Curs 20-21

UD 4. Eines d'automatització

Automatització en Java. Apache Maven.

Entorns de desenvolupament

IES Jaume II El Just
Tavernes de la Valldigna

Continguts

1. Introducció: Apache Maven

Maven és una altra eina que ens serveix també per a la construcció d'aplicacions Java i la gestió de les seues dependències, a l'igual que *ant*, però presenta un punt de partida diferent.

Maven pretén seguir diversos patrons per construir la infrastructura d'un projecte Java, per tal d'homogeneïtzar l'estructura de les aplicacions i bones pràctiques, millorant així la productivitat. Bàsicament es tracta d'una eina de gestió de projectes que proveeix una forma d'ajudar en diverses tasques, tals com:

- La construcció d'aplicacions,
- El manteniment de la documentació i informes,
- La gestió de dependències,
- La gestió dels sistemes de control de versions (VCS),
- L'alliberació de versions (releases)
- La distribució de l'aplicació

Això doncs, el principal avantatge que aporta Maven és que ens ofereix un suport al cicle de desenvolupament acord a estàndards i bones pràctiques per millorar-lo.

1.1. Instal·lació de Maven en Ubuntu

Maven no vé instal·lat per defecte en Ubuntu, però està disponible als repositoris.

Per tal de comprovar si tenim Maven instal·lat al nostre sistema, podem fer:

```
$ apt-cache policy maven
maven:
  Instal·lat: (cap)
  Candidat:   3.6.0-1~18.04.1
  Taula de versió:
     3.6.0-1~18.04.1 500
        500 http://es.archive.ubuntu.com/ubuntu bionic-updates/universe
            amd64 Packages
        500 http://es.archive.ubuntu.com/ubuntu bionic-updates/universe
            i386 Packages
        500 http://security.ubuntu.com/ubuntu bionic-security/universe
            amd64 Packages
        500 http://security.ubuntu.com/ubuntu bionic-security/universe
            i386 Packages
     3.5.2-2 500
        500 http://es.archive.ubuntu.com/ubuntu bionic/universe amd64
            Packages
        500 http://es.archive.ubuntu.com/ubuntu bionic/universe i386
            Packages
```

Com veiem ens indica que no tenim el paquet instal·lat, però que podem instal·lar la versió 3.6.0-1~18.04.1 des dels repositoris d'Ubuntu.

Per tal d'instal·lar el paquet, ho haurem de fer com a administradors:

```
$ sudo apt update
...
$ sudo apt install maven
[sudo] contrasenya per a eljust:
S'està llegint la llista de …paquets Fet
S'està construint l'arbre de dependéncies
S'està llegint la informació de l'…estat Fet
...
S'instal·laran els següents paquets extres:
  libwagon-file-java libwagon-http-shaded-java
S'instal·laran els paquets NOUS següents:
  libwagon-file-java libwagon-http-shaded-java maven
0 actualitzats, 3 nous a instal·lar, 0 a suprimir i 72 no actualitzats.
S'ha d'obtenir 1787 kB d'arxius.
Després d'aquesta operació s'empraran 2261 kB d'espai en disc
    addicional.
```

Com veiem, el paquet maven arrossega algunes dependències, el que ens dóna una visió de la magnitud del projecte front a ant. Indiquem que sí que volem realitzar la instal·lació, i esperem que aquesta finalitze.

Per tal de comprovar que tenim Maven instal·lat al sistema, fem:

```
$ mvn --version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 11.0.5, vendor: Private Build, runtime: /usr/lib/jvm/java
    -11-openjdk-amd64
Default locale: ca_ES, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-72-generic", arch: "amd64", family:
    "unix"
```

Veiem que ens ofereix informació sobre la versió i la ubicació de la instal·lació de Maven i de Java, així com informació genèrica sobre el sistema, com la configuració d'idioma i el sistema operatiu.

Podeu consultar de forma ampliada més informació sobre Maven als següents enllaços:

- https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html
- https://maven.apache.org/guides/getting-started/index.html
- https://www.baeldung.com/maven

2. Primer exemple amb Maven

Anem a crear un projecte de tipus *Hola Món* des de zero, amb les ajudes que ens proveeix Maven. Per a això farem ús del mecanisme d'*arquetipus* que ofereix. Un arquetipus no és més que un patró o model a partir del qual creem l'esquelet del nostre projecte.

2.1. Generació del projecte a partir de l'arquetipus

Anem a començar amb la creació d'aquest primer exemple. Per a això, llançarem la següent ordre:

```
$ mvn archetype:generate -DgroupId=com.ieseljust.edd -DartifactId=
    myHelloMVN -DarchetypeArtifactId=maven-archetype-quickstart -
    DinteractiveMode=false
```

Veiem els diferents paràmetres que hem proporcionat a Maven:

- **archetype:generate**: Indiquem a maven que volem executar l'objectiu (*goal*) ***generate*** del plugin ***archetype***. Recordem que en Maven, els *goals* són equiparables als *tasks* d'Ant. Aquest objectiu en concret ens genera un projecte simple basat en un arquetipus. Podem dir que un plugin és una col·lecció d'objectius amb un propòsit comú.
- **DgroupId**: Indiquem l'identificador únic de l'organització que crea el projecte, basat normalment en el domini completament qualificat de l'organització (*fully qualified domain*), en aquest cas *com.ieseljust.edd*.
- **DartifactID**: Indica el nom del recurs (artifact) que anem a generar. També ha de ser un nom únic per al projecte. En aquest cas, *myHelloMVN*.
- **DarchetypeArtifactId**: Indica el nom del recurs d'arquetipus a partir del qual agafar la plantilla per al nostre projecte. En aquest cas, indiquem el recurs *maven-archetipe-quickstart*.
- **DinteractiveMode**: Indica si volem utilitzar o no el mode interactiu. En aquest cas, hem posar el valor a *false*, per agilitzar la tasca i que agafe certs valors per defecte, en lloc de deixar que ens els pregunte l'eina.

Amb tot açò, l'eixida de l'ordre anterior (i després de descarregar algun programari addicional…) és la següent:

```
$ mvn archetype:generate -DgroupId=com.ieseljust.edd -DartifactId=
    myHelloMVN -DarchetypeArtifactId=maven-archetype-quickstart -
    DinteractiveMode=false
...
[INFO] Scanning for projects...
Downloading...

...
[INFO]
```

```
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
Downloading...
...
[INFO] ------------------------------------------------------------------------

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

[INFO] Parameter: basedir, Value: .../exemples_java/mvn1
[INFO] Parameter: package, Value: com.ieseljust.edd
[INFO] Parameter: groupId, Value: com.ieseljust.edd
[INFO] Parameter: artifactId, Value: myHelloMVN
[INFO] Parameter: packageName, Value: com.ieseljust.edd
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: .../exemples_java/mvn1/myHelloMVN
[INFO] ------------------------------------------------------------------------

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

[INFO] Total time:  30.233 s
[INFO] Finished at: 2020-01-02T09:13:05+01:00
[INFO] ------------------------------------------------------------------------

...
```

Com podem apreciar, ens mostra algunes alertes i missatges informatius, amb els valors que ha establert per a alguns paràmetres. Com podem comprovar també, es realitzen diverses descàrregues des de repo.maven.apache.org. Aquesta adreça és el repositori de Maven, i conté totes les plantilles i

eines que podem utilitzar amb Maven.

Bé, veiem ara què ens ha generat aquesta ordre:

```
$ tree
.
`-- myHelloMVN
    |-- pom.xml
    `-- src
        |-- main
        |   `-- java
        |       `-- com
        |           `-- ieseljust
        |               `-- edd
        |                   `-- App.java
        `-- test
            `-- java
                `-- com
                    `-- ieseljust
                        `-- edd
                            `-- AppTest.java
```

Com veiem, ha creat la carpeta del projecte *myHelloMVN* amb el fitxer *pom.xml*, que descriu el projecte segons el *Project Object Model (POM)*. Dins d'aquesta carpeta tenim la carpeta src, amb els fitxers font i de tests, degudament organitzats en carpetes segons el nom de domini completament qualificat.

2.2. El fitxer pom.xml

El fitxer pom.xml descriu la configuració del projecte en Maven, i proporciona la major part d'informació necessària per a la seua construcció. Pot arribar a ser un fitxer llarg i complex, però no és necessari entendre tot el seu contingut per traure tota l'efectivitat de Maven.

Veiem el contingut del nostre fitxer:

```
<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.ieseljust.edd</groupId>
 <artifactId>myHelloMVN</artifactId>
 <packaging>jar</packaging>
 <version>1.0-SNAPSHOT</version>
 <name>myHelloMVN</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>
```

El contingut del fitxer és bastant intuïtiu:

- **project**: L'etiqueta arrel del document no és més que la capçalera XML, on es defineix el tipus de document a través del seu espai de noms.
- **modelVersion**: Indica la versió de POM amb què es descriu el documet.
- **groupId**, **artifactId** i **name**: fan referència a l'identificador de l'organització (groupId), al nom del recurs (artifactId) i al de l'aplicació (name)
- **packaging**: Fa referència al tipus d'empaquetat per distribuir l'aplicació (jar).
- **version**: que indica la versió de l'aplicació. El sufix *SNAPSHOT*, indica que es tracta de l'últim codi al llarg d'una branca de desenvolupament, el que no garanteix l'estabilitat d'aquest. Generalment, s'usa quan estem en fase de desenvolupament i proves, fins que alliberem una *release* estable.
- **url**: Apunta a la url del projecte Maven.
- **Dependencies**: Indica les dependències que té la nostra aplicació. Com veiem, ho fa indicant per a cadascuna el seu identificador d'organització (**groupId**), de recurs (**artifactId**), i la versió. A més, especifica l'àmbit (**scope**) en què s'aplica la dependència. En aquest cas, l'arquetipus a partir del que hem generat la nostra aplicació incorpora la dependència de jUnit per tal de realitzar els tests.

Realitzant alguns ajustos al fitxer

Les últimes versions de Maven no suporten ja la compilació per a Java 1.5, pel que si no indiquem el contrari, obtindrem un missatge de compilació del tipus:

```
[ERROR] Source option 1.5 is no longer supported. Use 1.6 or later.
[ERROR] Target option 1.5 is no longer supported. Use 1.6 or later.
```

Per tal d'evitar açò, cal especificar un parell de propietats per al projecte. Concretament el source i el target del compilador de Maven. Per a això, editem el fitxem `pom.xml` i afegim la següent etiqueta <*properties*>, abans d'especificar l'etiqueta de dependències:

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

 <properties>
   <maven.compiler.source>1.6</maven.compiler.source>
   <maven.compiler.target>1.6</maven.compiler.target>
 </properties>
```

```
8
9   <dependencies>
10  [...]
```

2.3. Hola Món!

L'arquetipus que hem utilitzat per generar l'aplicació, *maven-archetype-quickstart*, ens crea directament un esquelet d'aplicació del tipus *Hola Món*, directament. Si accedim al fitxer src/main/java/com/ieseljust/app/App.java, veurem un codi bastant familiar:

```
1  package com.ieseljust.edd;
2
3  /**
4   * Hello world!
5   *
6   */
7  public class App
8  {
9      public static void main( String[] args )
10     {
11         System.out.println( "Hello World!" );
12     }
13 }
```

2.4. Compilació, neteja i construcció del projecte

Una vegada tenim l'esquelet de l'aplicació generat, ja podem realitzar la seua compilació i construcció.

Per tal de fer la compilació, **des de la carpeta arrel del projecte** (la que conté el pom.xml), llancem la següent ordre:

```
1  $ mvn compile
2  ...
3  [INFO] Scanning for projects...
4  [INFO]
5  [INFO] ----------------------< com.ieseljust.edd:myHelloMVN
       >----------------------
6  [INFO] Building myHelloMVN 1.0-SNAPSHOT
7  [INFO] --------------------------------[ jar
       ]---------------------------------
8  [INFO]
9  [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @
       myHelloMVN ---
10 [WARNING] Using platform encoding (UTF-8 actually) to copy filtered
       resources, i.e. build is platform dependent!
```

```
[INFO] skip non existing resourceDirectory /home/joamuran/Dropbox/Docencia/curs_19-20/EDD/Unitats/UD4. Automatitzacio/exemples_java/mvn1/myHelloMVN/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ myHelloMVN ---
[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 /home/joamuran/Dropbox/Docencia/curs_19-20/EDD/Unitats/UD4. Automatitzacio/exemples_java/mvn1/myHelloMVN/target/classes
[INFO] ------------------------------------------------------------------------

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

[INFO] Total time:  1.643 s
[INFO] Finished at: 2020-01-02T09:27:04+01:00
[INFO] ------------------------------------------------------------------------
```

Amb açò s'haurà generat una nova carpeta ***target*** a l'arrel del projecte, amb les classes generades, amb la següent estructura:

```
$ tree target
target
|-- classes
|   `-- com
|       `-- ieseljust
|           `-- edd
|               `-- App.class
|-- generated-sources
|   `-- annotations
`-- maven-status
    `-- maven-compiler-plugin
        `-- compile
            `-- default-compile
                |-- createdFiles.lst
                `-- inputFiles.lst
```

Per tal d'executar l'aplicació, hem d'indicar el classpath (opció -cp) i llençar l'aplicació, reemplaçant les barres del camí des del classpath fins la classe per punts (com/ieseljust/app/App -> com.ieseljust.app.App).

```
$ java -cp target/classes com.ieseljust.edd.App
Hello World!
```

Per altra banda, si el que volem és netejar el projecte, farem:

```
mvn clean
...
```

Que com veurem, ens haurà esborrat la carpeta target anterior.

Finalment, per fer la construcció i empaquetat en *jar* del projecte, farem:

```
$ mvn package
```

O si bé volem fer la neteja i la construcció tot d'una, podem fer:

```
$ mvn clean package
```

Veiem-ne el resultat de l'empaquetat:

```
$ mvn package
...
[INFO] Building jar: .../exemples_java/mvn1/myHelloMVN/target/
   myHelloMVN-1.0-SNAPSHOT.jar
[INFO]
   ------------------------------------------------------------------------

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

[INFO] Total time:  8.220 s
[INFO] Finished at: 2020-01-02T09:32:27+01:00
[INFO]
   ------------------------------------------------------------------------
```

Amb açò s'ha realitzat la compilació i els tests corresponents, generant tota l'estructura de directoris següent:

```
target
|-- classes
|   `-- com
|       `-- ieseljust
|           `-- edd
|               `-- App.class
|-- generated-sources
|   `-- annotations
|-- generated-test-sources
|   `-- test-annotations
|-- maven-archiver
|   `-- pom.properties
|-- maven-status
|   `-- maven-compiler-plugin
```

```
|       |-- compile
|       |   `-- default-compile
|       |       |-- createdFiles.lst
|       |       `-- inputFiles.lst
|       `-- testCompile
|           `-- default-testCompile
|               |-- createdFiles.lst
|               `-- inputFiles.lst
|-- myHelloMVN-1.0-SNAPSHOT.jar
|-- surefire-reports
|   |-- com.ieseljust.edd.AppTest.txt
|   `-- TEST-com.ieseljust.edd.AppTest.xml
`-- test-classes
    `-- com
        `-- ieseljust
            `-- edd
                `-- AppTest.class

20 directories, 10 files
```

Si ens fixem, dins la carpeta target es troba el fitxer `myHelloMVN-1.0-SNAPSHOT.jar`, amb l'aplicació empaquetada. Per tal d'executar l'aplicació des del `jar`, establirem aquest com al classpath i llançarem l'aplicació:

```
$ java -cp target/myHelloMVN-1.0-SNAPSHOT.jar com.ieseljust.edd.App
Hello World!
```

2.5. Entenent el cicle de vida de construcció

Disposeu informació ampliada sobre aquest punt en:

- https://rubensa.wordpress.com/2016/06/10/maven-lifecycle/

Un dels principals conceptes de Maven és el de *cicle de vida de construcció*, o *build lifecycle*, que defineix el procés de construcció i distribució de components.

Maven presenta tres cicles de vida al sistema:

- ***default***, que controla el desplegament del projecte,
- ***clean***, que controla la neteja del projecte,
- ***site***, que controla la generació de la documentació del projecte.

Cadascun d'aquests cicles es compon de diferents fases, que representen diferents estats del cicle de vida. Veiem a la **figura 1** la relació entre els cicles de vida i les seues respectives fases.

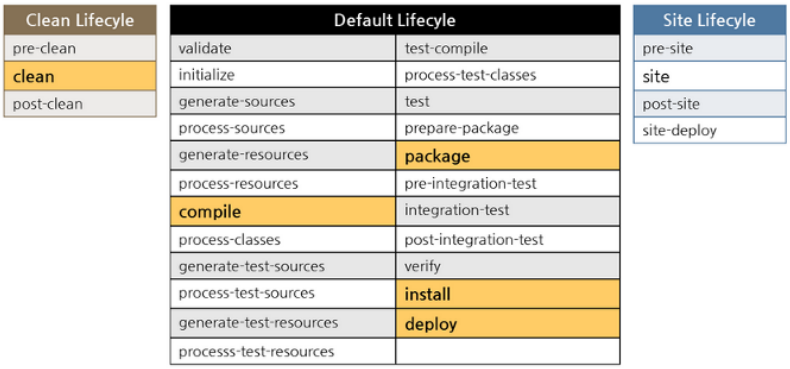

Figura 1: Cicles de vida Maven

Les diferents fases del cicle de vida s'executen de forma seqüencial per tal de completar aquest, però també es poden invocar de forma individual. Aquesta invocació individual d'una fase, implica la invocació de totes les fases anteriors. Per exemple, anteriorment, hem vist que per construir l'aplicació d'hola món hem fet `mvn package`, amb el que invocàvem la fase *package* del cicle de vida *default*. Aquesta fase, implica doncs totes les anteriors, de validació, inicialització, processat, compilació, generació i compilació de tests, fins arribar a la fase *package*.

Fases del cicle de construcció i plugins

Tot i que cada fase és responsable d'una part del cicle de vida de construcció, la forma de portar-la a terme pot variar entre projectes. Açò s'aconsegueix mitjançant la declaració d'objectius (goals) de plugins associats a cada fase de construcció al fitxer pom.xml. Un objectiu d'un plugin representa una tasca específica en la construcció i gestió d'un projecte, i pot estar associat a zero o més fases de construcció. Si no està associat a cap fase, aquest objectiu pot executar-se fora del cicle de vida, invocant-lo directament.

Disposeu de molta més informació al respecte a la bibliografía, però de moment, amb el que hem vist, ja podeu tindre una idea general del cicle de construcció i de la forma de treballar amb Maven.

2.6. El mode interactiu de Maven

A l'apartat anterior hem vist com crear un projecte Maven amb el mode batch. Maven permet també un mode interactiu, que ens va guiant en la creació del nostre projecte.

Per crear un projecte Maven des del mode interactiu, només cal indicar que volem executar l'objectiu ***generate*** del plugin ***archetype***, com hem fet abans, però sense cap paràmetre més.

```
$ mvn archetype:generate
```

Després d'alguns possibles avisos i missatges informatius, ens ofereix una llista de totes les plantilles disponibles a Maven (més de 2000). Per defecte, ens proposa la creació d'un projecte de tipus «maven-archetype-quickstart» (1466), que és el clàssic *Hola Món* que coneixem de l'exemple anterior. **Tingueu en compte que aquest número pot variar entre les diferents versions de Maven.**

```
Choose a number or apply filter (format: [groupId:]artifactId, case
    sensitive contains): 1466:
```

Després ens preguntarà el número de versió de l'aplicació:

```
Choose org.apache.maven.archetypes:maven-archetype-quickstart version:
1: 1.0-alpha-1
2: 1.0-alpha-2
3: 1.0-alpha-3
4: 1.0-alpha-4
5: 1.0
6: 1.1
7: 1.3
Choose a number: 7: 5
```

I algunes dades més que ja coneixem:

```
Define value for property 'groupId': ieseljust
Define value for property 'artifactId': holaMon
Define value for property 'version' 1.0-SNAPSHOT: :
Define value for property 'package' ieseljust: : com.ieseljust.edd
Confirm properties configuration:
groupId: ieseljust
artifactId: holaMon
version: 1.0-SNAPSHOT
package: com.ieseljust.edd
 Y: : Y
[INFO]
    ----------------------------------------------------------------------------

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

[INFO] Parameter: basedir, Value: .../exemples_java/mvn2
[INFO] Parameter: package, Value: com.ieseljust.edd
[INFO] Parameter: groupId, Value: ieseljust
[INFO] Parameter: artifactId, Value: holaMon
[INFO] Parameter: packageName, Value: com.ieseljust.edd
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: .../
```

```
      exemples_java/mvn2/holaMon
[INFO]
      ------------------------------------------------------------------------

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

[INFO] Total time:  02:12 min
[INFO] Finished at: 2020-01-02T09:46:04+01:00
[INFO]
      ------------------------------------------------------------------------
```

Amb açò, tindrem una estructura de projecte idèntica a la del projecte anterior i un fitxer `pom.xml` equivalent. Per tal de compilar-lo, haurem de fer les modificacions corresponents al fitxer `pom.xml` per tal d'evitar els errors amb les versions 1.5 de java, i llençar el `mvn compile` per compilar o el `mvn package` per obtindre el paquet.

3. El plugin Maven for Java per a VSCode

Com ja sabem, quan instal·lem el Java Extension Pack, una de les extensions que ens arrossega és l'extensió *Maven for Java*. Aquesta extensió ens ofereix un explorador de projectes Maven i afig les següent funcionalitats:

- Suport per a la generació de projectes a partir d'arquetipus Maven,
- Suport per generar POM de forma eficient,
- Oferix dreceres de teclat per llençar objectius de Maven comuns: *clean, validate, compile test, package, verify, install, site i deploy*,
- Manteniment d'un historial d'ordres per tornar a llançar ordres recents.

3.1. L'explorador de projectes Maven

Quan obrim una carpeta o espai de treball que conté un fitxer pom.xml, VSCode ens mostra una nova secció a la barra lateral anomenada *Maven Projects*, amb els projectes Maven de l'espai i els seus mòduls.

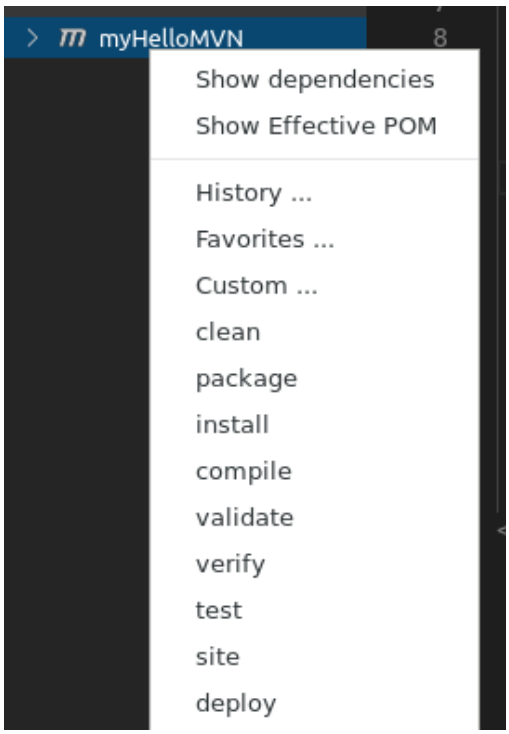

Figura 2: Explorador de Projectes Maven

Com veiem a la **figura 2**, després d'obrir la carpeta on tenim el projecte d'Hola Món, tenim el projecte *myHelloMVN* a la pestanya de *Maven Projects*. Si fem clic en el nom del projecte, veurem que ens obre el fitxer `pom.xml` que descriu aquest, i per altra banda, si fem clic al botó dret, veurem les diferents possibilitats que tenim per al projecte. Com podem veure, tenim diferents objectius a llençar: clean, package, install, compile, etc. A més, tenim accés a l'historial (per no tornar a escriure ordres llargues que ja hem utilitzat) i una opció anomenada *Effective POM*.

Què és l'*Efective POM*

Quan creem un projecte Maven a partir d'un arquetipus, definim per a aquest un conjunt de propietats tals com el groupID, l'artifactID i la versió. Aquests tres camps són els que es requereixen per a tot projecte, i són el que conformaría el POM més simple que podríem generar. El POM d'un projecte, a més, hereta certes propietats del que es coneix com el *Super POM*, que vindría a ser com la classe pare de la que hereten tots els POM. És a dir, un POM es compon dels atributs que hi definim explícitament en funció de l'arquetipus que hajam utilitzat, i a més, posseeix un conjunt de valors per defecte compartits per tots els projectes, i que no apareixen explícitament al fitxer `pom.xml`.

Tenint tot açò en compte, el POM Efectiu (*Effective POM*) s'entén com la combinació d'ambdós POM, el Super POM i el POM més simple. A efectes pràctics, si generem el POM efectiu del nostre projecte

(el genera en una pestanya a banda, sense modificar el nostre `pom.xml`), veurem que es tracta d'un fitxer XML bastant més extés que l'original, i en el que tindrem totes les característiques que teníem definides explícitament més totes aquelles que hi havía implícitament i que són comunes a tots els projectes.

Teniu més informació sobre l'*effective POM* en:

- https://books.sonatype.com/mvnref-book/reference/pom-relationships-sect-pom.html

3.2. Generació d'un projecte Maven des de VSCode

Visual Studio ens permet crear projectes Maven a partir d'arquetipus de diferents dipòsits. Simplement, el que fa és llençar per nosaltres les ordres que hem vist anteriorment a la terminal de VSCode.

Veiem com fer-ho. Partint d'una finestra neta del VSCode (*File > New Window*):

1. Busquem la pestanya desplegable «Maven Projects», i el símbol «+» que apareix a la seua dreta.
2. Fem clic en aquest símbol «+» per crear un projecte *Maven* nou.
3. A la paleta d'ordres ens apareixerant els diferents arquetipus de què disposem. Busquem *quickstart*, i seleccionem *maven-archetype-quickstart*.
4. Fet açò se'ns obrirà la terminal integrada de VSCode, i veurem com va generant-se el projecte tal i féiem per consola. En un moment donat ens demanarà la informació necessària per al nostre projecte, de la mateixa manera que es fa al mode interactiu.

Una vegada indicats els paràmetres, se'ns generarà el projecte. Si donem un cop d'ull a l'estructura de carpetes generada i al `pom.xml`, veurem que és el mateix que havíem definit per consola.

Actualització de la caché d'arquetipus

Una altra opció que podem realitzar amb el VSCode, és actualitzar la caché local dels diferents arquetipus des del magatzem central de Maven. Per a això, accedirem a la Consola d'Ordres amb `Ctrl`+`Shift`+`P`. Quan se'ns òbriga aquesta, començarem a escriure *Maven:…* i ja tindrem a la vista l'opció d'*Update Maven Archetype Catalog*.

4. Maven en Netbeans

Com a últim apartat d'aquest punt, anem a veure com gestionar projectes Maven des de l'IDE Netbeans.

Per tal de crear un nou projecte amb Maven, ho farem de la mateixa forma que amb Ant: *File -> New Project*, però ara escollint un projecte de tipus *Project from Archetype* de la categoría *Java with Maven*:

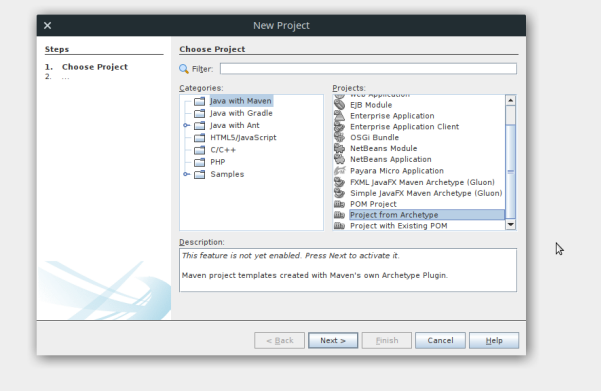

Figura 3: Creació d'un projecte Maven en Netbeans

En la següent finestra haurem d'indicar l'arquetipus. Veurem que tarda una miqueta a descarregar-se tots els arquetipus possibles. Al quadre de cerca de dalt, podem començar a escriure l'arquetipus per filtrar els resultats. Recordeu que volem un projecte a partir de l'arquetipus ***maven-archetype-quickstart***:

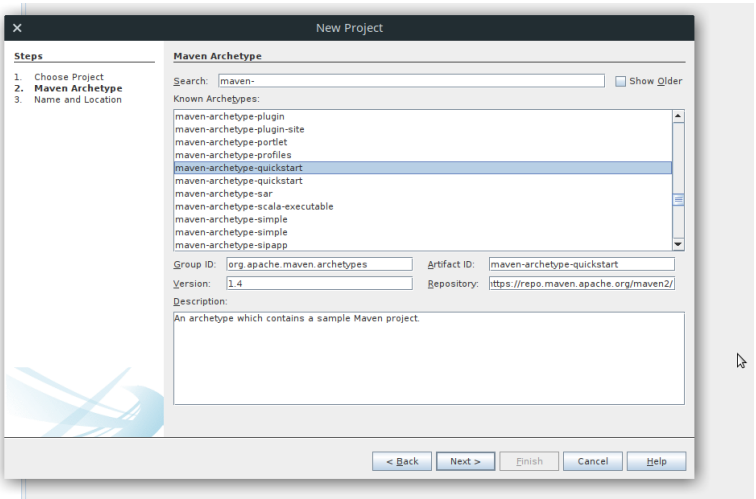

Figura 4: Selecció de l'arquetipus

Una vegada escollit l'arquetipus, triem el nom del projecte (HelloMaven), la carpeta de destí, i el nom de l'organització (GroupID: com.ieseljust.edd), així com la versió, i el nom del paquet (com.ieseljust.edd.helloMaven):

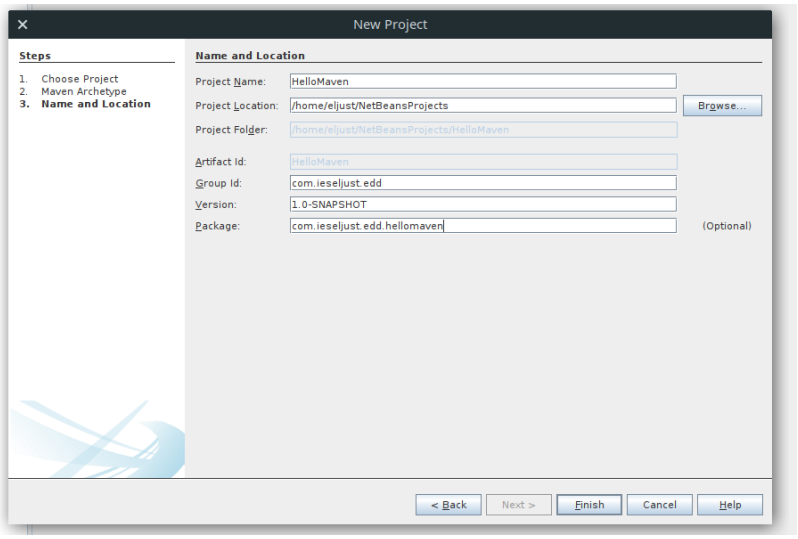

Figura 5: Selecció de l'arquetipus

Fet açò, veurem que ens genera la classe `App` dins el paquet que hem indicat amb el codi de l'Hola Món, i com dins el projecte es troba també, en la carpeta *Project Files* el fitxer `pom.xml`

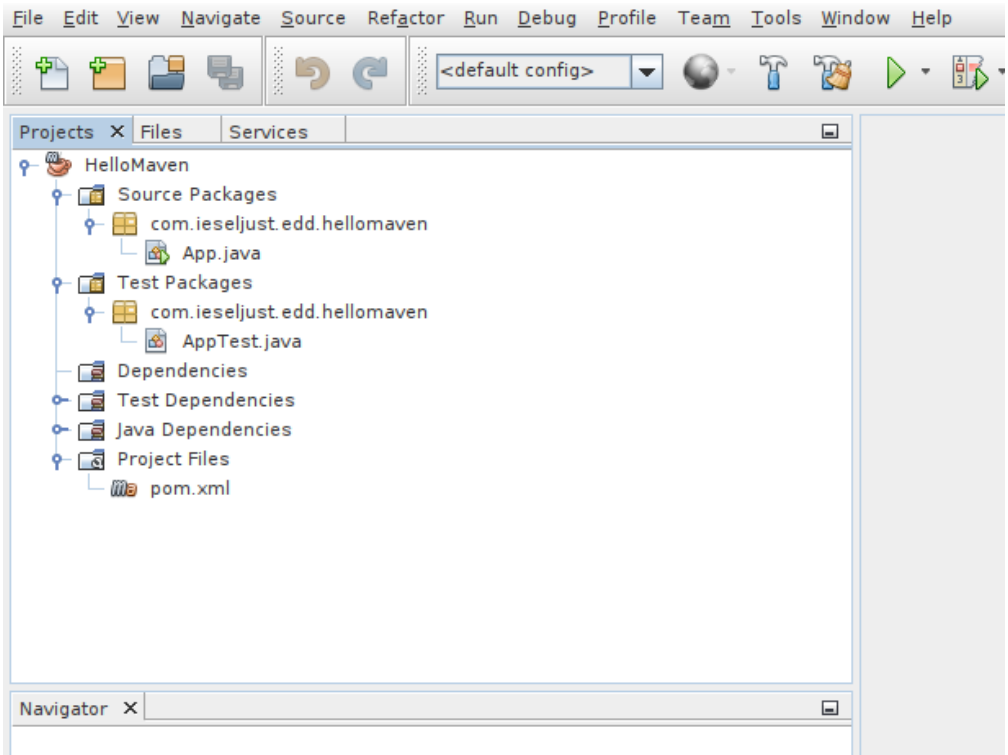

Figura 6: Projecte HelloMaven

> **TO-DO**
>
> Creeu un projecte Maven per al vostre codi de la calculadora. Per a això, partireu de l'arquetipus *quickstart* de maven per defecte, i haureu de crear les dues classes necessàries per al seu funcionament. Aneu en compte amb els noms dels paquets.
>
> Podeu utilitzar qualsevol dels mecanismes i IDEs què hem vist (eina `mvn` pròpia de Maven, el plugin de Maven per a VSCode, o bé Netbeans).