Maven

Maven is a project management tool/ build automation tool

Maven follows convention than configuration

mvn clean install – maven compile run tests and create jar file

we can add dependencies plugins apache-common etc

maven gets them from internet.

Maven is very tiny after installation

It uses plugin framework

It downloads plugins on the fly and uses them

To run junit test, for first time it downloads sure shark plugin

Maven will compile and dump into repos

Dependencies can be fetched from repos

Convention

Plugins

Pom.xml – project object model

Repositories

Directory/folder structure in java:

My project

src/main/java – main source code

src/main/resource – property file or config files

src/test/java – junit test

targets – copies all classes/jar files to targets

Maven automates build test deployment with very minimal convention.

If we follow the directory structure, maven takes care of test of things

Maven uses pom.xml - project object model

Using pom Compiles and builds a jar file out of project

Command to use – mvn

Maven downloads plugins on the fly and uses them

Repository: Maven will compile and create jar files into a repository

mvn clean

Or mvn install

Maven executes the life cycle phases

maven install - processing resources

compile

execute test

package project into jar/war files

install it

goals in maven are

Phase is like life cycle

Goal is like a task

compile:compile –

Left side is plugin

Right side is task

Goal is bound to Phase.

Phases	**Goal**
Process-resources	resources:resource
Compile	compile:compile
Process test resources	
Execute tests	
Package	jar:jar
Install	install:install

Maven jar:jar

First jar is Package life cycle phase and second jar is actual goal

Phase:goal

Maven makes life of java developer easy

Maven is project management tool java application tool

Developer might be using

Sprint mvc

Struts

Jpa – to connect to data base

War files

Deploy to tomcat websphere

Ant – we need to write config file manually

Maven archifacts – used to develop web applications

Power of Maven

POM

Build life cycle

Note: Maven downloads dependencies from internet (from maven repository)

In case of ant we need to download and place all in the machine

Ex: java ssh

Jcsh - java ssh

https://mvnrepository.com/artifact/com.jcraft/jsch/0.1.54

```
<!-- https://mvnrepository.com/artifact/com.jcraft/jsch -->
<dependency>
    <groupId>com.jcraft</groupId>
    <artifactId>jsch</artifactId>
    <version>0.1.54</version>
</dependency>
```

1) App.java App.class
2) App.Test.java App.Test.java.class
3) Run the unit test

4) Combine all jar files

Src/main

Src/test

Project Directory structure:

My_ project

-src/main/java – Java source code

-src/main/resources - property or xml configuration files

-src/test/java – test code (junit)

- targets – copies all config files into targets directory.

Example POM:

```
<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.project-group</groupId>
  <artifactId>project</artifactId>
  <version>1.0</version>
</project>
```

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

Projects notation in repository is 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.

POM contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, then executes the goal. Some of the configuration that can be specified in the POM are following:

project dependencies

plugins

goals

build profiles

project version

developers

mailing list

Before creating a POM, we should first decide the project group (groupId), its name(artifactId) and its version as these attributes help in uniquely identifying the project in 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

Goals in maven

clean

compile

install

package

jar

deploy

Build life cycle:

Prepare resources

The order of execution depends on the order in which the goal(s) and the build phase(s) are invoked. For example, consider the command below. The

clean and package arguments are build phases while the dependency:copy-dependencies is a goal.

mvn clean dependency:copy-dependencies package

Here the clean phase will be executed first, and then the dependency:copy-dependencies goal will be executed, and finally package phase will be executed

Clean phase example:

[root@reviewb consumerBanking]# ls -lrt target/classes/com/companyname/bank/App.class

-rw-r--r--. 1 root root 555 Jan 4 05:23 target/classes/com/companyname/bank/App.class

[root@reviewb consumerBanking]# mvn clean

[INFO] Scanning for projects...

[INFO]

[INFO] --

[INFO] Building consumerBanking 1.0-SNAPSHOT

[INFO] --

[INFO]

[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ consumerBanking ---

[INFO] Deleting /root/maven/consumerBanking/target

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 0.647 s

[INFO] Finished at: 2017-01-04T05:37:07+05:30

[INFO] Final Memory: 6M/56M

[INFO] --

[root@reviewb consumerBanking]# ls -lrt target/classes/com/companyname/bank/App.class

ls: cannot access target/classes/com/companyname/bank/App.class: No such file or directory

Sample 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/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.companyname.bank</groupId>

  <artifactId>consumerBanking</artifactId>

  <packaging>jar</packaging>

  <version>1.0-SNAPSHOT</version>

  <name>consumerBanking</name>

  <url>http://maven.apache.org</url>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>
```

```
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Command to execute:

mvn clean package

```
cd /root/maven/consumerBanking/target/classes
[root@reviewb classes]# java -classpath . com.companyname.bank.App
Hello World!
[root@reviewb classes]#
```

```
[root@reviewb classes]# cd /root/maven/consumerBanking/
[root@reviewb consumerBanking]# maven clean package
-bash: maven: command not found
[root@reviewb consumerBanking]# mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
```

[INFO] Building consumerBanking 1.0-SNAPSHOT

[INFO] --

[INFO]

[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ consumerBanking --
-

[INFO] Deleting /root/maven/consumerBanking/target

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @
consumerBanking ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered
resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory
/root/maven/consumerBanking/src/main/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @
consumerBanking ---

[INFO] Changes detected - recompiling the module!

[WARNING] File encoding has not been set, using platform encoding UTF-8,
i.e. build is platform dependent!

[INFO] Compiling 1 source file to
/root/maven/consumerBanking/target/classes

[INFO]

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources)
@ consumerBanking ---

[WARNING] Using platform encoding (UTF-8 actually) to copy filtered
resources, i.e. build is platform dependent!

[INFO] skip non existing resourceDirectory
/root/maven/consumerBanking/src/test/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @
consumerBanking ---

[INFO] Changes detected - recompiling the module!

[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!

[INFO] Compiling 1 source file to /root/maven/consumerBanking/target/test-classes

[INFO]

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ consumerBanking ---

[INFO] Surefire report directory: /root/maven/consumerBanking/target/surefire-reports

---

 T E S T S

Running com.companyname.bank.AppTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.028 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ consumerBanking ---

[INFO] Building jar: /root/maven/consumerBanking/target/consumerBanking-1.0-SNAPSHOT.jar

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 5.574 s

[INFO] Finished at: 2017-01-04T05:23:46+05:30

[INFO] Final Memory: 15M/56M

[INFO] --

Difference between ant and Maven

Ant	Maven
Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file.	Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file.
Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order.	Maven is **declarative**, everything you define in the pom.xml file.
There is **no life cycle** in Ant.	There is **life cycle** in Maven.
It is **a tool** box.	It is **a framework**.
It is **mainly a build tool**.	It is **mainly a project management tool**.
The ant scripts are **not reusable**.	The maven plugins are **reusable**.
It is **less preferred** than Maven.	It is **more preferred** than Ant.

Maven life cycle:

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

compile - compile the source code of the project

test - test the compiled source code 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.

verify - run any checks on results of integration tests to ensure quality criteria are met

install - install the package into the local repository, for use as a dependency in other projects locally

deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

Clean phase 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.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
<plugins>
  <plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-antrun-plugin</artifactId>
  <version>1.1</version>
  <executions>
    <execution>
      <id>id.pre-clean</id>
```

```
    <phase>pre-clean</phase>
    <goals>
      <goal>run</goal>
    </goals>
    <configuration>
      <tasks>
        <echo>pre-clean phase</echo>
      </tasks>
    </configuration>
  </execution>
  <execution>
    <id>id.clean</id>
    <phase>clean</phase>
    <goals>
     <goal>run</goal>
    </goals>
    <configuration>
      <tasks>
        <echo>clean phase</echo>
      </tasks>
    </configuration>
  </execution>
  <execution>
    <id>id.post-clean</id>
    <phase>post-clean</phase>
    <goals>
```

```
            <goal>run</goal>
          </goals>
          <configuration>
            <tasks>
              <echo>post-clean phase</echo>
            </tasks>
          </configuration>
        </execution>
      </executions>
      </plugin>
</plugins>
</build>
</project>

[root@reviewb consumerBanking]# cat superpom.xml  | grep -i phase
            <phase>clean</phase>
            <phase>process-test-resources</phase>
            <phase>process-resources</phase>
            <phase>package</phase>
            <phase>compile</phase>
            <phase>test-compile</phase>
            <phase>test</phase>
            <phase>install</phase>
            <phase>deploy</phase>
            <phase>site</phase>
            <phase>site-deploy</phase>
```

Superpom is maven’s default pom

All POMs extend maven super pom

```
[root@reviewb consumerBanking]# cat superpom.xml  | grep -i goal
          <goals>
            <goal>clean</goal>
          </goals>
          <goals>
            <goal>testResources</goal>
          </goals>
          <goals>
            <goal>resources</goal>
          </goals>
          <goals>
            <goal>jar</goal>
          </goals>
          <goals>
            <goal>compile</goal>
          </goals>
          <goals>
            <goal>testCompile</goal>
          </goals>
          <goals>
            <goal>test</goal>
          </goals>
          <goals>
```

```
  <goal>install</goal>
 </goals>
 <goals>
  <goal>deploy</goal>
 </goals>
 <goals>
  <goal>site</goal>
 </goals>
 <goals>
  <goal>deploy</goal>
 </goals>
```

```
[root@reviewb consumerBanking]# mvn clean compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building consumerBanking 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ consumerBanking ---
[INFO] Deleting /root/maven/consumerBanking/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @
consumerBanking ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
```

[INFO] skip non existing resourceDirectory /root/maven/consumerBanking/src/main/resources

[INFO]

[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ consumerBanking ---

[INFO] Changes detected - recompiling the module!

[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!

[INFO] Compiling 1 source file to /root/maven/consumerBanking/target/classes

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 3.239 s

[INFO] Finished at: 2017-01-04T05:42:37+05:30

[INFO] Final Memory: 12M/56M

[INFO] --

[root@reviewb consumerBanking]#

How to generate pom.xml :

archetype:generate -DgroupId=com.companyname.bank -DartifactId=consumerBanking -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

it generates a pom.xml

[root@reviewb 2]# mvn archetype:generate -DgroupId=com.companyname.bank -DartifactId=consumerBanking -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

[INFO] Scanning for projects...

[INFO]

[INFO] --

[INFO] Building Maven Stub Project (No POM) 1

[INFO] --

[INFO]

[INFO] >>> maven-archetype-plugin:2.4:generate (default-cli) > generate-sources @ standalone-pom >>>

[INFO]

[INFO] <<< maven-archetype-plugin:2.4:generate (default-cli) < generate-sources @ standalone-pom <<<

[INFO]

[INFO] --- maven-archetype-plugin:2.4:generate (default-cli) @ standalone-pom ---

[INFO] Generating project in Batch mode

[INFO] --

[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0

[INFO] --

[INFO] Parameter: basedir, Value: /2

[INFO] Parameter: package, Value: com.companyname.bank

[INFO] Parameter: groupId, Value: com.companyname.bank

[INFO] Parameter: artifactId, Value: consumerBanking

[INFO] Parameter: packageName, Value: com.companyname.bank

[INFO] Parameter: version, Value: 1.0-SNAPSHOT

```
[INFO] project created from Old (1.x) Archetype in dir: /2/consumerBanking
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 17.184 s
[INFO] Finished at: 2017-01-04T05:56:02+05:30
[INFO] Final Memory: 14M/56M
[INFO] ------------------------------------------------------------------------
[root@reviewb 2]# ls -lrt
total 0
drwxr-xr-x. 3 root root 32 Jan  4 05:56 consumerBanking
[root@reviewb 2]# cd consumerBanking/
[root@reviewb consumerBanking]# ls
pom.xml  src
[root@reviewb consumerBanking]# vi pom.xml
[root@reviewb consumerBanking]#


[root@reviewb consumerBanking]# cat 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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.companyname.bank</groupId>
  <artifactId>consumerBanking</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
```

```
  <name>consumerBanking</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Phase vs goal:

phase	**Goal**
process-resources	resources:resources
compile	compile:compile
process test resources	
execute test resoruces	
package	jar:jar
install	install:install

mvn install --> executes life cycle phase

goal is a task

single goal is bound to life cycle phase

if we select package as the phase using mvn package, all the earlier phases till package (process-resources,compile,process test resources, execute test resources,package) will be executed

or we can execute maven using goal also

mvn compile:compile syntax: mvn plugin:task

all the goals till compile will be executed.

jar is plugin jar is the task

[root@reviewb consumerBanking]# mvn jar:jar

[INFO] Scanning for projects...

[INFO]

[INFO] --

[INFO] Building consumerBanking 1.0-SNAPSHOT

[INFO] --

[INFO]

[INFO] --- maven-jar-plugin:2.4:jar (default-cli) @ consumerBanking ---

[WARNING] JAR will be empty - no content was marked for inclusion!

[INFO] Building jar: /2/consumerBanking/target/consumerBanking-1.0-SNAPSHOT.jar

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 1.503 s

[INFO] Finished at: 2017-01-04T06:03:16+05:30

[INFO] Final Memory: 7M/56M

[INFO] --

[root@reviewb consumerBanking]#

Repository manager:

How to improve the maven build performance:

Mvn install or build maven downloades third party binaries from repository and dump on the machine somewhere

.M2 directory

Maven takes time to download

Apache access tool

Spring framework

Hibernate

All jars if they are declared as dependencies they will be pulled from public repository.

To overcome this issue: we have maven local repos

Acts as proxy for public repos

Acts as internal project repos

Settings.xml maven config file, instead of pulling from public repos, pulls from local repos

Nexus is a easy maven repos from sonatype

Nexus is a repository

We can configure like: use only licensed apache jar files

Deploy using maven:

Define repository as belo

```
[...]
  <distributionManagement>
    <repository>
      <id>internal.repo</id>
      <name>MyCo Internal Repository</name>
      <url>Host to Company Repository</url>
    </repository>
  </distributionManagement>
[...]
```

Define server details in settings.xml

```
[...]
    <server>
      <id>internal.repo</id>
      <username>maven</username>
      <password>foobar</password>
    </server>
```

[...]

Goals Summary:

Validate – to validate project to check if any info is available.

Compile – compile a source code in src directory

Test – compile test code in test directory

Package - create jar

Integration-test – if any integration tests are there

Verify - to check if the package is valid

Install – will install the package into local repository to use it as a dependency in another project

Deploy - copies the package to remote repository.