Maven Build Tool

Table of Contents

Introduction

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation.

The most powerful feature is able to download the project dependency libraries automatically from maven central repo, maven remote repo or local repo.

Pre requisites

- Java
- XML

Maven Details	
Type	JAVA based Open source Build Technology
Vendor	Apache
Is Open Source?	Yes
Version	3.x
Operating system	Cross Platform
Software Download URL	
Is executable software?	No, download as zip, extract and use it.
Reference Websites	1)

Installation of Maven software

---

1) Download Maven software as a ZIP file using below url

http://maven.apache.org/download.cgi

2) Copy apache-maven-3.5.2-bin.zip it into C:\Maven drive and extract into this(C:\Maven) directory, we will see four folders namely

C:\Maven\apache-maven-3.5.2

1) \bin ----> It contains the batch files.

2) \conf ----> It contains the configuration files like settings.xml.

3) \lib ----> It contains the jar files.

Contents in pom.xml

```
<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/P
OM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
 <groupId>com.bt</groupId>
  <artifactId>maven-java-project</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies>

 <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
    <version>3.8.1</version>
     <scope>test</scope>
 </dependenc>
  </dependencies>

</project>
```

All POM files require the **project** element and three mandatory fields: **groupId, artifactId,version.**

Root element of POM.xml is **project** and it has three major sub-nodes .

groupId **:** This is an Id of project's group. This is generally unique amongst an organization or a project. For example, a banking group com.company.bank has all bank related projects.
artifactId : This is an Id of the project.This is generally name of the project. For example, consumer-banking. Along with the groupId, the artifactId defines the artifact's location within the repository.

version: This is the version of the project.Along with the groupId, It is used within an artifact's repository to separate versions from each other. For example: com.company.bank:consumer-banking:1.0com.company.bank:consumer-banking:1.1.

Maven Repositories

A Maven repository can be one of the following types:

- Maven Local Repository
- Maven Central Repository
- Remote Repository

Maven Local Repository

The **maven** local repository is a local folder that is used to store all your project's dependencies (plugin jars and other files which are downloaded by Maven). In simple, when you build a Maven project, all dependency files will be stored in your Maven local repository.

By default, Maven local repository is default to `.m2` folder:

1. Unix/Mac OS X – `~/.m2/repository` (`/Users/MithunReddy/.m2/repository`)
2. Windows – `C:¥Documents and`

`Settings¥{your-username}¥.m2/repository`

Normally, We will change the default local repository folder from default `.m2` to another more meaningful name as follows.

Find {M2_HOME}\conf\setting.xml, update localRepository to something else.

<!-- localRepository

The path to the local repository maven will use to store

artifacts. Default: ${user.home}/.m2/repository

<localRepository>/path/to/local/repo</localRepository>

-->

<localRepository>**D:/maven_custom_repo**</localRepository>

Maven Central Repository

The central repository is the repository provided by the Maven community. It contains a large repository of commonly used libraries. This repository comes into play when Maven does not find libraries in the local repository.

The central repository can be found at: http://search.maven.org/.

Default Lifecycle

validate validate the project is correct and all necessary information is available.

compile compile the source code of the project.

test run tests using a suitable unit testing framework. These tests should not require the code be packaged or deployed.

package take the compiled code and package it in its distributable format, such as a JAR, WAR, EAR….

i**nstall :** install the package into the local repository, for use as a dependency in other projects locally.

Deploy:copies the final package to the remote repository for sharing with other developers and projects.

Clean Lifecycle:

clean : remove all files generated by the previous build

Site Lifecycle

site generate the project's site documentation

Maven Goals

To invoke a maven build you set a life cycle goal.

mvn install : Invokes validate, compile, test, package,

install. mvn clean: Invokes just clean.

mvn clean compile : Clean old builds and execute validate,

compile. mvn compile install : Invokes validate, compile,

test, package, install. mvn test clean : Invokes validate,

compile, test then clean.

Maven with SonarQube

Note: For this we no need to create the sonar-project.properties file. It will take all the details from pom.xml.

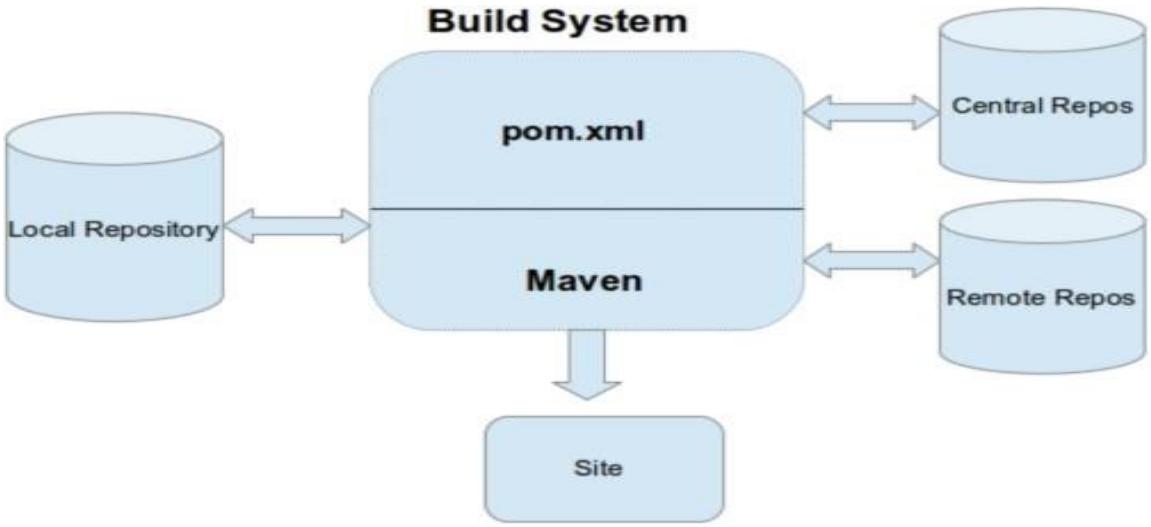
To generate SonarQube scanner report for maven projects need to add below lines in pom.xml.

Remote Repository

Enterprises usually maintain their own repositories for the libraries that are being used for the project. These differ from the local repository; a repository is maintained on a separate server, different from the

developer's machine and is accessible within the organization.

Maven Architecture



Maven Life cycles

There are 3 life cycles in maven.

a) clean

b) site

c) default

Default Lifecycle

validate validate the project is correct and all necessary information is available.

compile compile the source code of the project.

test run tests using a suitable unit testing framework. These tests should not require the code be packaged or deployed.

package take the compiled code and package it in its distributable format, such as a JAR, WAR, EAR….

i**nstall :** install the package into the local repository, for use as a dependency in other projects locally.

deploy copies the final package to the remote repository for sharing with other developers and projects.

Clean Lifecycle:

clean : remove all files generated by the previous build

Site Lifecycle

site generate the project's site documentation

Maven Goals

To invoke a maven build you set a life cycle goal.

mvn install : Invokes validate, compile, test, package,

install. mvn clean: Invokes just clean.

mvn clean compile : Clean old builds and execute validate,

compile. mvn compile install : Invokes validate, compile,

test, package, install. mvn test clean : Invokes validate,

compile, test then clean.

Maven with SonarQube

Note: For this we no need to create the sonar-project.properties file. It will take all the details from pom.xml.

To generate SonarQube scanner report for maven projects need to add below lines in pom.xml.

```
<profile
    s>
    <profile>
      <id>sonar</id>
      <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
        <sonar.host.url>http://localhost:9000/</sonar.host.url>
        <sonar-maven-plugin.version>3.3.0.603</sonar-maven-
        plugin.version>
    </properties>
      <build>
      <plugins>
       <plugin>
          <groupId>org.sonarsource.scanner.maven</groupId>
          <artifactId>sonar-maven-plugin</artifactId>
          <version>${sonar-maven-plugin.version}</version>
       </plugin>
```

```
        </plugins>
        </build>
    </profile>
</profiles>
```

Once you add you need to trigger the below command from the project directory to which you are generating report.

mvn

sonar:sonar

(OR)

mvn clean install sonar:sonar

In SonarQube dashboard you will find the project with the name which you mentioned in **<name>Maven-Java-Project</name>** tag and the key it will take from
<groupId>com.mt</groupId> and <artifactId>maven-java-project</artifactId>
as follows.

com.mt: maven-java-project , and version it will take from **<version>1.0</version>**
tag.

Note: If we want to generate sonarqube report for all Maven projects, we have to add above lines in each project, pom.xml. Instead of copying in each pom.xml, just copy into **settings.xml** which is available in **conf** directory.

In two locations settings.xml file may be

existed. The Maven install:

${maven.home}/conf/settings.xml A user's

install: ${user.home}/.m2/settings.xml

The former settings.xml are also called global settings, the latter settings.xml are referred to as user settings. If both files exists, their contents gets merged, with the user-specific settings.xml being dominant.

Maven with Remote Repositories

If we want to configure nexus remote repo with Maven, need to follow the below steps.

Remote Repository add

```
<repositories>
 <repository>
    <id>comt.mt</id>
    <url>https://maven.java.net/content/repositories/public/</url>
  </repository>
 </repositories>
```

To install the Oracle jdbc drivers :

2.1 ojdbc6.jar

```
$ mvn install:install-file -Dfile={Path/to/your/ojdbc6.jar}
    -DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=11.2.0 -
    Dpackaging=jar
```

2.2 ojdbc7.jar

$ mvn install:install-file -Dfile={Path/to/your/ojdbc7.jar}

-DgroupId=com.oracle -DartifactId=ojdbc7 -Dversion=12.1.0 -

Dpackaging=jar Full example to install a ojdbc7.jar

Terminal

> mvn install:install-file -Dfile=C:\\OracleJDBC\\ojdbc7.jar

-DgroupId=com.oracle -DartifactId=ojdbc7 -Dversion=12.1.0 -Dpackaging=jar

After installation, you must add below dependency in pom.xml as follows.

```
<!-- ojdbc6.jar example -->
    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc6</artifactId>
      <version>11.2.0</version>
    </dependency>

    <!-- ojdbc7.jar example -->
    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc7</artifactId>
      <version>12.1.0</version>
    </dependency>
```

---

mvn install:install-file -DlocalRepositoryPath=repo -DcreateChecksum=true - Dpackaging=jar -Dfile=/Users/BhaskarReddyL/Jars/BluePagesJavaToolKit-3.jar -DgroupId=com.mt -DartifactId=BluePagesJavaToolKit-3.jar -Dversion=1.0

pom.xml

```
<repositories>
      <repository>
            <id>com.mt</id>
            <url>file://${project.basedir}/repo</url>
      </repository>
</repositories>
```

What is the difference between -DskipTests and -Dmaven.test.skip=true ?
Ans)

---

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=com.mt -DartifactId=Maven- standalone-Project -DinteractiveMode=yes

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-webapp -DgroupId=com.mt -DartifactId=Maven-Web-
Project -

DinteractiveMode=yes mvn

eclipse:eclipse

mvn jetty:run

Generate the Java standalone project
==========

mvn -B archetype:generate \

```
    -DarchetypeGroupId=org.apache.maven.archetypes \
    -DgroupId=com.mt.devops.maven \
    -DartifactId=helloworld

----------------------------------------------------------------------------------------------------
--------------
```