**MAVEN HANDBOOK**

1. **What is Maven?**

* The Maven project is developed by Apache Software Foundation.
* Maven is a powerful build automation tool that is primarily used for Java-based projects.
* Maven prefers convention over configuration. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven Central Repository and stores them in a local cache.
* Maven can also help you build and manage projects written in C#, Ruby, Scala, and other languages.
* ***Project Object Model (POM)*** file 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.
* When we execute a maven command, we give maven a POM file to execute the commands. Maven reads the *pom.xml* file to accomplish its configuration and operations.
* Maven helps you tackle two critical aspects of building software –
* It describes how software is built
* It describes the dependencies.

1. **What it does?**

Maven is a build tool and does mainly following tasks: -

* Directory Structures to the applications
* Downloads the required dependencies [JARs]
* Compile the Source Code/Test scripts
* Packing the applications
* Run Applications
* Start/Stop the Server
* Deploying the Applications
* Undeployment of application
* Prepare Reports & Documents

1. **When should we use Maven?**

* If there are too many dependencies for the project.
* When the dependency version update frequently.
* Continuous builds, integration, and testing can be easily handled by using maven.
* When one needs an easy way to generate documentation from the source code, compiling the source code, packaging compiled code into JAR files or ZIP files.

1. **What is Build Tool?**

* A build tool takes care of everything for building a process. It does follow:
* Generates source code (if auto-generated code is used)
* Generates documentation from source code
* Compiles source code
* Packages compiled code into JAR of ZIP file
* Installs the packaged code in local repository, server repository, or central repository

**Note**: Maven is java-based tool, so the very first requirement is to have JDK installed on your machine.

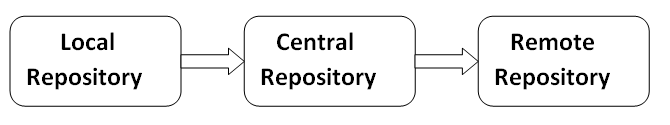
1. **What is a Compiler?**

* It is a software which will help to convert from high level language to low level language.
* . Java file 🡪 After compile 🡪 . Class file
* . Java files 🡪 .Class(bundle) 🡪 .Jar(bundle) 🡪 .war
* Group of .class files is called .jar file (Java Archive)
* Group of .jar files are called .war file (web Archive)
* .ear file 🡪Enterprise web archive
* Maven is an application build tool which will help us to build a java application.

Note: - SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, maven checks for a new SNAPSHOT version in a remote repository for every build.

1. **Maven Repository**

* A maven repository is a directory of packaged JAR file with pom.xml file.
* Maven searches for dependencies in the repositories.
* There are 3 types of maven repository:
* **Local Repository**
* **Central Repository**
* **Remote Repository**
* Maven searches for the dependencies in the following order:
* **Local repository** then **Central repository** then **Remote repository**.



* **If dependency is not found in these repositories, maven stops processing and throws an error.**

**1) Maven Local Repository**

* Maven local repository is located in your local system.
* It is created by the maven, when you run any maven command.
* By default, maven local repository is **%USER\_HOME%/.m2 directory.**
* For example: **C:\Users\DevOps\_IT\.m2**.

## Update location of Local Repository

* We can change the location of maven local repository by changing the **settings.xml file**.
* It is located in **MAVEN\_HOME/conf/settings.xml**,
* For example: **E:\apache-maven-3.6.1\conf\settings.xml**.

**2) Maven Central Repository**

* Maven central repository is located on the web.
* It has been created by the apache maven community itself.
* The path of central repository is: <http://repo1.maven.org/maven2/>.
* The central repository contains a lot of common libraries that can be viewed by this url :- https://central.sonatype.com/

**3) Maven Remote Repository**

* Maven remote repository is located on the web.
* Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.
* You can search any repository from Maven official website https://mvnrepository.com/

1. **Maven pom.xml file**

* POM is an acronym for Project Object Model.
* The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.
* Maven reads the pom.xml file, then executes the goal.
* It will provide all the project build configurations which are required by maven.
* When we create Maven project, then pom.xml file will be created automatically.
* In Maven projects, pom.xml file is able to provide no. of responsibilities:
* **Project Description**
* **Repository**
* **Dependency management**
* **Project Inheritance**
* **Build Configurations**
* **Build Profiles**

1. **Elements of maven pom.xml file**

* For creating the simple pom.xml file, we need to have following elements:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **project** | It is the root element of pom.xml file. |
| **modelVersion** | It is the sub element of project. It specifies the modelVersion.  It should be set to 4.0.0. |
| **groupId** | It is the sub element of project. It specifies the id for the project group. |
| **artifactId** | It is the sub element of project. It specifies the id for the artifact (project).  An artifact is something that is either produced or used by a project.  Examples of artifacts produced by Maven for a project include:  JARs, source and binary distributions, and WARs. |
| **version** | It is the sub element of project.  It specifies the version of the artifact under given group. |

**File: pom.xml**

**<project** xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.leenaIT.com/XMLSchema-instance"

 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd"**>**

**<modelVersion>**4.0.0**</modelVersion>**

**<groupId>**com.leenait.app1**</groupId>**

**<artifactId>**my-app**</artifactId>**

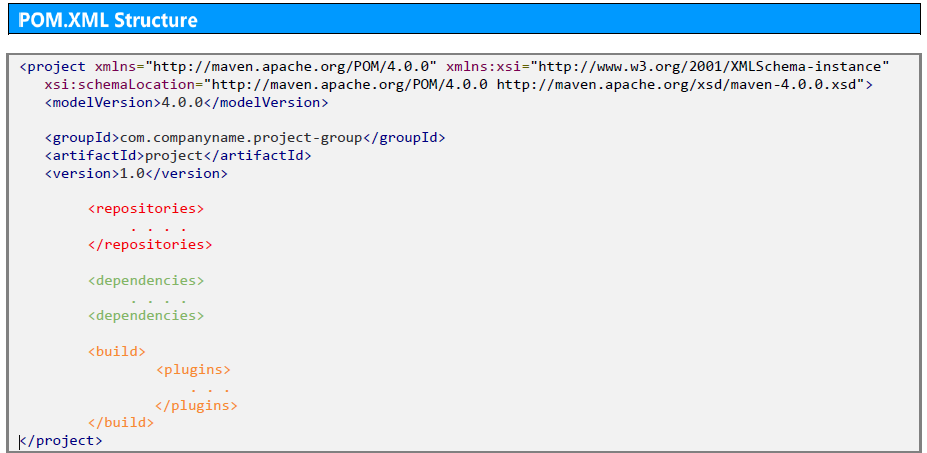
**<version>**1**</version>**

**</project>**

1. **Maven pom.xml file with additional elements**

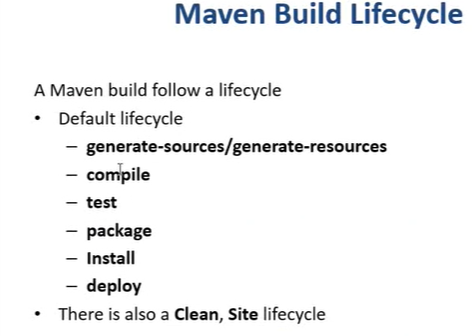
* Here, we are going to add other elements in pom.xml file such as:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **packaging** | defines packaging type such as jar, war etc. |
| **name** | defines name of the maven project. |
| **URL** | defines URL of the project. |
| **dependencies** | defines dependencies for this project. |
| **dependency** | defines a dependency. It is used inside dependencies. |
| **scope** | defines scope for this maven project.  It can be compiled, provided, runtime, test and system |





1. **Maven Build Life Cycle**

****

1. **Maven Commands:**

|  |  |
| --- | --- |
| mvn archetype: generate | It will create a project structure |
| mvn validate | It will validate the project structure |
| mvn compile | It will convert .java files into .class files |
| mvn test | It will test .class files |
| mvn package | It will help to create .jar, .war,. ear files |
| mvn install | It will install the package locally |
| mvn deploy | It will install the package remotely |
| mvn tomcat: run | It will run the package in tomcat server |
| mvn jetty: run | It will run the package in jetty server |

1. **What is Maven Snapshot?**

* Basically, while developing a large-scale application, the process involves a series of changes to be made in the application till it is confirmed to be ready as a final release. Because, there could be multiple teams working on an application across different set of modules.
* Consider 2 teams A & B working on 2 different modules and say team B will be dependent on the services provided by team A. Say team A is involved in working on some major critical fixes and those fixes will be checked in every day. So team B has to be notified in a way that there are still some pending work by team A and the final version has not yet been released by the team A. This is where the maven's SNAPSHOT comes into the picture.
* Snapshot is a special version which indicates the current development copy of the project which is being worked on. For each build, maven always checks out for a SNAPSHOT of the project.
* Hence, whenever maven finds a newer SNAPSHOT of the project, it downloads and replaces the older .jar file of the project in the local repository.
* Snapshot version always gets some updates/changes being made on the project. Also, Snapshot should exist only during development phase and it will not be having a release version. This implies that the build of the project will be changed at any time and it is still under the development process.

1. **Snapshot Vs Version**

* In case of SNAPSHOT, Maven will automatically fetch the latest SNAPSHOT (data-service: 1.0-SNAPSHOT) every time a project is built.
* In case of Version, maven once downloaded the mentioned version say JavaSamples:1.0, then it will never try to download a newer 1.0 available in repository. To download the updated JavaSamples,version is to be upgraded to 1.1.

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>JavaSamples</groupId>

<artifactId>JavaSamples</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<url>http://maven.apache.org</url>

</project>

1. **Maven Sample Project - Steps**

## Step 1: - Maven Archetype

* Maven uses archetype to create any project.
* Archetype is nothing but the tool provided by the maven to create a project. It is also defined as 'an organizational pattern from which all other things of the similar kind are made'.
* Using archetypes provides an easy and instrumental way to create projects to the developers, where the best practices are employed as per the standards defined at the organization level.
* Also, archetypes enables the developers to quickly create a project and start running the same within a matter of commands.
* In order to create a new simple maven java project, just use the maven archetype plugin in the command line.
* Currently there are more than 1500+ maven archetypes listed out in order to create projects under different frameworks like struts, spring, hibernate, web services (RESTful/SOAP) and many others.
* So, let us create one simple project in maven. Open the command prompt and navigate to the workspace folder in your local machine (**E.g. C:/Maven/workspace***).*

## Step 2: - We can create Sample Maven or web project as follows: -

1. **Syntax: -** Type the below maven command (its a single line command) and press enter.

mvn archetype:generate

-DgroupId=com.rknsit.services

-DartifactId=SampleMaven

-DarchetypeArtifactId=maven-archetype-quickstart

-DinteractiveMode=false

* This maven command will create a Java project with the below details:
* groupIdwill be **com.rknsit.app**, representing the package.
* artifactIdwill be Simplemaven (on the build, Sample-maven.jar will be created)
* archetypeArtifactId is nothing but the template used for creating this Java project.
* interactiveMode is used when the developer is aware of the actual spelling of the artifact id. In the above case, **maven-archetype-quickstart** is the one which is used and it is the proper one. If developer is not aware of this, then the interactiveMode is set to be TRUE so that it will scan the remote repositories for all available archetypes. This might take longer time.

## Step 3: - Project Folder structure

* We are done. A structure of the project has been created. (**E.g. C:/Maven/workspace**).
* Now if you traverse through your project folder, you will find **src** folder with App.java file containing the java code for printing ‘Hello World’.
* Run command **mvn validate** to validate the project structure.

## Step 4: - Compiling you project

* Open the Command Prompt and navigate to the root project folder,where it consists of the pom.xml file. E.g. *C:/Maven/workspace/Sample-maven*
* Run command **mvn compile** to compile your project.
* This will download all the dependencies and plugins required for the project running.
* Run command **mvn package.**
* This will package everything into a **JAR** file. This might download more plugins.
* **Note: -** whenever a maven phase is invoked, it executes in a sequence up to the invoked phase. Hence, in this case the phases above **package – validate, compile** and **test**phases will be executed.

## Step 5: - Packaging (Jar/war)

* When **mvn package** command is run, maven will validate and compile the source code, execute the junit test cases and packs/bundles it as per the instructions given in the tag <packaging> in the pom.xml file.
* **Note: -** If the packaging is specified as jar, then a jar with the package will be created. If the packaging is specified as war, then a war with the package will be created.
* The packaged jar file will be available under the target directory of the project. i.e. *C:\Maven/workspace/Sample-Maven/target*.

## Step 6: - Execute the compiled project

* To run the App.test java, open the command prompt and navigate to the folder *C:\Maven/workspace/Sample-Maven/target*and enter the command

*java -cp "target/CubeGenerator-1.0-SNAPSHOT.jar;." com.mavenpoint.App*

*java -cp "target/myfamily-1.0-SNAPSHOT.jar;." family.App*

1. **Maven Sample Project ( .jar file) – Steps**

* To create a simple maven project (. Jar file), open command prompt and run archetype: generate command with the following syntax:-
* **Step 1:- mvn archetype:generate** **-DgroupId=**com.cubegenerator **-DartifactId**=cubegenerator -**DarchetypeArtifactId=maven-acthetype-quickstart** **-DinteractiveMode=**False
* Now if you traverse through your project folder (here cubegenerator), you will find **src** folder with **App.java** file containing the java code for printing ‘Hello World’.
* Navigate to cubegenerator folder using **cd** folder.
* **Step 2:-** Run command **mvn validate** to validate the project structure.
* **Step 3:-** Run command **mvn compile** to compile your project.
* This will download all the dependencies and plugins required for the project running.
* **Step 4:-** Run command **mvn package.**
* This will package everything into a **JAR** file. This might download more plugins.
* **Step 5: -** Execute the compiled project
* To run the App.test java, open the command prompt and navigate to the folder enter the below given command:-

*java -cp "target/CubeGenerator-1.0-SNAPSHOT.jar;." com.cubegeneratort.App*

* **Step 6: -** Run command **mvn install** to Install the .jar file to local repository.

1. **Maven Sample Web Project ( .war file) – Steps**

* To create a simple maven project (. war file), open command prompt and run archetype: generate command with the following syntax:-
* **Step 1:- mvn archetype:generate** **-DgroupId=**com.cubegeneratorweb **-DartifactId**=cubegeneratorweb -**DarchetypeArtifactId=maven-acthetype-webapp** **-DinteractiveMode=**False
* Now if you traverse through your project folder (here cubegenerator), you will find **src/main/webapp** folder with **index.jsp** file containing the java code for printing ‘Hello World’.
* Navigate to cubegenerator folder using **cd** folder.
* **Step 2:-** Run command **mvn validate** to validate the project structure.
* **Step 3:-** Run command **mvn compile** to compile your project.
* This will download all the dependencies and plugins required for the project running.
* **Step 4:-** Run command **mvn package.**
* This will package everything into a **WAR** file (snapshot file).
* **Step 5: -** Deploy the war file to tomcat server
* Ensure tomcat server is in running state :- start tomcat.sh
* Access tomcat server by using **url:-** localhost:8009
* Update **pom.xml** file with the following plugins:-

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<version>1.1</version>

<configuration>

<url>http://localhost:8009/manager/text</url>

<server>TomcatServer</server>

<path>/cubegeneratorweb</path>

</configuration>

</plugin>

</plugins>

<finalName>cubegeneratorweb</finalName>

</build>

* Run **mvn tomcat:deploy** command to deploy the war file to tomcat server.
* Navigate to tomcat server, check for cubegeneratorweb.war file.
* Right click on cubegeneratorweb.war (Snashot) file and open in new tab.
* We can see the output (HelloWorld) of the war file.
* **Step 6: -** Run command **mvn install** to Install the .war file to local repository.
* **Step 7: -** Run command **mvn tomcat: redeploy** , if any changes made in index.jsp file.
* **Step 8: -** Run command **mvn tomcat: undeploy** , if you want to rollback the deployment.