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

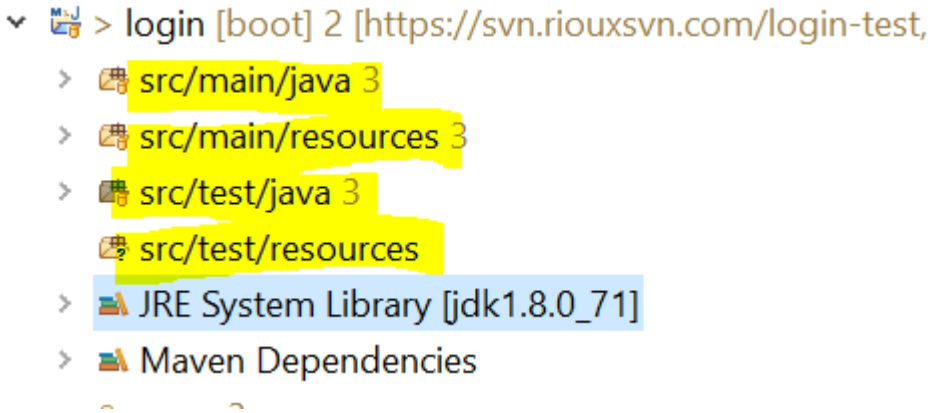
```
C:\Users\inkajm01>mvn --version
Apache Maven 3.5.0 (ff8f5e7444045639af65f6095c62210b5713f426; 2017-04-04T01:09:06+05:30)
Maven home: D:\mySoftware\sh-soft\apache-maven-3.5.0\apache-maven-3.5.0\bin\..
Java version: 1.8.0_71, vendor: Oracle Corporation
Java home: D:\myProjects\jdk_home\jdk1.8.0_71\jre
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
https://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>

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

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

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

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

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>

  <dependencyManagement>
  </dependencyManagement>

  <distributionManagement>
  </distributionManagement>

  <repositories>
  </repositories>

<pluginRepositories>
</pluginRepositories>

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

    <plugins>
    </plugins>

    <pluginManagement>
    </pluginManagement>

  </build>

<profiles>
</profiles>

</project>
```

These tags use to add libs(jars) into project there are five type of scope of dependency

compile, test, provided,runtime,system

To provide dependency into child project (inheritance of dependency)

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

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

To sharing the common repository to child

Final artifact name

Import plugin to call in life cycle phase

To sharing the common plugin to child project

Used to make profile oriented/dependent build

BASIC COMMADS

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 commad in one line by space like clean install

2) Using plugin goal command.

Systax:

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

example: mvn compiler:compile

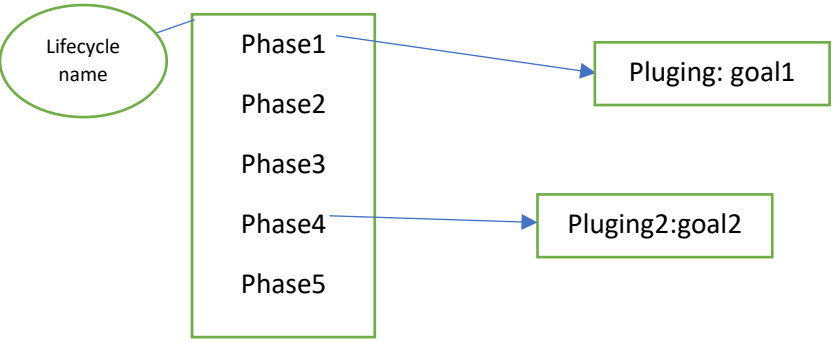
maven plugin is collection of goals and goal is small unit task to be done by maven.

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:deccribe -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] ------------------------------------------------------------------------
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[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] ------------------------------------------------------------------------
[INFO] Building ocs-core 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[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. Bydefault 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>
```

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>
```