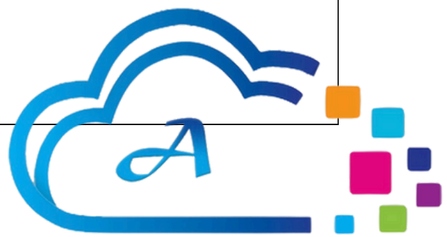
Introduction to Maven

What is Maven

- Maven is a **build automation** and **project management tool** primarily used in Java-based projects.
- It simplifies the build process by providing a standardized way to manage project dependencies, compile code, run tests, and package the application for deployment.
- Maven uses a configuration file called **pom.xml (Project Object Model)**, where you define project dependencies, plugins, and other project details.
- It automates the process of downloading necessary libraries from a central repository, making it easier to manage project dependencies.

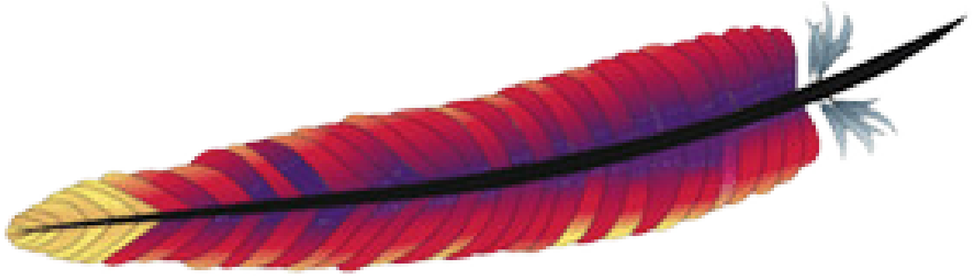


Advantages of the MVN

Its Platform Independent

Free & Open Source

- **Dependency Management:** Automatically handles the inclusion of project libraries and their transitive dependencies.
- **Build Management:** Compiles, tests, and packages the project.
- **Project Structure:** Enforces a standardized project structure.
- **Plugin Management:** Provides a wide range of plugins for various tasks like compiling code, testing, packaging, and deploying.
- **Reproducible Builds:** Ensures consistent builds across different environments.



Build tools for Prog languages

- **Java:** Maven, Gradle and Ant
- **.net :** MSBuild
- **JavaScript/TypeScript** : npm
- **Python** : setuptools, PyBuilder.
- **C/C++** : Make
- **PHP** : Composer

Understand libraries & dependencies

```java
import org.apache.commons.math3.util.ArithmeticUtils;
import org.apache.commons.math3.analysis.function.Sqrt;

public class MathCalculator {
    public static void main(String[] args) {
        // Basic arithmetic calculations
        int a = 10;
        int b = 5;
        int sum = a + b;
        int difference = a - b;
        int product = a * b;
        int quotient = a / b;

        // Print results
        System.out.println("Addition: " + a + " + " + b + " = " + sum);
        System.out.println("Subtraction: " + a + " - " + b + " = " + difference);
        System.out.println("Multiplication: " + a + " * " + b + " = " + product);
        System.out.println("Division: " + a + " / " + b + " = " + quotient);

        // Use Apache Commons Math for square root calculation
        Sqrt sqrtFunction = new Sqrt();
        double number = 25;
        double sqrtResult = sqrtFunction.value(number);

        // Print square root result
        System.out.println("Square root of " + number + " is " + sqrtResult);
    }
}
```

- **Code:** Performs basic arithmetic operations and calculates the square root of a number.
- **Library:** Apache Commons Math is used to perform the square root calculation.
- **Dependency:** Declared in pom.xml, this library provides advanced mathematical functions.

```
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Division: 10 / 5 = 2
Square root of 25.0 is 5.0
```

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>mathcalculator</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-math3</artifactId>
            <version>3.6.1</version>
        </dependency>
    </dependencies>
</project>
```

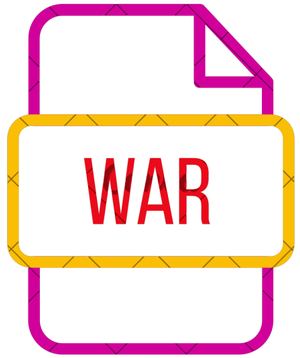
Imagine you're building a house.

- The **house represents** your software project.
- The **construction materials** (like bricks, wood, and concrete) are the libraries, dependencies, and code that you need to build the project.
- The **construction team** is your build system that puts everything together.

Now, if you're building a house without a plan or schedule, it can become chaotic. You might not know which materials to use first, when to bring in the plumbers, or how to keep everything organized.

Maven acts like your construction engineer/architect with a detailed blueprint (the pom.xml file) and a schedule.

Understand JAR & WAR



What is a .JAR File?

JAR stands for Java ARchive. It is a package file format used to bundle multiple Java class files, along with associated metadata and resources (like images, configuration files, etc.), into a single file for distribution.

- JAR includes the compiled .class files of a Java project.

What is a .WAR File?

WAR stands for Web Application Archive. It is a file used to package a Java-based web application for deployment on a web server or an application server like Apache Tomcat, JBoss, or GlassFish.

- A WAR file includes all the components of a web application, such as servlets, JavaServer Pages (JSPs), HTML files, JavaScript files, and images.

Compile .java Files to .class Files

- When you compile your **.java** files using the **javac** command, each **.java** file is converted into a corresponding **.class** file.
    - For example, **MathCalculator.java** — Compiled to **MathCalculator.class**

Combine .class Files into a .jar File

- Once you have multiple **.class** files (or even other resources like images or configuration files), you can package them into a **.jar** file.
- A **.jar (Java ARchive)** file is a compressed archive that can contain **.class** files, metadata, and other resources.

WAR typically Contains.

- Frontend Code (HTML, CSS, JavaScript)
- Backend Code (compiled .class files)
- Libraries (packaged as JAR files within WEB-INF/lib)
- Configuration Files (e.g., web.xml)
- Other Resources (e.g., images)

Typical Structure of a .war File

```
myapp.war
|-- META-INF/
|   |-- MANIFEST.MF
|-- WEB-INF/
|   |-- classes/
|   |   |-- com/
|   |       |-- example/
|   |           |-- MyServlet.class
|   |-- lib/
|   |   |-- some-library.jar
|   |-- web.xml
|-- static/
|   |-- css/
|   |-- js/
|   |-- images/
|-- index.html
```

Let's get started!

Are you ready?

Install Maven in Amazon Linux 2023

Install java

sudo dnf update -y

sudo dnf install java-17-amazon-corretto -y

Install Maven

wget https://dlcdn.apache.org/maven/maven-3/3.9.4/binaries/apache-maven-3.9.4-bin.tar.gz

tar -xvzf apache-maven-3.9.4-bin.tar.gz

sudo mv apache-maven-3.9.4 /opt/maven

Set Up Environment Variables

vim ~/.bashrc. and add below lines

export M2_HOME=/opt/maven

export PATH=$M2_HOME/bin:$PATH

source ~/.bashrc

Verify Installation : mvn -version

Maven Goals

1. validate → 2. compile → 3. test-compile → 4. test →
5. package → 6. verify → 7. install → 8. deploy

Maven Goals

validate : Validates that the project is correct and all necessary information is available. This might include checking if all the required properties are set and if the project's structure is correct.

compile: Compiles the source code of the project. This phase takes the source code from the src/main/java directory and compiles it into bytecode, usually placing the output in the target/classes directory.

test-compile: Compiles the test source code. This phase compiles the code located in the src/test/java directory.

test: Runs tests using a testing framework (e.g., JUnit). The compiled test code from the test-compile phase is executed against the compiled main code to validate the correctness of the code.

Maven Goals

package: Packages the compiled code into a distributable format, such as a JAR, WAR, or ZIP file. The output artifact is typically placed in the target directory.

verify :Runs checks on the results of integration tests to ensure quality criteria are met. This phase can involve running additional checks to ensure the integrity of the final artifact.

install : Installs the packaged project artifact into the local Maven repository. This makes the artifact available for other projects on the same machine.

deploy: Copies the final package to a remote repository, making it available to other developers and projects. This is typically used in a continuous integration or release process.

running "mvn clean install" will clean the project and then execute all the phases up to install

Thank you!

Have a great day ahead.