MAVEN

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 to local repo.

Maven definition:

Maven is a powerful *project build and management tool* that is based on POM (project object model). It is used for java projects build, dependency and documentation.

Pom.xml is the heart of maven.

Few contents of 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/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.mydept</groupId>
<artifactId>maven-java-project</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>
<name>maven-java-project</name>
<url>http://maven.apache.org</url>

<properties>
```

```
        <project.build.sourceEncoding>UTF-
        8</project.build.sourceEncoding>
</properties>
<dependencies>
        <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>3.8.1</version>
                <scope>test</scope>
        </dependency>
</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:1.0 com.company.bank:consumer-banking:1.1.

Maven Repositories

A Maven repository can be one of the following types:

1) Local Repository

2) Central Repository

3) Remote Repository

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 (/Users/Lenovo/.m2/repository)

2) Windows – C:\Users\{your-username}\.m2

Update Maven Local Repository or Custom repository:

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

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

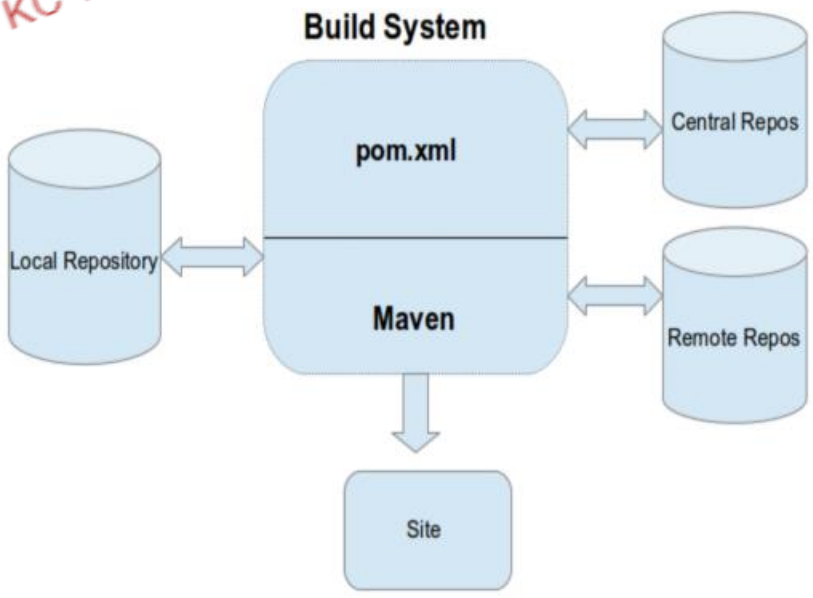
```
<!-- localRepository
| The path to the local repository maven will use to store artifacts.
|
| Default: ${user.home}/.m2/repository
<localRepository>/path/to/local/repo</localRepository>
-->
<localRepository>D:/maven_custom_repo</localRepository>
```

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/#browse.

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 Life cycles: clean, default and site.

Maven Goals:

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

mvn install: Invokes generate*, compile, test, package, integration-test, install.

mvn clean: Invokes just clean.

mvn clean compile: Clean old builds and execute generate*, compile.

mvn compile install: Invokes generate*, compile, test, integration-test, package, install.

mvn test clean: Invokes generate*, compile, test then clean.

*Multiple goals can be used with combinations.

Ex: mvn clean install

Maven Environment Setup

M2_HOME

JAVA_HOME

Create an environment variable for maven and set it to PATH variable.

mvn archetype:generate : This will download the maven required plugins to .m2/repository folder for the first time when you run this command. Also, it will create the project structure. It will ask the artifactid, groupid etc in the command prompt.

While creating project structure, we have to specify specific numbers to create different project structures like standalone, web projects.

mvn archetype:generate > file.txt : This command will generate all the numbers for different projects and creates

file.txt and loads the generated numbers to that file.

Much more explanation about Maven

groupid: Generally, it is the organization name.

artifactid: Generally, it is project name.

version: It is the version of the project. If version contains SNAPSHOT, then it means project is under development (not released) and its unstable.

modelVersion: It is the maven version. Mostly it is 4.0.0 always.

properties: These are required library versions need to be used by maven. Ex: java, spring, JUnit, log4j etc etc.

packaging: This is the type of packaging we need to build, like ear, war, jar or pom. Generally, parent project will contain pom as packaging which is parent of all packaging's. In Child projects(modules) we specify required packaging name like ear, war or jar.

target directory: target folder contains the result of the maven goal.

mvn complie: will compile the code and places generated .class files (results) in target.

mvn clean: will remove maven target directory.

mvn package: will generate the package and keeps the ear,war in target. This will generate the ear/war/jar with below format:

artifactid-version.packagingtype. Ex: helloworld-0.0.1-SNAPSHOT.war

To test maven, download spring-petclinic example from google.

https://github.com/spring-projects/spring-petclinic

goal clean: means, maven will delete the old targets(old build generated ear,war,jar files)

What are the build life cycles in Maven:

Ans) Maven has 3 build life cycles. 1)clean, 2) site and 3)default

Each life cycle has its own goals.

Clean important goals are: clean

Site important goals: site

Default important goals with order: validate, compile, test, package, install, deploy---------> These are very important

For example, if you run **mvn test**, then it will run validate, compile and then test.

validate: will validate project structure like src, java, WEBINF, web.xml etc files.

compile: used to compile

test: will run JUnit test cases. Ex: mvn test

package: Will create the packaging mentioned in packaging element and placed the artifact in target folder.

install: will keep the generated ear,jar,war under .m2/repository with group id and artifact id and also placed in target folder.

deploy: will keep the generated ear,jar,war in remote repository.

integration-test: process and deploy the package if necessary into an environment where integration tests can be ran..

The only goal we can skip in Maven life cycle is **test**.

Below is command to skip test (junit test cases):

mvn clean install -DskipTests (But it will compile JUnit test cases)

Below is command to skip test (JUnit test cases) and skip compilation of junit test cases: **mvn clean install -Dmaven.test.skip=true**

Difference between goals package and install:

package: To create the packaging mentioned in packaging element and place the generated artifact in **target** directory.

install: Will keep the generated ear,jar,war under .m2/repository with group id and artifact id and also places in target directory.

Maven Profiles:

For different environments like dev, stage, uat, prod we need different values from different property files. So, to read different property files we use profiles.

Ex: **mvn clean install -P staging**

In maven pom.xml,

Add the profiles elements. The name in pom.xml profiles and the name given in mvn command should be same.

Below is example of profiles in pom.xml:

```
<profiles>
    <profile>
        <id>local</id>
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
    </profile>
    <profile>
        <id>stagging</id>
        <properties>
            <env>stagging</env>
        </properties>
    </profile>
    <profile>
        <id>production</id>
        <properties>
            <env>production</env>
        </properties>
```

```
        </profile>
    </profiles>
```

Plugin management in maven:

Scopes in Maven: