**Apache Maven:**

**Apache Maven** is a powerful project management tool that is based on POM (project object model). It is used for project builds, dependency management, and documentation. It simplifies the build process like ANT. But it is much more advanced than ANT.

In short terms, Maven is a tool that can be used for building and managing any Java-based project. Maven makes the day-to-day work of Java developers easier and generally helps with the overall comprehension of any Java-based project.

## **What is Maven?**

Maven is a **build tool** mainly used for **Java projects**. It helps to **build, manage, and package** Java applications — especially **web apps**. Maven makes it easier to manage **complex projects**.

### **1. POM File (pom.xml)**

### POM is an XML file (pom.xml) that defines the project structure and configuration.

* It includes details like dependencies, plugins, goals, project version, and more.
* Maven reads the pom.xml to execute build tasks and manage the project.

### **2. Dependencies and Repositories**

* Dependencies are external Java libraries needed by the project.
* Repositories are storage locations for JAR files, either local or central.
* Maven checks the local repository first and downloads from the central one if needed.

### **3. Build Lifecycle, Phases, and Goals**

* Maven follows a build lifecycle with ordered phases like compile, test, package, etc.
* Each phase consists of goals, which are specific build tasks.
* Executing a phase runs all previous phases in the lifecycle automatically.

### **4. Build Profiles**

* Build profiles allow different build configurations for environments like dev, test, or prod.
* Profiles are defined using the <profiles> tag in pom.xml.
* Profiles can be activated via the command line or environment conditions.

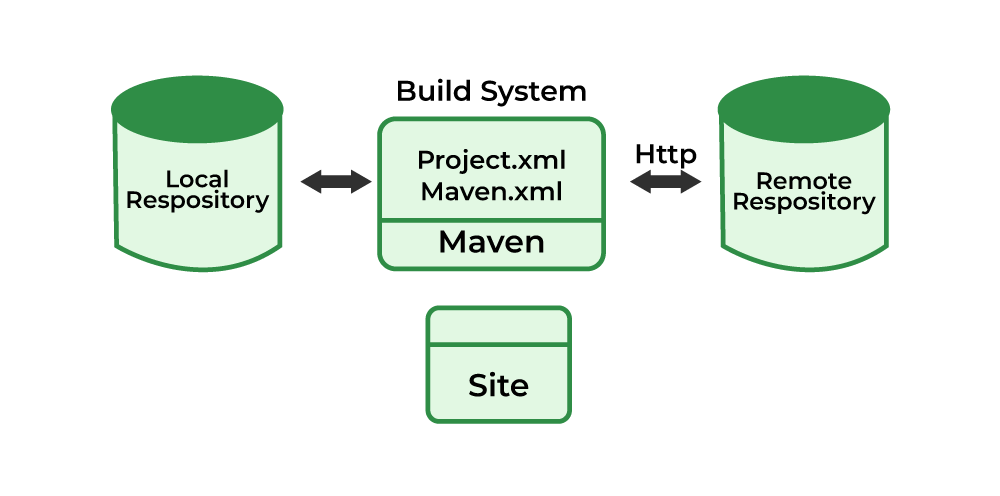
### **5. Build Plugins**

* Plugins add specific tasks to the Maven build process (e.g., compile, package, test).
* They are declared in the <build> section of the pom.xml file.
* Maven supports both standard and custom plugins written in Java.

## **What is Maven Architecture?**

* Maven repository is a place where the maven artifacts or dependencies of the JAR file is going to store which are written in the file called the POM.XML.
* POM.XML contains the Java classes, resources, and other dependencies. There are the two types of repositories like local repository and remote repository.

**DIAGRAM:**



* Maven reads the pom.xml file. Maven downloads the dependencies defined in the pom.xml file into the local repository from the central or remote repository.
* Maven executes the life cycles, phases, goals, and plugins defined in the pom.xml file.

## **What is Maven Used For?**

1. Maven simplifies project building by automating tasks like compiling, testing, packaging, and deploying without the need for scripting.
2. It manages dependencies efficiently by downloading required JARs from central repositories and keeping them updated.
3. Maven supports integration with version control systems like Git and Subversion and allows the use of plugins for extra features like code analysis and reporting.
4. It provides a standard project structure and supports multi-module projects, making it easier to manage large, related codebases.
5. Maven is highly customizable and informative, offering logs, test reports, and dependency lists to help developers maintain and understand their projects easily.

## **How to Install Maven?**

## Verify that your system has java installed or not. if not then install java (Link for Java Installation )

## Check java Environmental variable is set or not. if not then set java environmental variable(link to install java and setting environmental variable)

## Download maven from the official website.

## Unpack your maven zip at any place in your system.

## Add the bin directory of the created directory apache-maven-3.5.3(it depends upon your installation version) to the PATH environment variable and system variable.

## Open cmd and run **mvn -v** command. If Maven prints its version and Java info, installation is successful.

## Verify that your system has java installed or not. if not then install java (Link for Java

## **What is Maven pom.xml File?**

## Project Object Model(POM) is key to operate Maven. Maven reads pom.xml file to accomplish its configuration and operations.

It is an XML file that contains information related to the project and configuration information such as **dependencies**, **source directory**, **plugin**, **goals etc**. used by Maven to build the project.

The sample of pom.xml file is below:



**Elements used for Creating pom.xml file:**

* **project:** It is the main root element of the pom.xml file.
* **modelVersion:** Specifies the version of the POM model, usually 4.0.0 for Maven 2 and 3.
* **groupId:** A unique identifier for the project group, often matching the Java package name.
* **artifactId:** The name of the project you are building.
* **version:** The current version number of the project.

## **Pom.xml file Key Components:**

1. **dependencies:**dependencies is used to defines a list of dependency of project.
2. **dependency:**Defines each dependency with its groupId, artifactId, and version.
3. **name:**this element is used to give name to our maven project.
4. **scope:**this element used to define scope for this maven project that can be compile, runtime, test, provided system etc.
5. **packaging:**packaging is used to packaging our project to output types like JAR, WAR etc.

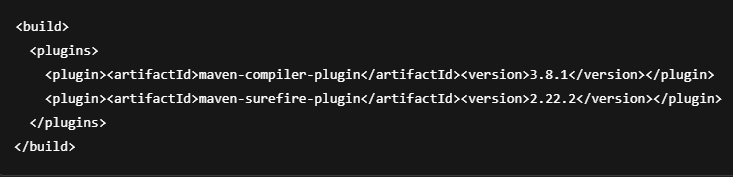
# **Maven Plugins:** It is a essential components in the Apache Maven build system designed to extend its functionality. They perform tasks such as compilation, testing, packaging, deployment, and others.

### **Types of Maven Plugin:**

* Build Plugins
* Reporting Plugins

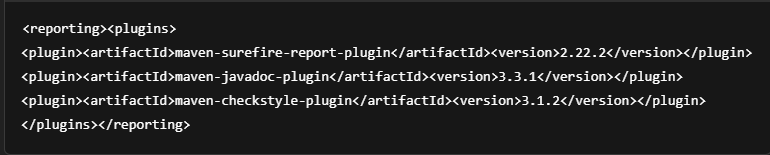
## **Maven Build Plugins:**

These plugins are executed during the build process and these are defined in the **<build>**section in the**pom.xml**file**.**For example, Compiler Plugin, Surefire Plugin, and Assembly Plugin.



## **Maven Reporting Plugins:**

* These plugins are used for generating reports about the project. These are defined in the **<**reporting**>**section in the **pom.xml**file.
* Examples include the Surefire Report Plugin for generating test reports and Javadoc Plugin for generating API documentation.



### **Different types of Maven Plugins:**

* **Compiler Plugin:** It is used for Compiles the project source code.
* **Surefire Plugin:** It is used for Run the Project unit Tests.
* **Failsafe Plugin:** It is used for Runs the project integration tests.
* **JAR Plugin:** It is used for Packages the compiled code into a JAR file.
* **War Plugin:** It is used for Packages the compiled code into a WAR file.
* **Assembly Plugin:** It is used for Creates distributions from the project.
* **Javadoc Plugin**: It is used for Generates Javadoc documentation for the project.
* **Checkstyle Plugin**: It is used for Checks the code against coding standards.

### **Working Process of Plugins in a Maven Project:**

* Define the project configuration and project structure and build plugins configurations in the POM file
* Now configure the required plugins in the maven project.
* After that maven build process is started by help of different phases in the maven build like validate, compile, test, package, verify, install, and deploy.

**Advantages:**

* Automate common build tasks like compiling, testing, and packaging.
* Easy to add and configure within the pom.xml file.
* Large number of standard plugins available for many needs.
* Support for custom plugin development for specific tasks.
* Help maintain consistency across multiple projects.

**Disadvantages:**

* Plugins can sometimes be complex to configure correctly.
* Poorly written plugins may slow down the build process.
* Dependency on plugins can lead to version conflicts or compatibility issues.
* Limited control compared to fully custom build scripts.
* Learning curve for understanding plugin goals and configuration