**APACHE MAVEN**

**WHAT IS MAVEN ?**

* It is a build tool which is provided by Apache software foundation
* It is completly a open source
* Maven is a powerful project management tool that is based on POM (project object model).
* It is used for project build, dependency, and documentation.
* Maven is mostly used for the java projects to build web application packages.
* Maven provides different features which make it easy to build the web-application packages we manage complex projects easily.
* maven makes the day-to-day work of Java developers easier and generally helps with the comprehension of any Java-based project

**POM Files:**

* Project Object Model(POM) Files are 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.
* When you should execute a maven command you give maven a POM file to execute the commands. Maven reads pom.xml file to accomplish its configuration and operations.

**Dependencies and Repositories:**

* Dependencies are external Java libraries required for Project and repositories are directories of packaged JAR files.
* The local repository is just a directory on your machine’s hard drive.
* If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository.

**Build Life Cycles, Phases, and Goals:**

* A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals.
* Maven command is the name of a build lifecycle, phase, or goal. If a lifecycle is requested executed by giving the maven command, all build phases in that life cycle are executed also.
* If a build phase is requested executed, all build phases before it in the defined sequence are executed too.

**Build Profiles:**

* Build profiles a set of configuration values that allows you to build your project using different configurations.
* Example, you may need to build your project for your local computer, for development and test. To enable different builds you can add different build profiles to your POM files using its profiles elements which are triggered in a variety of ways.

**Build Plugins:**

* Build plugins are used to perform a specific tasks like compilation,testing,deployment,reporting,optimization. you can add a plugin to the POM file.
* Reuseable software components
* Extend functionality to maven.

**What Is Maven pom.xml File(Project Object Model File):**

POM means Project Object Model 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.

* project : It is the root element of the pom.xml file.
* Group ID: Client name or project ;eg. Axis,SBI,Union bank etc
* Artifact ID: Features of group ID ;eg. loan,gold loan,credit card,internet banking.
* Version ID: 8.2.0

Major release:If 8 changes to 9 = project got released

Minor release: If 2 changes to 3 = New feature is added

Hot fix/Bug fix: Developers still working on code.

**What is Pom.xml file Key Components:**

1. Dependencies: Dependencies element is used to defines a list of dependency of project.
2. Dependency: Dependency defines a dependency and used inside dependencies tag. Each dependency is described by 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 element is used to packaging our project to output types like JAR, WAR etc.

Libraries = Dependencies

**Features of Maven:**

1. We can easily build a project using maven.
2. We can add jars and other dependencies of the project easily using the help of maven.
3. Maven provides project information (log document, dependency list, unit test reports, etc.)
4. Maven is very helpful for a project while updating the central repository of JARs and other dependencies.
5. With the help of Maven, we can build any number of projects into output types like the JAR, WAR, EAR without doing any scripting.
6. Using maven we can easily integrate our project with a source control systems (such as Subversion or Git).
7. Maven also helps in managing the project’s build lifecycle, including tasks like compiling, testing, packaging, and deploying the code.
8. Maven provides a standard project structure, making it easy for developers to understand the layout of the project and locate specific files.
9. Maven supports multi-module projects, allowing developers to work on multiple related projects simultaneously and manage their dependencies efficiently.
10. Maven plugins can be used to add additional functionality to the build process, such as code coverage analysis, static code analysis, and more.
11. Maven is highly customizable, allowing developers to configure the build process to meet their specific needs and requirements.
12. Maven simplifies the process of managing project dependencies, ensuring that the correct versions of libraries and frameworks are used throughout the project.

**MAVEN BLOCK DIAGRAM:**



**Fig: MAVEN BLOCK DIAGRAM**

**MAVEN BUILD LIFE-CYCLE:**

There are 3 steps:

1. **DEFAULT:** Takes code from developers & perform few functions like compile,Validate,Test,Packaging,Install,Verify

* COMPILE: converts sorce code to machine language.
* VALIDATE: checks the compiled source code.
* TEST: Tests the source code.
* PACKAGE: Generates package for code.
* INSTALL: Install all packages generated by package block.
* VERIFY: It will verify generated package.

2**. CLEAN:** Performed before compilation

* PRE-CLEAN: Check for jar/war/ear
* CLEAN: Delete older jar/war/ear.
* POST-CLEAN: New generated war/jar/ear files will be saved.

3. **SITE:** It is like a folder where we will deploy our application.

* PRE-SITE: It receives POST-CLEAN files.
* SITE: It receives PRE-SITE files.
* POST-SITE: It receives files from SITE.
* SITE-DEPLOY: Place where we need to copy the files.

**MAVEN REPOSITARIES:**

Maven Repositories is a directory where all project jars, libraries, WARs, plugins, and other artifacts are stored and can be retrieved by Maven for use in the project.

**Types of Maven Repositories:**

1. **Local Repository: (own laptop)**  
    A local repository is a directory on the developers' machine where Maven stores all the artifacts resolved from remote repositories or created by the developer.   
     
    When a dependency is needed, Maven first check this local repository. If dependency is found It is used, Otherwise Maven fetches it from the remote repositories and caches it locally.
2. **Central Repository:(online repo - GitHub)**  
    The central is a repository provided by the Maven community and maintained by the Apache Software Foundation. It contains a large number of commonly used libraries and is the default repository used by Maven When It needs to resolve dependencies.  
      
    When a dependency is not found in the local repository, Maven queries the central repository to download it.
3. **Remote Repository:(Organiztn repo)**

The Remote repositories are repositories other then the central repository that can be set up by third parties or any organizations to host their own artifact. They are specified in the project's pom.xml file in the Maven Settings.

If an artifact is not available in the local repository, Then Maven will check these remote repositories for the required dependencies.

**MAVEN GOALS AND MAVEN COMMANDS:**

1. mvn compile

2. mvn clean

3. mvn test

4. mvn validate

5. mvn run

**RELEASES:**

* It is final version of that build which will not change.
* Looks like => 8.2.0 => only numerical value present.
* Releases maintain versions.

**SNAPSHOTS:**

* It is under development version build compile/artifacts can change.
* Looks like =>( 8.2.0 - SNAPSHOT )
* Does't maintain any version of software.

**WHAT IS BUILD TOOL IN DEVEOPS:**

* It ia software program that automates process of transmits source code into executable format and deployment.
* It is mainly used for java based
* Organise project specific files
* Compile the source code
* Generates documentation for the source code
* Runs Junit and integration test cases
* Create software deployable package (war/ear/jar)

**What is Junit?**

* Junit is a framework for automating unit testing, whenever we add new features to the software
* we need to retested with all functionalities, developers write Junit test cases.

**What is integration testing?**

* Integration test cases are written by QA team, integrations testing is testing end to end flow
* QA teams use tools like selenium, QAT for automating integration testing. This can be integrated with build tool like maven.

**What is war/ear/jar ?**

* **WAR** stands WebArchive, it a format to package web applications
* **EAR** stands for Enterprise Archive, this format is for EJB based applications
* **JAR** stands for Java Archive

**MAVEN IN DEVEOPS SHOULD BE UTILIZED IN 3 SCENARIOUS:**

1. If initiative has no significant dependencies.

2. If dependencies version needs to be upgraded frequently.

3. The task involves rapid documentation,compilation, & building of source code as jar or zip files.

**MAVEN INSTALLATION STEPS:**

1. Take 1 EC2 instance & login.
2. Connect to Linux environment.
3. Insttall Java
4. Install Git (we need to install in cloud)
5. Clone Git URL
6. Cd filename (pom.xml)
7. mvn clean package => war/jar/ear =>stored in target directory.

**PACKAGES:** Packages are stored in **Target directory**.

**DIFFERENCES BETWN APACHE MAVEN AND APACHE ANT:**

|  |  |  |
| --- | --- | --- |
|  | **MAVEN** | **ANT** |
| **Definition** | It is a framework based on the concept of POM. | It is a Java library and command-line toolbox. |
| **Convention** | It has built-in conventions to place source code, compiled code, etc. | It does not have any formal conventions. |
| **Project structure Info** | It does not require information about the project structure to be provided in the pom.xml file. | It requires information on the project structure to be provided in the build.xml file. |
| **Lifecycle** | It has a lifecycle. | It does not have a lifecycle. |
| **Nature** | It is declarative in nature (only source should be present in the default directory) | It is procedural in nature (manually tell exactly what to do and when to do it). |
| **Type** | It is primarily a project management tool. | It is primarily a project management tool. |
| **Dependency** | It can self download the dependencies from a central repository for building projects. | It has no built-in support for dependency management. |
| **Reusability** | It consists of reusable plugins. | It consists of scripts that are not reusable. |
| **Preference** | It is less preferred. | It is more preferred. |
| **Complexity** | It is more complex. | It is simple and reliable. |
| **Flexibility** | It is less flexible and maintainable. | It is more flexible and maintainable. |
| **Reports** | It creates reports. | It does not create reports |
| **Interface** | It provides common interface. | It does not provide common interface. |
| **Naming Conventions** | It has standard naming conventions. | It does not have standard naming conventions. |
| **Popularity** | It is more popular in users. | It is less popular in users. |
| **Scripts** | Scripts are not complicated. | Scripts are complicated. |
| **Ease of Use** | It is easy to use for developers. | It is complex to use at first for developers. |
| **Documentation** | It has more documentation. | It has less documentation. |
| **Reliability** | It is not reliable. | It is more reliable than Maven. |
| **Conflicts** | Conflicts are not handled for the same library. | Conflicts are handled for the same library. |