Maven
========

other build tools are :: gradle,ant , ivy , buildr and etc..

type:: Java based Build tool

Build Process:: The process of keeping application ready for execution or deployment or release is called
Build process...

=> Every build process contains multiple repeatative operations like
- -> Project creation
- -> packages creation
- -> source code development
- -> compilation
- -> deployment directotry structure creation
- -> add jar files/dependencies/ libraries to CLASSPATH and to other places
- -> running unit testcases (Juni test cases)
- -> starting servers
- -> packing the project content into jar /war file
- and etc..
- -> deployment of the web application
- -> deleting the temporary folders and files

jar file :: Java archive -- represents library/api/standalone java project
war :: Web application archieve -- represents java web application (website)

=> until we get satisfactory results we need to perform the above operations of build process for multiple times .. Instread doing those operations manually it is recomanded to automate them by taking the support of Build tools

List of Build Tools

- =>Ant (Another Neat Tool)
- => Maven (1)
- => Gradle (2)
- => Ms Build (only for .net projects)
- => Ivy
- =>BuildR
- => Leigningen
- and etc..

- => Most of the Java Projects prefer using Maven as the Build tool
- => Some java projects and most of the non-java projects prefer using gradle
- => Maven can be used only for java Proejcts build process
- => Gradle can be used for both java and non-Java Projects build process

note: maven tool can be used only in Java Projects where as Gradle tool can be used for both java and non java Projects

Maven advantages
=================

=> Provides archetypes as the Project Templates
[These archetypes can be used to create different types of Projects with Directory ready made structures]

1000+ archetypes are given by maven , but two archetypes of are very popular
- -> maven-archetype-quickstart --> for standalone apps
- -> maven-archetype-webapp --> for web applications

->Stadandard drietories in the above archetypes are (projects with templates)
- src/main/java --> To place java packages and source files (.java files)
- src/test/java ---> to place unit test code packages and source files
- src/main/resources --> To place helper files like properties files, java script files and etc..

All these folders will be added to CLASSPATH automatically

note: Programmer's testing on his own piece of code is called Unit Testing

- target --> To see the outputs /resultant content
- maven dependencies ---> For jar files/dependencies

=> Maven takes care of Library MAnagement/Dependency MAnagement by interacting with multiple repositories (getting the libraries/jar files)
(storage place)

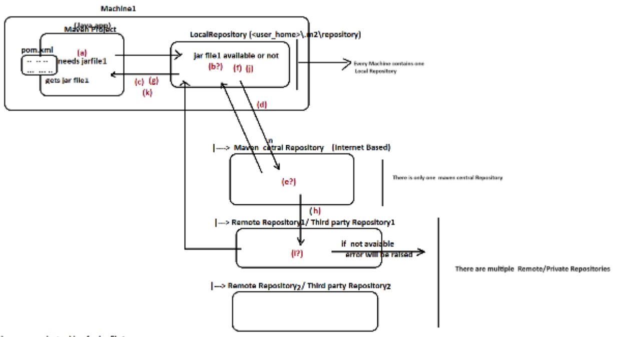
note:: we give inputs and instructions to maven build tool using pom.xml

POM : Project Object Management

->maven project can get the jar files(Libraries) required for the project directly from 3 types of Repositories

These repositories maintain libaries/ jar files , plugins, maven projects

- a) Local Repository (with in ur system <user_home>\.m2\repository))
  [C:\Users\NATARAJ\.m2\repository]
- b) Maven Cetral Repository [Internet Based -- https://repo.maven.apache.org/maven2/]
- c) Remote Reposotories/ Third Party Reposotories/ Private Repositories
  [Specific to certain organizations]
  eg:: IBM Repository , Jboss Repository and etc..



Machine1
(Java app)
Maven Project
pom.xml
(a)
needs jarfile1
gets jar file1
(c) (g)
(k)
LocalRepository (<user_home>\.m2\repository)
jar file1 available or not
(b?) (f) (j)
Every Machine contains one Local Repository
(d)
|---> Maven central Repository (Internet Based)
(e?)
There is only one maven central Repository
(h)
|---> Remote Repository1/ Third party Repository1
(i?)
if not available error will be raised
There are multiple Remote/Private Repositories
|---> Remote Repository2/ Third party Repository2

(a) maven project asking for jar file1
(b?) maven project first searches in maven Local Repository of the system

(c) if avalaibe
Brings the jar file1 from Local Repository and keeps in the Maven Project

(d) if not avaiable
(e) Maven tool searches for jar file1 in the maven central Repository

if avaiable
(f) collects the jar file1 from the central repository and keeps in Local repository
(g) adds the jar file1 to the Maven project

if not avaiable
(h) maven tool goes specified remote/third party repository for searching jar file1
(i?) searches for jar file1 in the remote/thrid party Repository

if available
(j) keeps the jar file1 in Local Repository
(k) adds the jar file1 to the maven project

if not available
-> maven tool throws the error

note:: Plugin is a patch /pluggable software that will provide additional facilities to the existing facilities of the software / software application

=> Maven tool supports transitive dependency management

-> if u specify any main jar file info in the Project , the maven tool will take care finding out all the dependent jar files of the hierarchy (dependents and their dependents , so on...) and downloads them to projects from various maven repositories

eg:: If we specify spring-context-support jar file information in the maven project it will give total 1+6 jar files(1 main jar file , 6 - dependent jar file) to the project

=> All the inputs and instructions to maven Project will be given using pom.xml file
pom :: Project object model

=> Maven tool provides built -in decompiler to get the source code(.java file) from the byte code (.class file)

=> Maven tool can run Junit test cases to generate the test repots (Jnuit can be integrated with maven)

=> Maven tool can generate Project Report having info about entire project (information about packages classes and other things)

=> maven tool can pack the code into jar file or war file and others..

=> In latest versions of eclipse IDE (2020+) , the maven is built-in tool in IDE..

=> Maven is self intelligent tool i.e it gives many facilties by convention.. becoz the maven is life cycle based declarative tool
(based on naming conventions and instructions given by pom.xml file we can achieve the results

=>Compiler is given to give java source code to byte code
eg: javac (built in with IDE)
=>DeCompiler is given to get java source code from byte code
eg: JIT,online decompilers and etc..

Important maven terminologies
==============================
- a) Archetypes (Project Templates)
- b) Artifact (Project name/jar file name/plugin name)
- d) Repository (Storage Place)
- e) POM (Project Object Model)
- f) Dependency (jar file)
- g) Transitive Dependency (Both main and dependent jar files)
- h) Build Process (Keeping the Project ready for execution)
- and etc..

Maven

type:: Java based Build tool

other build tools are :: gradle,ant, ivy, buildr and etc..

Build Process:: The process of keeping application ready for execution or deployment or release is called

Build process...

=> Every build process contains multiple repeatative operations like

->Project creation

-> packages creation

-> source code development

-> compilation

-> deployment directotry structure creation

-> add jar files/dependencies/libraries to CLASSPATH and to other places

-> running unit testcases (Juni test cases)

-> starting servers

-> packing the project content into jar /war file

and etc..

-> deployment of the web application

-> deleting the temporary folders and files

jar file:: Java archive -- represents library/api/standalone java project

war :: Web application archieve -- represents java web application (website)

=> until we get satisfactory results we need to perform the above operations of build process

for multiple times.. Instredthose operations manually it is recomanded to automate them by taking the support of Build tools

List of Build Tools

=>Ant (Another Neat Tool)

=> Maven (1)

=> Gradle (2)

=> Ms Build (only for .net projects)

=> Ivy

=> BuildR

=> Leigningen

and etc..

=> Most of the Java Projects prefer using Maven as the Build tool

=> Some java projects and most of the non-java projects prefer using gradle

=> Maven can be used only for java Proejcts build process

=> Gradle can be used for both java and non-Java Projects build process

Maven advantages

==================

=> Provides archetypes as the Project Templates

note: maven tool can be used only in

Java Projects where as Gradle tool can be used for both java and non java Projects

with

Directory

[These archetypes can be used to create different types of Projects

ready made structures]

1000+ archetypes are given by maven, but two archetypes of are very popular

-> maven-archetype-quickstart --> for standalone apps

-> maven-archetype-webapp ---> for web applications

->Stadandardrietories in the above archetypes are (projects with templates)

src/main/java ---> To place java packages and source files (.java files) src/test/java -----> to place unit test code packages and source files

src/main/resources ---> To place helper files like properties files, java script files and etc..

target ---> To see the outputs /resultant content maven dependencies ---> For jar files/dependencies

=> Maven takes care of Library Management/Dependency Management by interacting with multiple repositories (getting the libraries/jar files)

(storage place)

->maven project can get the jar files (Libraries) required for the project directly from 3 types of Repositories

maintain libaries/

These repositories

jar files, plugins, maven projects

a) Local Repository (with in ur system <user_home>\.m2\repository)) [C:\Users\NATARAJ\.m2\repository]

All these folders will be added to CLASSPATH automatically

note: Programmer's testing on his own piece of code is called Unit Testing

note:: we give inputs and instructions to maven build tool using pom.xml

POM: Project Object Management

b) Maven Cetral Repository [Internet Based -- https://repo.maven.apache.org/maven2/]

c) Remote Reposotories/ Third Party Reposotories/ Private Repositories [Specific to certain organizations]

eg:: IBM Repository, Jboss Repository and etc..

Machine1

Maven PiBect

LocalRepository (<user_home>\.m2\repository)

pom.kml

....

(a) needs jarfile1

jar file1 available or not

(b?) (f) (j)

gets jar file1

(c) (g)

(k)

(d)

Every Machine contains one

Local Repository

|----> Maven cetral Repository (Internet Based)

(e?)

(h)

|---> Remote Repository1/ Third party Repository1

There is only one maven central Repository

if not avaiable

(i?)

error will be raised

There are multiple Remote/Private Repositories

|---> Remote Repository2/ Third party Repository2

(a) maven project asking for jar file1

(b?) maven project first searches in maven Local Repository of the system

(c) if avaialbe

Brings the jar file1

from Local Repository

and keeps in the Maven Project

(d) if not avaiable

(e) Maven tool searches for jar file1 in the maven central Repository

if avaiable

(f) collects the jar file1

from the central repository

and keeps in Local repository

(g) adds the jar file1 to the Maven project

if not avaiable

(h) maven tool goes specified remote/third party repository for searching jar file1

(i?) searches for jar file1 in the remote/thrid party Repository

if available

if not available

(j) keeps the jar file1

-> maven tool throws the error

in Local Repository

note:: Plugin is a patch /pluggable software that will provide additional facilities to the existing facilities of the software/software application

(k) adds the jar file1

to the maven project

=> Maven tool supports transitive dependency management

-> if u specify any main jar file info in the Project, the maven tool will take care finding out

all the dependent jar files of the hierarchy (dependents and their dependents, so on...) and

downloads them projects from various maven repositories

eg:: if we specify spring-context-support jar file information in the maven project

it will give total 1+6 jar files(1 main jar file, 6 - dependent jar file) to the project

=> All the inputs and instructions to maven Project will be given using pom.xml fle pom :: Project object model

=> Maven tool provides built-in decompiler to get the source code(.java file) from the byte code (.class file)

=> Maven tool can "Junit test cases to generate the test repots (Jnuit can be integrated with maven)

=> Maven tool can generate Project Report having info about entire project (information about packages classes and other things)

=> maven tool can pack the code into jar file or war file and others..

=> In latest versions of eclipse IDE (2020+), the maven is built-in tool in IDE..

=> Maven is self intelligent tool i.e it gives many facilties by convention.. becoz the maven is life cycle based declarative tool (based on naming conventions and instructions given by

pom.xml file we can achieve the results

=>Compiler is given to give java source code to byte code

eg: javac (built in with IDE)

=>DeCompiler is given to get java source code from byte code

eg: JIT,online decompilers and etc..

Important maven terminologies

==============================

a) Archetypes (Project Templates)

b) Artifact (Project name/jar file name/plugin name)

d) Repository (Storage Place)

e) POM (Project Object Model)

f) Dependency (jar file)

g) Transitive Dependency (Both main and dependent jar files)

h) Build Process (Keeping the Project ready for execution)

and etc..