**MAVEN TOOL**

**🡪**Maven is a powerful project management tool primarily used for building the java projects.

🡪Maven is developed by the Apache company.

🡪Maven is the latest tool while there in an older version of Apache maven which is known as Apache ANT.

🡪Maven is a build Automation tool.

🡪Maven is open source which means free to use without any cost.

🡪Mostly Maven is based on POM (Project Object Model).

🡪It is used for project build, documentation and dependency.

🡪 The Apache Maven is more advanced than the Apache Ant.

🡪Maven as a build automation tool it automates the source code compilation &dependency management, assembles binary code into packages and executes the test scripts.

🡪Here test scripts are nothing but used in the automation testing.

🡪Maven makes the day-to-day work of java developers more easier and helps with the comprehensions of any java based project.

**What is a build tool in devOps?**

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

🡪let us see the build tools used for the several languages.

**Build Tools**

1. Java- Apache Maven, Apache ANT (older version).
2. Python- pybuilder
3. .Net- MS build (Microsoft build engine)
4. Node.js- gulp, grunt, Gradle, wen pack.

**What is maven?**

🡪Maven is mostly used java based projects to build web application packages.

🡪Maven provides different features which is helpful to build the web application packages and manage complex projects easily.

🡪Maven basically consists of five steps.

1. **POM files**:

🡪Project object files are xml files that contains the information about the project and configuration information such as dependencies and source directories, plugins, goals etc.

🡪when we are trying to execute the commands in maven use pom files.

🡪Maven read the pom .xml files to accomplish its configurations and operations.

1. **Dependencies and Repositories**:

🡪Maven use dependencies which are external libraries required for project repositories and directories of packaged JAR (java application)

files.

🡪The local repository is just a directory in our system, if the dependencies are not found in the local repository maven downloads it from the central server and adds to **the local repository.**

1. **Build Lifecycles, 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.

🡪A build phase is requested executed, all build phases before it in the defined sequence are executed too.

1. **Build profiles:**

🡪 Build profiles a set of configuration values that allows you to build your project using different configurations.

1. **Build Plugins:**

🡪 Build plugins are used to perform a specific goal. you can add a plugin to the POM file. Maven has some standard plugins you can use, and you can also implement your own in Java.

**How does maven work?**

Pom.xml 🡪build🡪war/jar/ear🡪packages (deployable format)🡪copy tomcat (web server).

🡪ear – Enterprise Applications archive

🡪war – web applications

🡪jar – java applications

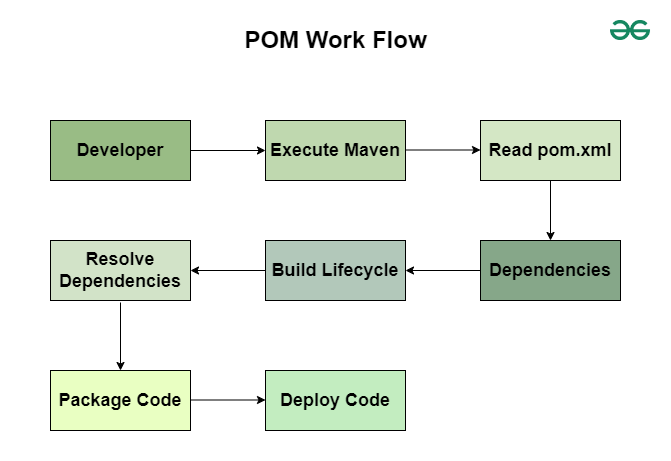
🡪The above mentioned are the different types of packaging formats used for deploying applications.

🡪Jar is used for packaging standalone applications.

🡪war is used for packaging web applications.

🡪Ear is used for packaging enterprise applications that has multiple modules.

**Workflow of Maven**



* Developers 🡪code 🡪push 🡪git hub (remote repository) 🡪 build(maven) 🡪fetch 🡪build 🡪generate ear/war/jar 🡪packages 🡪copy the packages on to the web server (tomcat) 🡪EU (access the applications)
* Here the developers will write the code after the finalized code which is source code is pushed into git hub that is remote repository and then it is used to build by maven tool and fetches and generates the ear/war/jar , packages and these packages copy is sent to the web server which is known as tomcat and later we can access the application.
* Let us see some more details about the maven.
* **Maven**

1. Maven works for project object model pom.xml |pom.xml2.
2. The most important thing to notice here is the pom .xm1 and pom.xm12 both are same where pom.xm1 is the latest version and pom.xm12 is the older version.
3. Pom.xm1 is used to manage the java application.
4. Developers will write the pom.xm1 file which is nothing but an source code.
5. Pom.xm1 is also known as **super pom** and **parent pom.**
6. Maven contains all the dependency libraries.
7. Dependency libraries are nothing but it is an internal or external features included in the project.
8. Pom.xm1 should be unique.
9. Each project has one pom.xm1 file.

* **Plugins**

1. Plugins are used to perform a specific goal.
2. Installing the dependencies which is external features is known as plug-ins.
3. Basically, there are two types of plugins
4. In built plugins
5. Added plugins

**In built plugins**

**🡪**In maven, inbuilt plugins are default plugins which are pre-installed and automatically executed in build life cycle.

🡪These plugins handle tasks like compilation, testing, packaging, installation, and deployment.

🡪It is used for only one time in a project.

**Added plugins**

**🡪**Added plugins in Maven are external or additional plugins that are manually specified in the pom.xml file to extend Maven’s functionality beyond the default (inbuilt) plugins.

🡪These plugins help with tasks like code analysis, logging, Docker integration, and more.

🡪It is reusable

1. Maven has a life cycle.

* **Repositories**
* There are three types of repositories present in it.

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

Maven uses repositories to store and retrieve dependencies required for a **Central repository**

* The **default public** repository project.
* provided by Maven.
* It Stores thousands of open-source libraries (e.g., Spring, Hibernate).
* It is Located at [**Maven Central**](https://repo.maven.apache.org/maven2/).
* If a dependency is not found in the local repository, Maven downloads it from the **Central Repository**.
* It is online repository.

**Remote Repository**

* A private or third-party repository (e.g., Nexus, JFrog Artifactory).
* Used for storing internal organization dependencies.
* Developers push artifacts here for sharing across teams.
* Defined in pom.xml under <repositories>.
* It is organizational repository by IBM

**Local Repository**

* The local repository is a storage location on your machine where Maven caches dependencies downloaded from remote repositories (like Maven Central).
* It helps reduce download time and allows Maven to work offline**.**

**Applications**

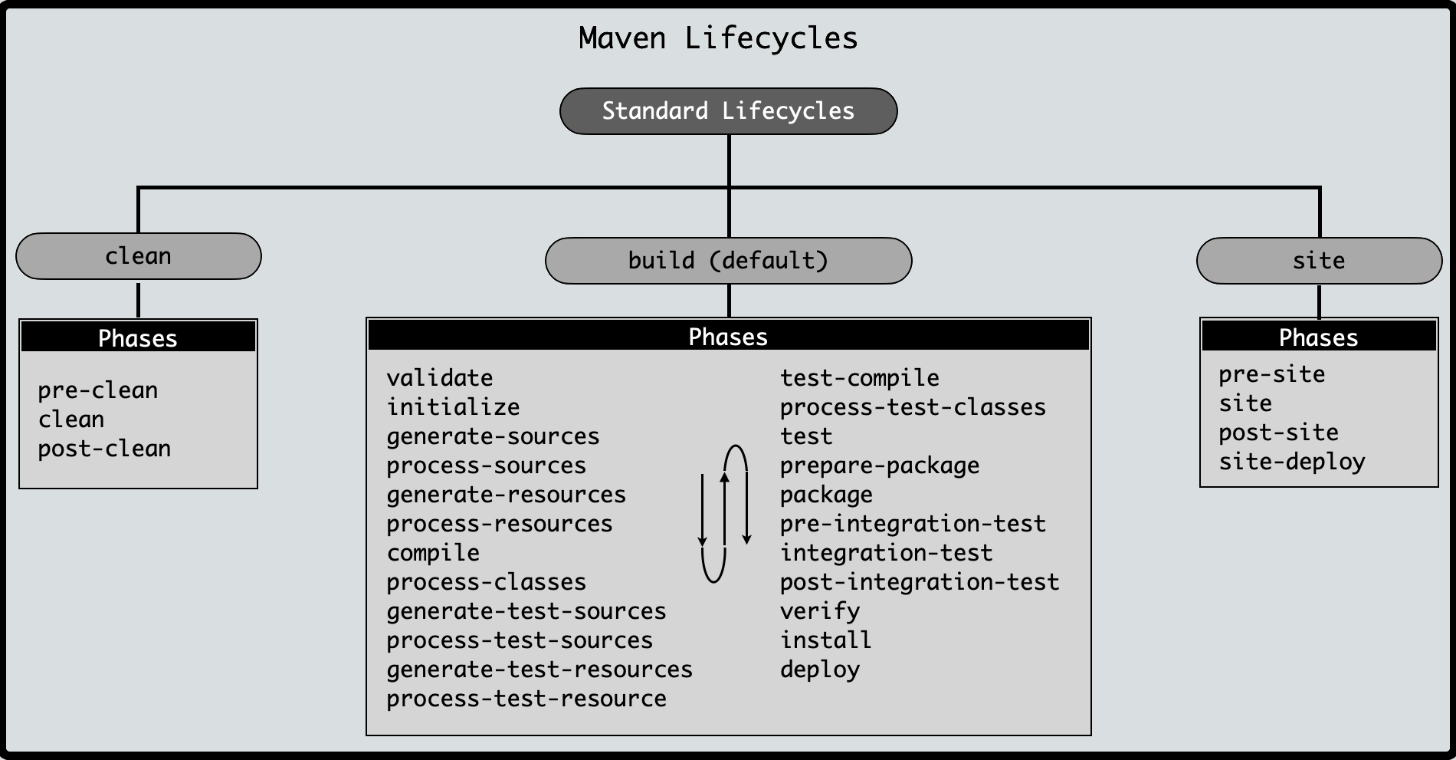
* Maven in devOps should be utilised in 3 scenarios.

1. If the initiative has no 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.
4. It manages the dependencies which means it will downloads the jar files.
5. Compiling the .java files to .class files.
6. It executes the unit test using junit which means java unit.
7. It creates jar or war package files.

**Apache ANT**

* It is the older version of maven.
* It is also developed by Apache company.
* It can build any kind of projects.
* It has no life cycle.
* The developers will use the build.xml files.
* Junit (java unit) test cases not there in Apache ant where it is present in maven.
* Here the scripts are not reusable, until unless we need to update it repeatedly.
* In Apache ANT, it stands for **Another Neat Tool.**

**MAVEN LIFE CYCLE**



🡪Maven follows a **build lifecycle** that defines the sequence of steps required to build, test, and deploy a project. The lifecycle consists of **phases**, and each phase has a specific purpose.

1. **Default**

* It fetches the code from developers and perform few functions.
* It handles the entire project build process.

1. **Compile:** It compiles the entire source code.
2. **Validate:** It validates the complied code.
3. **Test:** It test the source code.
4. **Package:** It will generate the package for our source code.
5. **Install :** install all the packages and compiled into jar or war files.
6. **Verify:** It will verify the generated package.
7. **Clean**

* It Removes old build files before starting a new build.
* It is performed before compilation.

1. **Pre-clean:** It checks for jar/war/ear files.
2. **Clean:** deletes the older ear/jar/war files.
3. **Post-clean:** the new generated war/ear/jar files will be saved automatically.
4. **Site**

* It is like a folder where we will deploy our applications.
* Generates project documentation and reports**.**

1. **Pre-site:** It will receives the post clean files.
2. **Site:** It will receive the pre-site files.
3. **Post-site:** It receives the file from site.
4. **Site: Deploy:** It will deploy to where address of server we need to copy the files.

**DIFFERENCE BETWEEN THE MAVEN AND ANT**

|  |  |
| --- | --- |
| **MAVEN** | **ANT** |
| 1. It is the latest version which is developed by Apache company | 1. It is the older version which is developed by Apache company |
| 1. It is mainly used for java based project applications | 2.It can build any kind of projects. |
| 1. It has a build life cycle | **3.**It has no life cycle |
| 1. The files are in the form of pom.xm1 (project object model) | **4.**The files are in the form of build.xm12 |
| 1. It has the test case of java unit (junit) | **5.**It has no testcases |
| 1. Scripts are not reusable | **6.**scripts are reusable |
| 1. Extensive plugin support | 7.limited plugin support |
| 1. Easier due to predefined steps | 8.requires manual configuration |
| 1. <dependencies> in pom.xml | 9.should be downloaded manually |
| 1. Best for Standard Java projects, Microservices | 10.best for Custom build processes, Legacy projects |
|  |  |

**CONCLUSION**

* **Maven** is best for **structured projects** with **automatic dependency management**.
* **Ant** is useful for **custom builds** where you need **manual control**.