**Project documentation: mvn site**

**What is Maven?**

Maven is a robust project management tool based on the POM architecture (project object model).

**Maven** build **lifecycle** goes through a set of **stages**, they are called build **phases**.

**Why Maven?**

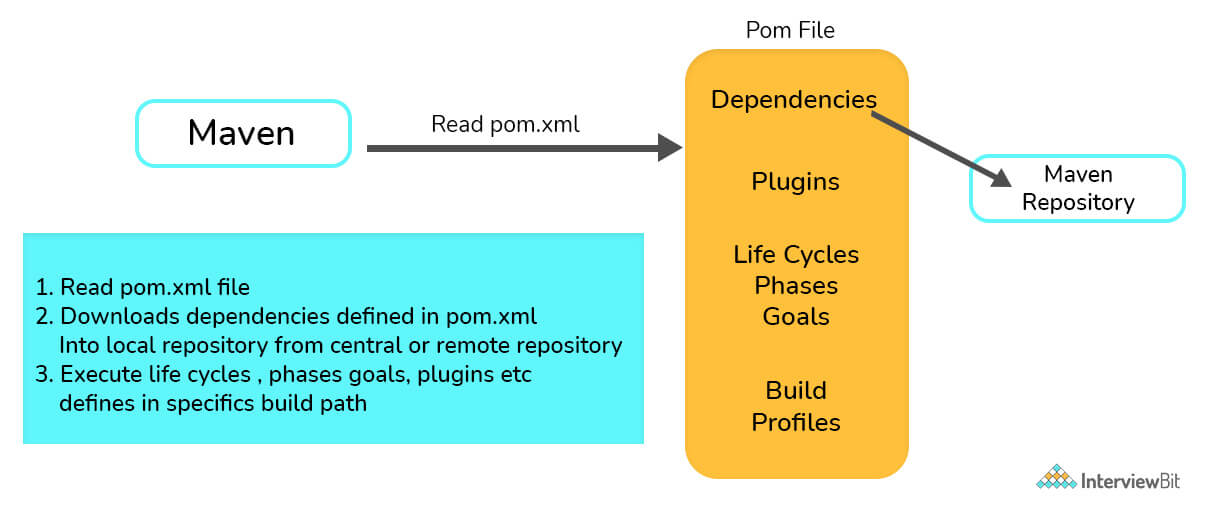
It is used for project build, as well as for managing dependencies and documentation. In simple terms, Maven is a tool that can be used to create and manage any Java-based project.

Maven is a build automation tool that is mostly used in Java projects but Maven can also be used to create and manage projects written in other languages such as C#, Ruby, Scala, and others.

**When should one use Maven?**

* When the project has a large number of dependencies. Then, using Maven, you can easily manage those dependencies.
* When the version of a dependency changes frequently. To update dependencies, simply update the version ID in the pom file.
* Maven makes it simple to handle continuous builds, integration, and testing.
* When you need a quick way to generate documentation from source code, this is the tool you use. It helps in compiling source code, and then packaging it into JAR or ZIP files

**How does Maven work?**



* Reading the pom.xml file is the first step.
* The dependencies mentioned in pom.xml are then downloaded from the central repository into the local repository.
* Finally, it builds and generates a report based on the requirements, as well as handles life cycles, phases, goals, plugins, and other tasks.

## **Maven Build Lifecycle**

A Build Lifecycle is a well-defined sequence of phases, which define the order in which the goals are to be executed. Here phase represents a stage in life cycle.

The Maven lifecycle is the set of tasks that are executed when the maven build is run.

There are three built-in lifecycles:

* default: handles project build and deployment
* clean: to clean the project and remove all files generated by the previous build
* site: to create the project's site documentation

## **Maven Phase**

**Maven build lifecycle goes through a set of stages, they are called build phases**. Each phase is responsible for a specific task.

Here are some of the most important phases in the default build lifecycle:

* validate: check if all information necessary for the build is available
* compile: compile the source code
* test-compile: compile the test source code
* test: run unit tests
* package: package compiled source code into the distributable format (jar, war, …)
* integration-test: process and deploy the package if needed to run integration tests
* install: install the package to a local repository
* deploy: copy the package to the remote repository

## **Maven Goal**

**A build phase is made up of a set of goals, and each goal is responsible for a specific task.**

When we run a phase, all goals bound to this phase are executed in order.

Here are some of the phases and default goals bound to them:

* compiler:compile – the compile goal from the compiler plugin is bound to the compile phase
* compiler:testCompile is bound to the test-compile phase
* surefire:test is bound to the test phase
* install:install is bound to the install phase
* jar:jar and war:war is bound to the package phase

|  |  |
| --- | --- |
| Phase | Goal |
| process-resources | resources:resources |
| compile | compiler:compile |
| process-test-resources | resources:testResources |
| test-compile | compiler:testCompile |
| test | surefire:test |
| package | jar:jar |
| install | install:install |
| deploy | deploy:deploy |

## **What is Build Profile?**

A Build profile is a set of configuration values, which can be used to set or override default values of Maven build. Using a build profile, you can customize build for different environments such as Production v/s Development environments.

## **What are Maven Plugins?**

Maven is actually a plugin execution framework where every task is actually done by plugins. Maven Plugins are generally used to −

* create jar file
* create war file
* compile code files
* unit testing of code
* create project documentation
* create project reports

mvn [plugin-name]:[goal-name]

Ex: mvn compiler:compile

**External Dependency**:

As you know, Maven does the dependency management using the concept of Repositories. But what happens if dependency is not available in any of remote repositories and central repository? Maven provides answer for such scenario using concept of **External Dependency**.

For example, let us do the following changes to the project created in ‘Creating Java Project’ chapter.

* Add **lib** folder to the src folder.
* Copy any jar into the lib folder. We've used **ldapjdk.jar**, which is a helper library for LDAP operations.

<dependency>

<groupId>ldapjdk</groupId>

<artifactId>ldapjdk</artifactId>

<scope>system</scope>

<version>1.0</version>

<systemPath>${basedir}\src\lib\ldapjdk.jar</systemPath>

</dependency>

**Discuss the core concepts of Maven.**

* **POM Files:**Project Object Model (POM) files are XML files that include information about the project and configuration information used by Maven to construct the project, such as dependencies, source directory, plugin, goals, and so on. When you want to run a maven command, you provide it with a POM file to run. To complete its configuration and functions, Maven reads the pom.xml file.
* **Dependencies and Repositories:** Repositories are folders containing bundled JAR files, and dependencies are external Java libraries necessary for Project. The local repository is simply a folder on your computer's hard drive. Maven retrieves dependencies from a central Maven repository and places them in your local repository if they aren't found in the local Maven repository.
* **Build Life Cycles, Phases, and Goals:** A build life cycle is made up of a series of build phases, each of which contains a set of goals. A build lifecycle, phase, or goal is referred to as a Maven command. When a lifecycle is asked to be run using the maven command, all of the build steps in that life cycle are likewise run. When a build phase is requested to be executed, it is followed by all build phases in the given sequence.
* **Build Profiles:** Build Profiles are a set of configuration parameters that allow you to build your project using a variety of setups. For example, you might need to develop and test your project on your local computer. You can add different build profiles to your POM files using its profile elements to enable different builds, which can be triggered in a variety of ways.
* **Build Plugins:** Build Plugins are used to accomplish a certain task. A plugin can be added to the POM file. Maven comes with various pre-installed plugins, but you can also write your own in Java.

**What elements are used for creating a pom.xml file?**

* **Project:** The root element of the pom.xml file is the project.
* **ModelVersion:** It identifies which version of the POM model you're working with. For Maven 2 and Maven 3, use version 4.0.0.
* **GroupId:** GroupId is the project group's identifier. It is unique, and you will most likely use a group ID that is similar to the project's root Java package name.
* **ArtifactId:** It is used for naming the project you're working on.
* **Version:** The version number of the project is contained in the version element. If your project has been released in multiple versions, it is helpful to list the versions.
* **Dependencies:** This element is used to establish a project's dependency list.
* **Dependency:** Dependency is used inside the dependencies tag to define a dependency. The groupId, artifactId, and version of each dependency are listed.
* **Name:** This element is used to give our Maven project a name.
* **Scope:** This element is used to specify the scope of this maven project, which can include compile, runtime, test, among other things.
* **Packaging:** The packaging element is used to package our project into a JAR, WAR, and other output formats

**What is a 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:

* Local Repository
* Remote Repository
* Central Repository

**What are the different types of Maven repositories? Discuss.**

The three types of repositories of Maven are:

* Local repository
* Central repository
* Remote repository

Maven scans these repositories for dependencies. Maven looks in the Local repository first, then the Central repository, and finally the Remote repository if the Remote repository is defined in the POM.

* **Local Repository:** Local repository is a directory on the developer's device. The local repository contains all of Maven's dependencies. Even though several projects rely on dependencies, Maven only needs to download them once.
* **Central Repository:** The Maven community has built the central Maven repository. Maven searches this central repository for any dependencies that aren't available in your local repository. The dependencies are subsequently downloaded into your local repository by Maven.
* **Remote Repository:** Maven may download dependencies from a remote repository hosted on a web server. It is frequently used to host internal organization projects. The dependencies are subsequently downloaded into your local repository by Maven

**In Maven, what do you mean by Clean, Default, and Site?**

**What is Maven Build Lifecycle?**

The three built-in build life cycles are:

* **Clean**: The clean lifecycle is in charge of project cleaning.
* **Default**: The project deployment is handled by the default lifecycle.
* **Site**: The creation of the project's site documentation is referred to as the site lifecycle.

**What are the different phases of the default life cycle?**

The different phases of the default lifecycle are:

* **Validate**: Make sure the project is correct and that you have all of the necessary information.
* **Test**: Test the compiled source code using an appropriate unit testing framework. These tests should not demand that the code be packed or deployed; instead, take the compiled code and package it in a manner that can be distributed, such as a JAR.
* **Compile:** Compile the project's source code.
* **Verify**: Perform any necessary checks on integration test findings to ensure that quality criteria are met.
* **Install:** Adds the package to the local repository, allowing it to be used as a dependency in other projects.
* **Deploy:** Copies the entire package to the remote repository for sharing with other developers and organizations, and is done in the build environment

**What are Maven plugins used for? What are the types of Maven plugins?**

Maven Plugins are used for:

* Creating JAR files.
* Creating WAR files.
* Compiling the source code files.
* Unit testing of the code.
* Creating the project documentation.
* Creating project reports.

Maven plugins are divided into two categories:

* **Build plugins:**  These plugins are used throughout the build process and are configured in the pom.xml file's <build/> element.
* **Reporting plugins:** These plugins are configured in the pom.xml's <reporting/> element and run during stage generation.

### “Maven uses convention over configuration” - Why is it so?

* Because developers simply need to establish a Maven project while employing convention,  Maven uses convention over configuration. The rest of the structure is generated automatically. In the case of configuration, build processes are to be created manually.
* For setting up a project, creating artifacts, releasing code, and running unit tests, Maven has a number of conventions.

### What is Maven's inheritance order?

In Maven, the order of inheritance is:

* Settings
* CLI parameters
* Parent POM
* Project POM

### What is POM?

Project Object Model (POM) refers to the XML files with all the information regarding project and configuration details.

* It has the description of the project, information regarding the versioning and configuration management of the project[.](https://www.simplilearn.com/tutorials/maven-tutorial/maven-project-in-eclipse)
* The XML file is in the project home directory. When we tend to execute a task, Maven searches for the POM in the current directory.

### What are the minimum required elements for POM?

The minimum required elements for POM are:

* project root
* modelVersion – should be 4.0.0
* groupId – project’s group id
* artifactId – artifact (project) id
* version – version of the artifact

### What is meant by the term ‘Build Tool’?

A build tool is essential for the process of building. It is needed for the following procedures:

* Generating source code.
* Generating documentation from the source code.
* Compiling of source code.
* Packaging of the compiled codes into JAR files.
* Installing the packaged code in a local repository, server, or central repository.

### What are the steps to install Maven on Windows?

To install Maven on Windows, observe the following steps:

* Download Maven first, and then extract it.
* In the environment variable, add JAVA\_HOME, and MAVEN\_HOME.
* Then, add the environment path in the Maven variable.
* Lastly, verify [the Maven installation](https://youtu.be/Hff-1uDH1Ts) by checking its version. The command mvn -version will display the version installed in the system.

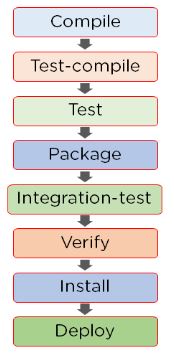
### What is the command to install JAR files in the Local Repository?

* mvn install is used to install JAR files in the local repository.

### How do you know the version of Maven being used?

* mvn –version is used to check the version of Maven present in the system.

### What are the different phases in the Maven Build Lifecycle?



### What are the different conventions used while naming a project in Maven?

The full name of a project in Maven includes:

  <GroupId>: <artifactId>: <version>

For example:

org.apache.maven: maven: 2.0.1

### What is a Maven Artifact?

Maven Artifact refers to a file, usually a JAR that gets deployed to a Maven repository. The tool creates one or more artifacts, such as a compiled JAR and a source JAR.

Every artifact has its groupID, an artifact ID, and a version string. These three together identify the artifact. For example:

com.your.package, any name, and a version string (to uniquely identify).

### What is meant by the term ‘Dependencies and Repositories’ in Maven?

* Dependencies refer to the Java libraries that are needed for the project. Repositories refer to the directories of packaged JAR files.
* If the dependencies are not present in your local repository; then Maven downloads them from a central repository and stores them in the local repository.

### Where are Maven dependencies stored?

* All the JARS, dependency files, etc. that are downloaded by Maven are saved in the Maven local repository.
* The Maven local repository is a folder location on the local system where all the artifacts are locally stored.