MAVEN

DRAWBACKS:-

We have many problems with project development like

1. We need to add set of jar files in each project, these jar files must also include dependencies
2. If we don’t create right project structure, the project will not be executed
3. We need to write scripts for any tasks

WHAT IS BUILD TOOL:-

Source code = the actual program/code written by programmer in human readable format ex: - .java files

Byte code = the binary file which is machine language ex:- .class files

Build tool is a program that takes source code and compile and then finally creates an executable file

Source code (.java files) will be compiled then we get .class files. These .class files are packaged into jar/war files

MAVEN INTRODUCTION:-

Maven is a powerful **build automation tool** which is basically used for java projects. (Building means taking your source code then compiling and archiving into single file)

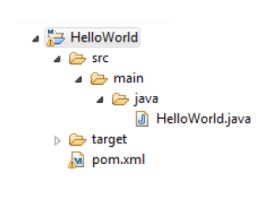
Maven is a powerful **project management tool** that is based on POM (project object model).

Maven helps in code building, dependency management, document creation, site publication

Maven will download jar, plugin dynamically

Maven contains lifecycle

Mavens supports 1700 directory structures



MAVEN WORK FLOW:-

We install & configure maven and select a directory structure. We configure pom.xml now when the code is done, we build the source code using maven. During that time maven will generate target folder which has .class file for all source code and download all dependencies, libraries, jar…. Finally we will get build artifact (war/jar/ear) as an outcome

MAVEN REPOSITORIES:-

A maven repository is a place where all the java related jars and plugins are maintained

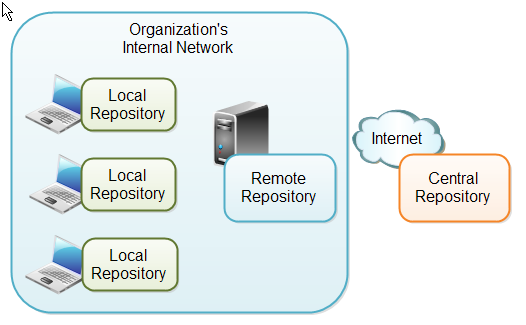
There are 3 types of maven repositories

1. Local repo = this repository is created whenever you run mvn command for the first time. The local repo present in local system of developer. maven by default downloads the libraries in (.m2) which is in users home directory

Ex: - c:\user\raju\.m2\repository

1. Public/Central repo = it is the centralized repository maintained by maven community. Maven knows the central repo location
2. Remote repo = A repository which is maintained within the organization (your own companies libraries) ex: - jfrog/nexus. You have to manually configure remote repository location in pom.xml

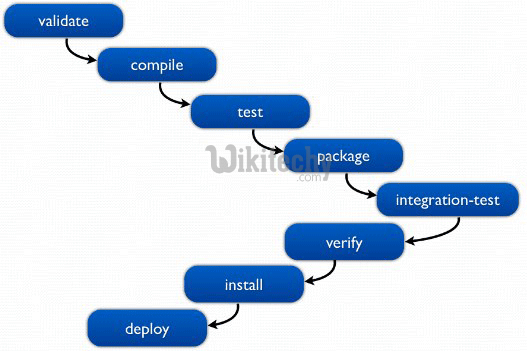
Maven first searches in local repository for dependencies, if not found then searches in central repository, even then maven didn’t find then it searches in remote repository as how you mention in pom.xml



MAVEN BUILD LIFECYCLES:-

There are 3 lifecycles in maven

1. default:- in this default lifecycle we have 6 phases



1. Validate = checks if pom.xml exists

We know that dependencies are not mentioned in source code. So we write all dependencies in pom.xml. When you run **#mvn validate** then it will check dependencies download it and store in /home/user/.m2/repository/Dependency

1. Compile = it will compile the source code

When you run **#mvn compile,** it will compile the source code and generate a .class file under target directory

1. Test = tests the compiled source code using unit testing framework like junit
2. Package = takes the compiled code and package into “jar/war/ear” as how you defined in pom.xml
3. Verify =
4. Install = install phase will push artifact(jar/ear/war) to your local repository (.m2)
5. Deploy = this phase will push artifacts from local repo(.m2) to remote repository

\*To use maven phases

#mvn phase (or) #mvn clean phase

\*To execute any mvn commands you need to be present in pom.xml location

1. Clean:-
2. Site:-

MAVEN POM.XML:-

POM means Project Object Model. The pom.xml contains 4 sections

1. **project** = it is all about project information
2. ***model version*** = which version of pom.xml are you using
3. ***group id*** = grouping structure which is unique for each project
4. ***artifact id*** = name of the artifact ex:- helloworld.war
5. ***version*** = version of the artifact, it maintains version of each artifact
6. ***packaging*** = the outcome of the build weather jar/war/ear

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>com.infosys.project1</groupId>

<artifactId>HelloWorld</artifactId>

<version>1.0</version>

<packaging>jar</packaging>

</project>

1. **dependencies** = in this section you can have n dependency which are used for application
2. ***group id*** = grouping structure of dependency found in their site
3. ***artifact id*** = name of the dependency
4. ***version*** = specific version of dependency

Ex:- junit\_1.0.jar

<dependencies>

<dependency>

<groupId>org.apache.junit</groupId>

<artifactId>junit</artifactId>

<version>1.0</version>

</dependency>

</dependencies>

1. **build** =
2. **distribution management** = the remote repository location need to be specified here/ where we download/push artifacts to
3. ***snapshot*** = partial completed jar files is called snapshot (or) it indicates that project is in development
4. ***release/final*** = official release of jar is called release/indicating that project is in production

In case of version maven downloads only once and never try to download newer version but in case of snapshot maven will automatically fetch the latest snapshot every time when a project is built

Ex: - there is a developer "Adam" who is working from January to June. "Adam" can officially release the code in June. Say there is other developer "john" who is working from February to September. Now imagine if john wants a jar file from Adam then he has to wait till June where john work will delayed. So john asks Adams partial jar and then works on it

\* In our remote repositories we maintain two types of repositories

In pom.xml under project you have something like this <version>1.0snapshot</version>. This tells maven to push it to snapshot repository. If nothing is mentioned in version then it is telling maven to push to release version

<repositories>

<repository>

<id>my-internal-site</id>

<name>sonarnexus</name>

<url>https://host:8081/repositories/releases</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

Enable = false means which means maven will not look for updates for this repository

There are 2 pom.xml

1. parent pom.xml:-
2. child pom.xml:-

To generate project architecture

#mvn archetype:generate -DgroupId=groupid -DartifactId=artifactid

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=booleanValue

Ex:-

mvn archetype:generate -DgroupId=com.javatpoint -DartifactId=CubeGeneratorWeb

-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

Maven has plugin called “exec” which runs any o.s commands

Ex:- mvn package exec:exec

EX: - There is a calculator application which has 4 modules addition, subtraction, multiplication & division

Now on each module each developer works and push it to source code repository also each developer tests their code that is called unit testing

Ex:-the developer who worked on addition module will check weather his code is working correct

* after every developer push the code maven will build the source code and generate a build artifact and that is deployed on test servers now testers test the application which is called integration testing
* dev build = building one component (building only addition module)
* incremental build = integrating and building latest code
* qa build = whole component build