Creating Java jar files from command line
Maven way of code compilation
    Setting up POM file
    Code compilation using Maven
Maven integration with Jenkins

What is Maven?

- Maven is a powerful build management tool.
- This tool primarily used to build Java projects.
- Maven can also be used to build and manage projects written in C#, Ruby, Scala and other languages.
- Maven tool is hosted and developed by Apache Software Foundation.
- Maven software tool is developed with Java.

What Maven does?

- Maven helps us to build the project.
- We can add Jars and other dependents of the project easily using Maven.
- Also, with help of Maven we can build any number of projects into output types like JAR & WAR etc.

Why Maven ?

- Maven has 70% of build tool market for Java applications right now.
    - 20% Maketshare from Gradle.
    - 10% Marketshare from ANT
- Used very common in larger companies.
- Project started in 2002 and launched in 2004.

- Above all Maven is an Open source build tool.

Maven Advantages

- Quick Project setup.
- Mature Dependent Management
- Mature Project lifecycle.
- Robust plugin Community

Maven Disadvantages

- Projects are described in XML document - which is considered as "Outdated"
- Gradle uses Groovy, which can offer more flexibility in-terms of build.

Core concept of Maven

POM Files : Project object Model

- POM files are XM file that contains information related to the project and configuration information such as dependencies, source directory, plugins etc used by Maven to build the project.
- When you execute Maven command it actually reads "pom.xml" file to accomplish its configuration.

Dependencies and Repositories

- Dependencies : External Java libraries required for Project.
- Repositories : Directories of Packaged JAR files.

Installation process of Maven

- Verify that your system has java installed or not. if not then install java.

Setting up the environment in Java - GeeksforGeeks

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented etc. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.The latest version is Java 15. Below are the environment settings for both

https://www.geeksforgeeks.org/setting-environment-java/

How to Install Java on CentOS 8

Java is one of the most popular programming languages used to build different types of applications and systems. There are two different implementations of Java, OpenJDK and Oracle Java, with almost no differences between them, except that Oracle Java has a few additional commercial features.

https://linuxize.com/post/install-java-on-centos-8/

Java installation in linux

```
yum install java-11-openjdk-devel
java --version
```

Maven installation

- Install through yum

```
yum install maven -y
```

- Install through tar & zip files.

Maven - Download Apache Maven

Apache Maven 3.6.3 is the latest release and recommended version for all users. The currently selected download mirror is https://mirrors.ocf.berkeley.edu/apache/. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are backup mirrors (at the end of the mirrors list) that should be available.

https://maven.apache.org/download.cgi

Check Maven version

```
mvn --version
```

Compiling Java code from command line

- Before seeing how Maven is going to compile and build Java code, lets try to understand how to create a code and compile it with Java itslef.
- Let's create simple classic "Hello world" Java code and compile it.

> Note : Java code should be written with .java extension

- Creating file "HelloWorld.java" under any of the directory in your machine.
- Then write some piece of Java code which could print the "Hello World"

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

- Then this code should be compiled before running (Which creates compile file)

```
javac HelloWorld.java
```

- Once after this command you might see new file which is getting created.

```
HelloWorld.class
HelloWorld.java
```

- "Helloworld.class" is file which is not human readable, It can be interpretated by your machine at run time.
- Let's execute our compiled file now.

```
  java HelloWorld
```

Creating Java jar files from command line

> What is Jar files ? (Jar - Java Archive)

- Jar is a package file format.
- Jar file have the extenison .jar which contains (libraries, resources and metadata files)
- Essentially jar is an Zip file containing compressed version of Java libraries and other related files.

> What is War files ? (War - Web Archive)

- These archive files have the .war extension and are used to package web applications.

> Generate & Execute jar file ?

- Generate

```
jar cf myjar.jar HelloWorld.class
```

- Execute

```
java -classpath myjar.jar HelloWorld
```

- You can also try moving this file to different folder and unzip it.

```
mkdir temp
mv myjar.jar temp
unzip myjar.jar
```

Maven way of code compilation

Setting up POM file

- Now let's try compiling the java code with Maven.
- To do this we might need a simple POM file.
- The file should be created in same location where your source code is located.
- The file name should be "pom.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://ma
    <modelVersion>4.0.0</modelVersion>

    <groupId>my-java-application</groupId>
    <artifactId>hello-world</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>11</java.version>
        <maven.compiler.source>${java.version}</maven.compiler.source>
        <maven.compiler.target>${java.version}</maven.compiler.target>
    </properties>

</project>
```

Elements used for Creating pom.xml file

1. **project-** It is the root element of the pom.xml file.
2. **modelVersion-** modelversion means what version of the POM model you are using. Use version 4.0.0 for maven 2 and maven 3.
3. **groupId-** groupId means the id for the project group. It is unique and Most often you will use a group ID which is similar to the root Java package name of the project like we used the groupId com.project.loggerapi.
4. **artifactId-** artifactId used to give name of the project you are building.in our example name of our project is hello-world
5. **version-** version element contains the version number of the project. If your project has been released in different versions then it is useful to give version of your project.

Code compilation using Maven

- Now our pom.xml file is ready.
- Before executing it lets clean up the environment

```
mvn clean
```

- Then lets do build the java code which is going to compile the code and give us the jar file.

```
mvn package
```

- Once after compilation you can find folder called target.
- you can find 2 files under target.

```
ls -al target

hello-world-0.0.1-SNAPSHOT.jar
maven-archiver
 - pom.properties
```

- Jar file have snapshot version added.
- maven-archiver/pom.properties will have all the details about your build

```
Generated by Maven
#Fri Mar 05 07:57:03 EST 2021
version=0.0.1-SNAPSHOT
groupId=my-java-application
artifactId=hello-world
```

- When you unzip the file you can all your java source code.

```
unzip hello-world-0.0.1-SNAPSHOT.jar
Archive:  hello-world-0.0.1-SNAPSHOT.jar
   creating: META-INF/
  inflating: META-INF/MANIFEST.MF
   creating: META-INF/maven/
   creating: META-INF/maven/my-java-application/
   creating: META-INF/maven/my-java-application/hello-world/
  inflating: META-INF/maven/my-java-application/hello-world/pom.xml
  inflating: META-INF/maven/my-java-application/hello-world/pom.properties
```

- To clean all your build you can run clean command again which will remove the artifacts.

```
mvn clean
```

Maven integration with Jenkins

Create simple standard job which can build your code and extract jar file.

```
# echo "Build jar file starting"
cd /data/Maven-workspace/java-program && /usr/bin/mvn clean && sudo /usr/bin/mvn package
```

Create pipeline job which can build and clean your java code.

```
pipeline {
  agent any
    stages {

      stage("Generic Inputs") {
        input {
          message "Select what task you want to perform ?"
          parameters {
            booleanParam(defaultValue: 'false', description: 'Build', name: 'build')
            booleanParam(defaultValue: 'false', description: 'Clean', name: 'clean')
            }
          }
          steps {
          script {
            env_build = "${build}"
            env_clean = "${clean}"
            }
          }
        }


      stage("Build") {
        steps {
          script {
            if (env_build == 'true') {
            sh '''
            echo "Build java artifact"
            cd /data/java-files && /usr/bin/mvn clean && sudo /usr/bin/mvn package
            '''
            }
          }
        }
      }

      stage("Clean") {
        steps {
          script {
            if (env_clean == 'true') {
            sh '''
            echo "Build java artifact"
            cd /data/java-files && /usr/bin/mvn clean
            '''
            }
          }
        }
      }
    }
  }
```