Maven

**Maven**  is a project management and comprehension tool that provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

In case of multiple development teams environment, Maven can set-up the way to work as per standards in a very short time. As most of the project setups are simple and reusable, Maven makes life of developer easy while creating reports, checks, build and testing automation setups.

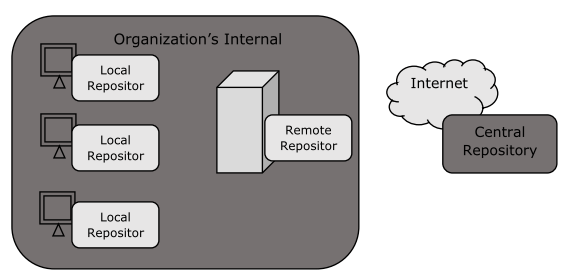
Maven provides developers ways to manage the following:

To summarize, Maven simplifies and standardizes the project build process. It handles compilation, distribution, documentation, team collaboration and other tasks seamlessly. Maven increases reusability and takes care of most of the build related tasks.

**What is a Maven Repository?**

In Maven terminology, a repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

Maven repository are of three types. The following illustration will give an idea regarding these three types.



* Local repository
* Central repository
* Remote repository

**Local Repository :-**

Maven local repository is a folder location on your machine. It gets created when you run any maven command for the first time.

Maven local repository keeps your project's all dependencies (library jars, plugin

jars etc.).When you run a Maven build, then Maven automatically downloads all

the dependency jars into the local repository. It helps to avoid references to dependencies stored on remote machine every time a project is build.

Maven local repository by default get created by Maven in %USER\_HOME%

directory. To override the default location, mention another path in Maven

settings.xml file available at %M2\_HOME%\conf directory.

**<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"**

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

**xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0**

**http://maven.apache.org/xsd/settings-1.0.0.xsd">**

**<localRepository>C:/MyLocalRepository</localRepository>**

**</settings>**

When you run Maven command, Maven will download dependencies to your custom path.

**Central Repository :-**

Maven central repository is repository provided by Maven community. It contains

a large number of commonly used libraries. When Maven does not find any dependency in local repository, it starts searching in central repository using following URL: http://repo1.maven.org/maven2/

Key concepts of Central repository are as follows:

 This repository is managed by Maven community.

 It is not required to be configured.

 It requires internet access to be searched.

To browse the content of central maven repository, maven community has

provided a URL: http://search.maven.org/#browse%7C47. Using this library, a

developer can search all the available libraries in central repository.

**Remote Repository :-**

Sometimes, Maven does not find a mentioned dependency in central repository

as well. It then stops the build process and output error message to console. To

prevent such situation, Maven provides concept of Remote Repository, which

is developer's own custom repository containing required libraries or other

project jars. For example, using below mentioned POM.xml, Maven will download dependency (not available in central repository) from Remote Repositories mentioned in the same pom.xml.

**What is 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.

As an example, a typical Maven Build Lifecycle consists of the following

sequence of phases. There are always pre and post phases to register goals, which must run prior to, or after a particular phase.

When Maven starts building a project, it steps through a defined sequence of phases and executes goals, which are registered with each phase.

**Maven has the following three standard lifecycles:**

* + clean
  + default(or build)
  + site

A goal represents a specific task which contributes to the building and managing

of a project. It may be bound to zero or more build phases. A goal not bound to

any build phase could be executed outside of the build lifecycle by direct

invocation.For example, consider the command below.

**mvn clean dependency:copy-dependencies package**

Here the clean phase will be executed first, followed by the **dependency:copy-**

**dependencies** goal, and finally package phase will be executed.

**Clean Lifecycle**

When we execute mvn post-clean command, Maven invokes the clean lifecycle consisting of the following phases.

* + pre-clean
  + clean
  + post-clean

In the following example, We'll attach maven-antrun-plugin:run goal to the pre-clean, clean, and post-clean phases. This will allow us to echo text messages displaying the phases of the clean lifecycle.

You can try tuning mvn clean command, which will display pre-lean and clean. Nothing will be executed for post-clean phase.

**Default (or Build) Lifecycle**

This is the primary life cycle of Maven and is used to build the application. It has

the following 23 phases.

**Maven Default life cycle**

1.validate

2. compile

3. test

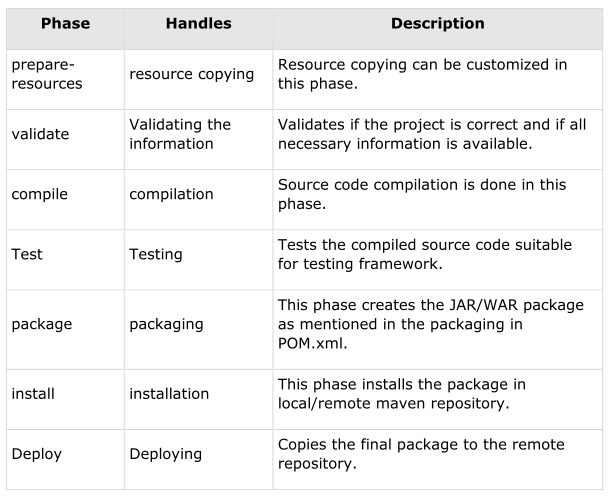
4. integrate -test

5. Package

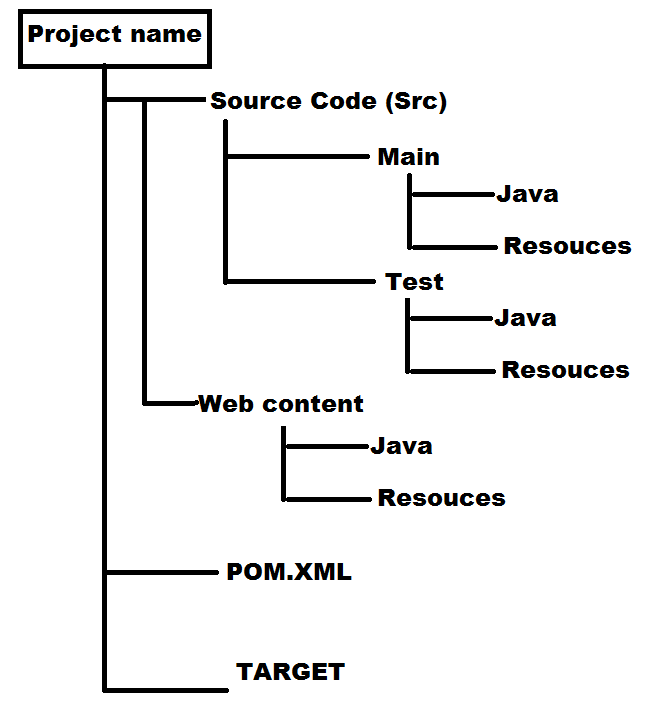
6.install

7.Deploy

**Maven Default life cycle**



**MAVEN FOLDER STRUCTER :**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **Element Description** | | |  |
| **Project** | It is the root element of pom.xml file. | | |
| **Model Version** | 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. | | |
| **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 compile, provided, runtime, test and system. | |

**POM.XML File Structure**

<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>com.companyname.projectgroup</groupId>

<artifactId>project</artifactId>

<version>1.0</version>

<packaging>jar</ packaging >

<Properties>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</ maven.compiler.target >

</properties>

<dependencies>

<dependency>

<groupiId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

</dependencies>

<profile>

   <id>openncp</id>

   <activation>

   <activeByDefault>true</activeByDefault>

   </activation>

   <properties>

<sonar.jdbc.url>jdbc:mysql://ipaddress:3306/sonar?useUnicode=true&amp;char acterEncoding=utf8&amp;rewriteBatchedStatements=true</sonar.jdbc.url>

    <sonar.jdbc.driver>com.mysql.jdbc.Driver</sonar.jdbc.driver>

    <sonar.jdbc.username>sonar</sonar.jdbc.username>

    <sonar.jdbc.password>sonar</sonar.jdbc.password>

    <sonar.host.url>http://sonar.gnomon.com.gr/sonar/</sonar.host.url>

    </properties>

</profile>

<distributionManagement>

<repsitory>

<id>releases</id>

<url>http://localhost:8081/nexus/contenct/repositories/thirdparty/ </url>

<repository>

<snapshotRepository>

<id>snapshots</id>

<url> http://localhost:8081/nexus/contenct/repositories/snapshots/</url>

</snapshotRepository>

</distributionManagement>

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-pulgin</artifactId>

<version>2.2</version>

<configuration>

<port>9000</port> //configure port number

<path>/spring5-webmvc-demo</path> //configure application root URL

</configuration>

</plugin>

</plugins>

Now open command console, go to the folder containing pom.xml and execute

the following mvn command.

**Maven Installation**

In linux machine

In linux machine using with wget command to download the java jdk , apache maven tar files from respectively sites.

In Windows open the browser download the java jdk , apache maven tar files from respectively sites.

Open winscp tool:-

copy the java jdk and apache maven file in to linux machine

**step1:**

#cd /opt

Create 3 directories

#mkdir java maven mvn-test

move the java jdk ,apache maven files respectively directories

step2:

java environmental variables configuration

#mv jdk. **xx**.tar.gz /opt/java

#tar -xvf jdk. **xx**.tar.gz

#cd java/jdk. **xx**

#pwd

/opt/java/jdk.**xx**

copy the above path

#vi /etc/profile.d/java.sh

export Java\_Home=/opt/java/jdk.**xx**

export PATH=${JAVA\_HOME}/bin:${PATH}

wq!

#source /etc/profile.d/java.sh

#java --version

java environmental variables configured successfully

Apache maven environmental variables configuration

#mv apache-maven. **xx**.tar.gz /opt/maven/

#tar -xvf apache-maven. **xx**.tar.gz

#cd maven/apache-maven. **xx**

#pwd

/opt/maven/ apache-maven.**xx**

copy the above path

#vi /etc/profile.d/maven.sh

export M2\_Home=/opt/maven/ apache-maven.**xx**

export PATH=${M2\_HOME}/bin:${PATH}

wq!

#source /opt/maven/ apache-maven.**xx**

#mvn --version

maven environmental variables configured successfully

**Maven working Process**

#cd /opt

#mkdir mvn-test

#cd mvn-test

#mvn archetype: generate

(using with **archetype: generate** command to creating project structure )

choose a number or apply filter format:1140

(1-1400 project structure are having the apache central repository)

choose a number:6

Define value for property 'group ID' : devops

Define value for property 'artifactory ID' :Project1

'version 1.0' snapshot: ENTER

'Package devops' : ENTER

Y ENTER