

Learn Maven:

- 1) Why
- 2) Installation
- 3) Maven Concept
- 4) 3.1) Basic command
- 5) 3.2) Maven life cycle
- 6) 3.3) Maven plugin
- 7) Maven Repository
- 8) Maven inheritance (parent)
- 9) Maven aggregation(module)
- 10) Maven profile

WHY

Its is build tool for java provide many feature (1. build artifacts 2. dependency management etc..) it is next of ant build tool. Current alternative of maven and ant both is Gradle.

Installation

- 1) Download from below link.
<https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip>
- 2) Unzip the file
- 3) Create MAVEN_HOME env variable
- 4) Add % MAVEN_HOME %/bin into PATH variable

Verification of installation

`mvn -version` or `mvn -v`

```
C:\Users\inkajm01>mvn --version
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-04T01:09:06+05:30)
Maven home: D:\mySoftware\sh-soft\apache-maven-3.5.0\apache-maven-3.5.0\bin\..
Java version: 1.8.0_71, vendor: Oracle Corporation
Java home: D:\myProjects\jdk_home\jdk1.8.0_71\jre
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

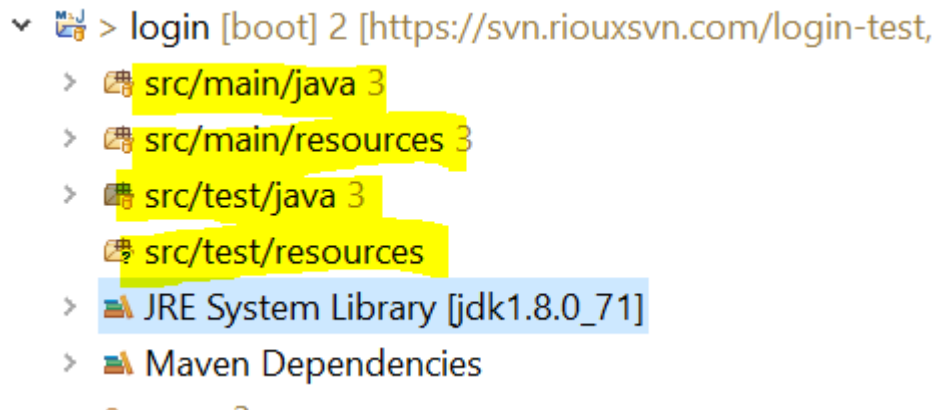
C:\Users\inkajm01>mvn -v
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-04T01:09:06+05:30)
Maven home: D:\mySoftware\sh-soft\apache-maven-3.5.0\apache-maven-3.5.0\bin\..
Java version: 1.8.0_71, vendor: Oracle Corporation
Java home: D:\myProjects\jdk_home\jdk1.8.0_71\jre
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Maven Concept

- 1) Maven also follow directory structure in java project.

src/main/java	-for application java file
src/main/resources	-for configuration and static files.
src/test/java	-for junit testing java file
src/test/resources	-for junit testing configuration file,

example:



- 2) Every maven project has pom.xml file which is main configuration file of maven project.
- 3) Every pom has a parent pom called super pom, like every java class parent class is Object class. (convention over configuration approach)
- 4) You can create you parent pom to share common configuration across multiple project. (example spring boot)
- 5) Every maven project has coordinates to uniquely identify the artifacts.

BASIC Structure of POM file

```
<?xml version="1.0"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
```

```
  <parent>
    <groupId>com.amir</groupId>
    <artifactId>ocs-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
```

This parent tag use for inheritance feature to apply common dependency and config into child project

```
  <groupId>com.amir</groupId>
  <artifactId>ocs-web</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
```

These tags known as coordinates of maven project to as uniquely identifier.

```
  <name>ocs-web Maven Webapp</name>
```

Artifacts types after build the project

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

```
<properties>  
</properties>
```

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

These tags use to add libs(jars) into project there are five type of scope of dependency
compile, test, provided, runtime, system

```
<dependencyManagement>  
</dependencyManagement>
```

To provide dependency into child project (inheritance of dependency)

```
<distributionManagement>  
</distributionManagement>
```

To deploy the project artifacts into our remote repository using deploy command(phase) of default life cycle

```
<repositories>  
</repositories>
```

This section for defining our remote repository or maven central repository to download the libs

```
<pluginRepositories>  
</pluginRepositories>
```

To sharing the common repository to child

```
<build>  
  <finalName>ocs-web</finalName>
```

Final artifact name

```
  <plugins>  
  </plugins>
```

Import plugin to call in life cycle phase

```
  <pluginManagement>  
  </pluginManagement>
```

To sharing the common plugin to child project

```
</build>
```

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

Used to make profile oriented/dependent build

```
</project>
```

BASIC COMMANDS

General syntax of maven command.

- 1) Using life cycle phase command
mvn <phase-name> [args]
example :
mvn clean or mvn install or mvn site

note: in eclipse we don't need to put mvn as prefix (clean or install) we can merge two command in one line by space like clean install

2) Using plugin goal command.

Syntax:

`mvn <plugin-identifier:goal-name> [args]`

example: `mvn compiler:compile`

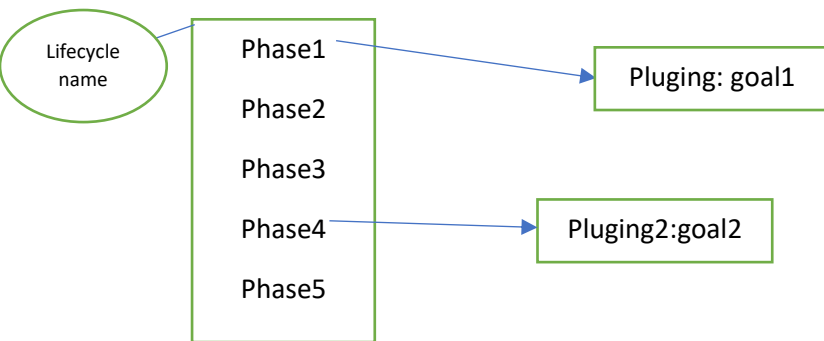
maven plugin is collection of goals and goal is small unit task to be done by maven.

Maven life cycle

Maven contains inbuilt 3 build lifecycle 1) default 2) clean 3) site.

Default life cycle contains validate, initialize, package, install, deploy command.

Build life cycle: it is collection of phases who attached sequentially. Phases can also attached to plugin goals to perform unit task.



To see life cycle phases and associated plugin use following command of help plugin.

`mvn help:deccrbe -Dcmd=<any phase name>`

Example:

`mvn help:describe -Dcmd=clean`

```

D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core>mvn help:describe -Dcmd=clean
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-help-plugin:3.2.0:describe (default-cli) @ ocs-core ---
[INFO] 'clean' is a phase within the 'clean' lifecycle, which has the following phases:
* pre-clean: Not defined
* clean: org.apache.maven.plugins:maven-clean-plugin:2.5:clean
* post-clean: Not defined

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.746 s
[INFO] Finished at: 2020-05-12T16:05:44+05:30
[INFO] Final Memory: 13M/212M
[INFO] -----

```

Site life cycle:

```

D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core>mvn help:describe -Dcmd=site
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-help-plugin:3.2.0:describe (default-cli) @ ocs-core ---
[INFO] 'site' is a phase within the 'site' lifecycle, which has the following phases:
* pre-site: Not defined
* site: org.apache.maven.plugins:maven-site-plugin:3.3:site
* post-site: Not defined
* site-deploy: org.apache.maven.plugins:maven-site-plugin:3.3:deploy

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----

```

Default life cycle:

```

D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core>mvn help:describe -Dcmd=install
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-help-plugin:3.2.0:describe (default-cli) @ ocs-core ---
[INFO] 'install' is a phase corresponding to this plugin:
org.apache.maven.plugins:maven-install-plugin:2.4:install

It is a part of the lifecycle for the POM packaging 'jar'. This lifecycle includes the following phases:
* validate: Not defined
* initialize: Not defined
* generate-sources: Not defined
* process-sources: Not defined
* generate-resources: Not defined
* process-resources: org.apache.maven.plugins:maven-resources-plugin:2.6:resources
* compile: org.apache.maven.plugins:maven-compiler-plugin:3.1:compile
* process-classes: Not defined
* generate-test-sources: Not defined
* process-test-sources: Not defined
* generate-test-resources: Not defined
* process-test-resources: org.apache.maven.plugins:maven-resources-plugin:2.6:testResources
* test-compile: org.apache.maven.plugins:maven-compiler-plugin:3.1:testCompile
* process-test-classes: Not defined
* test: org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test
* prepare-package: Not defined
* package: org.apache.maven.plugins:maven-jar-plugin:2.4:jar
* pre-integration-test: Not defined
* integration-test: Not defined
* post-integration-test: Not defined
* verify: Not defined
* install: org.apache.maven.plugins:maven-install-plugin:2.4:install
* deploy: org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy

[INFO] -----
[INFO] BUILD SUCCESS

```

Yellow highlighted parts are associated plugin name with phase name.

When we hit any phase command ...then maven execute that life cycle up to that phase only.

Means install command will not execute deploy phase.

Maven Plugin

It is execution library where task done by maven engine.

Every goal of maven task is responsible for unit task. By default super pom contains all plugin which is required for maven life cycle.

Example of clean plugins.

```

<plugin>
  <artifactId>maven-clean-plugin</artifactId>
  <version>2.5</version>
  <executions>
    <execution>
      <id>default-clean</id>
      <phase>clean</phase>
      <goals>
        <goal>clean</goal>
      </goals>
    </execution>
  </executions>
</plugin>

```

```
        </execution>
    </executions>
</plugin>
```

when we run clean life cycles console show every plugin names,goal name,and plugin id & project name which this build is running.

```
D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ ocs-core ---
[INFO] Deleting D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core\target
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.854 s
[INFO] Finished at: 2020-05-12T16:32:03+05:30
[INFO] Final Memory: 5M/150M
[INFO] -----
```

Here

maven-clean-plugin is plugin name:

2.5 is version of plugin

(default-clean) id name of plugin

ocs-core project name for which this build is running.

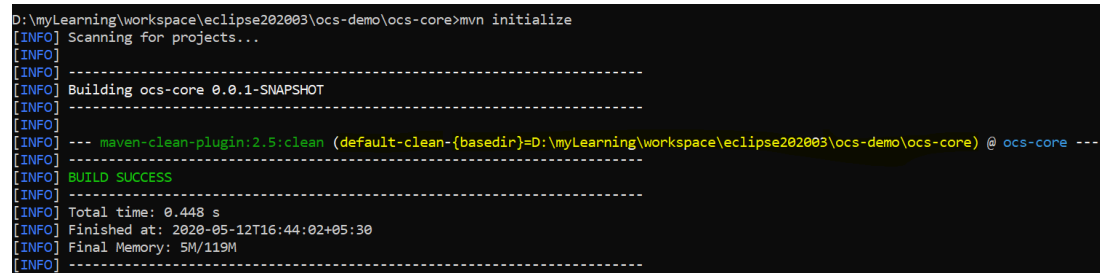
If you want to print anythings from maven , you can put inside the plugin `<id>default-clean</id>`

Like `<id>default-clean =${project.build.directory}</id>`

Or `<id>default-clean =${basedir}</id>`

Example:

```
<plugin>
  <artifactId>maven-clean-plugin</artifactId>
  <version>2.5</version>
  <executions>
    <execution>
      <id>default-clean-{basedir}=${basedir}</id>
      <phase>validate</phase>
      <goals>
        <goal>clean</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```



```
D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core>mvn initialize
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-clean-plugin:2.5:clean (default-clean-{basedir}=D:\myLearning\workspace\eclipse202003\ocs-demo\ocs-core) @ ocs-core ---
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.448 s
[INFO] Finished at: 2020-05-12T16:44:02+05:30
[INFO] Final Memory: 5M/119M
[INFO] -----
```

POM dependency elements

```
<dependencies>
```

```
  <dependency>
```

```
    <groupId>junit</groupId>
```

```
    <artifactId>junit</artifactId>
```

```
    <version>4.11</version>
```

```
    <scope>test</scope>
```

```
  </dependency>
```

```
</dependencies>
```

Scope element tell what role of will be added artifacts for building project.


```
<scope>compile/provided/runtime/test/system/import</scope>
```

Basically, every maven build project having following stage of task.

- 1) Compiling src
- 2) Compiling testSrc
- 3) Running application
- 4) Running testClass
- 5) Building artifacts(jar/war)

- | | | |
|-------------|-------------------------|---|
| a) compile | - Available for {1,3,5} | --almost all log4j |
| b) provided | - Available for {1,3} | --jsp and servlet api |
| c) runtime | - Available for {3,5} | --all transitive jar (mysql jar) |
| d) test | - Available for {2,4} | --junit |
| e) system | - Available for {1,3} | --external dependency |
| f) import | - Available for {1,3,5} | -- it's only available for the dependency type pom. |

Example of System scope:

```
dependency>  
  <groupId>extDependency</groupId>  
  <artifactId>extDependency</artifactId>  
  <scope>system</scope>  
  <version>1.0</version>  
  <systemPath>${basedir}\war\WEB-INF\lib\extDependency.jar</systemPath>  
</dependency>
```

MAVEN Repository

To store/fetch artifacts we require some location that is known as repository.

if by default maven provide two repositories (local and central) for all project.

Maven repository can be classified into two categories

- 1) offline or local
- 2) online (central or remote)

Local repository

Default path of local repository : \${user.home}/.m2/repository

You can change the local repository local using setting.xml file.

```
<!--localRepository
```

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

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

```
-->
```

Possible location of settings.xml

`${MAVEN_HOME}/conf/settings.xml`

Or `${user.home}/.m2/settings.xml` (having highest priority)

While you build the project you can provide custom settings file using

`mvn -s filename` or `mvn --settings filename`

attribute of mvn command.

`-f, --file <file>`

Forces the use of an alternate POM file

`-s, --settings <arg>`

Alternate path for the user settings file

LATER we will be discussed more about settings.xml and how many elements can be config in settings.xml

Online repository (central)

Online repository which is provides by maven is known as central repository.

By default maven provide one central repository for artifacts and one central repository for plugins using super POM you can see in the effective pom

`mvn help:effective-pom`

1) Repository for artifacts

```
<repositories>
```

```
<repository>
```

```
<snapshots>
```

```
<enabled>>false</enabled>
```

```
</snapshots>
```

```
<id>central</id>
```

```
<name>Central Repository</name>
```

```
<url>https://repo.maven.apache.org/maven2</url>
```

```
    </repository>
</repositories>
```

2) Repository for plugins

```
<pluginRepositories>
```

```
  <pluginRepository>
```

```
    <releases>
```

```
      <updatePolicy>never</updatePolicy>
```

```
    </releases>
```

```
    <snapshots>
```

```
      <enabled>false</enabled>
```

```
    </snapshots>
```

```
    <id>central</id>
```

```
    <name>Central Repository</name>
```

```
    <url>https://repo.maven.apache.org/maven2</url>
```

```
  </pluginRepository>
```

```
</pluginRepositories>
```

Normally every artifacts can be SNAPSHOT or release one
By default snapshot one is not enable to search only release one is enable.

Snapshot means: still development is in progress.
Release means: development is completed. If version does not consist snapshot word means it is coming under release categories.

Release example: `<version>0.0.1</version>`

Snapshot example: `<version>1.0.SNAPSHOT</version>`

Note: maven keep these two repositories open for all so no authentication is required.

Why it called central repository just because of id value:

```
<id>central</id>
```

If you defined your own repository with same id value, then maven central will not work to search artifacts only your user-define repository will work.

We can config the credential of repository using these id element `<id>central</id>`

If repository is password protected then you can configure the credential in settings.xml .

Example of configuration.

```

<servers>
  <server>
    <id>central</id>
    <username>username</username>
    <password>password</password>
  </server>
</servers>

```

This id element value must match with id value of repository then only this will work.

Deep dive into settings.xml

```
<settings>
```

Settings is root tag/element of maven configuration.

sub-root element list

- 1) <localRepository> default value : \${user.home}/.m2/repository (covered)
- 2) <interactiveMode> default value is true --for created project using cmd
- 3) <offline> default value is false. -if you want to local repo only
- 4) <pluginGroups> used to maintain groupid for plugins
- 5) <proxies> used to connect the internet via proxy rule.
- 6) <servers> used to configure credential of remote repo <already covered>
- 7) <mirrors> used to mirror(replace) the repository url.
- 8) <profiles> build profiling. <will cover later>
- 9) <activeProfiles> build profiling active selection.

```
</settings>
```

<mirrors> elements

```
<mirrors>
```

```
  <mirror>
```

```
    <id>mirrorId</id>
```

It can any unique name like: myMirrorId

```
    <mirrorOf>repositoryId</mirrorOf>
```

It must be equal to value of repository id: like central

```
    <name>Mirror Name</name>
```

Any meaning full Name

```
    <url>http://my.repository.com/repo/path</url>
```

Alternative repository Id or instance-id

```
  </mirror>
```

```
</mirrors>
```

Maven aggregation(module)

Its use to create first parent project and then all child project(module one),when we will be the parent project first it will build all child project and aggregate it with parent one.

Note: parent type packaging must be pom type.

Parent type project pom contains modules element for all module or child project.

Example of parent pom.

```
-----  
<project>  
  <modelVersion>4.0.0</modelVersion>  
  <groupId>com.amir</groupId>  
  <artifactId>ocs-demo</artifactId>  
  <version>0.0.1-SNAPSHOT</version>  
  <packaging>pom</packaging>  
  <modules>  
    <module>ocs-core</module>  
    <module>ocs-web</module>  
  </modules>  
</project>
```

Child project will be normal maven project with packaging type (jar or war).

Example:

```
-----  
<project >  
  <modelVersion>4.0.0</modelVersion>  
  <parent>  
    <groupId>com.amir</groupId>  
    <artifactId>ocs-demo</artifactId>  
    <version>0.0.1-SNAPSHOT</version>  
  </parent>
```

```
<groupId>com.amir</groupId>
<artifactId>ocs-core</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>ocs-core</name> //if packaging is not mentioned then default value is jar
<url>http://maven.apache.org</url>
</project>
```

Another example of child or module

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>com.amir</groupId>
    <artifactId>ocs-demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <groupId>com.amir</groupId>
  <artifactId>ocs-web</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>ocs-web Maven Webapp</name>
  <url>http://maven.apache.org</url>
</project>
```

In every child or module parent element/tag must be there.

Packaging type can be any value except pom default value is jar

