**11-03-2025**

**Maven**

**What is maven?**

* Maven is a popular open-source, build tool developed by apache company to build, publish and deploy several projects at once.
* Maven is written in java and is used to build projects written c#, scala, ruby, etc.
* The tool is used to build and manage any java-based project. It simplifies the day to day work of java developers and helps them in their projects.

**What is build tool in devops?**

It is a software program that automates the process of transforming the source code into deployable and executable format.

Build tools:

1. Java – apache maven, apache ant (older version)
2. Python – pybuilder
3. .net - msbuild (Mircosoft build engine)
4. Nodejs – gulp, grunt, gradle, web pack

As a build automation tool, it automates the source code compilation & dependency management, assembles binary codes into packages & executes the test scripts.

**Maven Repository:**

Maven repositories refer to the directories of packaged JAR files that contain metadata. The metadata refers to the POM files relevant to each project. This metadata is what allows maven to download dependencies.

There are three types of repositories in maven:

1. Local repository
2. Remote repository
3. Central repository

**Local repository:**

Local repository refers to the machine of the developer where all the project material is saved. This repository contains all the dependency jars.

**Remote repository:**

The remote repository refers to the repository present on a web server which is used when maven needs to download dependencies. This repository works same as the central repo. Whenever anything is needed from remote repo it is first downloaded to the local repo and then it is used.

**Central repository:**

Central repository refers to the maven community that comes into action when there is a need of dependencies, and those dependencies cannot be found in the local repository. Maven downloads the dependencies from here in the local repository whenever needed.

**Project Object Model (POM):**

* POM refers to the xml files that have all information regarding projects and configuration details.
* pom.xml is the fundamental configuration file used in Apache Maven, a popular build automation tool for Java projects.
* It defines the project's dependencies, plugins, goals, and various configurations required for building and managing the Java application.
* pom.xml used to manage the java application.
* Developers will write the pom.xml
* pom.xml is also known as super pom or parent pom.
* It contains all the dependency libraries
* pom.xml should be unique and each project can contain one pom.xml file.

**JAR, WAR, and EAR files:**

In Java, applications are packaged into JAR, WAR, and EAR files for deployment and distribution. Each format serves a different purpose based on the type of application.

1. Jar (java archive):

A JAR (Java ARchive**)** file is a compressed package containing compiled Java classes, resources, and metadata. It is mainly used for standalone applications and libraries.

1. War (web application archive):

A WAR (Web Application Archive) file is used to package Java web applications that follow the Servlet model. It is used when deploying java web applications on servlet container (e.g.,tomcat).

1. Ear (enterprise archive):

EAR (Enterprise Archive) is a file format used by Jakarta EE for packaging one or more modules into a single archive so that the deployment of the various modules onto an application server happens simultaneously and coherently. It also contains xml files called deployment descriptors which describe how to deploy the modules.

**Plugins:**

* Maven plugins are tools that extend and enhance the build process by performing specific tasks like compilation, testing, packaging, deployment, and reporting.
* Maven plugin refers to the group of goals. These goals may or may not be of the same phase. Installing the dependencies (external features) is known as plugins.
* The plugins are used to perform specific goal
* There are used to:

1. Automates repetitive tasks (e.g., compiling, testing, deploying).
2. Adds custom functionalities to the build process.
3. Handles project lifecycle phases (e.g., compile, test, package).
4. Manages external tools like Docker, Javadoc, or Checkstyle.

**Types of plugins:**

1. **Build plugins**

The build plugins are executed during different phases of the build lifecycle.

Eg: maven-jar-plugin (creates jar files)

1. **Reporting plugins**

These plugins generate reports and analyze project quality.

Eg: maven-javadoc-plugin

This plugin generates Javadoc documentation for your project.

**How does maven work**

pom.xml ----> build ----> war/jar/ear ----> packages (deployable format) ----> copy tomcat (web server)

* Read pom.xml file
* Download dependencies defined in pom.xml into local repository from central repository
* Create and generate a report according to the requirements, and execute life-cycles, phases, goals, plugins etc.

**Maven life cycle:**

Maven life cycle is a collection of steps that are to be followed to build a project.

There are three steps:

1. **Default**: handles project deployment

It fetches the source code from developers and performs few actions.

* Compile: compiles the entire source code
* Validate: validates the complied code
* Test: tests the source code
* Package: generates the package for the source code
* Install: install all the packages generated by packages
* Verify: it will verify the generated package

1. **Clean**: handles project cleaning

Performs before compilation

* Pre-clean: check for jar/war/ear files
* Clean: delete the older ear/jar/war files
* Post-clean: the new generated jar/war/ear files will be saved automatically

1. **Sites:** handles the creation of project site’s documentation

* Pre-site: it receives the post clean files
* Site: it receives the pre site files
* Post-site: it receives the files from site
* Site-deploy: to where (address of the server) we need to copy the files.

Maven in devops should be utilized in 3 scenarios:

1. If the initiative has number of significant dependencies.
2. If the dependencies version needs to be upgraded frequently.
3. The task involves rapid documentation, compilation & building of source code as jar or zip files.

Ant:

It is an older version of maven

* Ant (another neat tool) is a procedural build tool used for compiling, testing, and packaging Java applications.
* Developed by apache company
* Can build any kind of projects
* Has no life cycle
* Requires manual script writing for dependencies and tasks
* Flexible but complex (no built-in dependency management)
* Good for customized build processes

Differences between maven and ant:

|  |  |  |
| --- | --- | --- |
| Feature | Maven | Ant |
| Approach | Declarative(config-driven) | Procedural (task-driven) |
| Configuration File | pom.xml | build.xml |
| Dependency Management | Yes (built-in via Maven Central) | No (manual configuration required) |
| Project Structure | standardized | Customizable (no fixed structure) |
| |  | | --- | | Ease of Use | | |  | | --- | | Easier due to conventions |  |  | | --- | |  | | Requires scripting for each task |
| Reusability | High (plugins and conventions) | Low (requires writing scripts) |
| Use Case | Best for standardized Java projects | Best for custom build processes |
| documentation | It has more documentation | It has less documentation |
| Lifecycle | It has a lifecycle. | It does not have a lifecycle. |