

# DEMO MAVEN

## AVEC ECLIPSE (WINDOWS 10)

### Table des matières

<b>1.</b>	<b><u>INTRODUCTION .....</u></b>	<b><u>4</u></b>
<b>2.</b>	<b><u>PREREQUIS.....</u></b>	<b><u>4</u></b>
<b>3.</b>	<b><u>TELECHARGEMENT ET INSTALLATION .....</u></b>	<b><u>5</u></b>
<b>3.0</b>	<b>TELECHARGEMENT ET INSTALLATION JDK .....</b>	<b>5</b>
<b>3.1</b>	<b>TELECHARGEMENT ET INSTALLATION MAVEN .....</b>	<b>7</b>
<b>3.2</b>	<b>TELECHARGEMENT ET CONFIGURATION ECLIPSE.....</b>	<b>8</b>
<b>4.</b>	<b><u>CREATION PROJET MAVEN.....</u></b>	<b><u>12</u></b>
<b>4.0</b>	<b>CREATION PROJET SIMPLE MAVEN .....</b>	<b>12</b>
<b>4.1</b>	<b>CREATION CLASSE JAVA DANS LA PARTIE SOURCE.....</b>	<b>16</b>
<b>4.2</b>	<b>CREATION D'UNE CLASSE JAVA JUNIT TEST CASE .....</b>	<b>1ERROR! BOOKMARK NOT DEFINED.</b>
<b>5.</b>	<b><u>DEPEDANCE .....</u></b>	<b><u>23</u></b>
<b>5.0</b>	<b>AJOUT D'UNE DEPENDANCE .....</b>	<b>23</b>
<b>5.1</b>	<b>DEPENDANCE TRANSITIVE .....</b>	<b>25</b>
<b>5.2</b>	<b>EXCLUSION DE DEPANDANCES.....</b>	<b>26</b>

<b><u>6.</u></b>	<b><u>COMMANDES MAVEN.....</u></b>	<b><u>26</u></b>
<b>6.0</b>	<b>COMMANDE CLEAN.....</b>	<b>30</b>
<b>6.1</b>	<b>COMMANDE COMPILE.....</b>	<b>32</b>
<b>6.2</b>	<b>COMMANDE TEST-COMPILE .....</b>	<b>35</b>
<b>6.3</b>	<b>COMMANDE TEST.....</b>	<b>38</b>
<b>6.4</b>	<b>COMMANDE INSTALL .....</b>	<b>42</b>
<b><u>7.</u></b>	<b><u>PROJET MULTI-MODULES .....</u></b>	<b><u>44</u></b>
<b>7.0</b>	<b>CREATION D'UN PROJET PARENT.....</b>	<b>44</b>
<b>7.1</b>	<b>CREATION DU MODULE SERVEUR .....</b>	<b>47</b>
<b>7.2</b>	<b>CREATION DU MODULE CLIENT .....</b>	<b>50</b>
<b>7.3</b>	<b>DEPENDANCES ENTRE PROJETS ET MODULES .....</b>	<b>53</b>
<b><u>8.</u></b>	<b><u>DEPLOIEMENT .....</u></b>	<b><u>59</u></b>
<b>8.0</b>	<b>CREATION D'UN REPERTOIRE GITHUB .....</b>	<b>59</b>
<b>8.1</b>	<b>CONFIGURATION DU POM ET COMMANDE DEPLOY .....</b>	<b>59</b>

## 1. INTRODUCTION

Ce document comprend la procédure à suivre pour installer, configurer et exécuter les fonctions principales de maven en utilisant eclipse.

## 2. PREREQUIS

- Maven 3 (dans notre cas 'apache-maven-3.3.9')
- Eclipse Kepler + (dans notre cas eclipse LUNA)
- JDK 1.5 + (dans notre cas JDK 1.8)

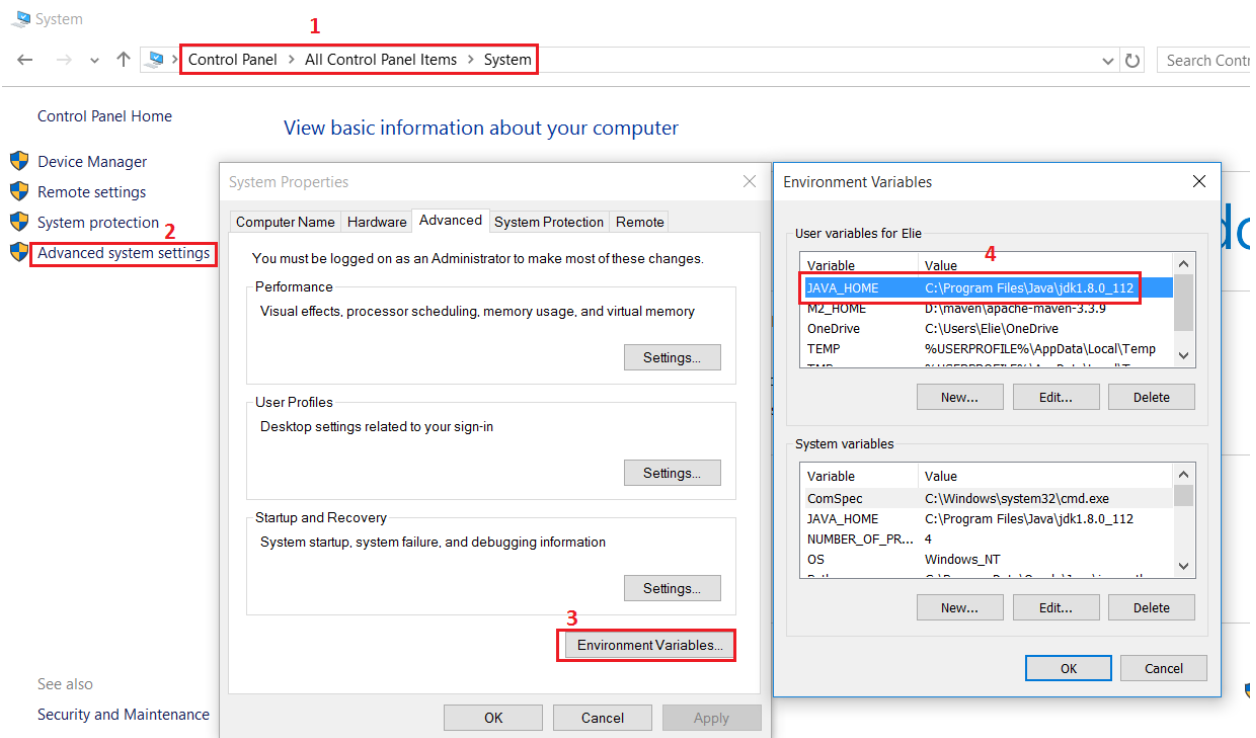
### 3. TELECHARGEMENT ET INSTALLATION

#### 3.0 TELECHARGEMENT ET INSTALLATION JDK

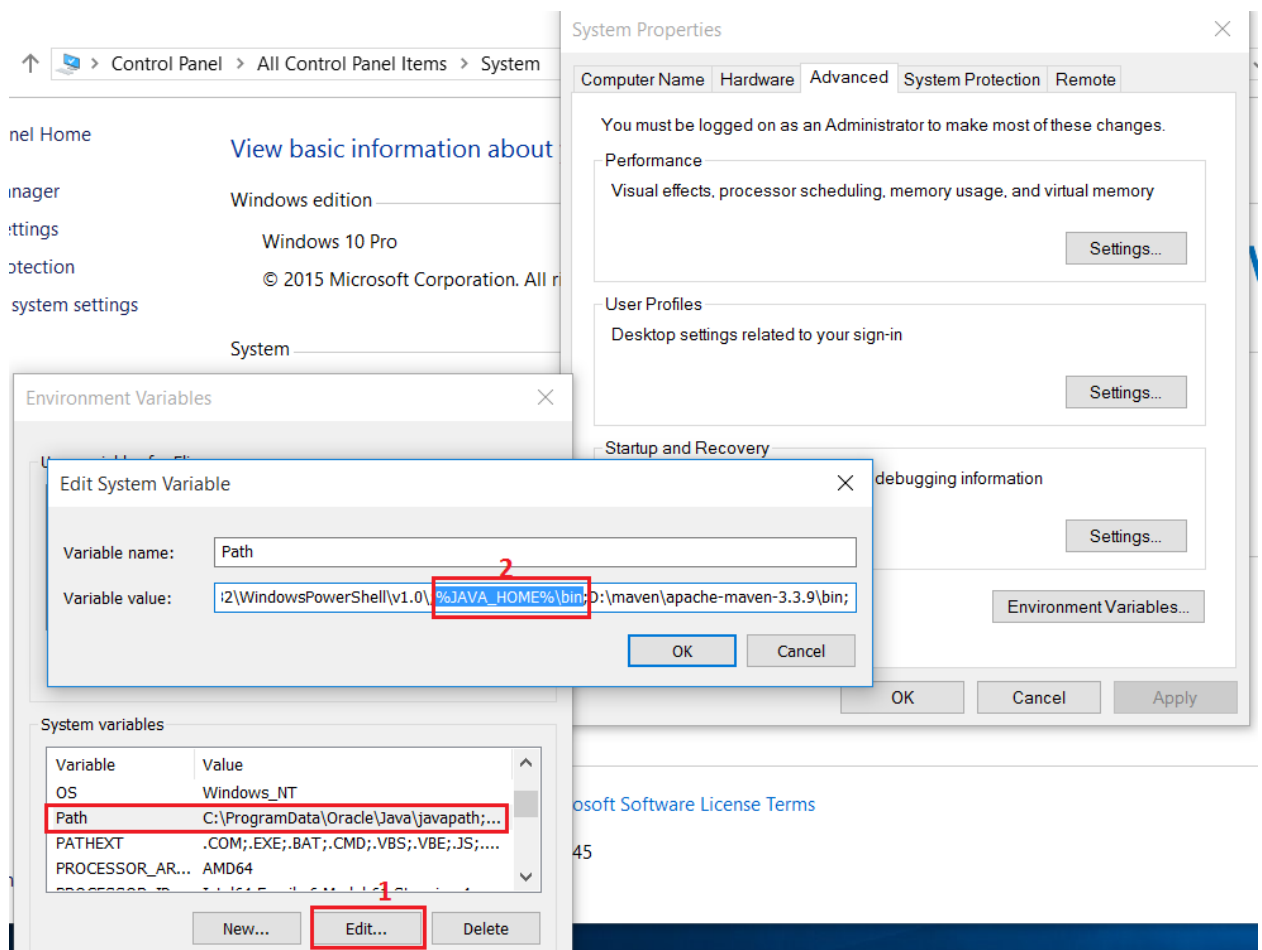
Maven est un outil écrit en Java : Java doit donc être installé sur la machine.

Donc il faut :

- Télécharger le JDK (version 1,5 ou plus) sur le site : <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Décompresser l'archive dans un répertoire du système
- Créer la variable d'environnement JAVA\_HOME qui pointe sur le répertoire contenant le JDK :



- Ajouter le chemin JAVA\_HOME/bin à la variable PATH du système :



- Vérifier que le JDK est bien installé sur notre système en exécutant la commande `java -version`:

```

Command Prompt

Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Elie>java -version
java version "1.8.0_112"
Java(TM) SE Runtime Environment (build 1.8.0_112-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.112-b15, mixed mode)

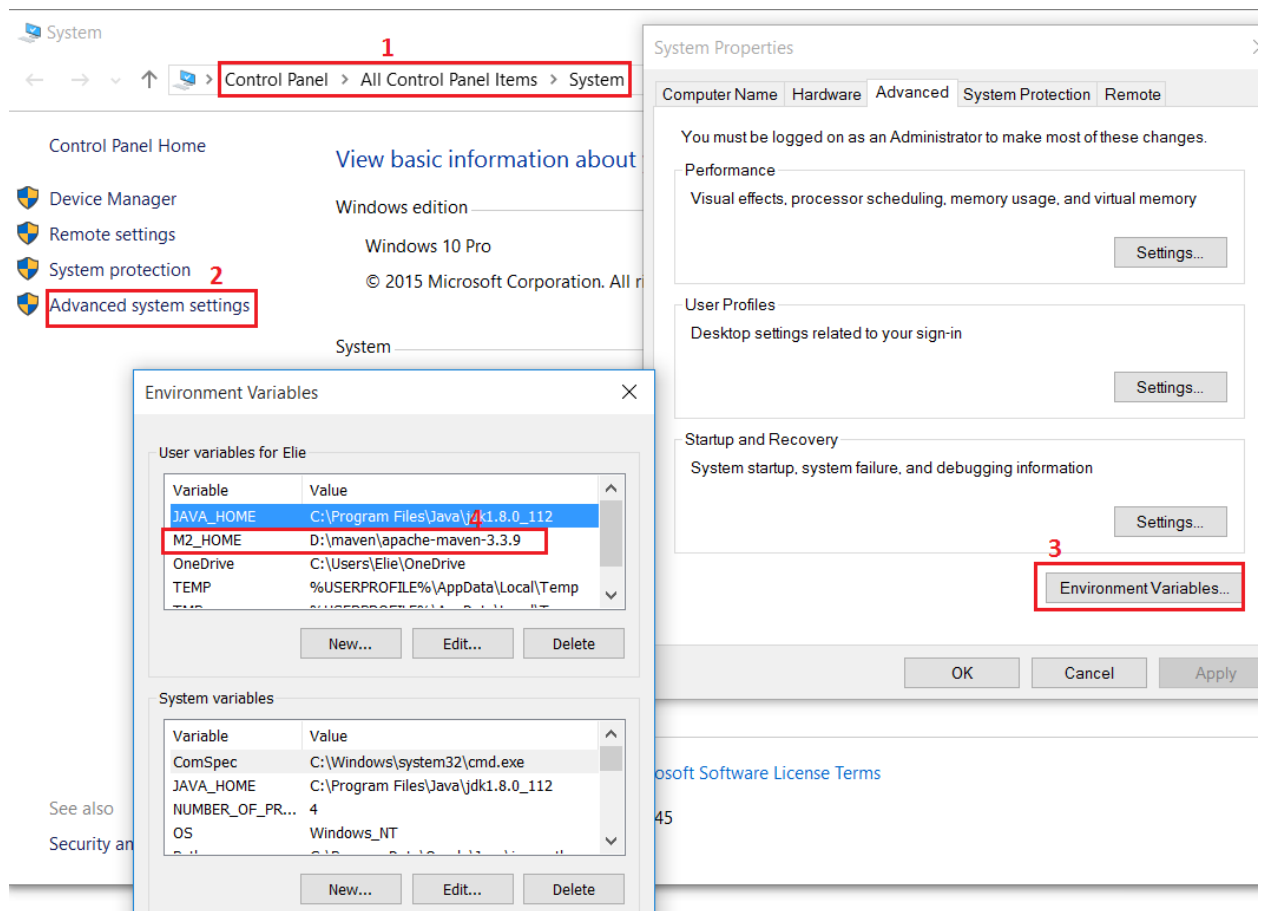
C:\Users\Elie>

```

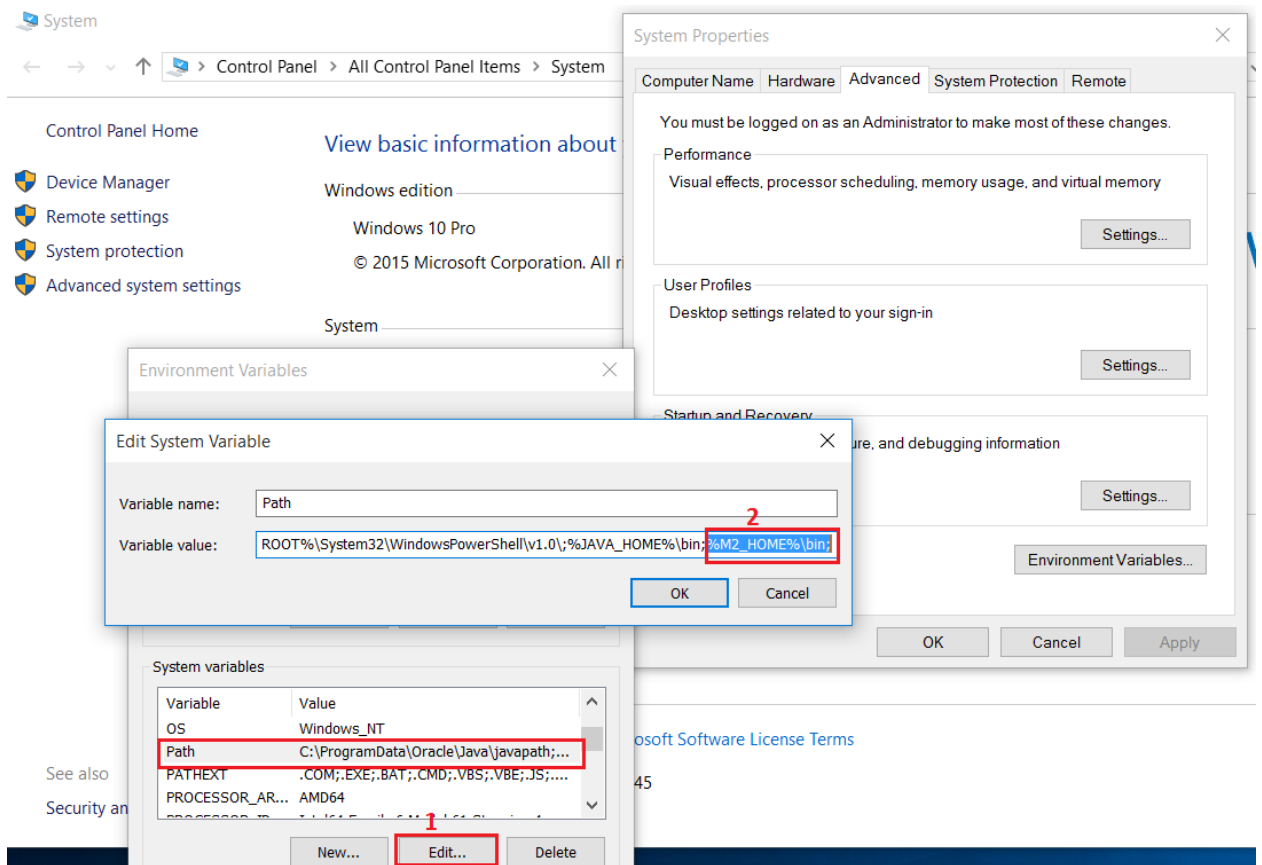
### 3.1 TELECHARGEMENT ET INSTALLATION MAVEN

Pour installer Maven il faut :

- Télécharger l'archive (apache-maven-3.3.3-bin.zip) sur le site : <http://maven.apache.org/download.cgi>
- Décompresser l'archive dans un répertoire du système
- Créer la variable d'environnement M2\_HOME qui pointe sur le répertoire contenant Maven



- Ajouter le chemin M2\_HOME/bin à la variable PATH du système



Pour vérifier l'installation, il faut lancer la commande mvn -version :

```

C:\Users\Elie>mvn -version
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Elie>mvn -version
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T18:41:47+02:00)
Maven home: D:\maven\apache-maven-3.3.9
Java version: 1.8.0_112, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_112\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "dos"

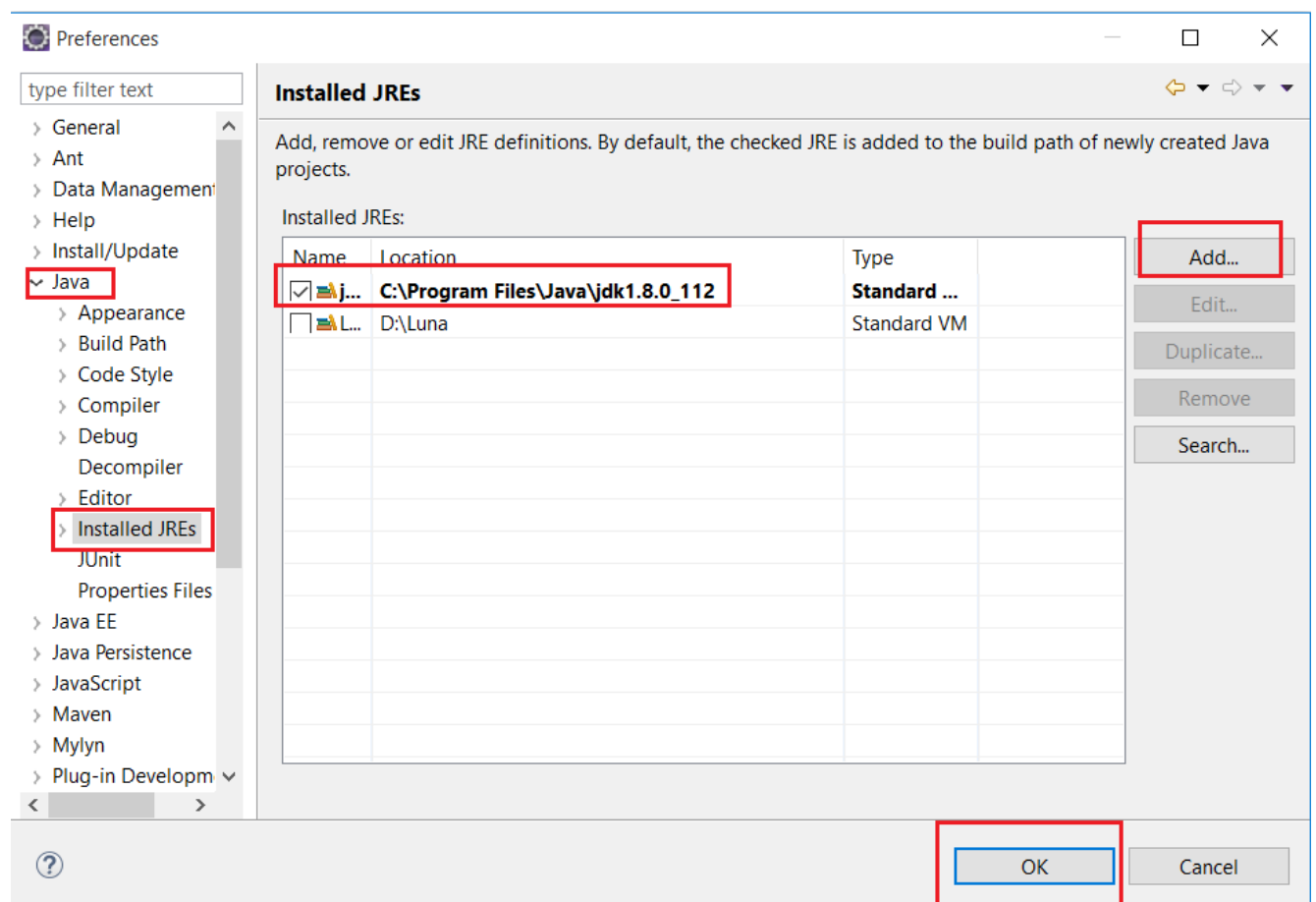
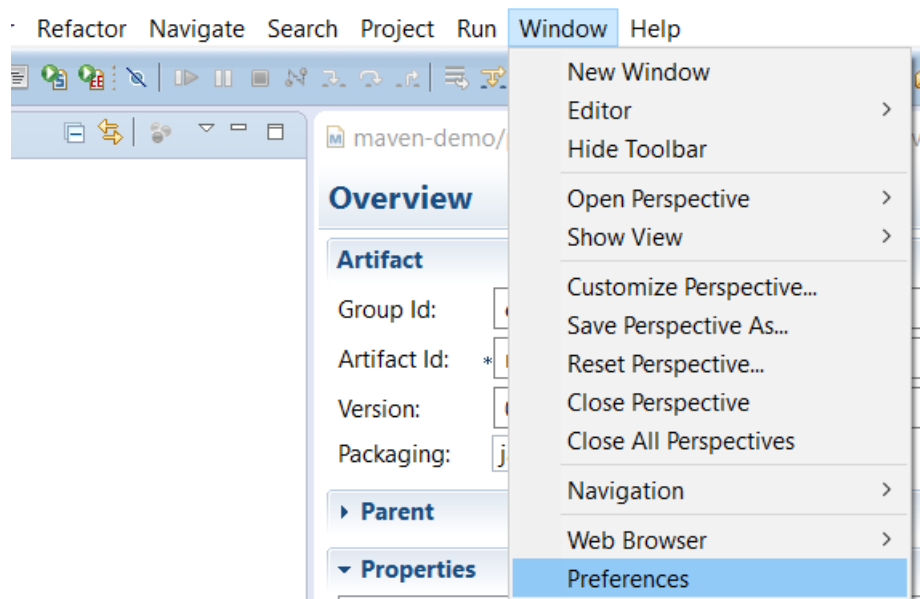
C:\Users\Elie>
  
```

## 3.2 TELECHARGEMENT ET CONFIGURATION ECLIPSE

Pour télécharger eclipse, il faut aller au site : <http://www.eclipse.org/downloads/packages/> , choisir 'Eclipse IDE for Java EE Developers' et télécharger la version désirée.

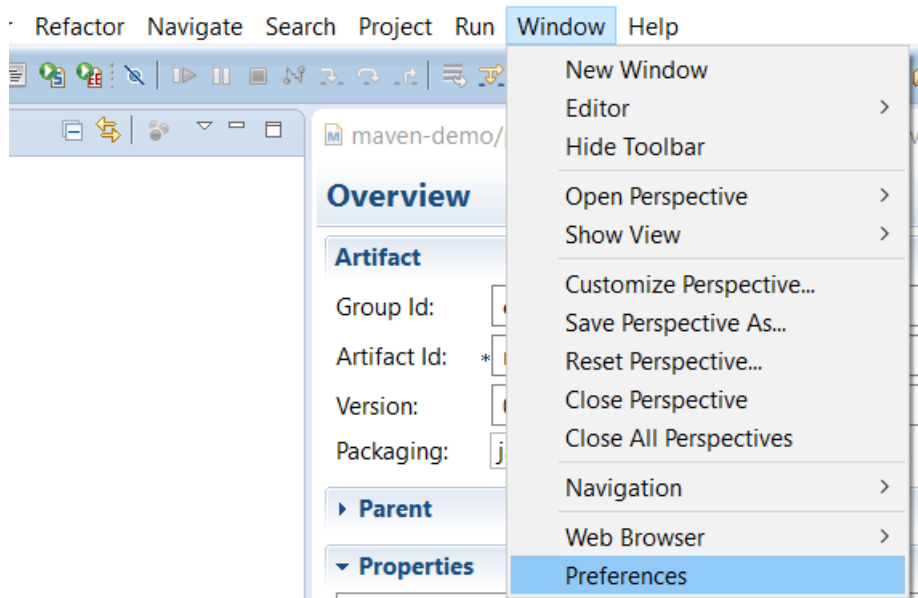
Choisir le JDK avec lequel on développera les projets :

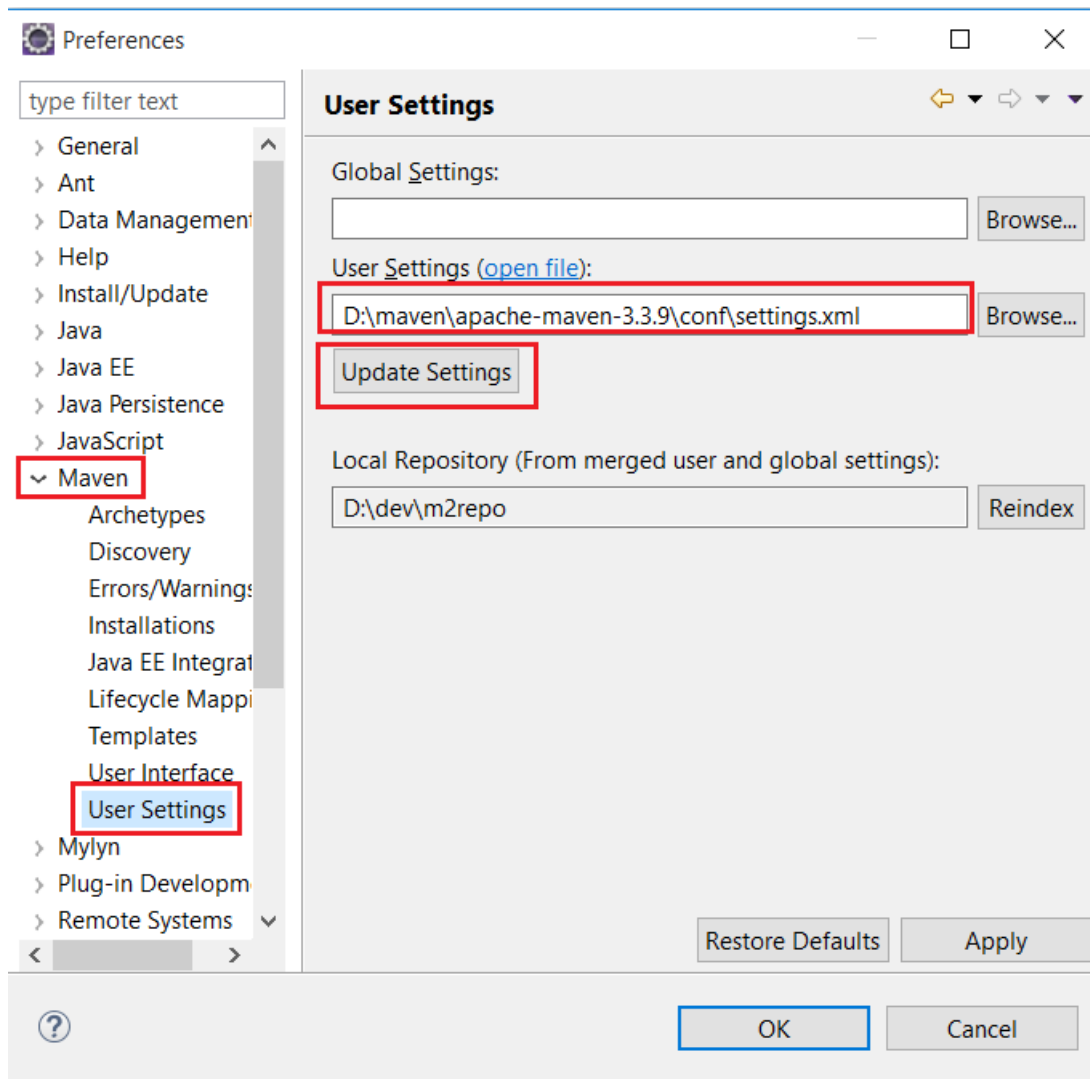
Windows → Préférences → Java → Installed JRE → Add → choisir le repertoire de notre JDK → OK :





Pour configurer Maven : Window → Preference → Maven → User Settings → Browse → choisir le path du fichier settings.xml → update settings → OK:

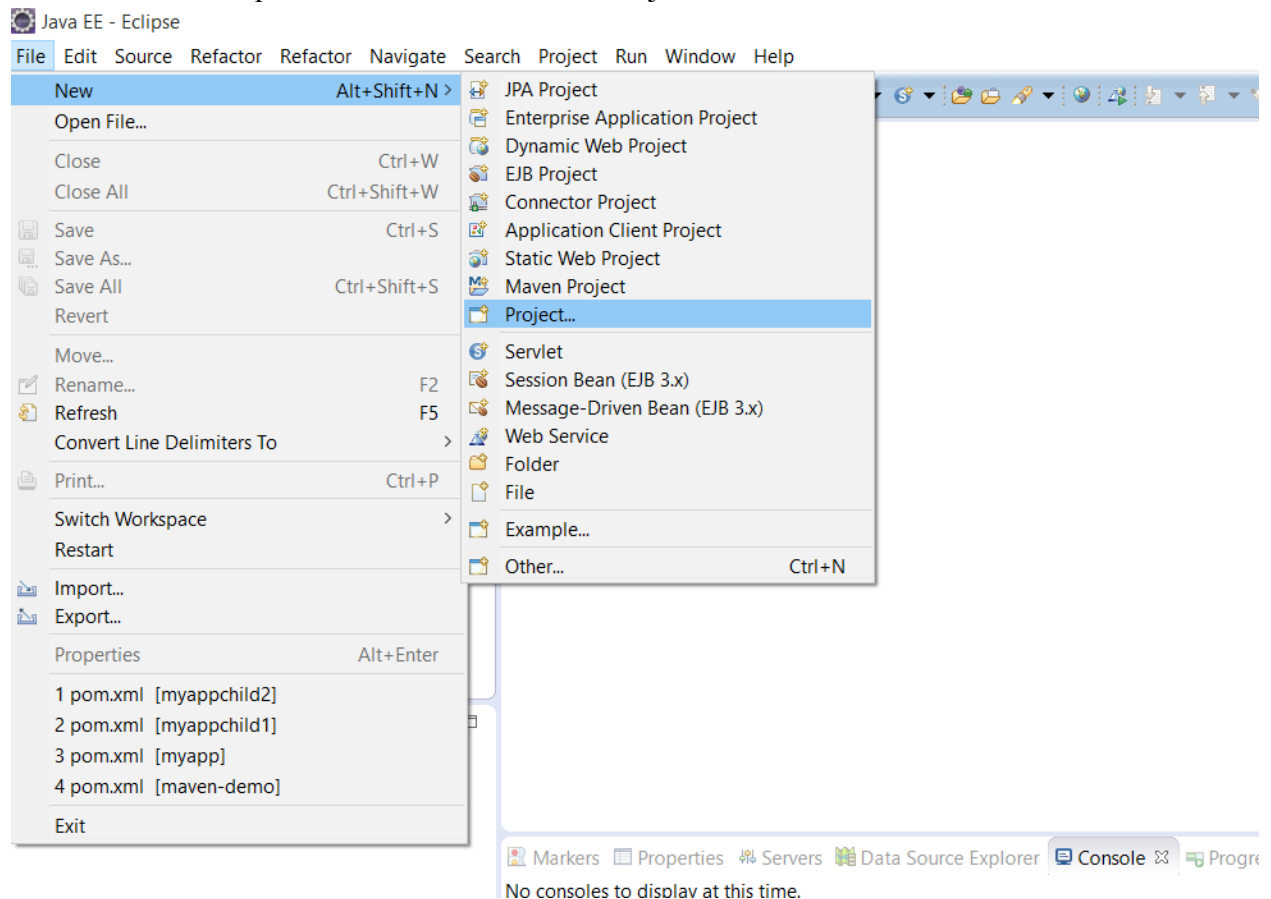




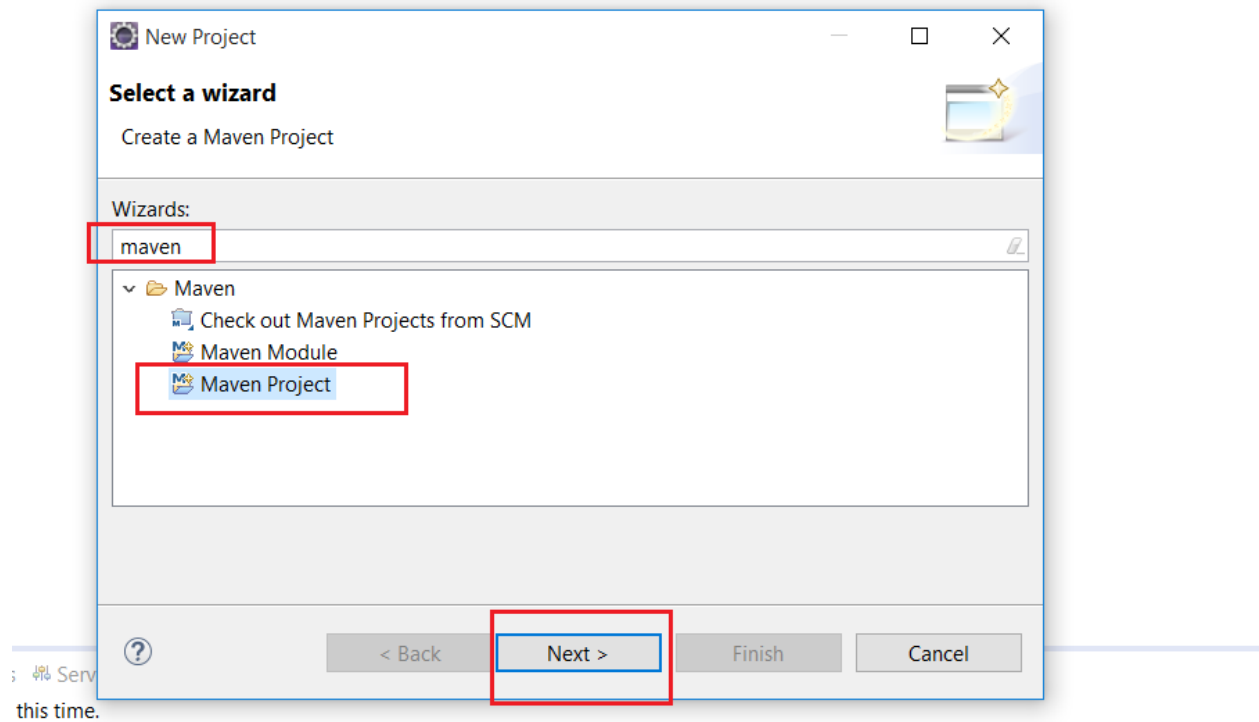
## 4. CREATION PROJET MAVEN

### 4.0 CREATION PROJET SIMPLE MAVEN

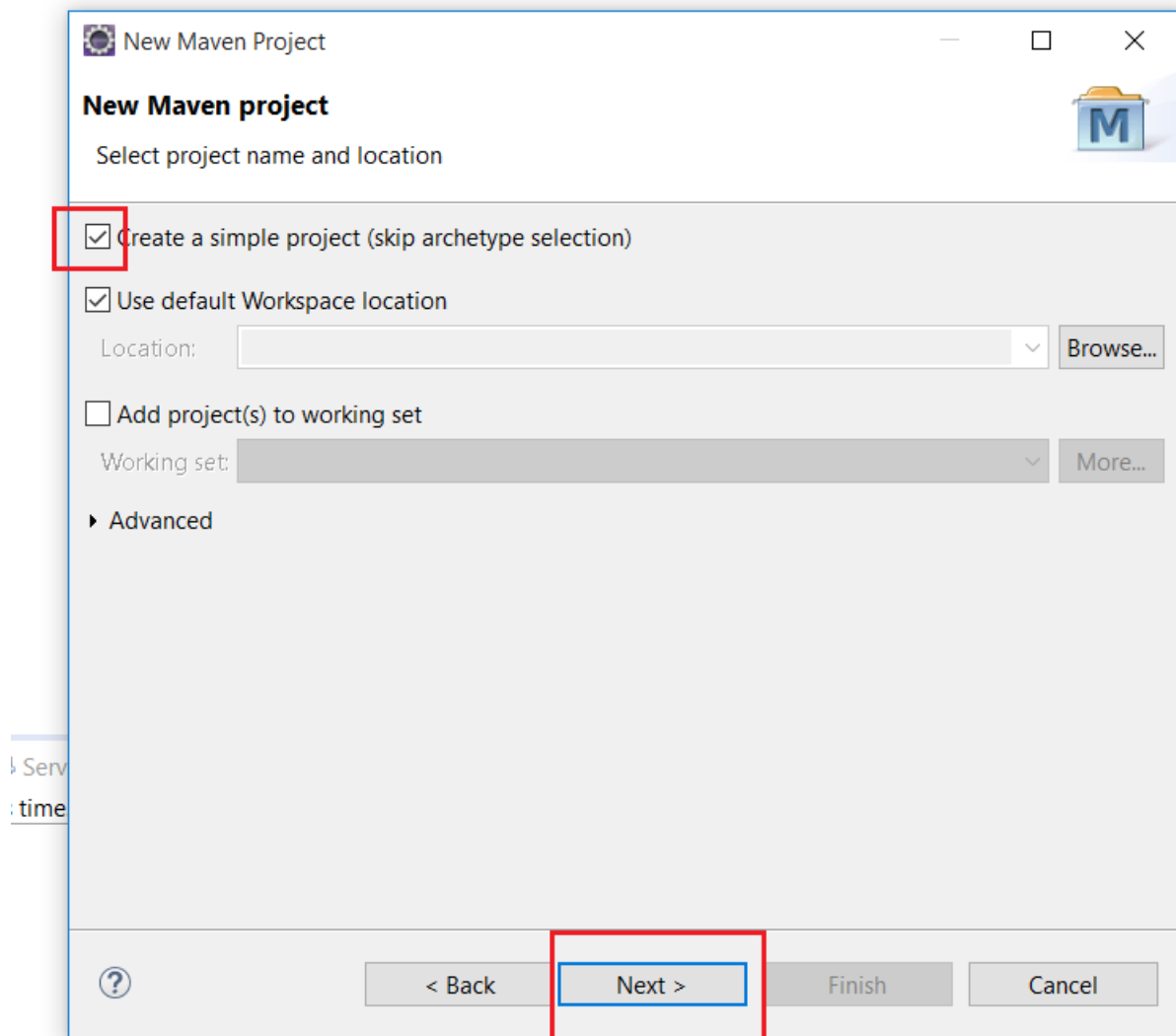
Sur le menu on clique sur 'File' → 'New' → 'Project'



On tape 'maven' et on choisit 'Maven Project' → 'Next'



On sélectionne 'Create simple project' → 'Next'



On saisit le 'group Id', 'Artifac Id' ; la version et Packaging étant remplis par défaut,  
Puis on clique 'Finish'

**New Maven Project**

Configure project

**Artifact**

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

**Parent Project**

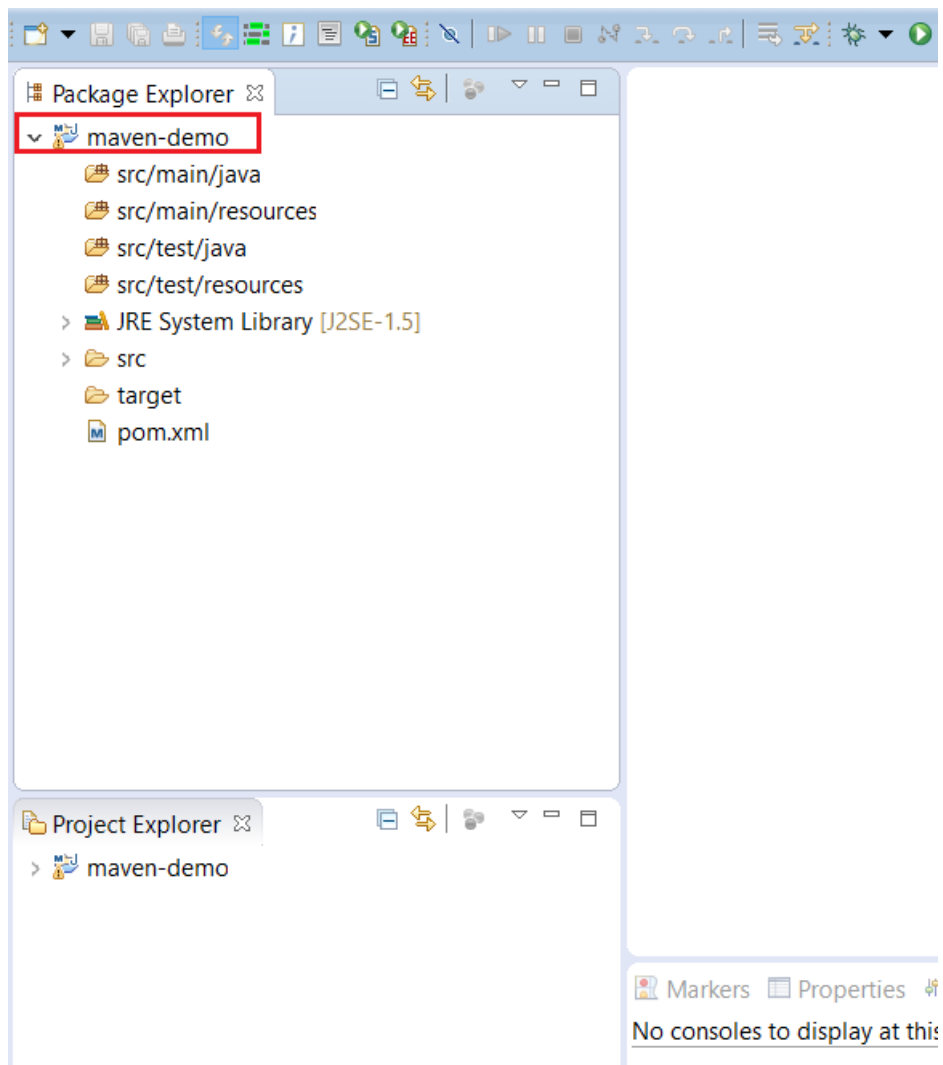
Group Id:

Artifact Id:

Version:

► Advanced

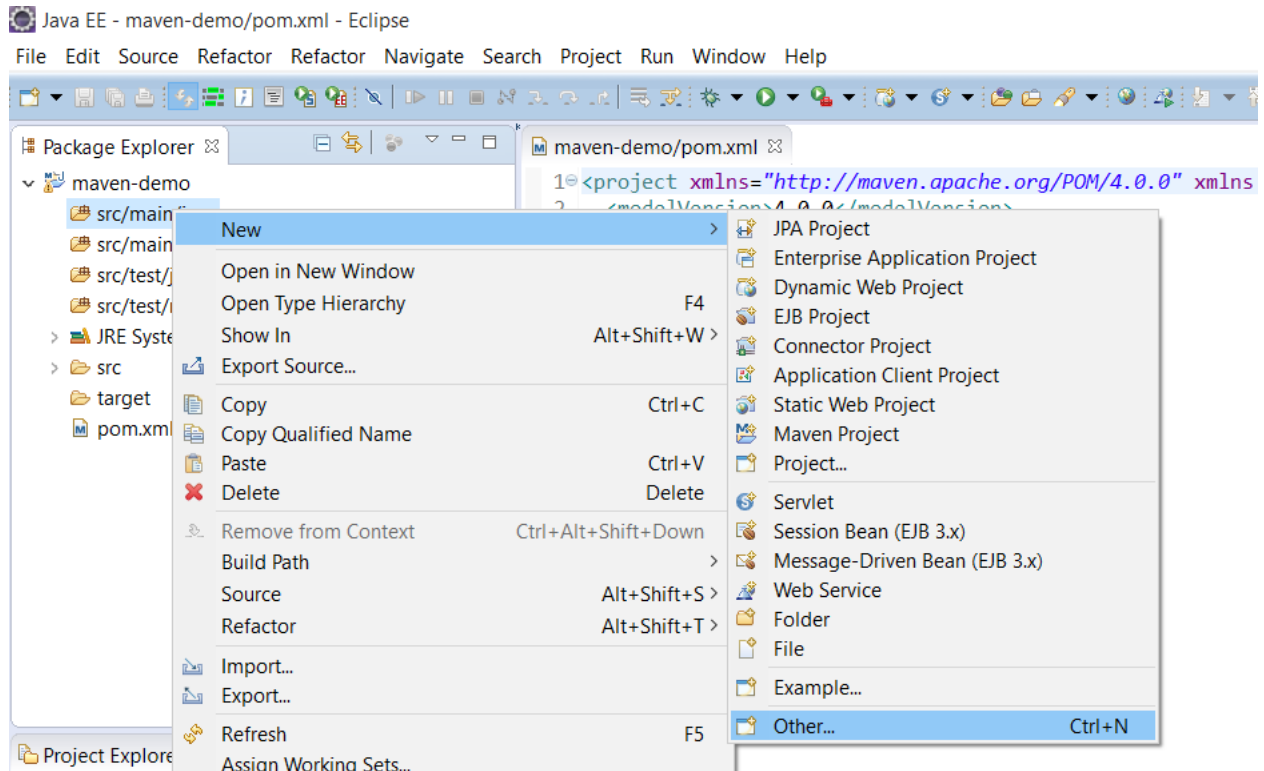
Eclipse va créer le projet Maven 'maven-demo'



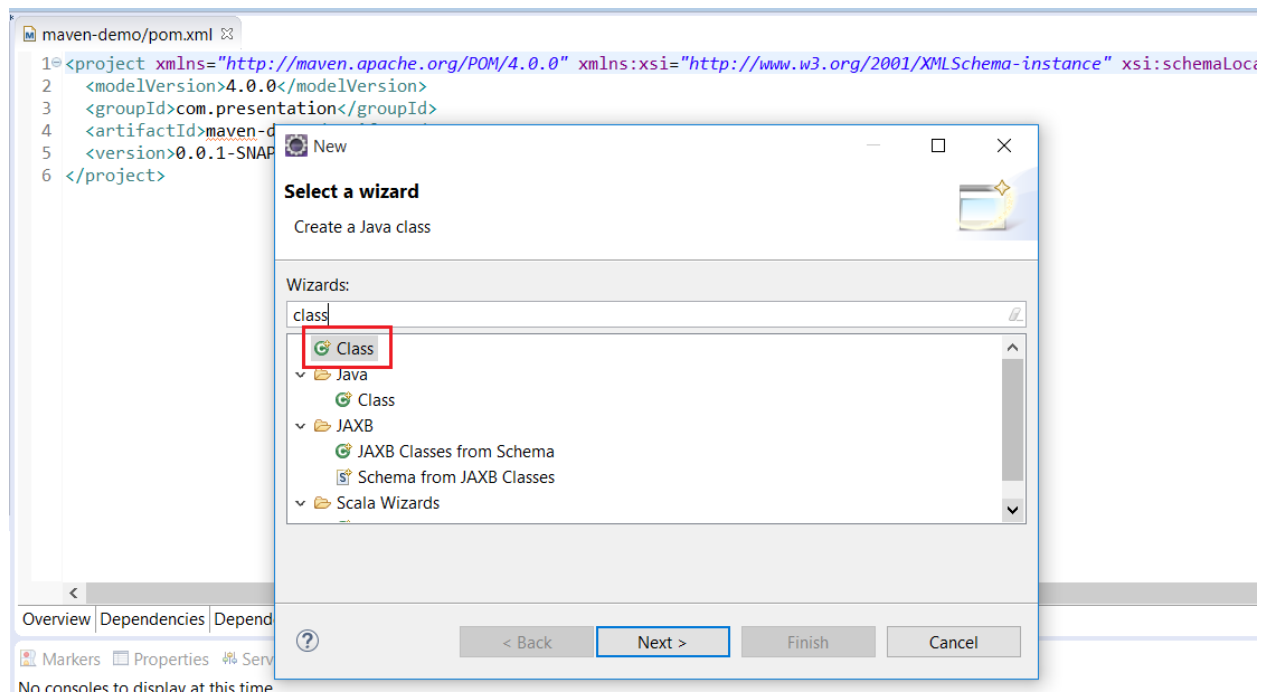
#### 4.1 CREATION CLASSE JAVA DANS LA PARTIE SOURCE

On va créer une classe java 'App.java':

On fait right click sur 'src/main/java' → 'New' → 'Other'

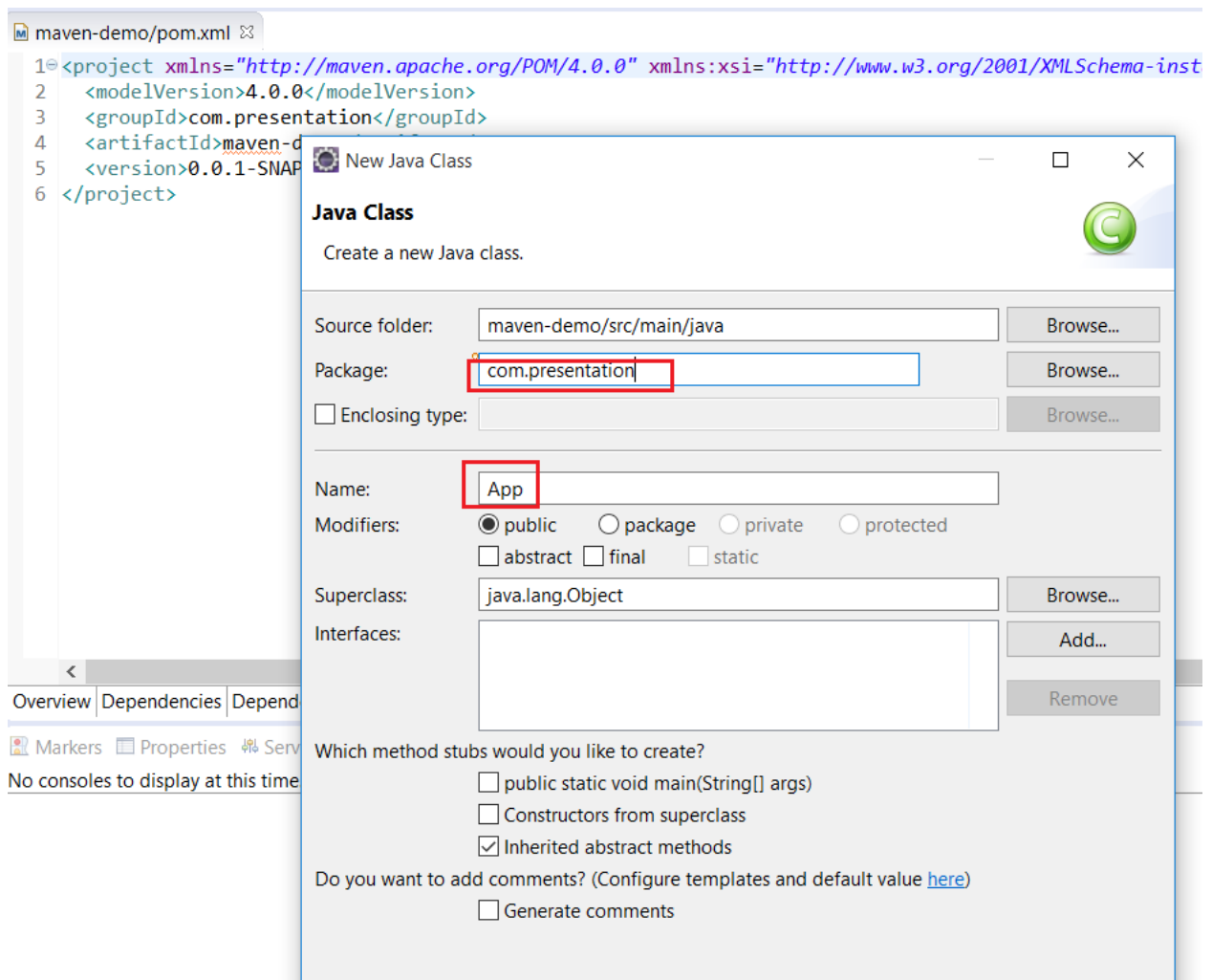


On tape 'Class' et on choisit 'Class' puis on clique 'Next'

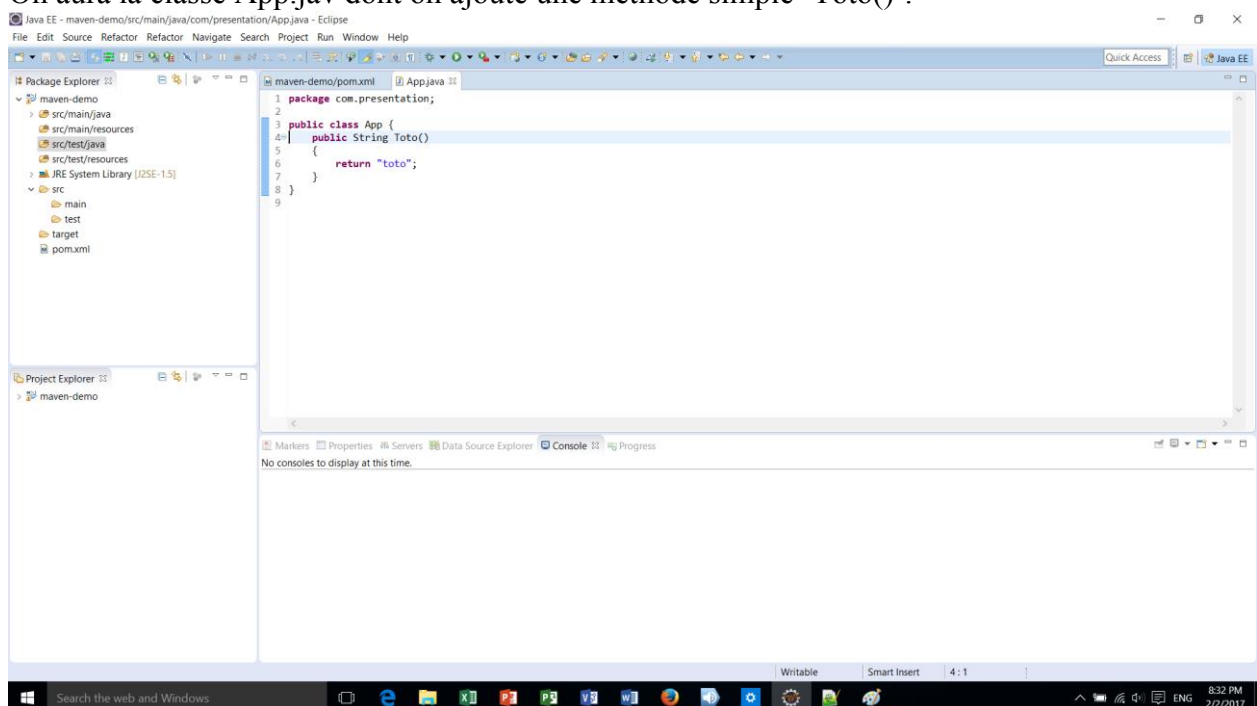


On saisit le package par la même valeur que celle du group Id.  
Comme on saisit le nom de la classe, 'App' dans notre exemple.  
On clique 'Finish'.

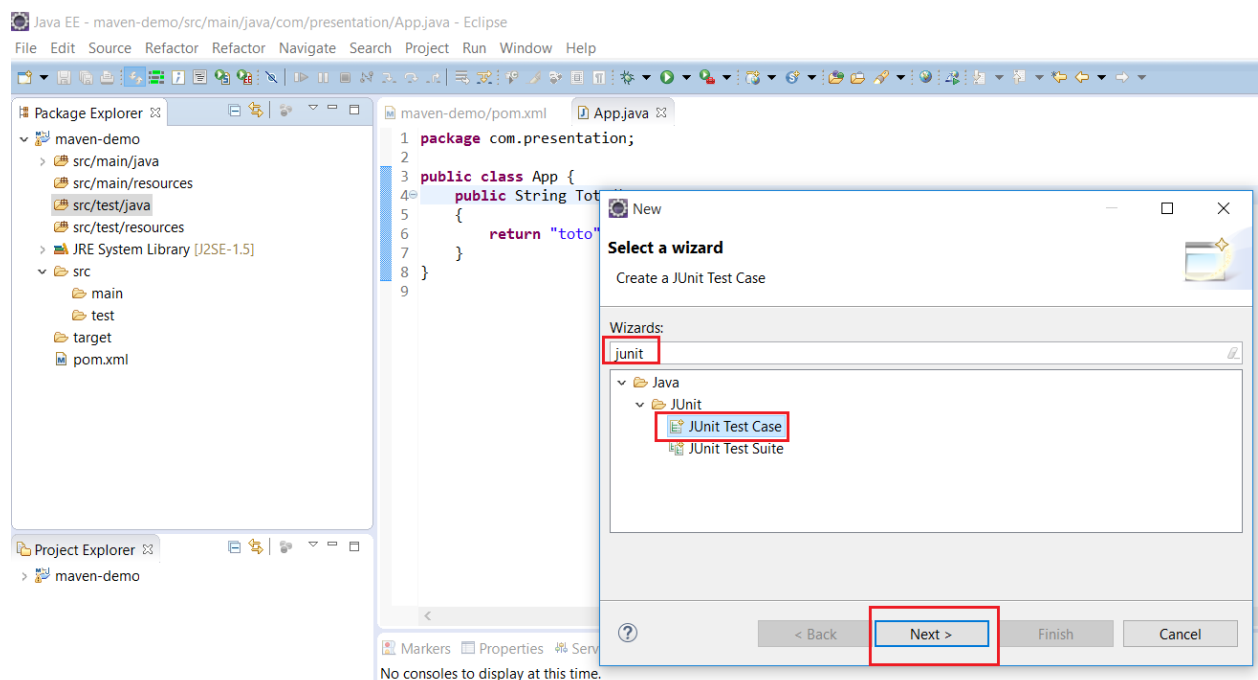
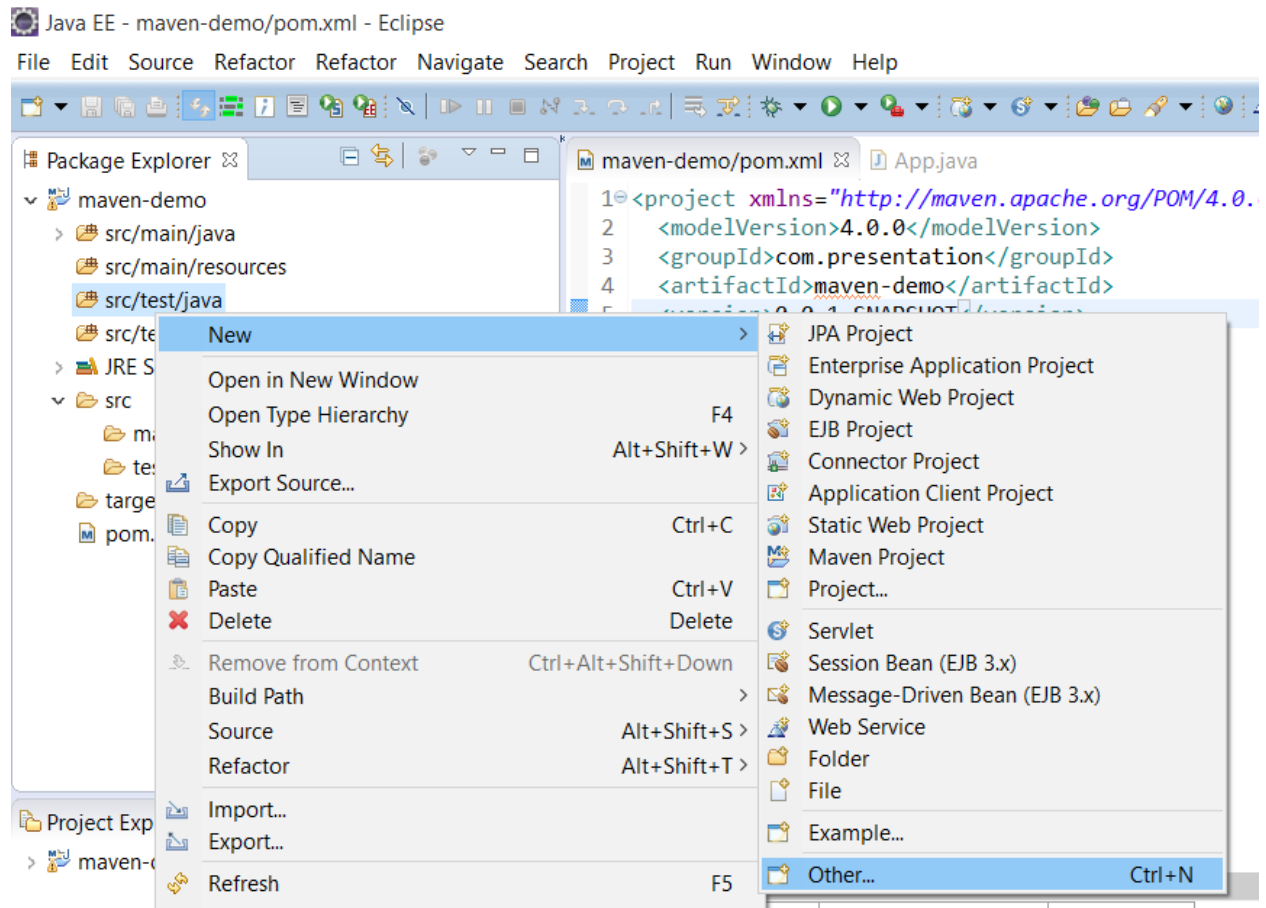





On aura la classe App.java dont on ajoute une méthode simple 'Toto()'.



## 4.2 CREATION D'UNE CLASSE JAVA JUNIT TEST CASE



 New JUnit Test Case

**JUnit Test Case**

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

Package:

Name:

Superclass:


Which method stubs would you like to create?

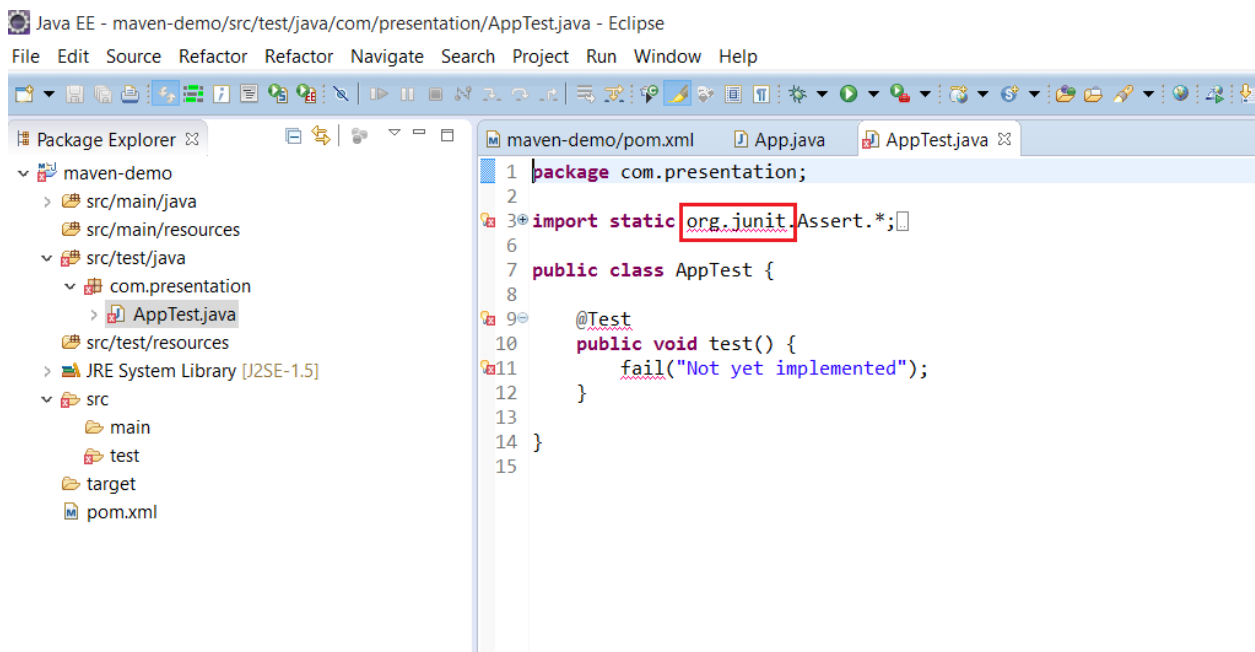
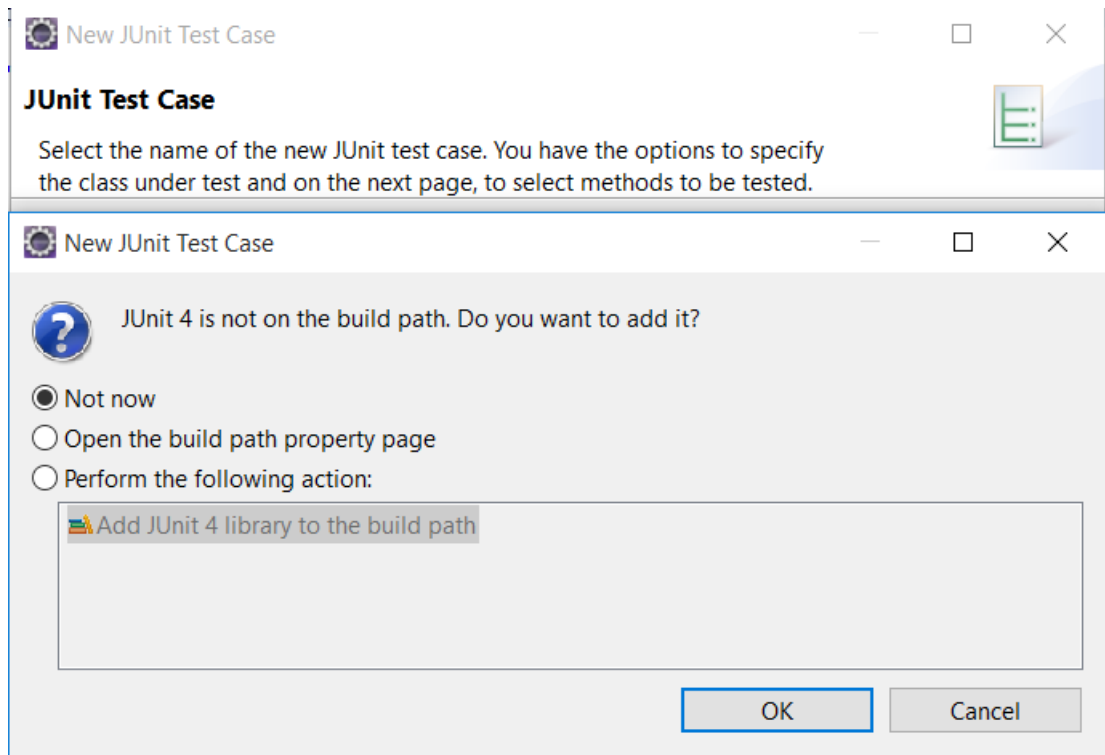
☐ setUpBeforeClass() ☐ tearDownAfterClass()  
☐ setUp() ☐ tearDown()  
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test:





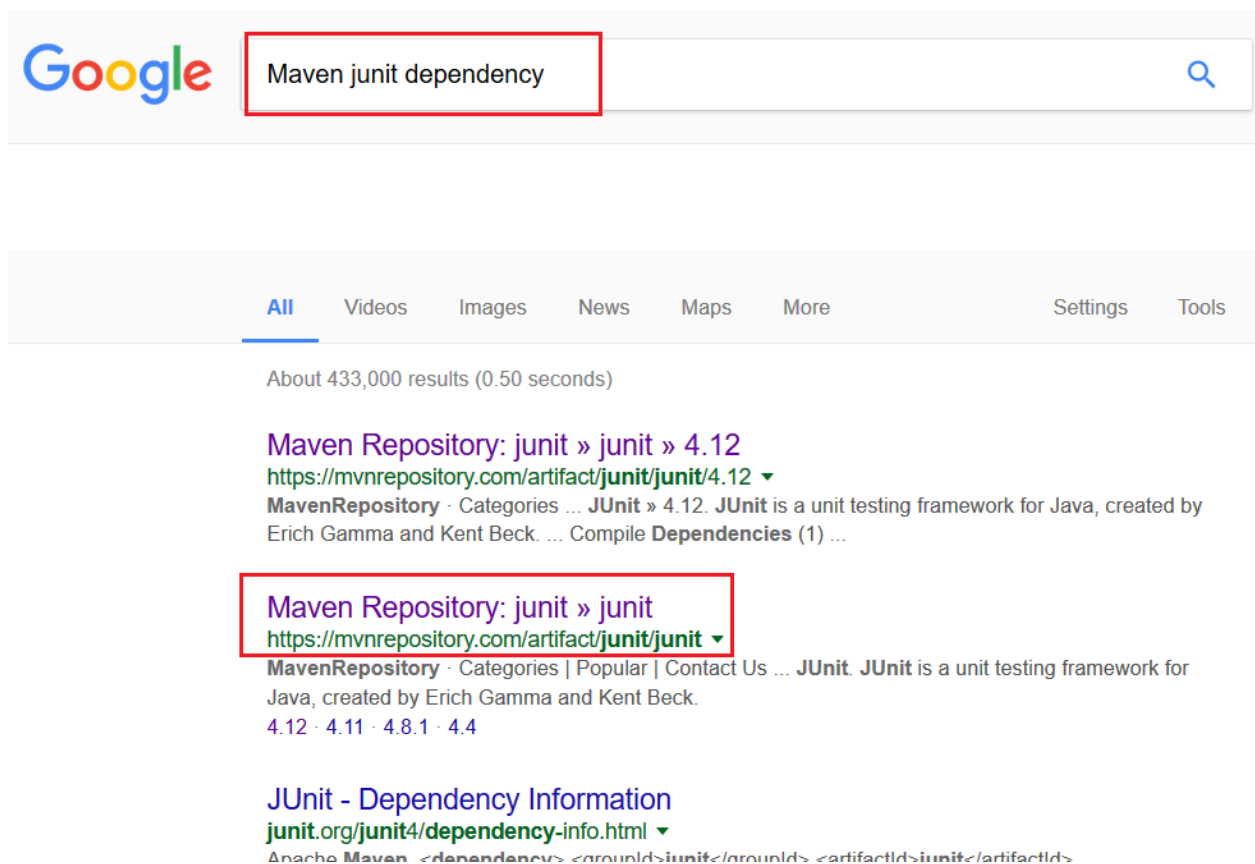
On remarque des erreurs, parce qu'on n'a pas le Junit.jar dans notre buildpath.

## 5. DEPEDANCE

### 5.0 AJOUT D'UNE DEPENDANCE

Normalement si on n'a pas Maven, il faut télécharger le jar et l'ajouter au buildPath.  
Mais comme ayant Maven, Il faut faire les etapes suivantes :

1. chercher dans google la dependance et choisir celle ayant  
l'url : <https://mvnrepository.com/artifact/...>



2. Choisir la version convenable



## JUnit

JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.

License	EPL 1.0
Categories	Testing Frameworks
Tags	testing
Used By	55,211 artifacts

	Version	Repository	Usages	Date
4.12.x	4.12	Central	16,535	(Dec, 2014)
	4.12-beta-3	Central	28	(Nov, 2014)
	4.12-beta-2	Central	31	(Sep, 2014)
	4.12-beta-1	Central	30	(Jul, 2014)
4.11.x	4.11	Central	18,887	(Nov, 2012)
	4.11-redhat-1	Redhat GA	18	(Apr, 2014)
	4.11-20120805...	Alfresco Public	4	(Sep, 2012)
	4.11-beta-1	Central	22	(Oct, 2012)
	4.10	Central	7,987	(Sep, 2011)

### 3. Copier le code XML de la dépendance



## JUnit » 4.12

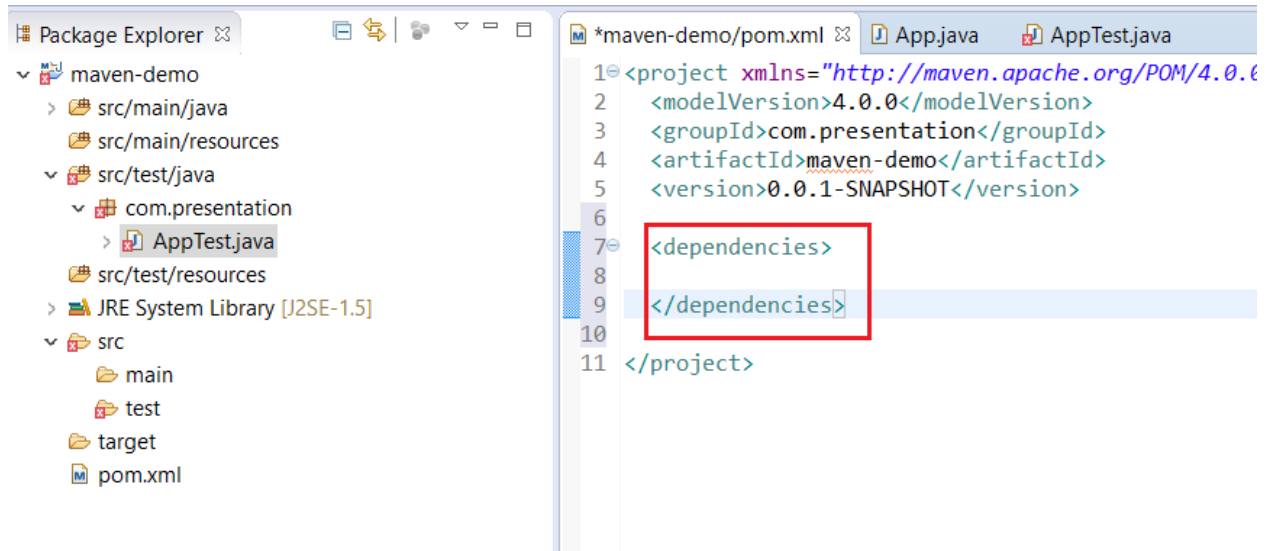
JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.

License	EPL 1.0
Categories	Testing Frameworks
Organization	JUnit
HomePage	<a href="http://junit.org">http://junit.org</a>
Date	(Dec 04, 2014)
Files	<a href="#">Download (JAR)</a> (195 KB)
Repositories	Central Sonatype Releases
Used By	55,211 artifacts

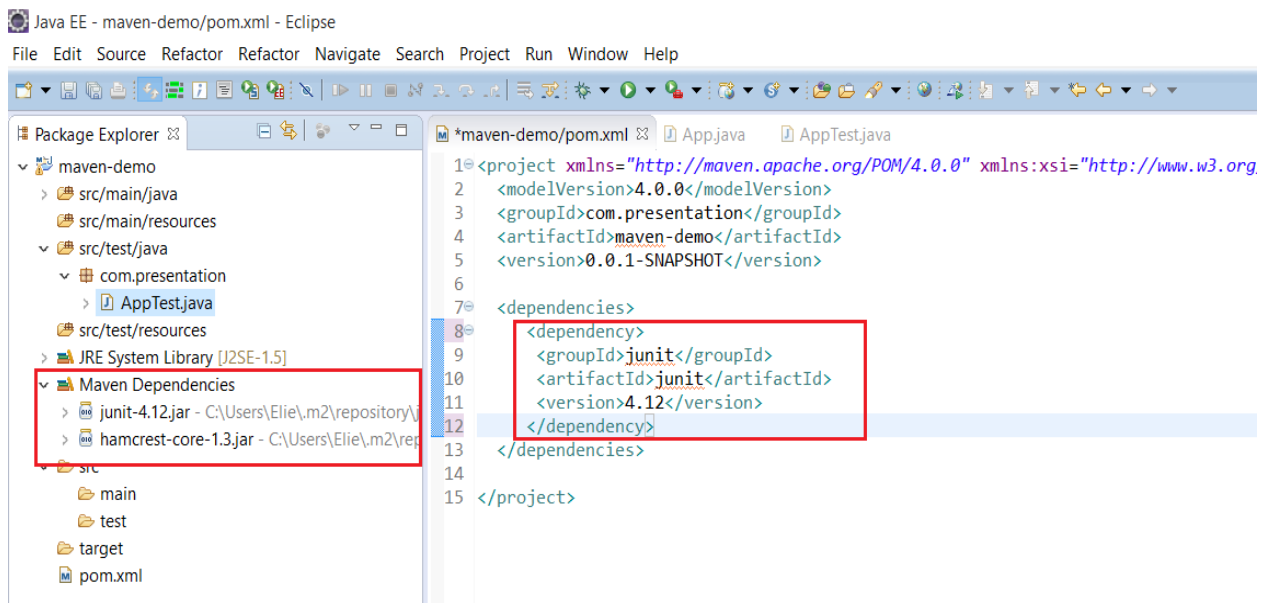
Maven	Gradle	SBT	Ivy	Grape	Leiningen	Buildr
<pre>&lt;!-- https://mvnrepository.com/artifact/junit/junit --&gt; &lt;dependency&gt;   &lt;groupId&gt;junit&lt;/groupId&gt;   &lt;artifactId&gt;junit&lt;/artifactId&gt;   &lt;version&gt;4.12&lt;/version&gt; &lt;/dependency&gt;</pre>						

☒ Include comment with link to declaration

### 4. Ouvrir les balises <dependencies> dans notre pom.xml pour coller nos dépendances nécessaires.



- On colle notre dépendance JUnit et on sauvegarde ; On remarque l'apparition de la librairie 'Maven Dependencies' contenant les jars ajoutés.



```

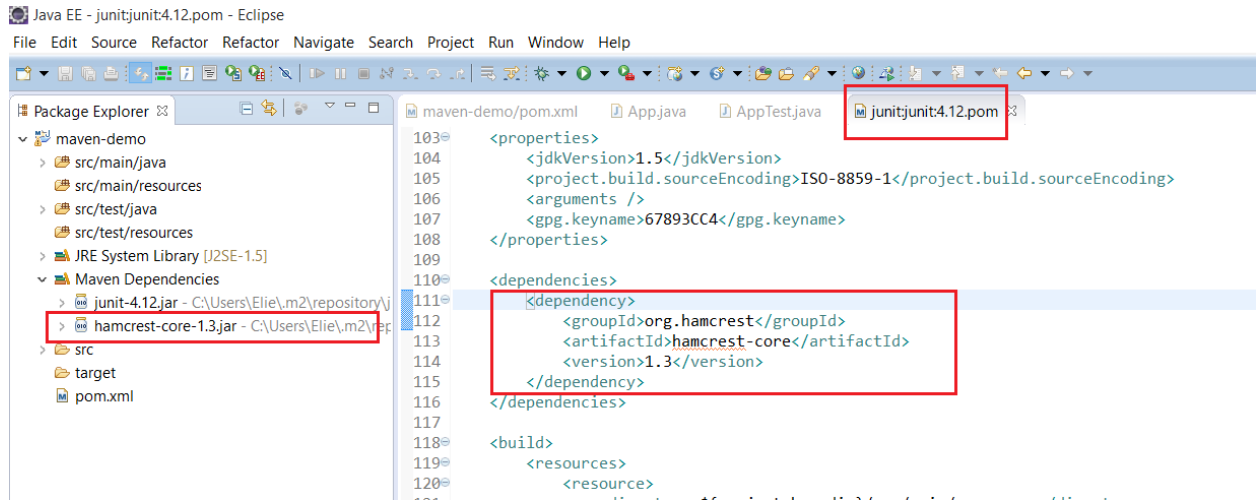
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
  </dependency>
</dependencies>

```

## 5.1 DEPENDANCE TRANSITIVE

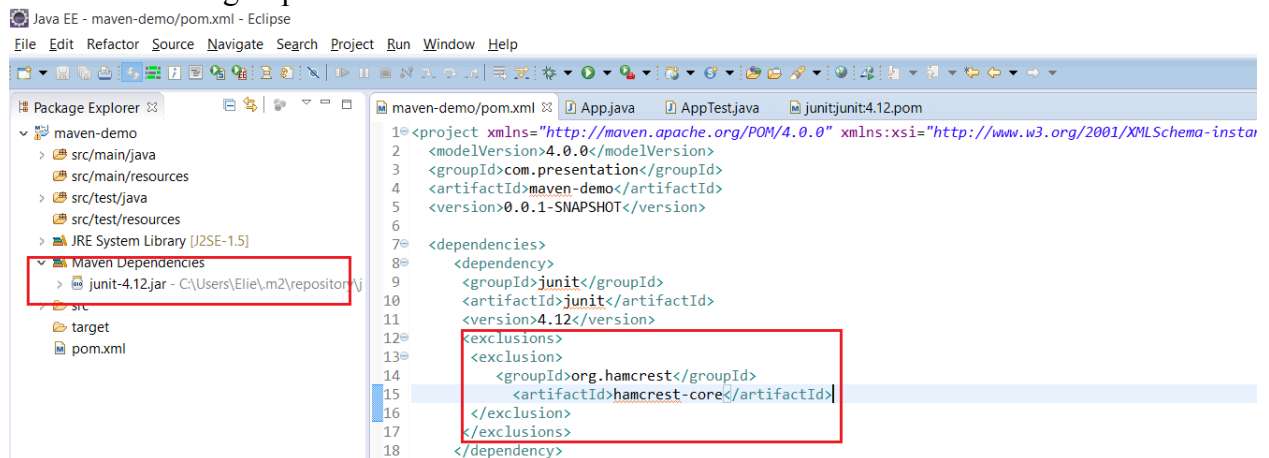
Les dépendances dans maven sont transitives :

On remarque dans le pom.xml de la junit une dépendance vers hamcrest-core ce qui explique la présence du hamcrest-core.jar dans le 'Maven Dependencies'.



## 5.2 EXCLUSION DE DEPANDANCES

On peut aussi exclure des dépendances en utilisant les tags <exclusions><exclusion> Ou on ajoute l'artefact id et le group Id de celle à exclure :



## 6. COMMANDES MAVEN

Pour créer un jar de notre projet :

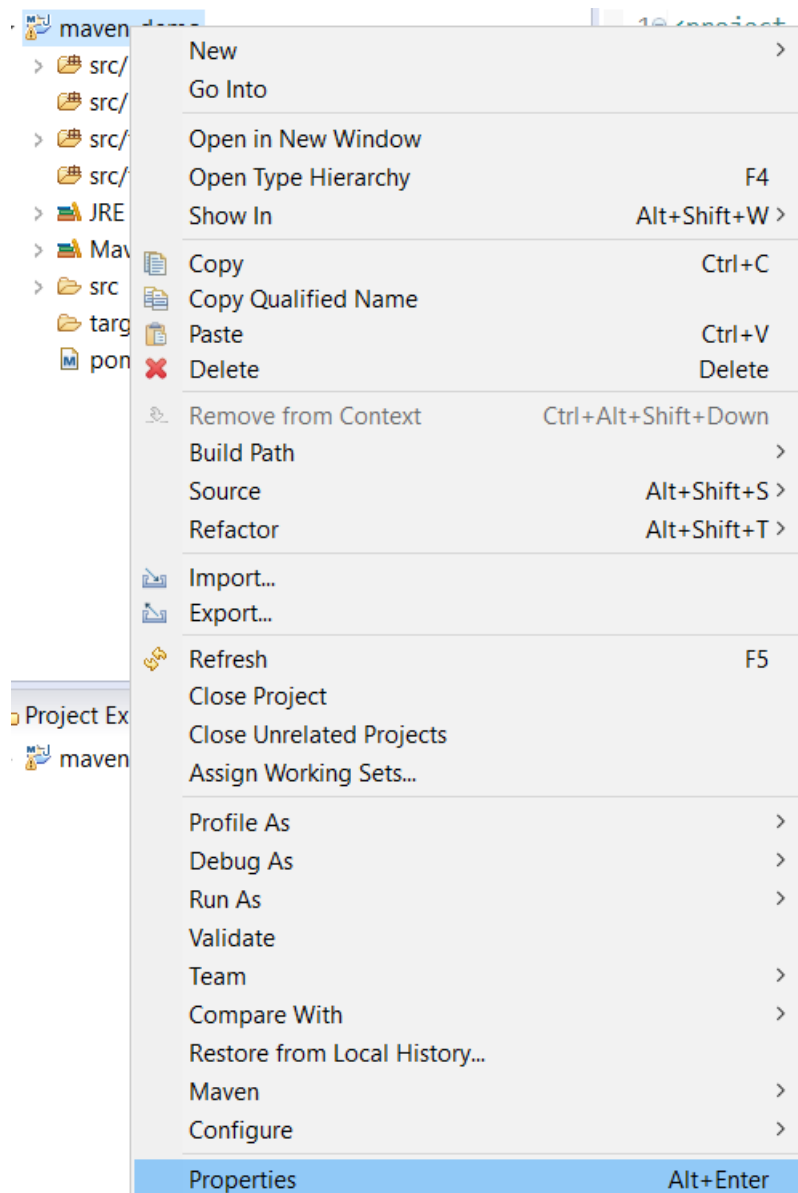


## Maven-demo.jar

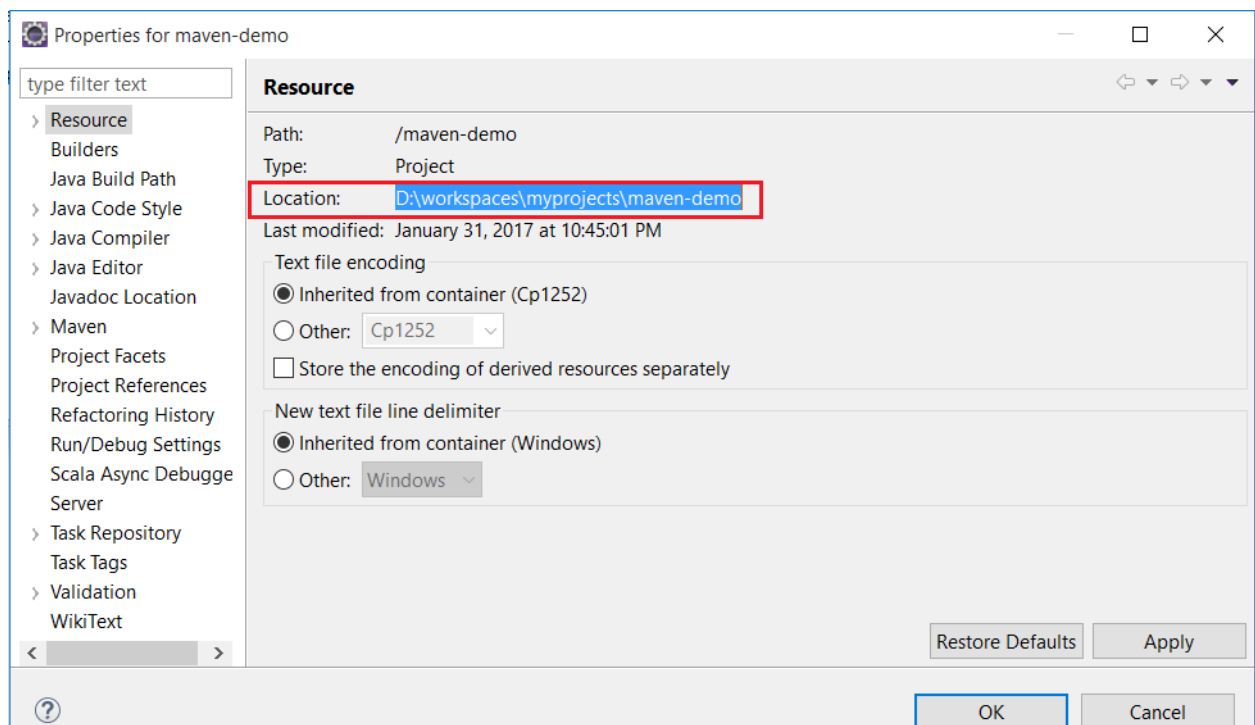
- App.java → App.class
- AppTest.java → AppTest.class
- Run tests
- Create jar

1. Compiler mes fichiers .java en .class
2. Exécuter les tests
3. Créer le jar

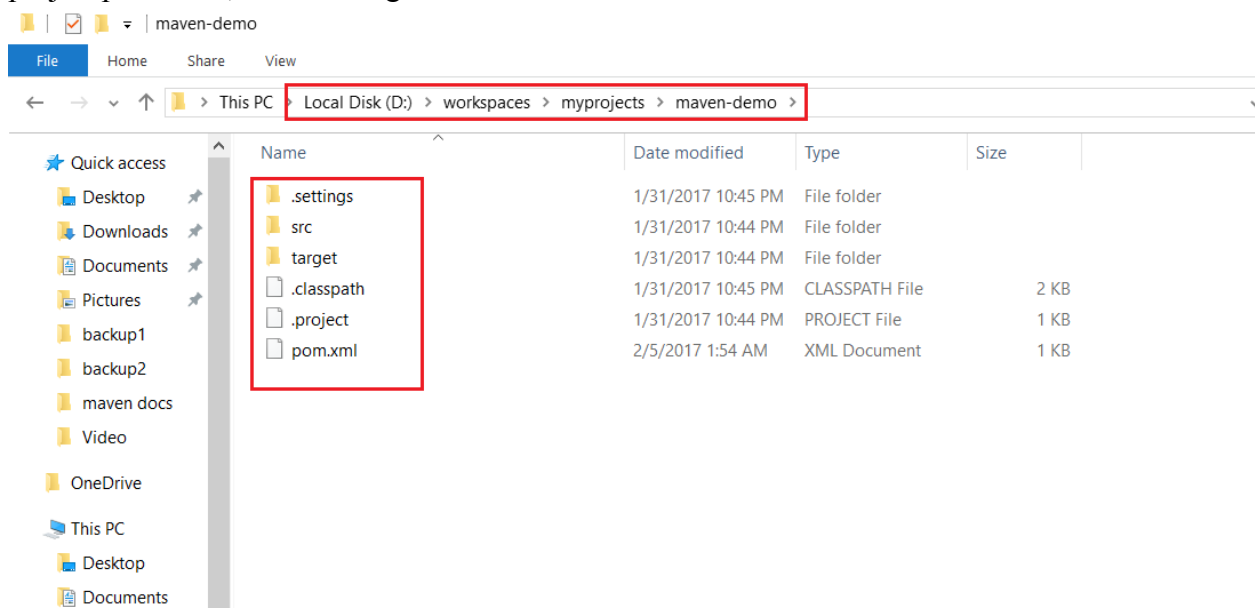
Pour connaître le chemin du répertoire de notre projet :  
Clic droit sur notre projet Maven → properties



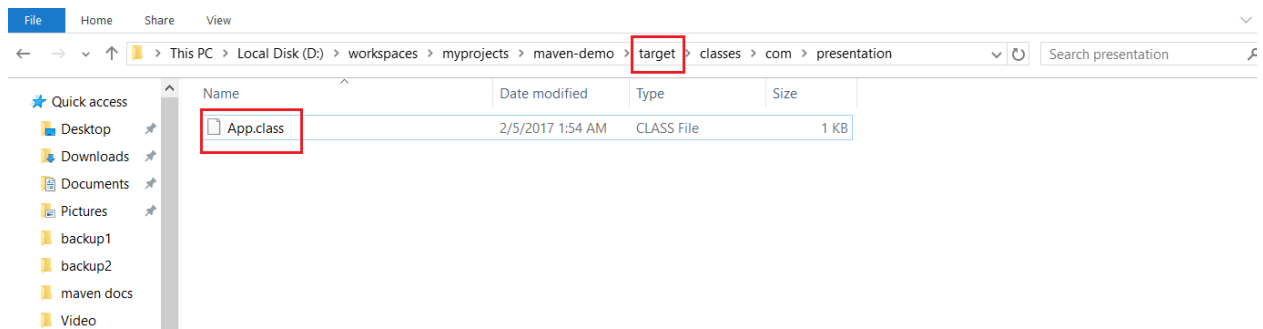
Copier l'URL du projet Maven



Coller le path ; on remarque plusieurs fichiers et répertoires dans la répertoire maven-demo de notre projet 'pom.xml', 'src' et 'target'.



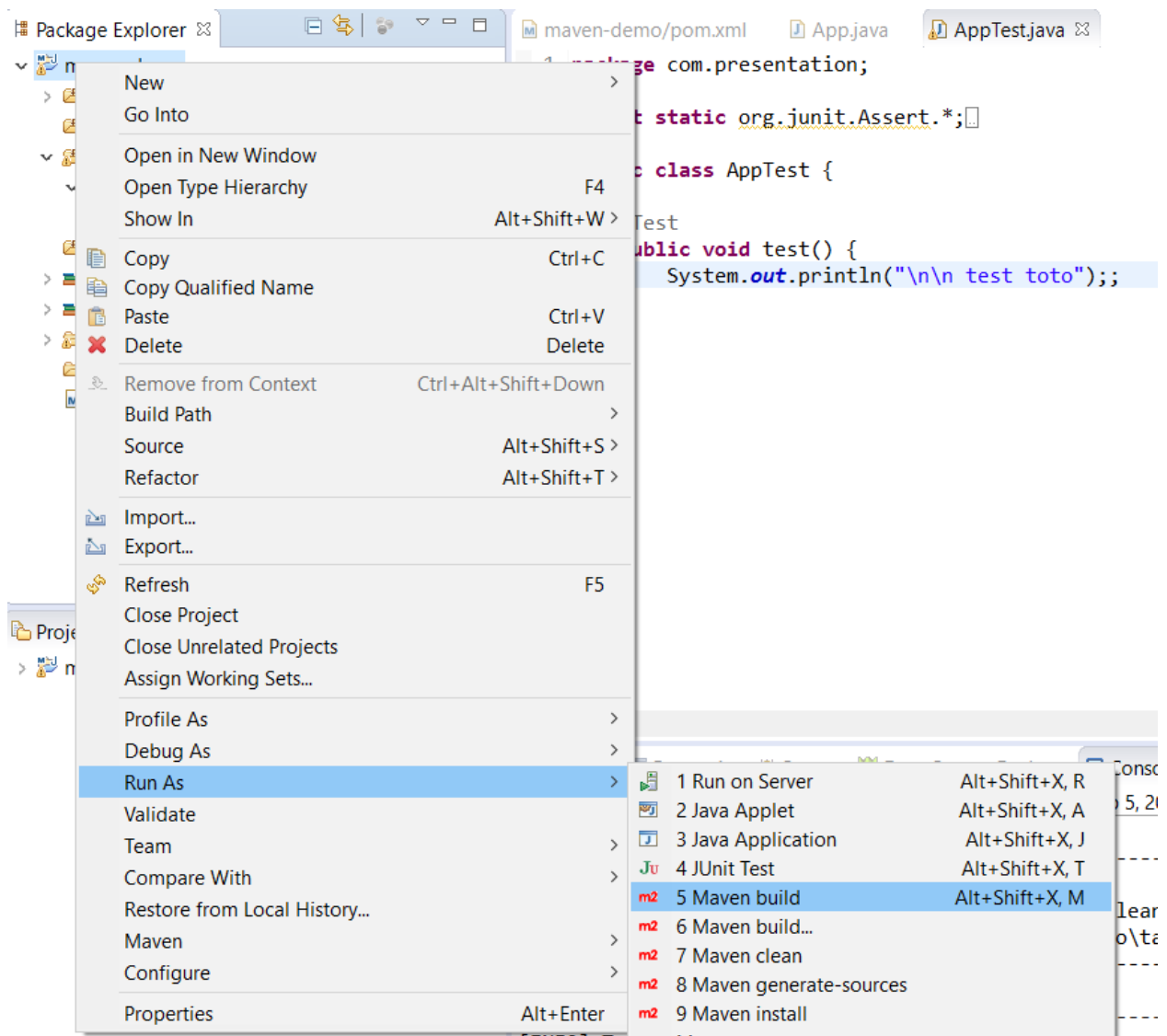
Le répertoire 'target' contient les classes compilées.



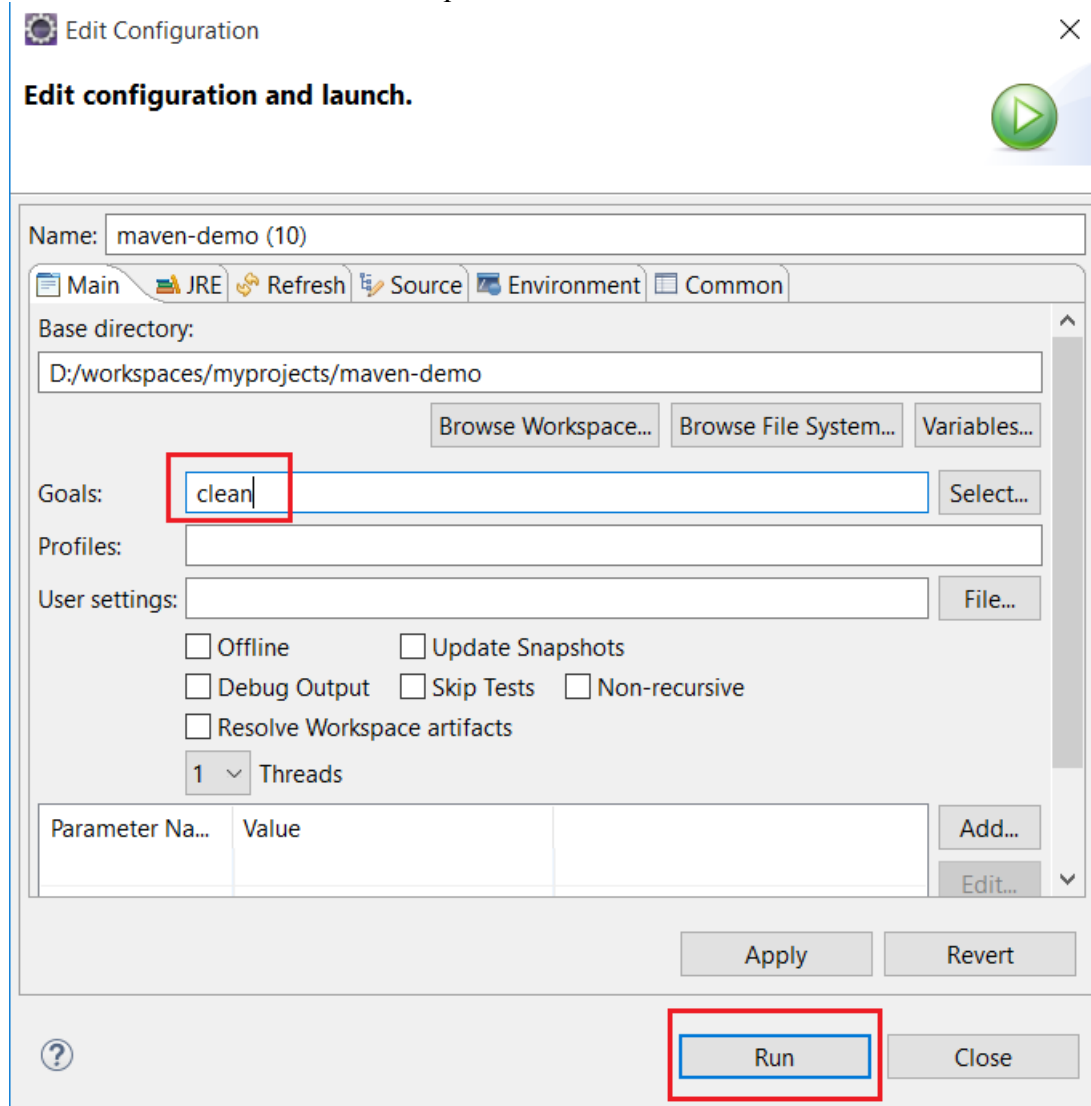
## 6.0 COMMANDE CLEAN

Commande clean :

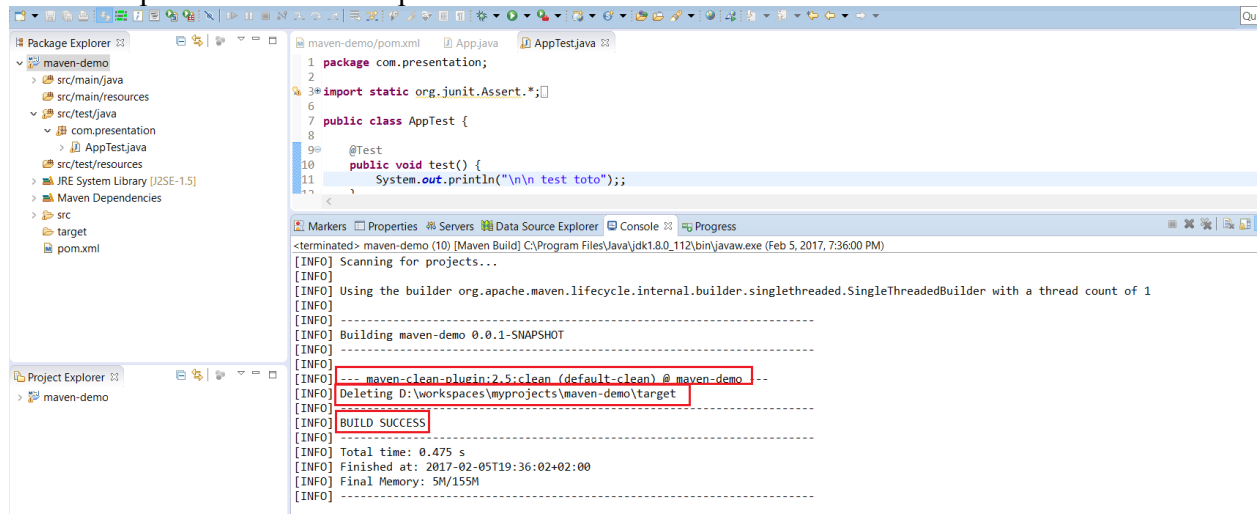
Clic droit sur notre projet maven → Run As → Maven build



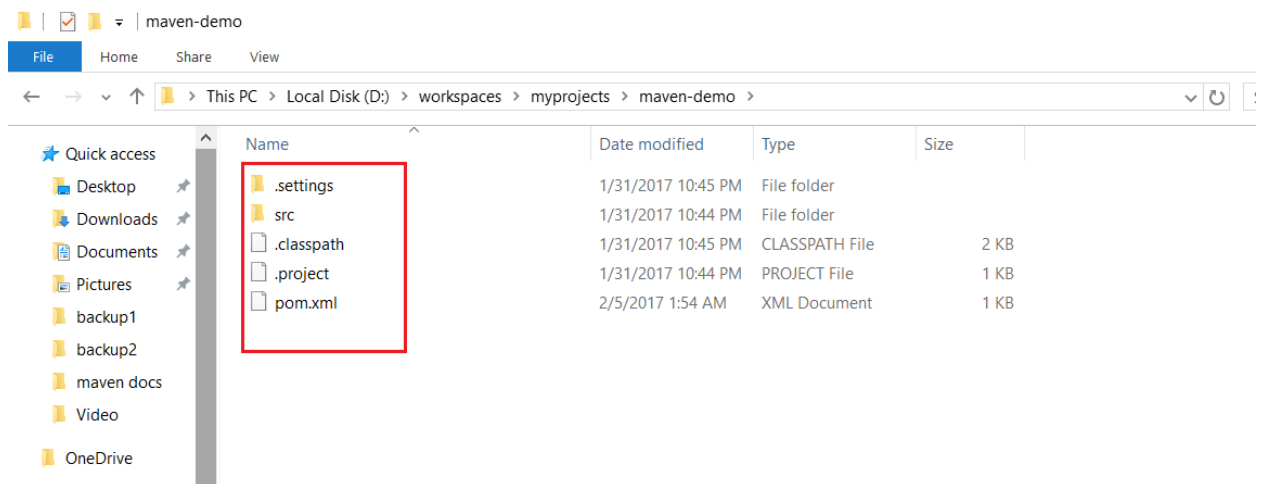
Ecrire 'clean' dans 'Goals' → cliquer sur 'Run'



On remarque dans la console que c'est un succès :

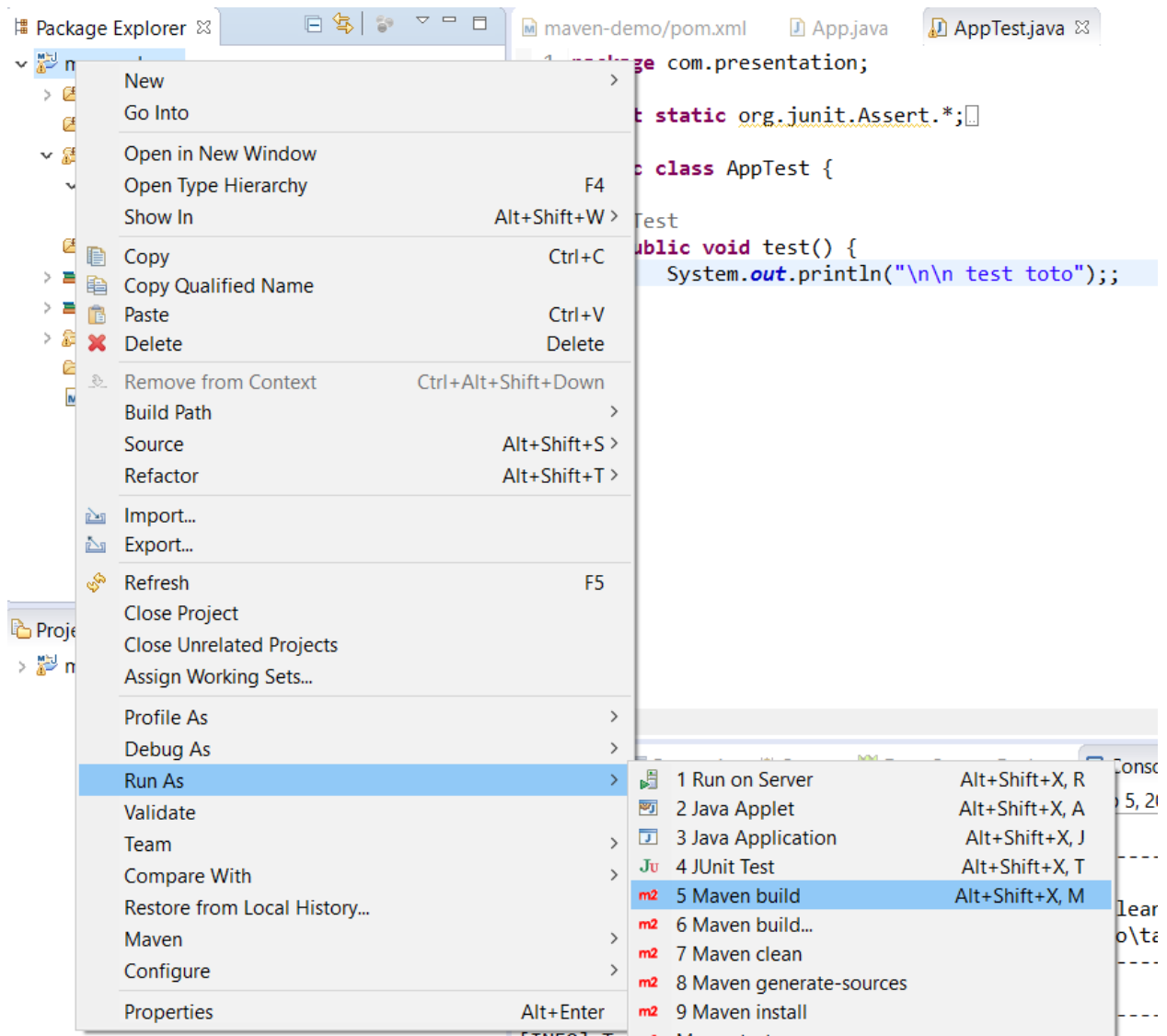


On remarque que le répertoire 'target' a disparu.



## 6.1 COMMANDE COMPILE

Commande compile (compilation des fichiers sources) :



Ecrire 'compile' dans 'Goals' → cliquer sur 'Run'

**Edit Configuration**

**Edit configuration and launch.**

Name:

Base directory:

Goals:

Profiles:

User settings:

☐ Offline ☐ Update Snapshots

☐ Debug Output ☐ Skip Tests ☐ Non-recursive

☐ Resolve Workspace artifacts

Threads

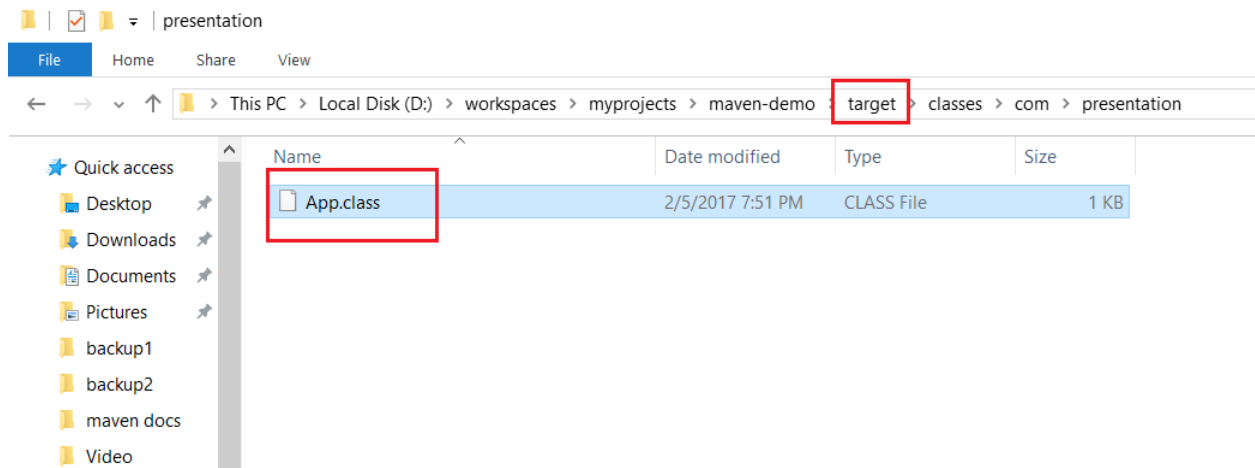
Parameter Na...	Value

Markers Properties Servers Data Source Explorer Console Progress

```
<terminated> maven-demo (11) [Maven Build] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (Feb 5, 2017, 7:51:36 PM)
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread count of 1
[INFO]
[INFO] -----
[INFO] Building maven-demo 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven-demo ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ maven-demo ---
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\workspaces\myprojects\maven-demo\target\classes
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 2.891 s
[INFO] Finished at: 2017-02-05T19:51:40+02:00
[INFO] Final Memory: 10M/123M
[INFO] -----
```

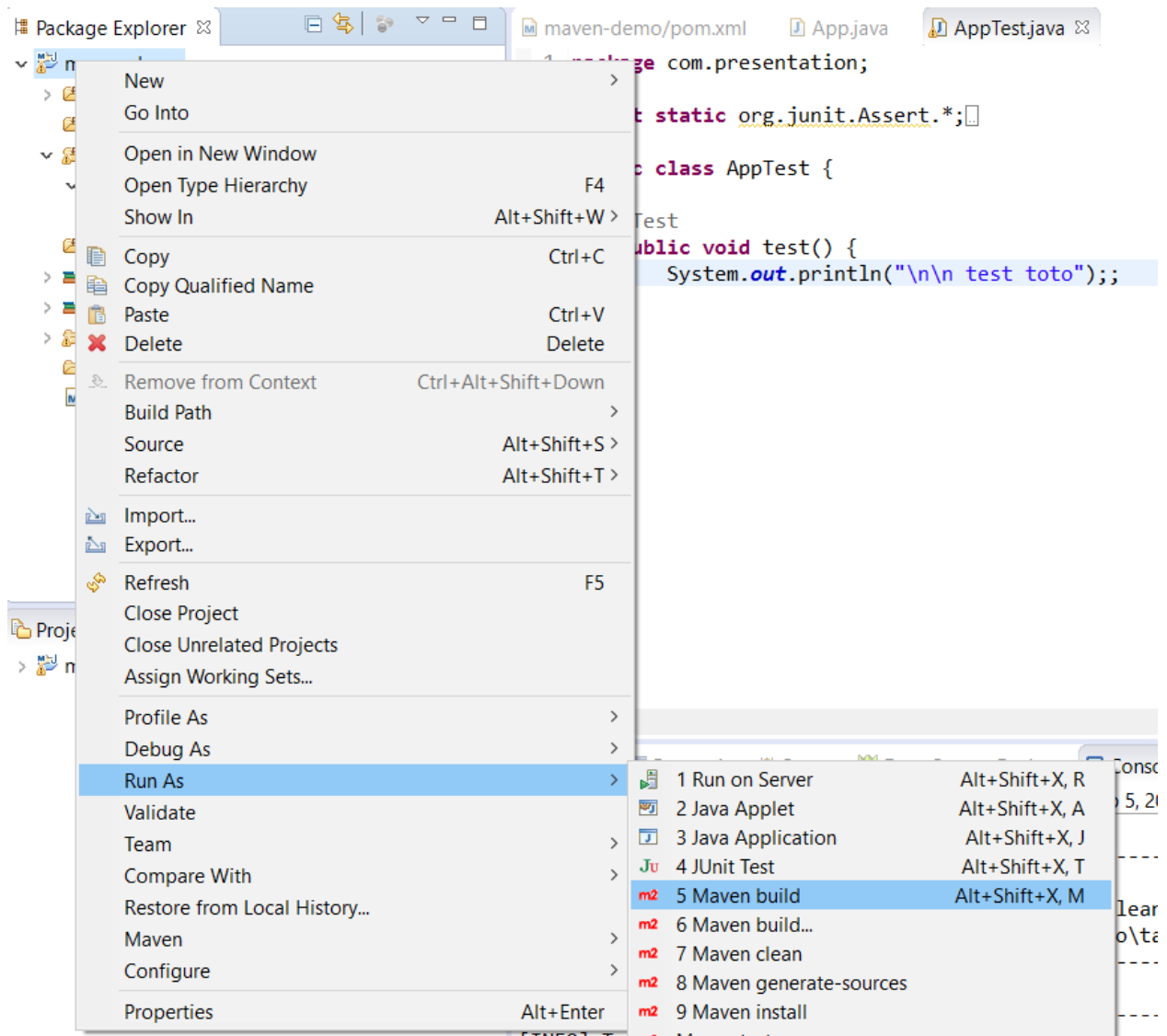
On remarque l'apparition du repertoire 'target' ; La classe App.java est compilé en 'App.class'.



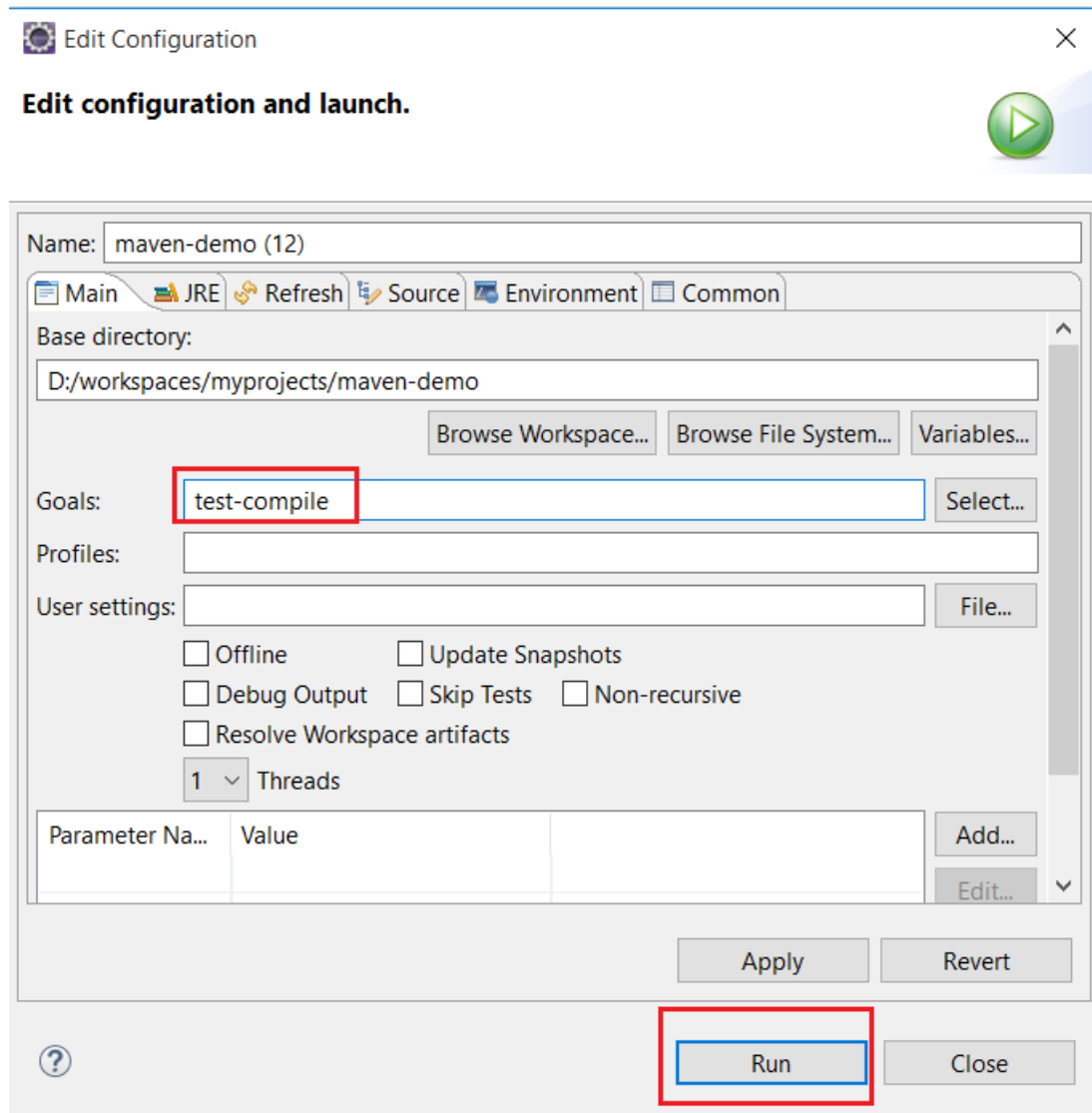


## 6.2 COMMANDE TEST-COMPILE

Commande test-compile (compilation des fichiers tests) :

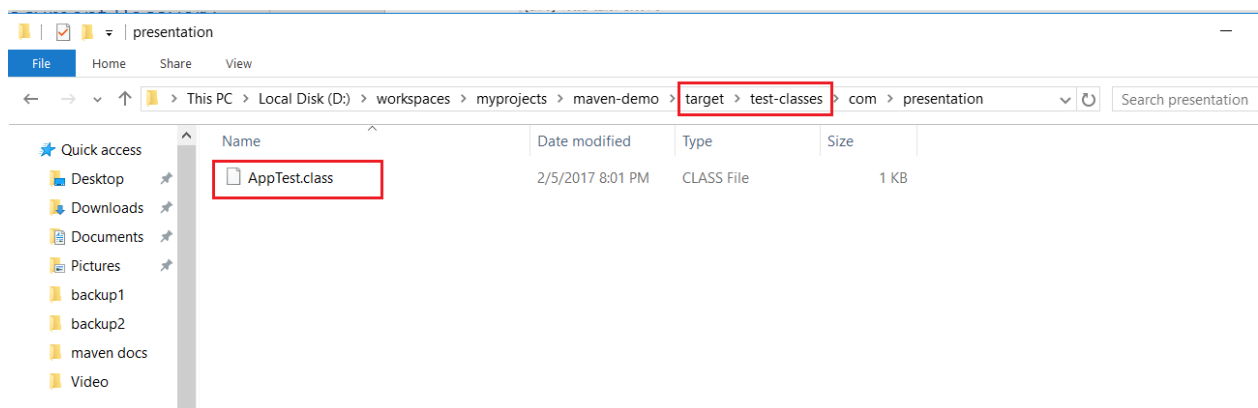


Ecrire 'test-compile' dans 'Goals' et cliquer sur 'Run' :

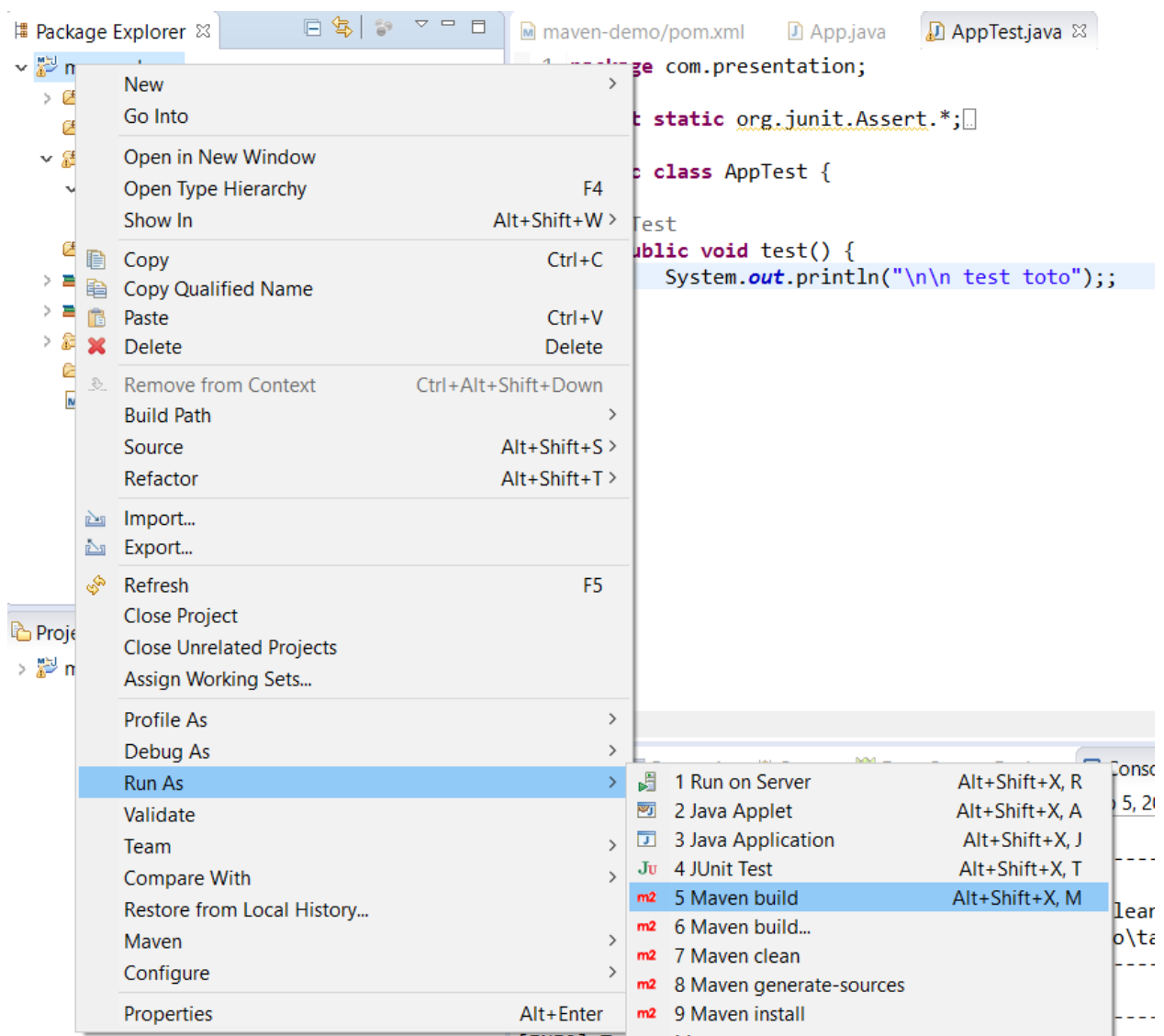


```
Markers Properties Servers Data Source Explorer Console Progress
<terminated> maven-demo (12) [Maven Build] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (Feb 5, 2017, 8:01:50 PM)
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread count of 1
[INFO]
[INFO] -----
[INFO] Building maven-demo 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven-demo ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ maven-demo ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ maven-demo ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @ maven-demo ---
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\workspaces\myprojects\maven-demo\target\test-classes
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO]
[INFO] Total time: 1.664 s
[INFO] Finished at: 2017-02-05T20:01:53+02:00
[INFO] Final Memory: 11M/155M
[INFO] -----
```

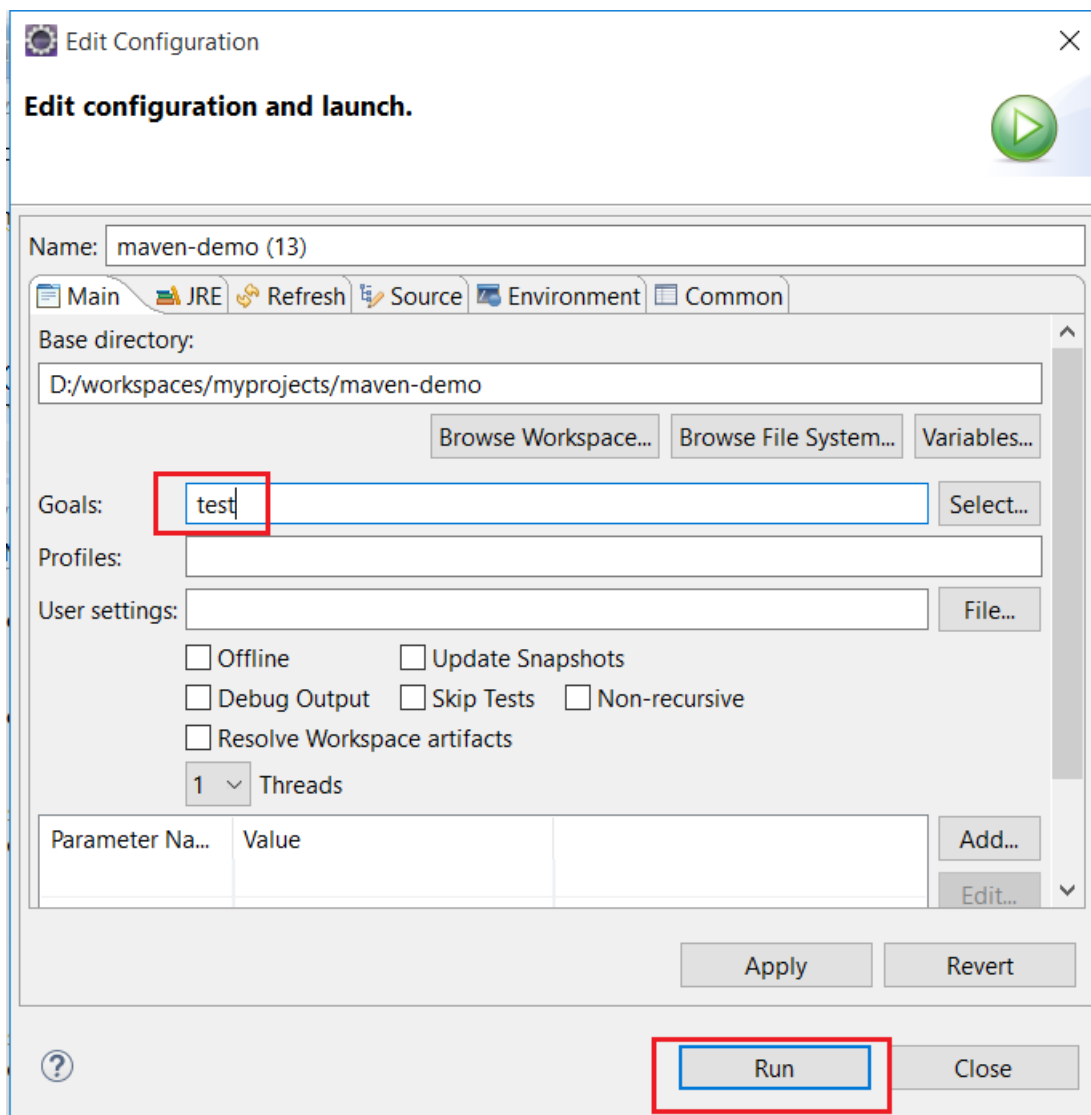
On remarque que la classe AppTest.java est compilé en 'AppTest.class'.



### 6.3 COMMANDE TEST



On tape 'test' dans 'Goals' et on clique 'Run' :



Dans notre exemple on remarque que notre test a échoué

```
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-demo ---
[INFO] Surefire report directory: D:\workspaces\myprojects\maven-demo\target\surefire-reports

-----
T E S T S
-----
Running com.presentation.AppTest
Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.062 sec <<< FAILURE!
initializationError(com.presentation.AppTest) Time elapsed: 0.014 sec <<< ERROR!
java.lang.NoClassDefFoundError: org/hamcrest/SelfDescribing
    at java.lang.ClassLoader.defineClass1(Native Method)
    at java.lang.ClassLoader.defineClass(ClassLoader.java:763)
    at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
    at java.net.URLClassLoader.defineClass(URLClassLoader.java:467)
    at java.net.URLClassLoader.access$100(URLClassLoader.java:73)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:368)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:362)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:361)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    at org.junit.internal.builders.JUnit4Builder.runnerForClass(JUnit4Builder.java:10)
    at org.junit.runners.model.RunnerBuilder.safeRunnerForClass(RunnerBuilder.java:59)
    at org.junit.internal.builders.AllDefaultPossibilitiesBuilder.runnerForClass(AllDefaultPossibilitiesBuilder.java:26)
    at org.junit.runners.model.RunnerBuilder.safeRunnerForClass(RunnerBuilder.java:59)
    at org.junit.internal.requests.ClassRequest.getRunner(ClassRequest.java:33)
    at org.apache.maven.surefire.junit4.JUnit4Provider.execute(JUnit4Provider.java:250)
    at org.apache.maven.surefire.junit4.JUnit4Provider.executeTestSet(JUnit4Provider.java:141)
    at org.apache.maven.surefire.junit4.JUnit4Provider.invoke(JUnit4Provider.java:112)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.apache.maven.surefire.util.ReflectionUtils.invokeMethodWithArray(ReflectionUtils.java:189)
    at org.apache.maven.surefire.booter.ProviderFactory$ProviderProxy.invoke(ProviderFactory.java:165)
    at org.apache.maven.surefire.booter.ProviderFactory.invokeProvider(ProviderFactory.java:85)
    at org.apache.maven.surefire.booter.ForkedBooter.runSuitesInProcess(ForkedBooter.java:115)
    at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:75)
Caused by: java.lang.ClassNotFoundException: org.hamcrest.SelfDescribing
<
Caused by: java.lang.ClassNotFoundException: org.hamcrest.SelfDescribing
    at java.net.URLClassLoader.findClass(URLClassLoader.java:381)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:331)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    ... 29 more

Results :

Tests in error:
  initializationError(com.presentation.AppTest): org/hamcrest/SelfDescribing

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

[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 3.976 s
[INFO] Finished at: 2017-02-05T20:16:37+02:00
[INFO] Final Memory: 7M/123M
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on project maven-demo: There are test failures.
[ERROR]
```

L'erreur est due à l'exclusion de la dépendance 'hmcrest-core'. On enlève l'exclusion et on exécute la commande test de nouveau.

```

1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://maven.apache.org/xsi:schemaLocation"
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>com.presentation</groupId>
4   <artifactId>maven-demo</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6
7   <dependencies>
8     <dependency>
9       <groupId>junit</groupId>
10      <artifactId>junit</artifactId>
11      <version>4.12</version>
12    </dependency>
13  </dependencies>
14
15
16 </project>

```

## Le test est un succès

```

<terminated> maven-demo (8) [Maven Build] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (Feb 5, 2017, 8:24:38 PM)
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ maven-demo ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ maven-demo ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @ maven-demo ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-demo ---
[INFO] Surefire report directory: D:\workspaces\myprojects\maven-demo\target\surefire-reports

-----
T E S T S
-----
Running com.presentation.AppTest

test toto
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.055 sec

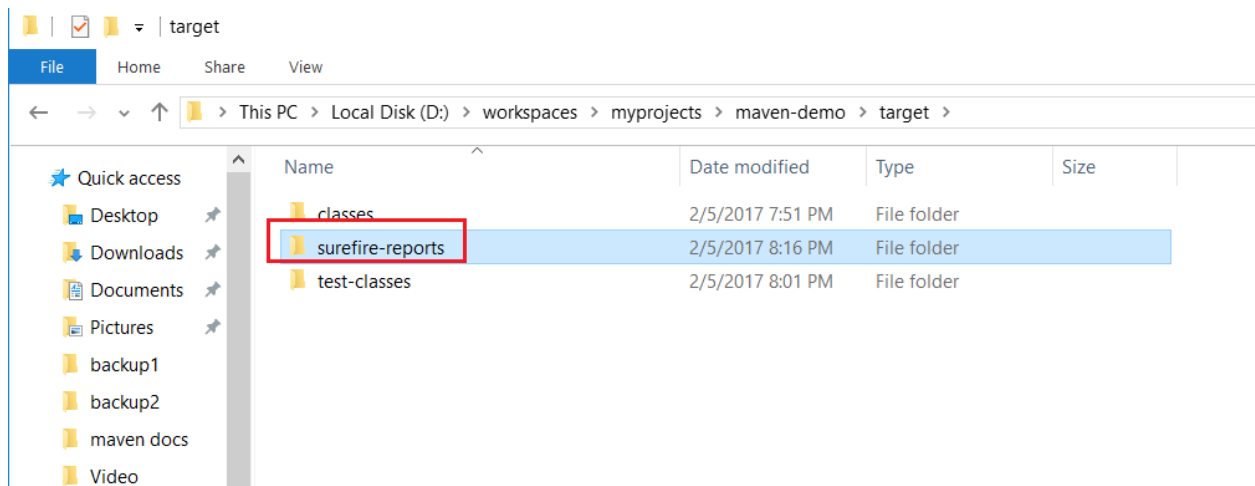
Results :

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

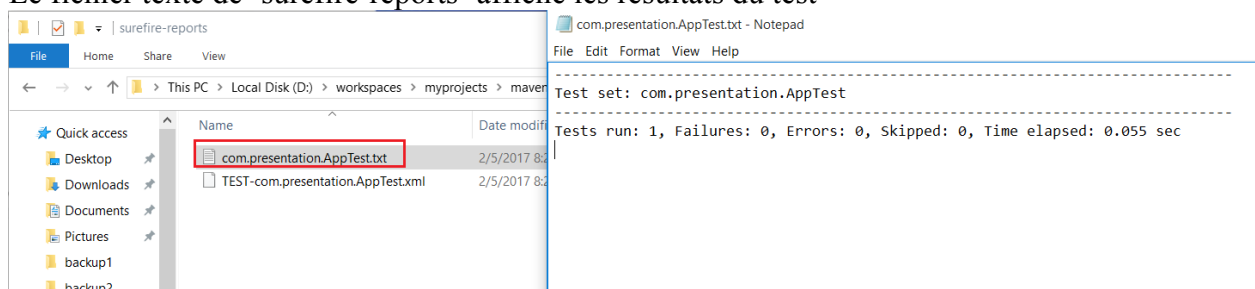
[INFO] BUILD SUCCESS
[INFO] Total time: 1.884 s
[INFO] Finished at: 2017-02-05T20:24:41+02:00
[INFO] Final Memory: 8M/155M
[INFO]

```

On remarque l'apparition d'un nouveau répertoire 'surefire-reports' dans le répertoire 'target'



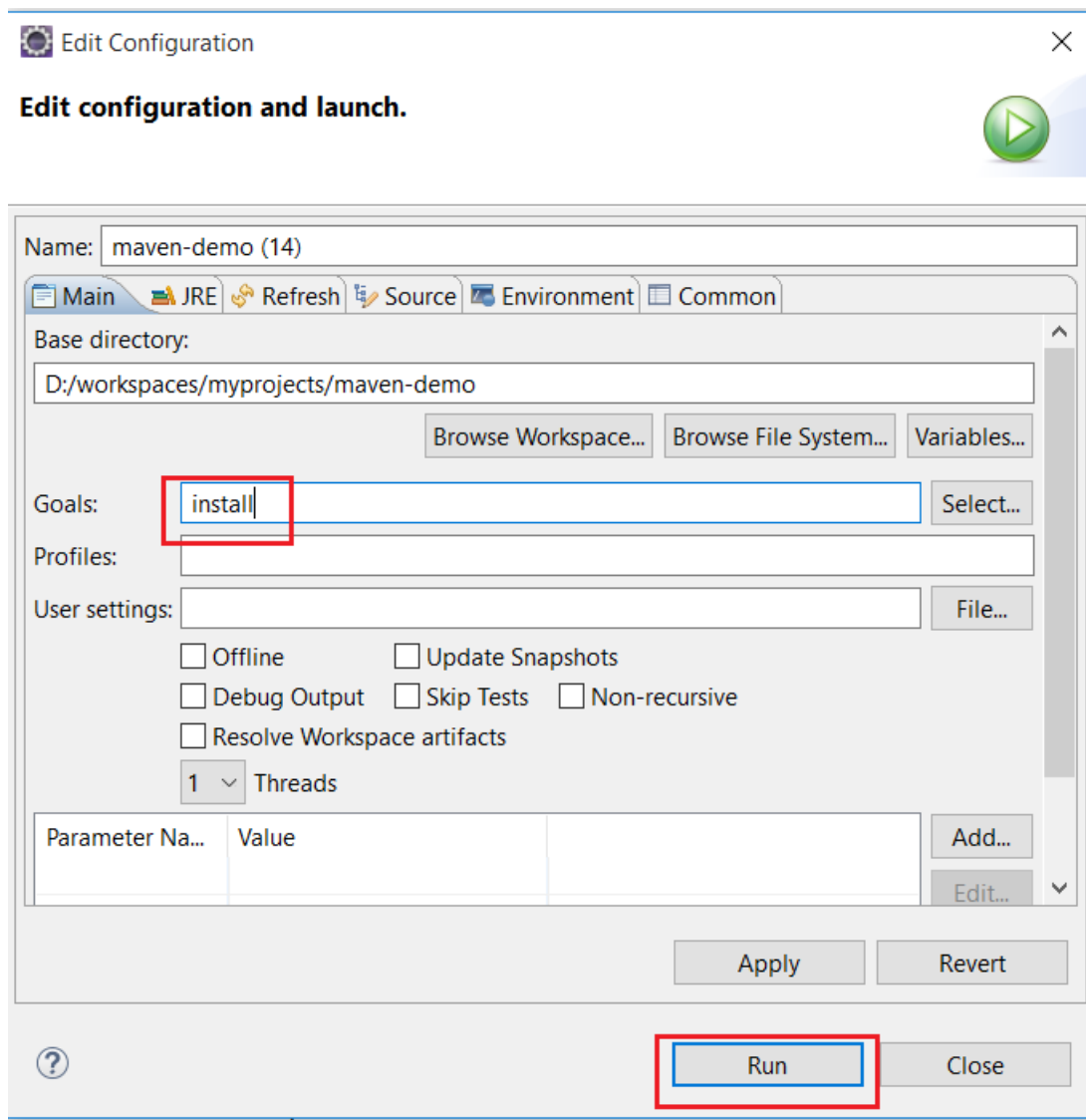
Le fichier texte de 'surefire-reports' affiche les résultats du test



## 6.4 COMMANDE INSTALL

On tape 'install' dans 'Goals' et on clique sur 'Run'





```

Markers Properties Servers Data Source Explorer Console Progress
<terminated> maven-demo (14) [Maven Build] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (Feb 5, 2017, 9:03:19 PM)
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-demo ---
[INFO] Surefire report directory: D:\workspaces\myprojects\maven-demo\target\surefire-reports

-----
T E S T S
-----
Running com.presentation.AppTest

test toto
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.081 sec

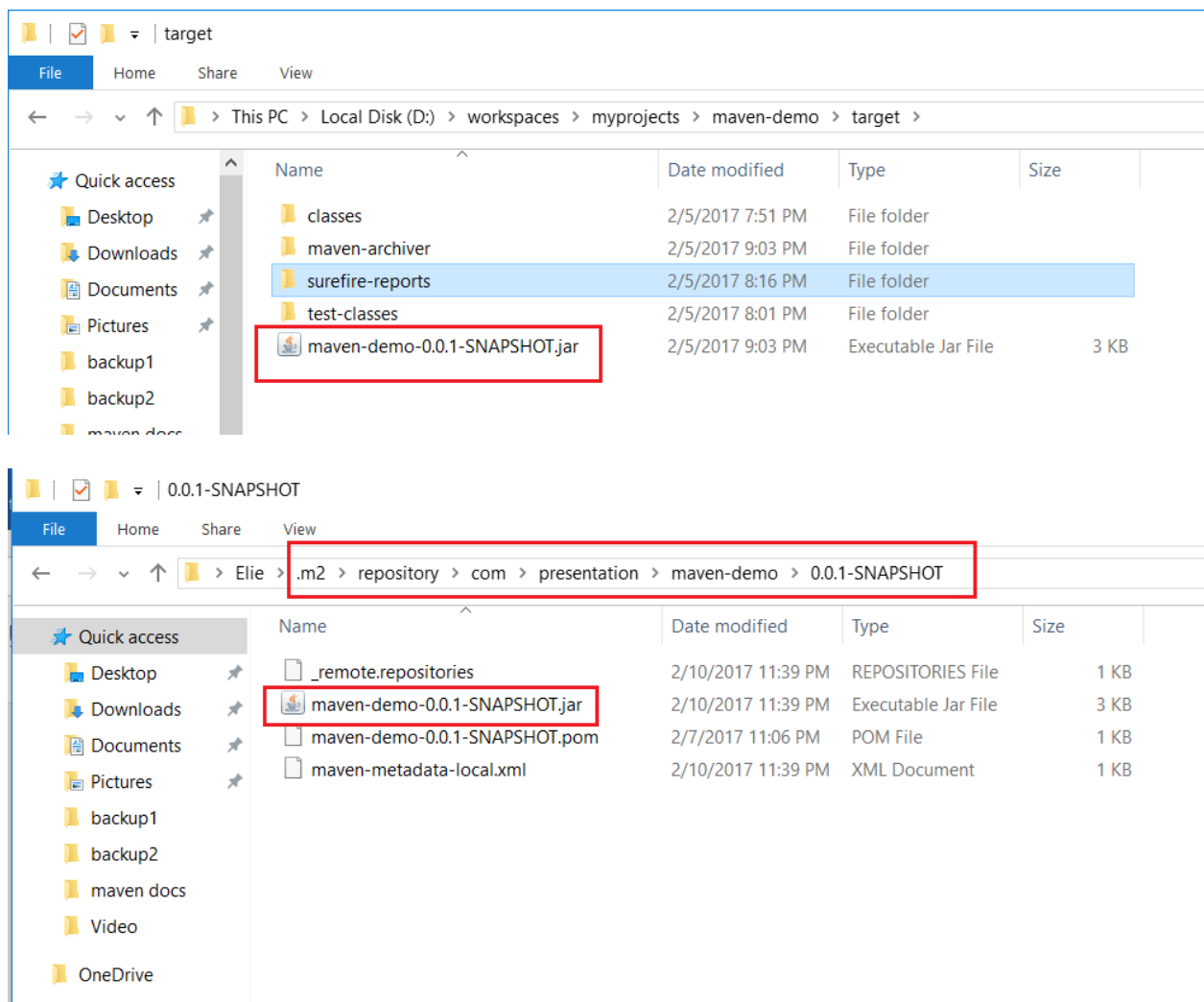
Results :

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

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ maven-demo ---
[INFO] Building jar: D:\workspaces\myprojects\maven-demo\target\maven-demo-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ maven-demo ---
[INFO] Installing D:\workspaces\myprojects\maven-demo\target\maven-demo-0.0.1-SNAPSHOT.jar to C:\Users\Elie\.m2\repository\com\presentation\maven-demo\0.0.1-SNAPSHOT\
[INFO] Installing D:\workspaces\myprojects\maven-demo\pom.xml to C:\Users\Elie\.m2\repository\com\presentation\maven-demo\0.0.1-SNAPSHOT\maven-demo-0.0.1-SNAPSHOT.pom
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO]
[INFO] Total time: 3.559 s
[INFO] Finished at: 2017-02-05T21:03:24+02:00
[INFO] Final Memory: 9M/155M
[INFO]

```

Le jar est créé dans 'target' et dans notre dépôt local



## 7. PROJET MULTI-MODULES

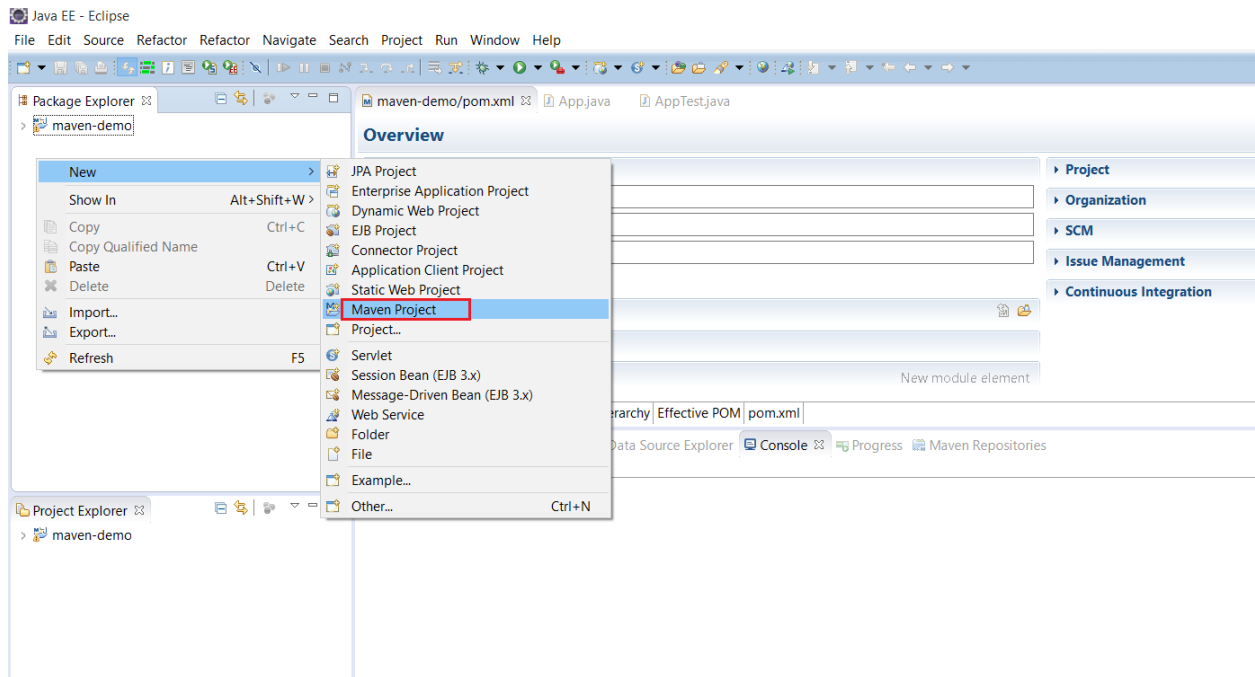
Exemple : web server et son client ; le client dépend du serveur et il faut qu'il y a une sorte de synchronisation entre les deux. Maven va faire ce travail.

On va créer un projet parent ayant 2 modules :

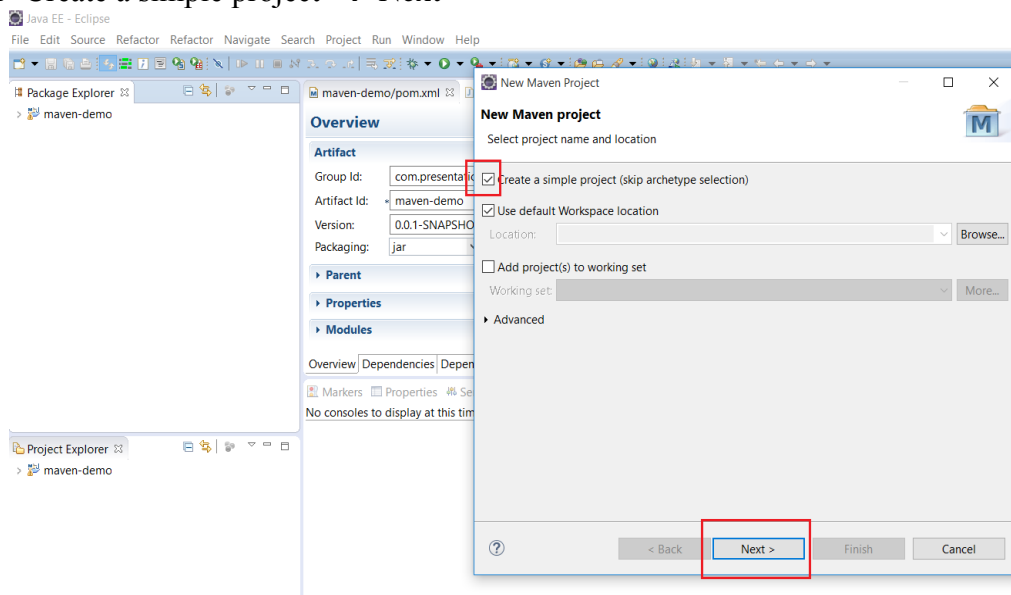
- a. Web server
- b. Client

### 6.0 CREATION D'UN PROJET PARENT

Clique droit dans package explorer → New → Maven Project



Cocher 'Create a simple project' → Next



Saisir 'multi.module' comme group Id, 'parent-project' comme artifact Id et choisir 'pom' pour packaging.

**New Maven Project**

Configure project

**Artifact**

Group Id: multi.module

Artifact Id: parent-project

Version: 0.0.1-SNAPSHOT

Packaging: pom

Name:

Description:

**Parent Project**

Group Id:

Artifact Id:

Version:

Browse... Clear

Advanced

< Back Next > **Finish** Cancel

Configuration de la java compiler pour que tous les modules puissent l'utiliser cela :

Java EE - parent-project/pom.xml - Eclipse

File Edit Refactor Navigate Search Project Run Window Help

Package Explorer

- maven-demo
  - parent-project
    - src
      - pom.xml

Project Explorer

- parent-project
  - maven-demo

```
1<?xml version="1.0" encoding="UTF-8"?>
2<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">
3  <groupId>multi.module</groupId>
4  <artifactId>parent-project</artifactId>
5  <version>0.0.1-SNAPSHOT</version>
6  <packaging>pom</packaging>
7
8
9  <build>
10    <pluginManagement>
11      <plugins>
12        <plugin>
13          <groupId>org.apache.maven.plugins</groupId>
14          <artifactId>maven-compiler-plugin</artifactId>
15          <version>3.1</version>
16          <configuration>
17            <source>1.7</source>
18            <target>1.7</target>
19          </configuration>
20        </plugin>
21      </plugins>
22    </pluginManagement>
23  </build>
24</project>
```

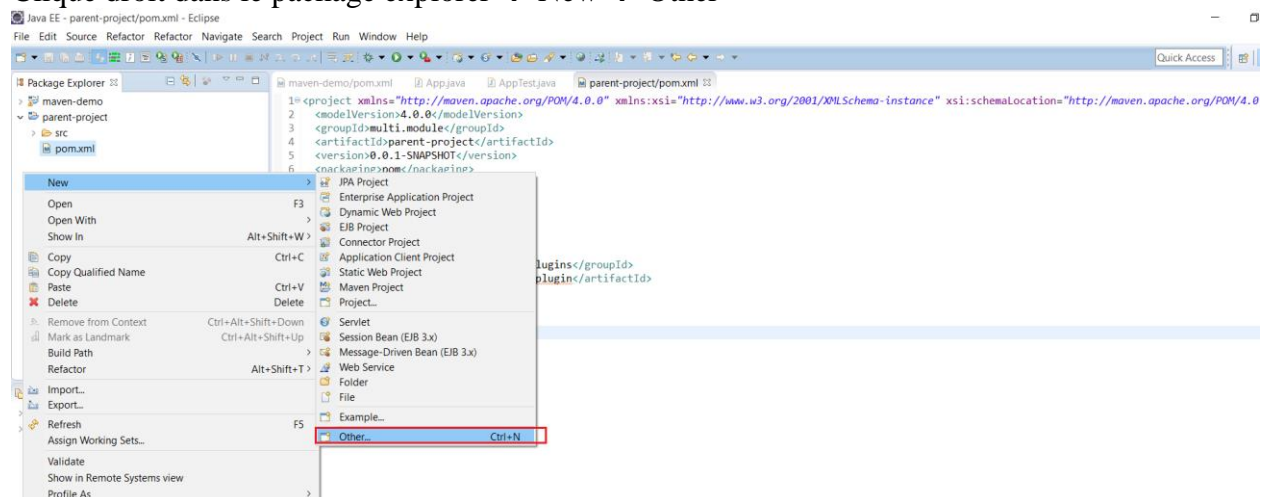
```

<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>

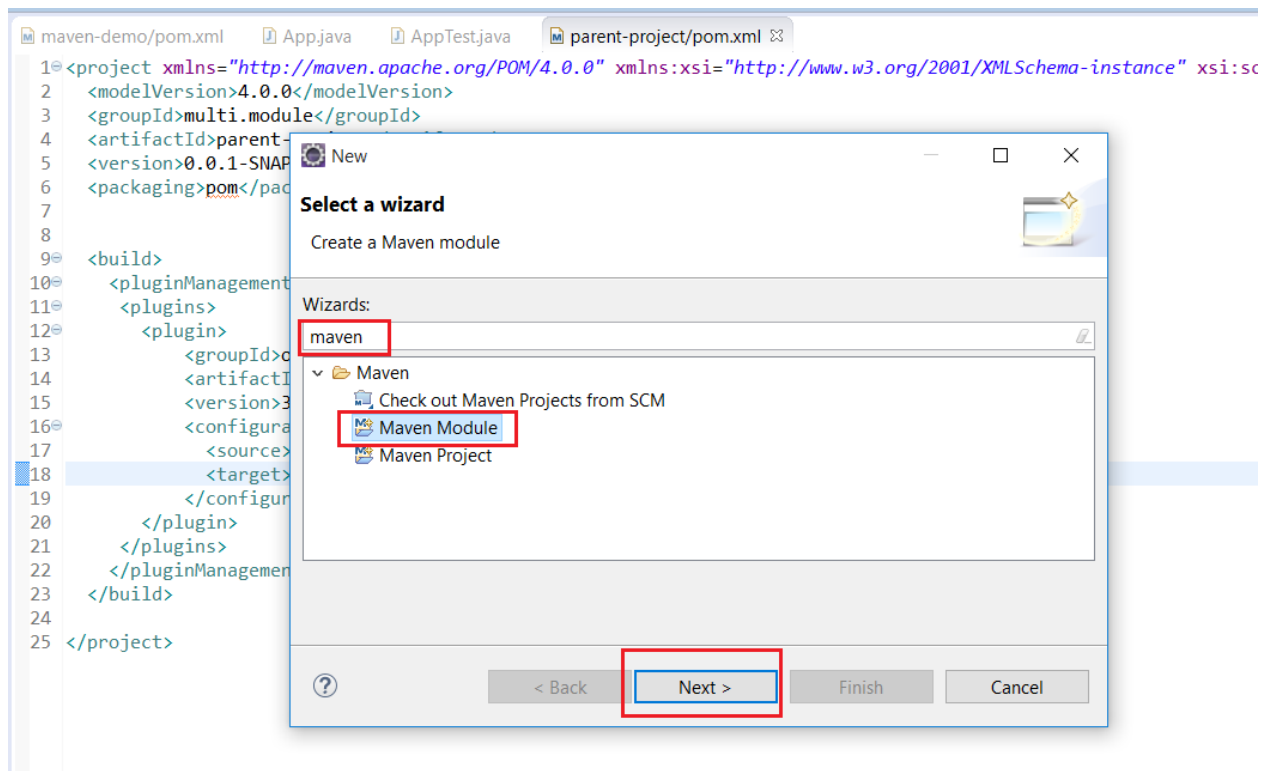
```

## 6.1 CREATION DU MODULE SERVEUR

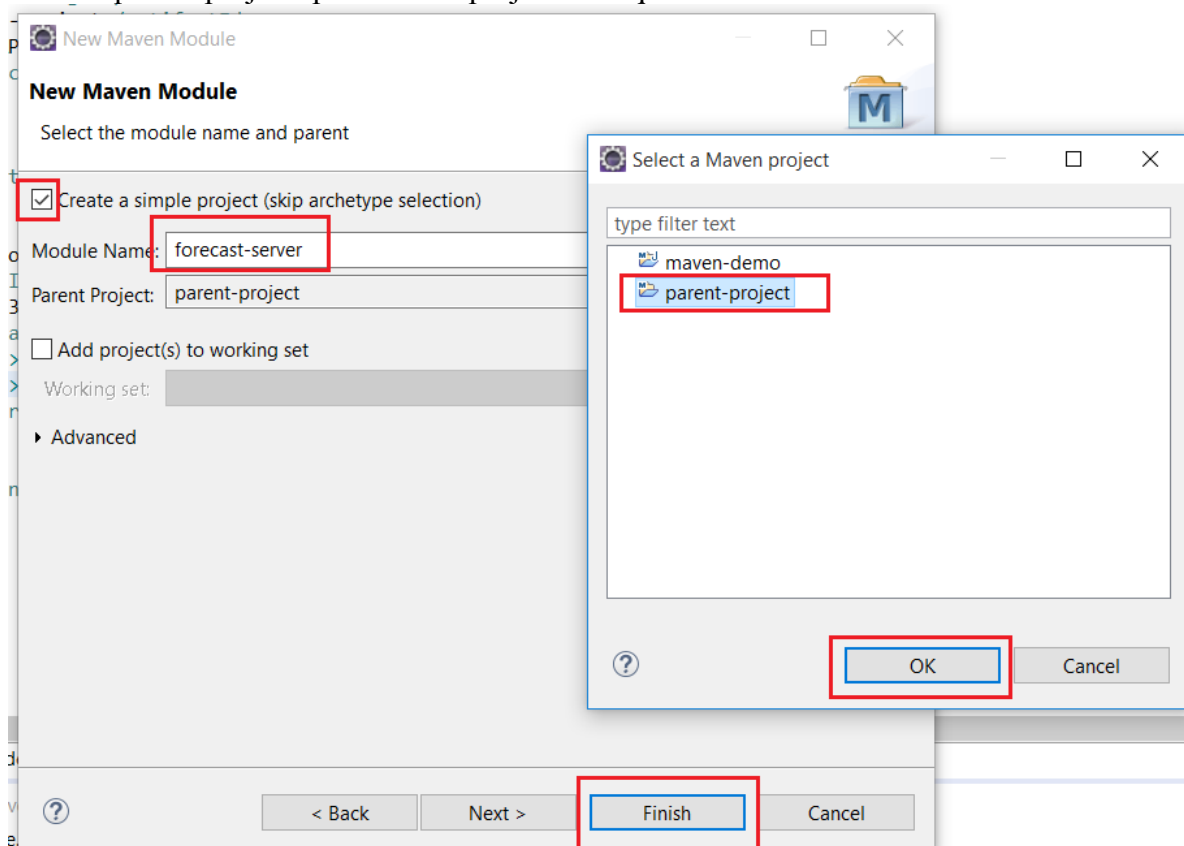
Clique droit dans le package explorer → New → Other



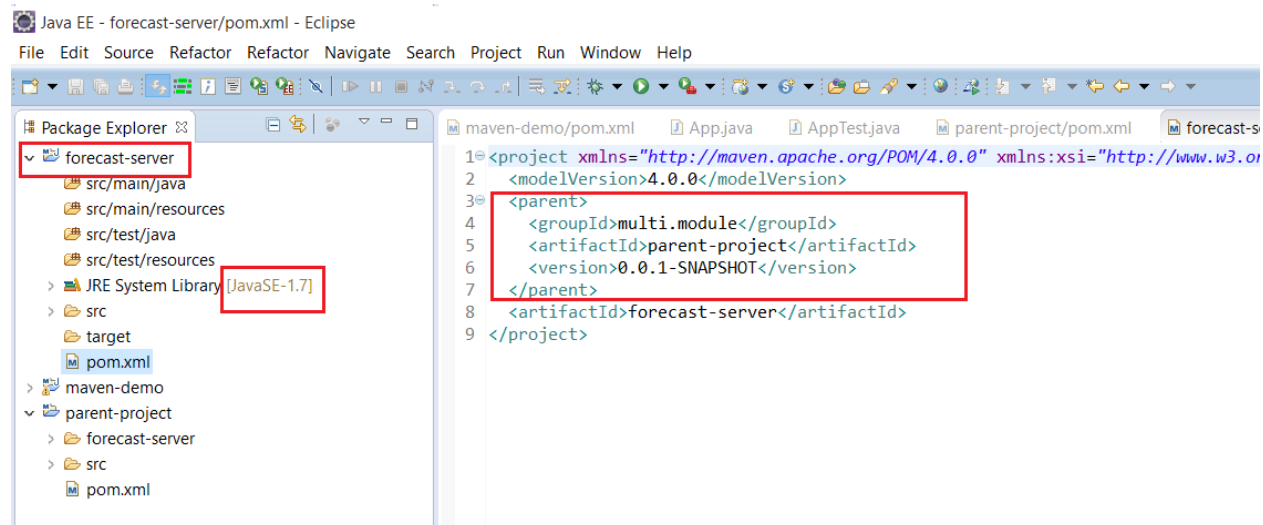
Saisir 'maven' → choisir 'Maven Modules' → Next



Cocher 'Create simple project' → saisir le module name 'forecast-server' → cliquer sur browse → choisir 'parent-project' pour Parent project → cliquer Finish



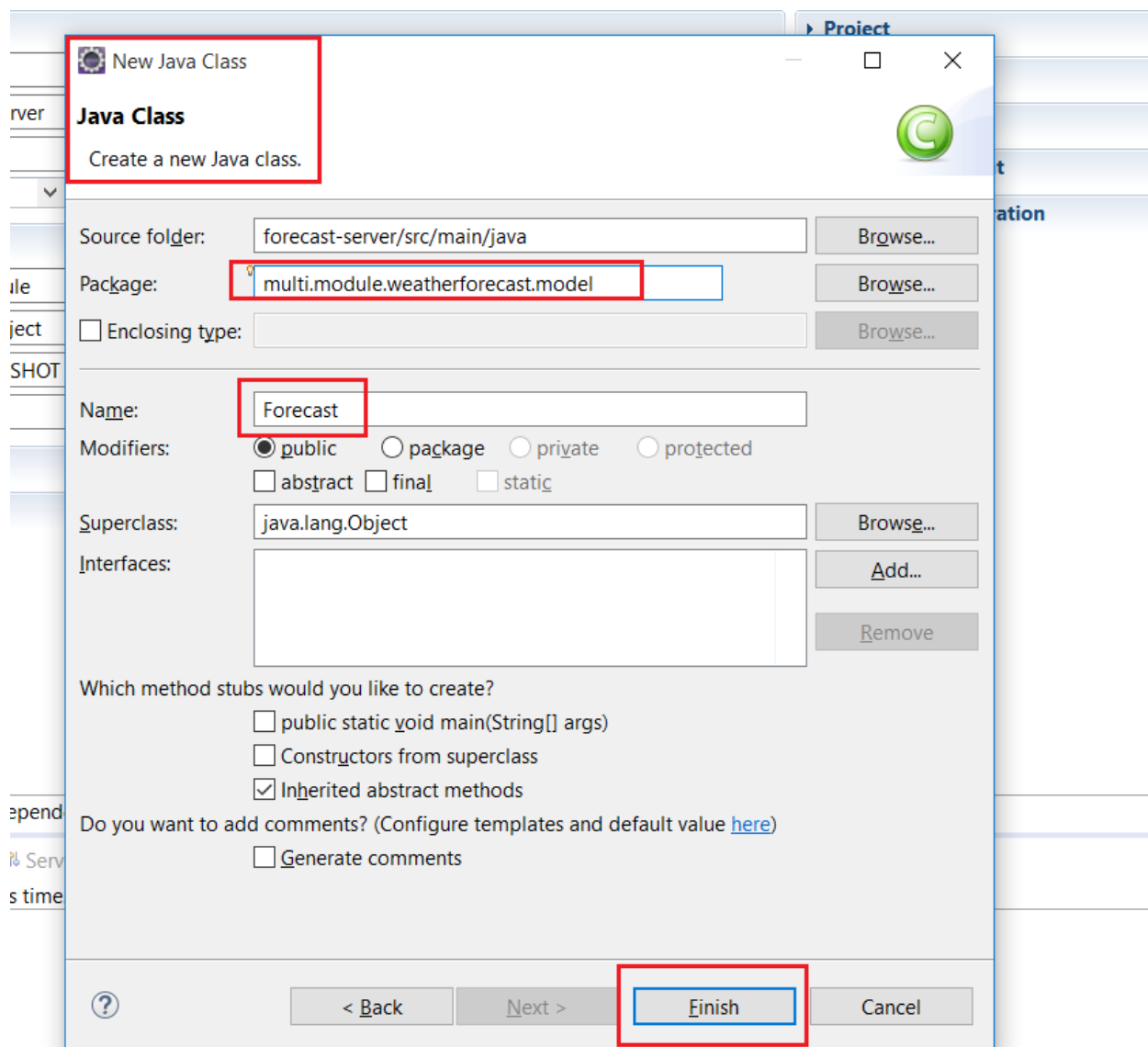
Le module forecast-server étant créé, on remarque qu'on le javaSE 1.7 comme on a déjà configuré dans le POM du parent. On remarque aussi que dans le POM de notre module, on a la balise <parent>



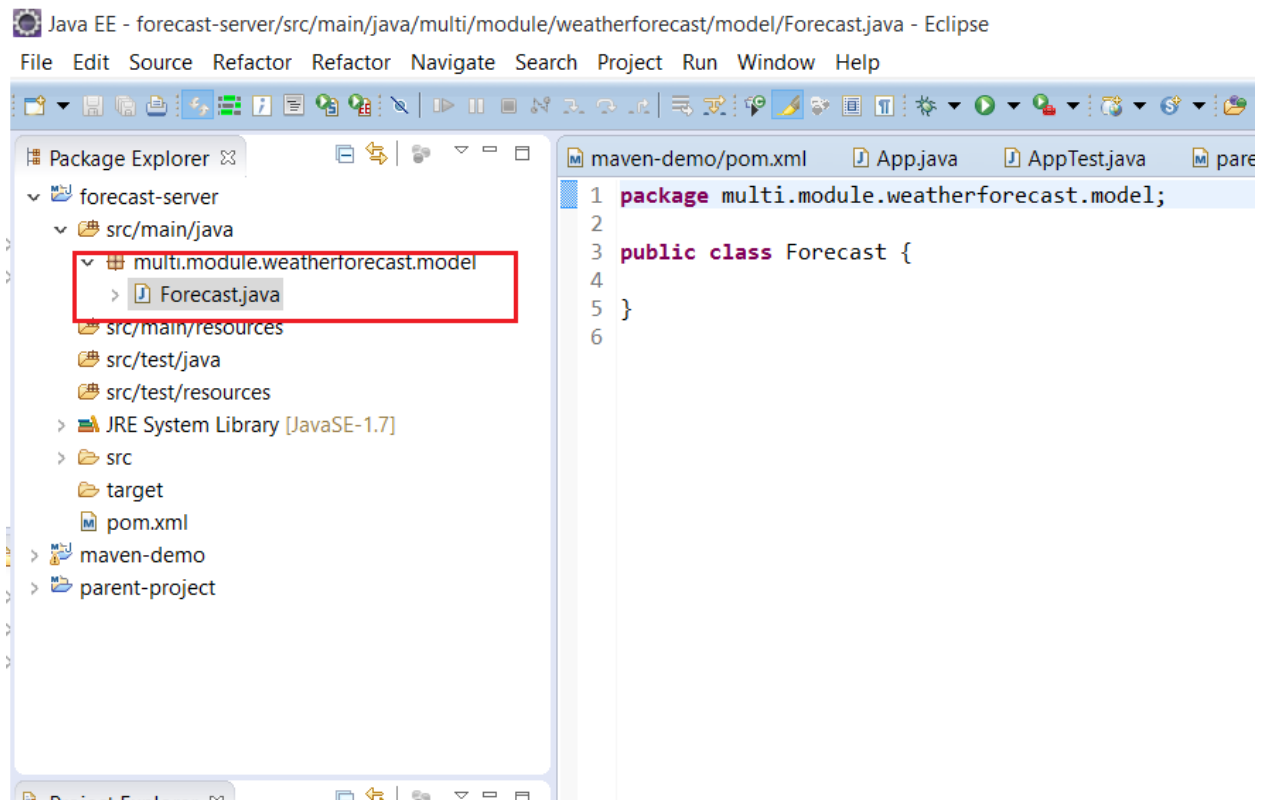
Ajoutons une classe java à notre serveur :

Package :multi.module.weatherforecast.model

Classe: Forecast



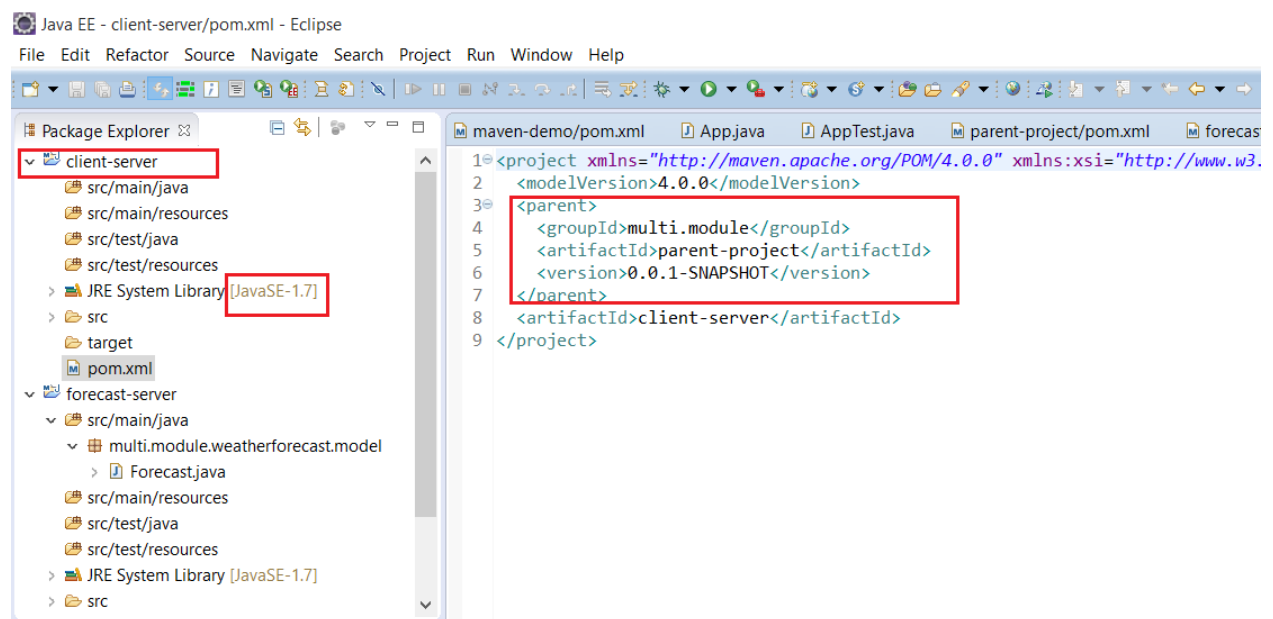
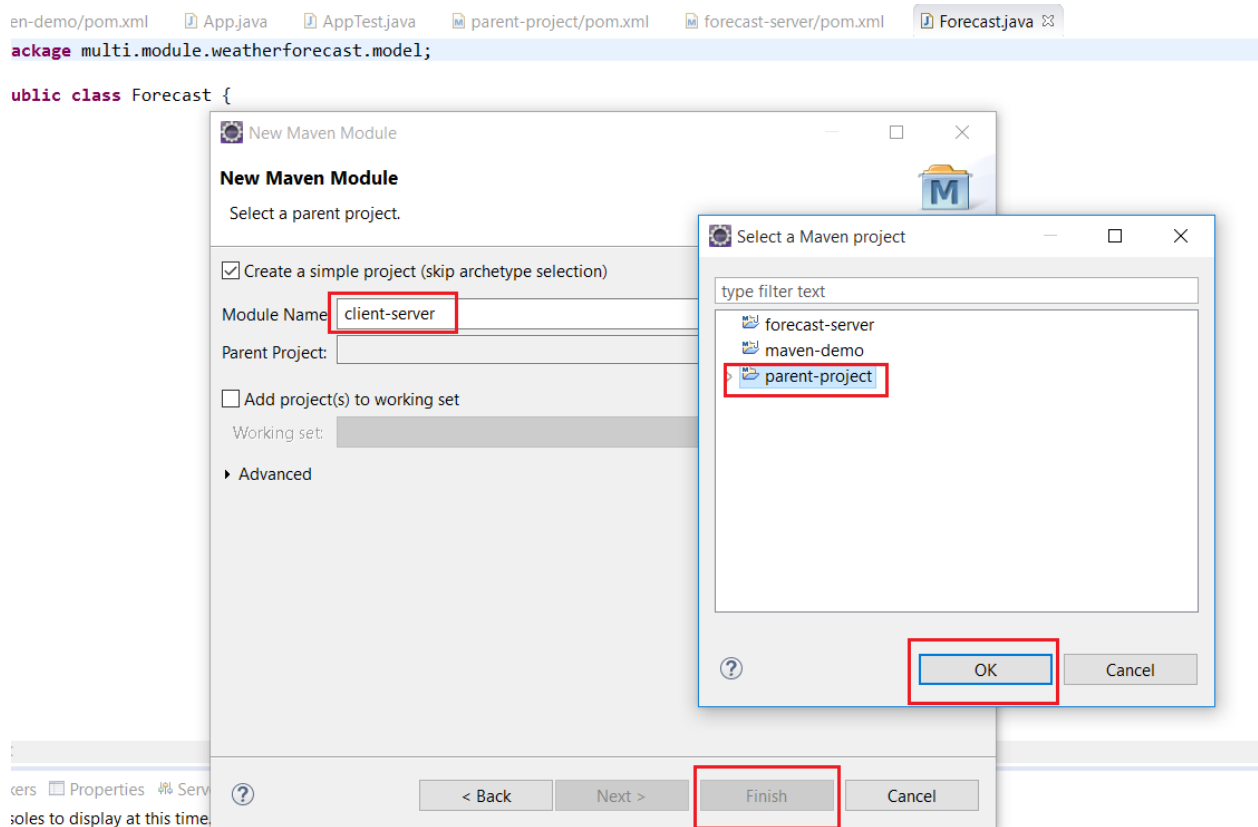




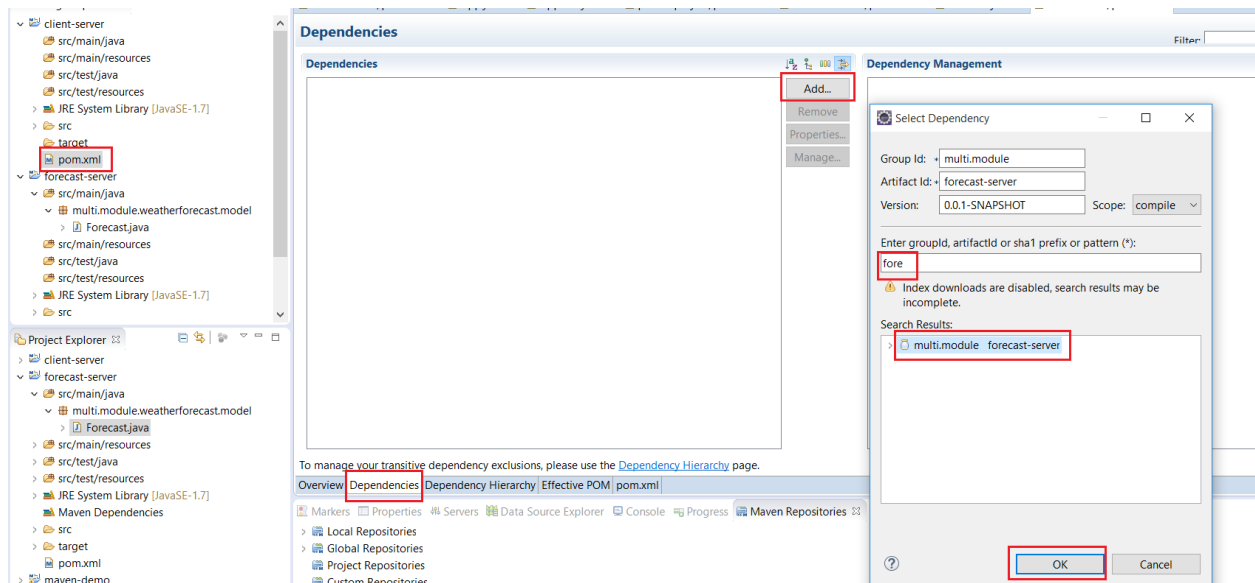
## 6.2 CREATION DU MODULE CLIENT

Module name: client-server

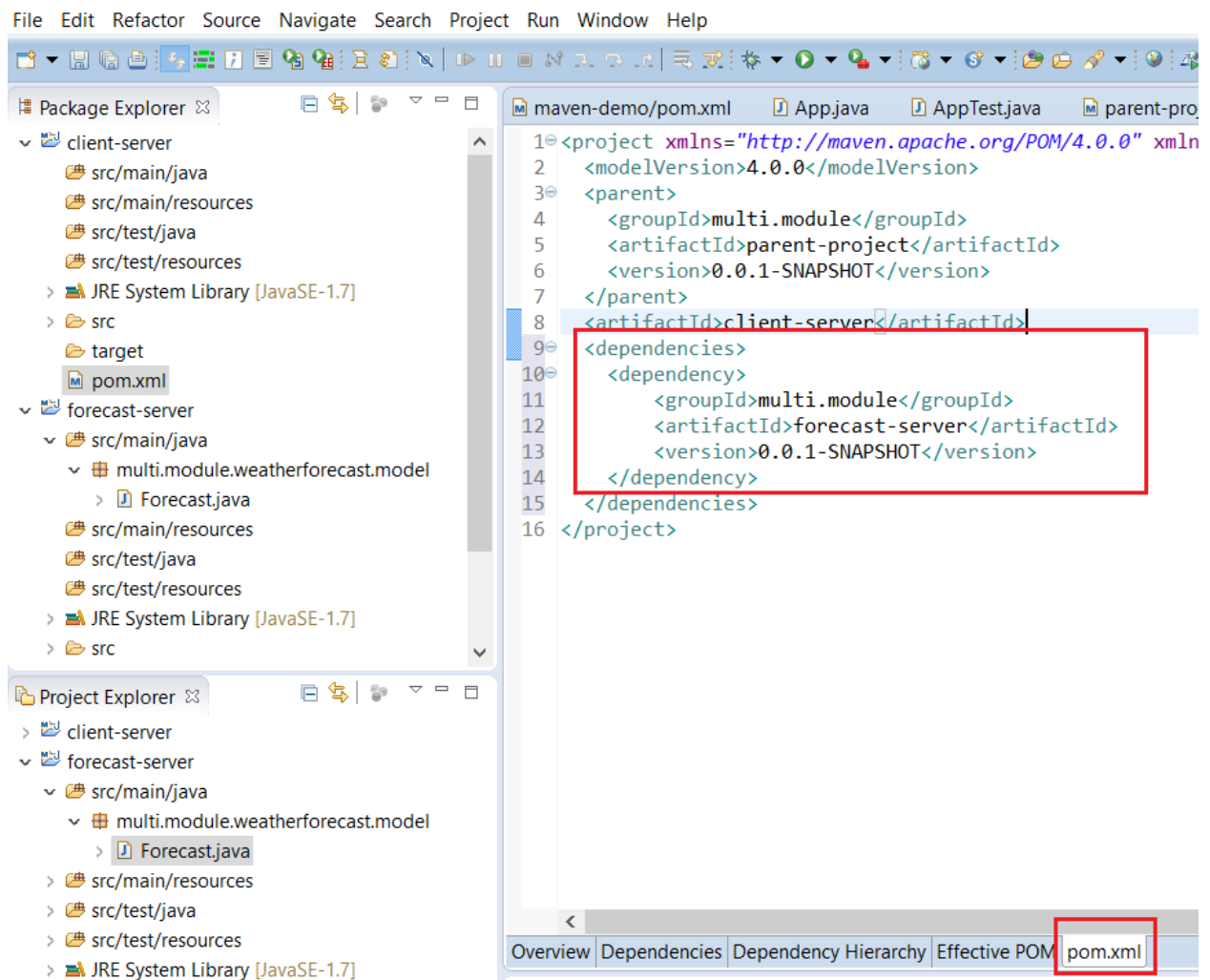
Parent Package : parent-package



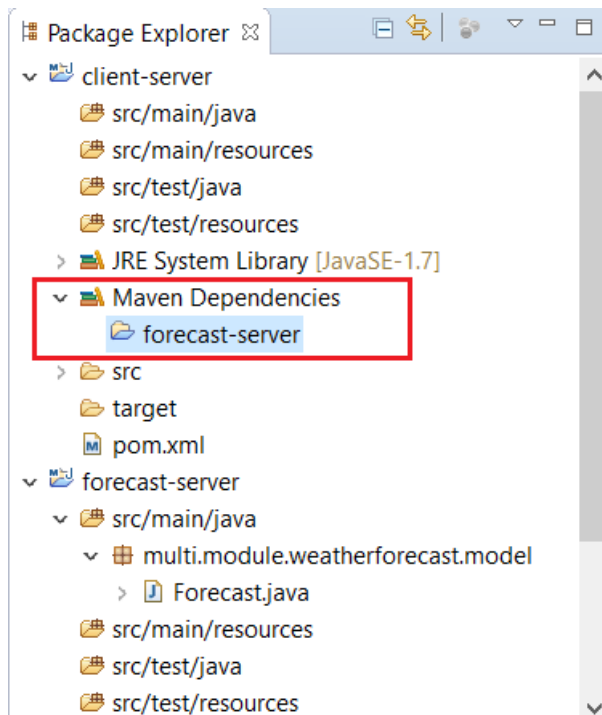
Déclaration d'une dépendance entre serveur et client (client dépend du serveur) :  
 Double cliquer sur le pom.xml du client-server → choisir la TAB 'Dependencies' → Add... → choisir 'forecast-server' → OK



Java EE - client-server/pom.xml - Eclipse



Lorsqu'on sauvegarde, on remarque l'apparition de 'Maven Dependencies' et une référence au projet forecast-server :

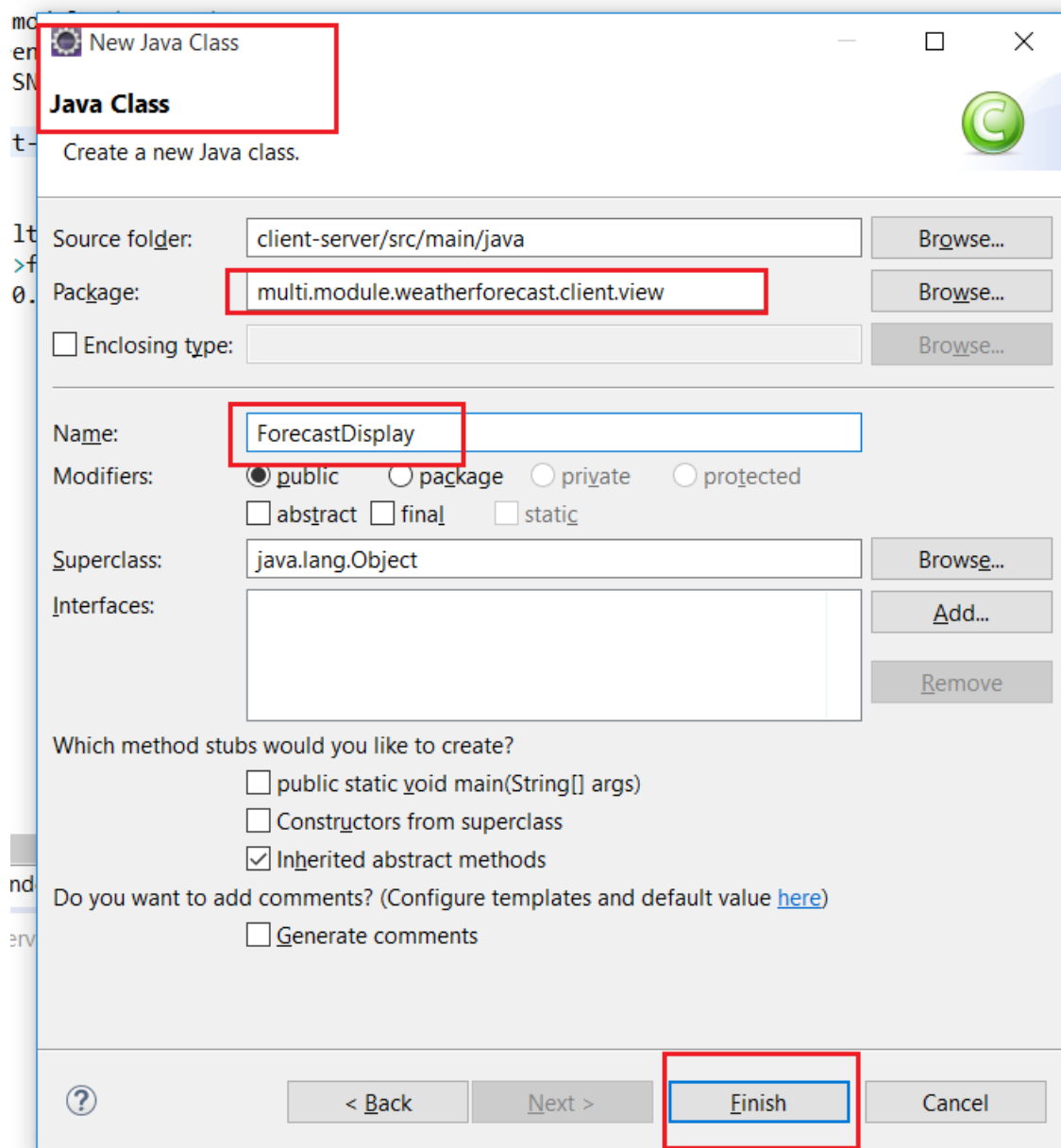


### 6.3 DEPENDANCES ENTRE PROJETS ET MODULES

Ajoutons une classe java à notre serveur :

Package :multi.module.weatherforecast.client.view

Classe: ForecastDisplay



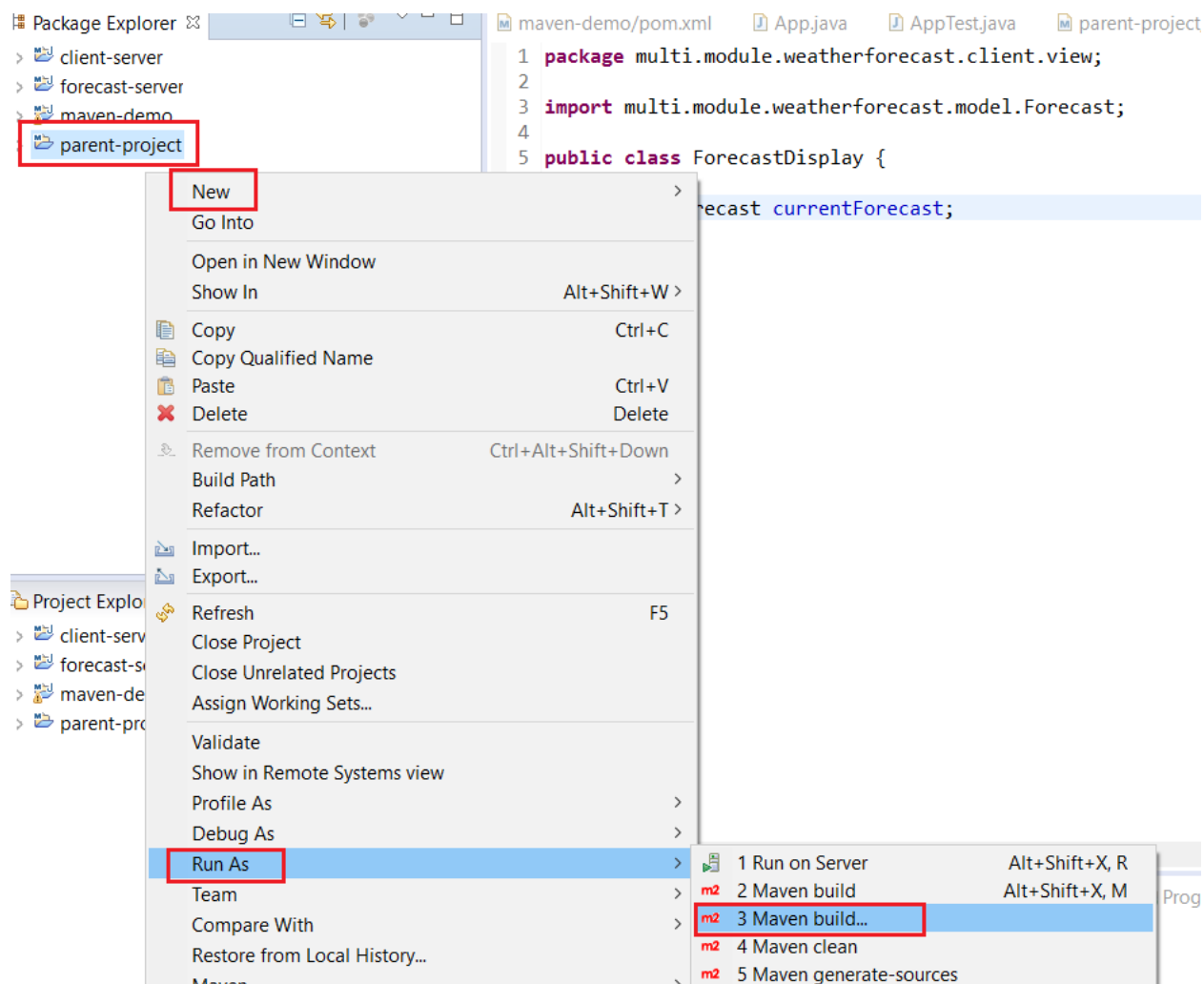
On Remarque qu'on a pu utiliser la classe Forecast du projet du module forecast-server et cela grace à la dependance ajoutée.

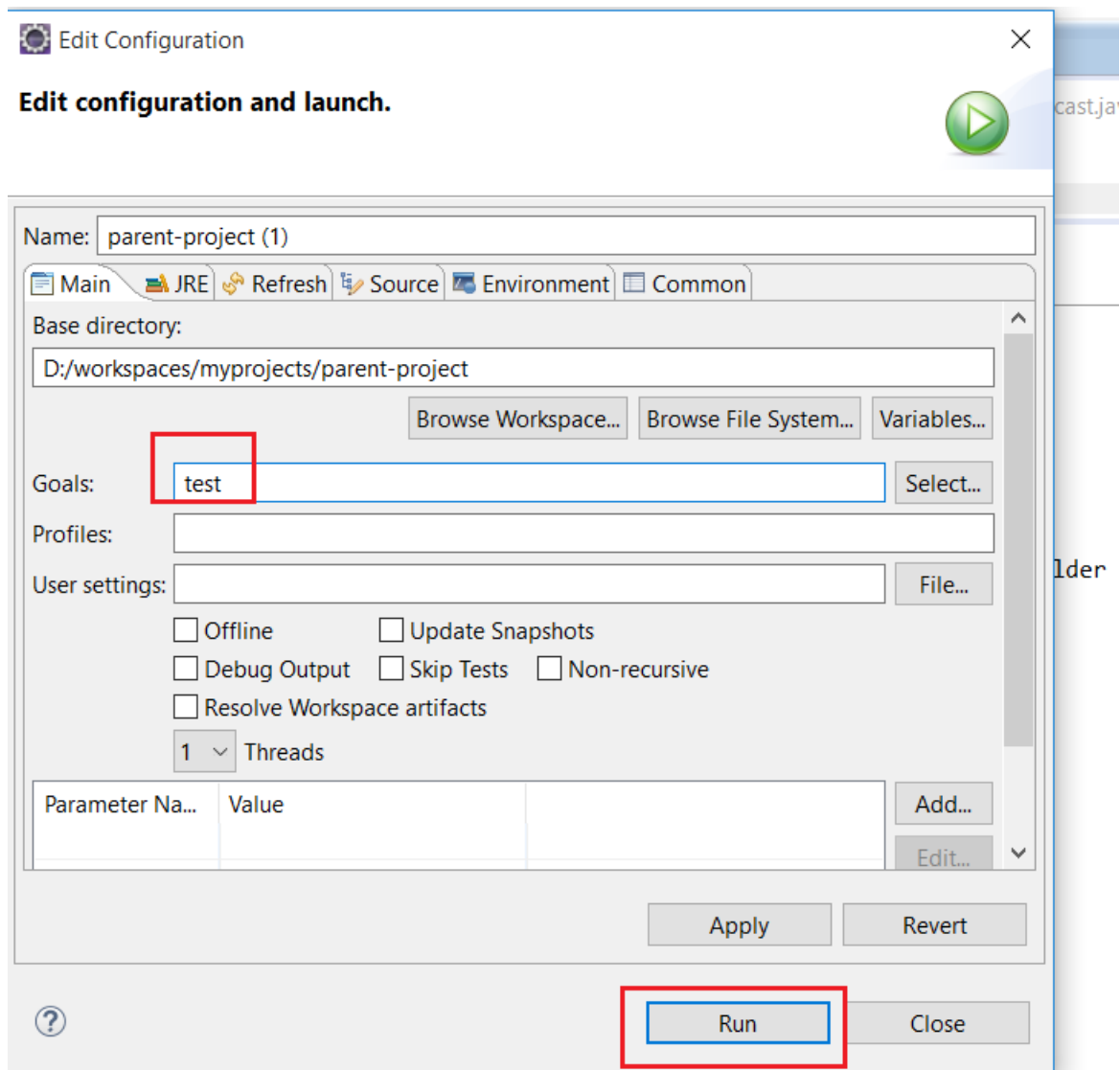
```

1 package multi.module.weatherforecast.client.view;
2
3 import multi.module.weatherforecast.model.Forecast;
4
5 public class ForecastDisplay {
6
7     public Forecast currentForecast;
8
9 }
10

```

Lorsqu'on fait build du projet parent, le build des modules fils sera déclenché automatiquement ; On peut remarquer cela dans la 'console'.





phase "build". You must specify a valid lifecycle phase or a goal in the format

```

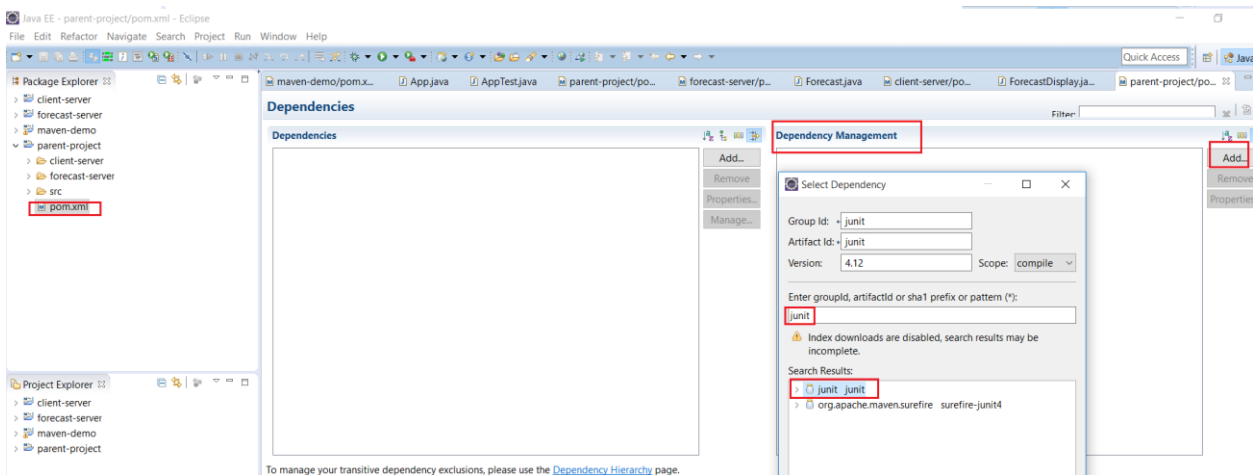
Markers  Properties  Servers  Data Source Explorer  Console  Progress  Maven Repositories
<terminated> parent-project (1) [Maven Build] C:\Program Files\Java\jdk1.8.0_112\bin\javaw.exe (Feb 12, 2017, 8:04:15 PM)
[INFO] Reactor Build Order:
[INFO]
[INFO] parent-project
[INFO] forecast-server
[INFO] client-server
[INFO]
[INFO] Using the builder org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder with a thread count of 1
[INFO]
[INFO] Building parent-project 0.0.1-SNAPSHOT
[INFO]
[INFO]
[INFO] Building forecast-server 0.0.1-SNAPSHOT
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ forecast-server ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ forecast-server ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file to D:\workspaces\myprojects\parent-project\forecast-server\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ forecast-server ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ forecast-server ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ forecast-server ---
[INFO]
[INFO] Building client-server 0.0.1-SNAPSHOT
[INFO]

```

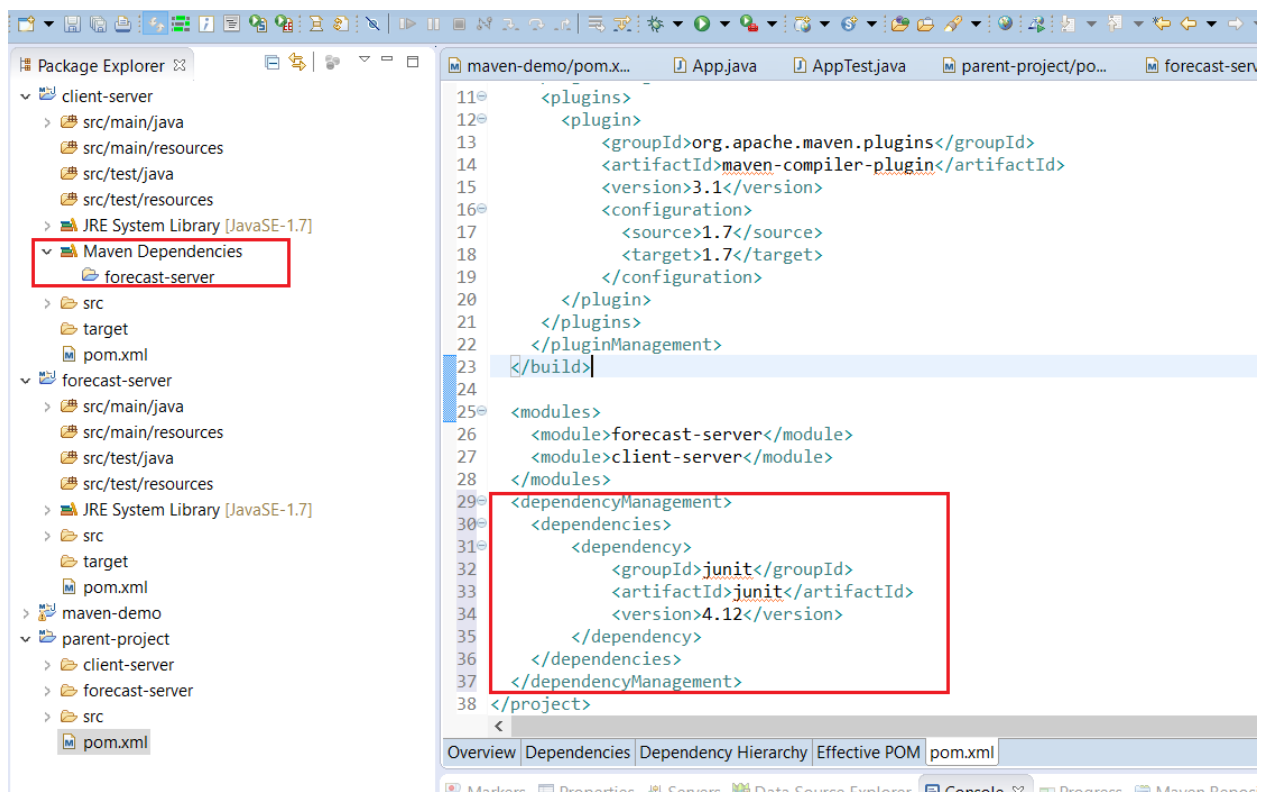
### Dependency Management:

On l'ajoute dans le pom.xml du projet parent et tous les modules fils héritent la dépendance incluse dans la 'dependency management'.

Pour ajouter la dependency management on clique sur le pom.xml du parent → on choisit le TAB 'dependencies' → Add dans 'Dependency Management' → on tape junit → on choisit 'junit-junit' → OK  
→ sauvegarder





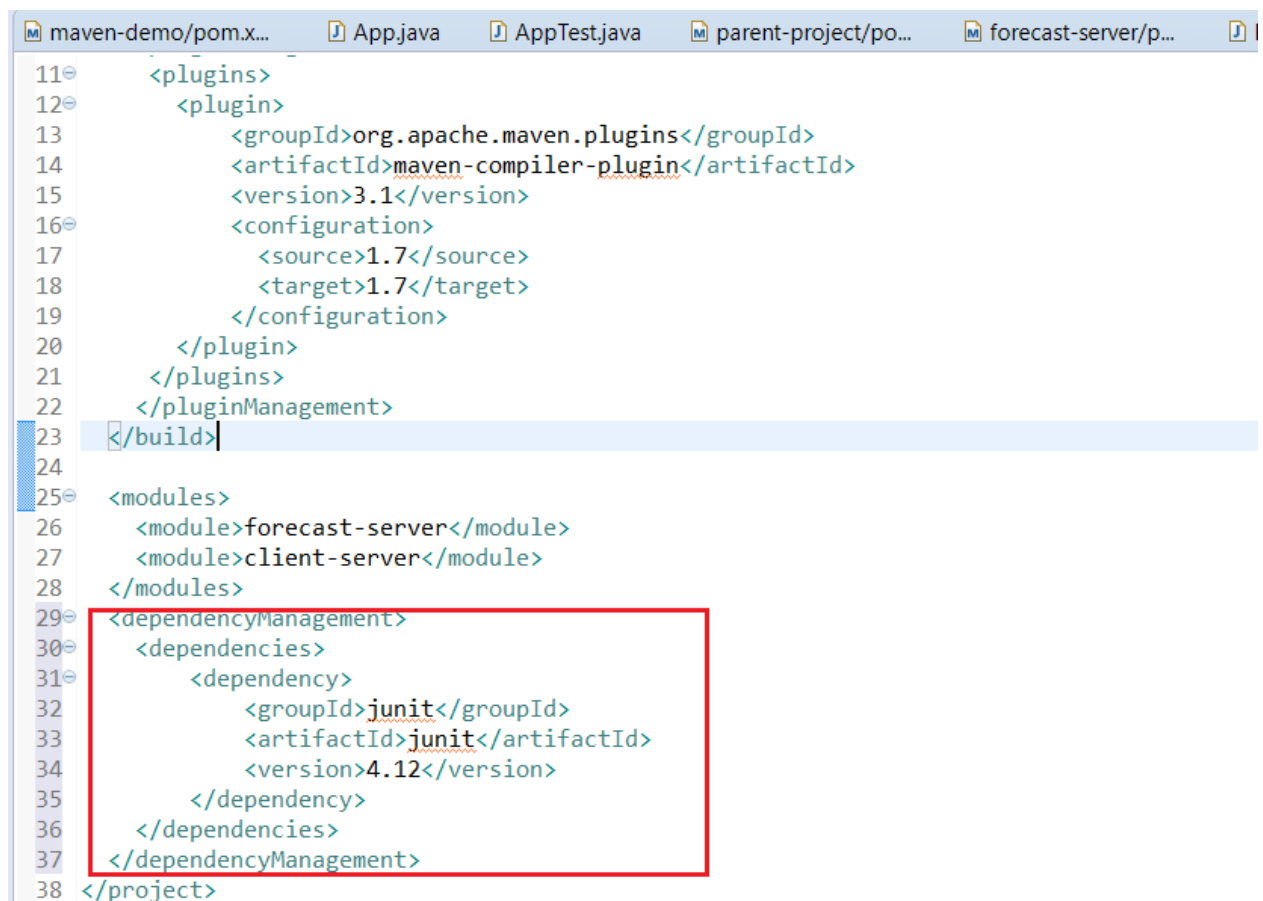


The screenshot shows an IDE with the Package Explorer on the left and the pom.xml file on the right. The Package Explorer shows a tree structure with 'forecast-server' highlighted under 'Maven Dependencies'. The pom.xml file shows the XML configuration for the 'maven-demo' project, with the 'dependencies' section highlighted in red.

```

11<  <plugins>
12<    <plugin>
13<      <groupId>org.apache.maven.plugins</groupId>
14<      <artifactId>maven-compiler-plugin</artifactId>
15<      <version>3.1</version>
16<      <configuration>
17<        <source>1.7</source>
18<        <target>1.7</target>
19<      </configuration>
20<    </plugin>
21<  </plugins>
22<  </pluginManagement>
23< </build>
24
25<  <modules>
26<    <module>forecast-server</module>
27<    <module>client-server</module>
28<  </modules>
29<  <dependencyManagement>
30<    <dependencies>
31<      <dependency>
32<        <groupId>junit</groupId>
33<        <artifactId>junit</artifactId>
34<        <version>4.12</version>
35<      </dependency>
36<    </dependencies>
37<  </dependencyManagement>
38< </project>

```



The screenshot shows the pom.xml file with the 'dependencies' section highlighted in red.

```

11<  <plugins>
12<    <plugin>
13<      <groupId>org.apache.maven.plugins</groupId>
14<      <artifactId>maven-compiler-plugin</artifactId>
15<      <version>3.1</version>
16<      <configuration>
17<        <source>1.7</source>
18<        <target>1.7</target>
19<      </configuration>
20<    </plugin>
21<  </plugins>
22<  </pluginManagement>
23< </build>
24
25<  <modules>
26<    <module>forecast-server</module>
27<    <module>client-server</module>
28<  </modules>
29<  <dependencyManagement>
30<    <dependencies>
31<      <dependency>
32<        <groupId>junit</groupId>
33<        <artifactId>junit</artifactId>
34<        <version>4.12</version>
35<      </dependency>
36<    </dependencies>
37<  </dependencyManagement>
38< </project>

```

## 8. DEPLOIEMENT

### 8.0 CREATION D'UN REPERTOIRE GITHUB

Au début il faut créer un répertoire dans GitHub avec le nom 'maven-repo' :

Search GitHub Pull requests Issues Gist

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: elyzgheib

Repository name: maven-repo

Great repository names are short and memorable. Need inspiration? How about **fuzzy-chainsaw**.

Description (optional)

☒ Public  
Anyone can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

☒ Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None ⓘ

Create repository

### 8.1 CONFIGURATION DU POM ET COMMANDE DEPLOY

Puis il faut faire un 'deploy' dans un répertoire temporaire :  
On Configure notre pom.xml en utilisant la balise <distributionManagement> :

```

<distributionManagement>
  <snapshotRepository>
    <id>rep-staging</id>
    <url>file://${project.build.directory}/maven-repo</url>
  </snapshotRepository>
  <repository>
    <id>rep-staging-release</id>
    <url>file://${project.build.directory}/maven-repo</url>
  </repository>
</distributionManagement>

```

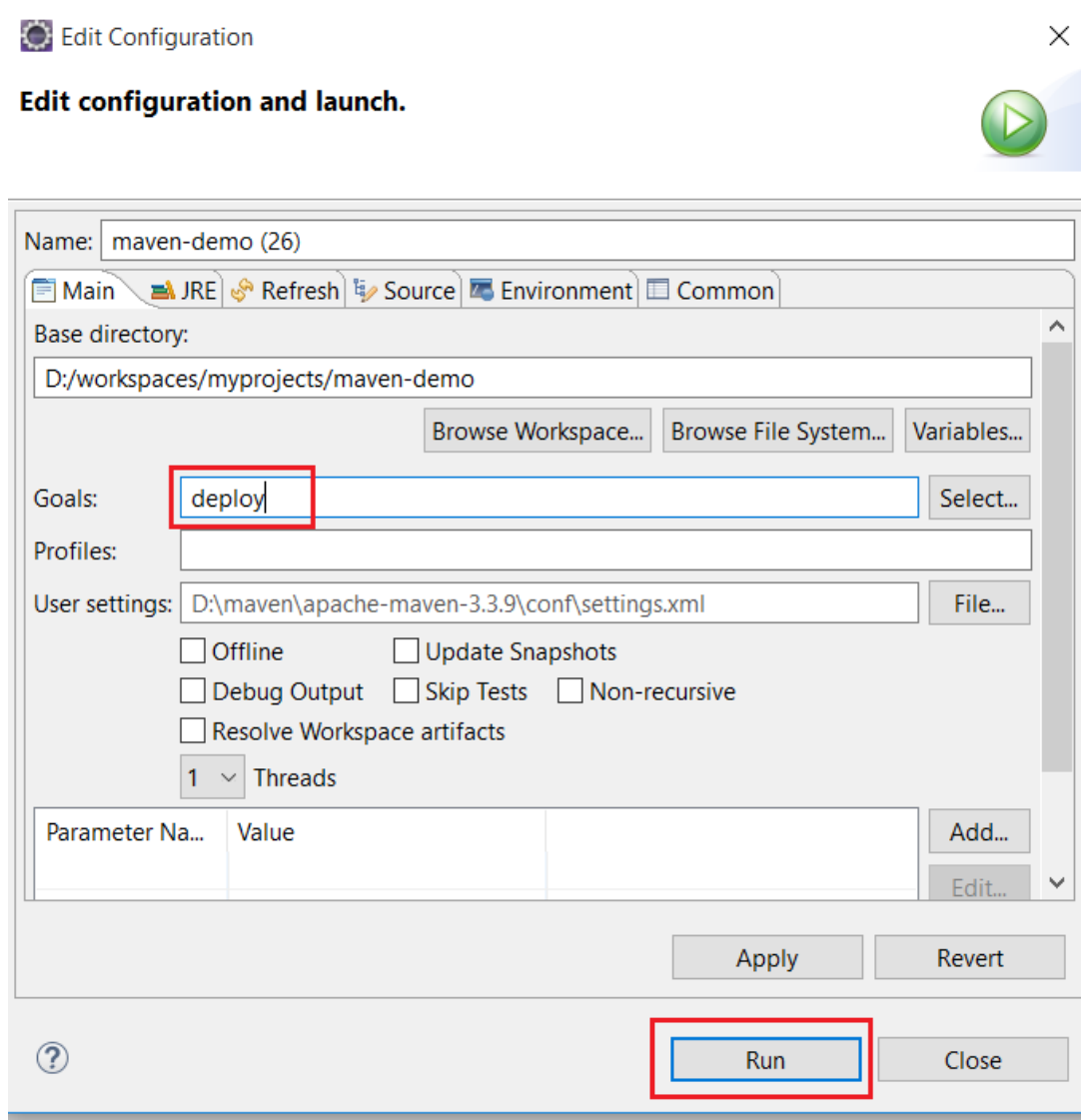
```

<distributionManagement>
  <snapshotRepository>
    <id>rep-staging</id>
    <url>file://${project.build.directory}/maven-repo</url>
  </snapshotRepository>
  <repository>
    <id>rep-staging-release</id>
    <url>file://${project.build.directory}/maven-repo</url>
  </repository>
</distributionManagement>
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-deploy-plugin</artifactId>
    <version>2.8.2</version>
    <configuration>
      <altDeploymentRepository>rep-
staging::default::file://${project.build.directory}/maven-repo</altDeploymentRepository>
    </configuration>
  </plugin>
</plugins>
</build>

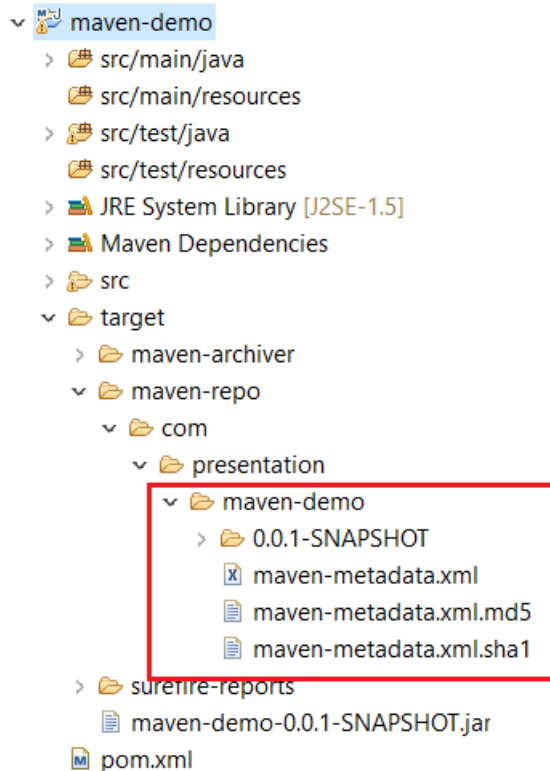
```

Puis on exécute la commande deploy. :

Clic droit sur le projet maven-demo → Run As → Maven build... → on tape 'deploy' dans goals  
→ Run :



On sélectionne le projet, et on clique 'F5' pour rafraîchir ; on remarque l'apparition du répertoire temporaire dans le répertoire 'target' :



Maintenant on va faire un 'upload' de l'artefact créé à github :  
On ajoute les infos d'authentification au fichier settings.xml :

```
<server>
  <id>github</id>
  <username>***nom profil github***</username>
  <password>***mot de passe github***</password>
</server>
```

Puis on va donner les détails de notre serveur ajouté dessus au projet en référant à son ID en ajoutant le code suivant dans le pom.xml:

```
<properties>
  <github.global.server>github</github.global.server>
</properties>
```

Et le code suivant :

```
<plugin>
  <groupId>com.github.github</groupId>
  <artifactId>site-maven-plugin</artifactId>
  <version>0.12</version>
  <configuration>
    <!-- git commit message -->
    <message>Maven artifacts for ${project.version}</message>
    <!-- disable webpage processing -->
    <noJekyll>true</noJekyll>
    <!-- matches distribution management repository url above -->
    <outputDirectory>${project.build.directory}/maven-repo</outputDirectory>
    <!-- remote branch name -->
```

```

<branch>refs/tags/${project.version}</branch>
<!-- If you remove this then the old artifact will be removed and new
      one will replace. But with the merge tag you can just release by changing
      the version -->

<merge>true</merge>
<includes>
  <include>*/*</include>
</includes>
<!-- github repo name -->
<repositoryName>maven-repo</repositoryName>
<!-- github username -->
<repositoryOwner>elyzgheib</repositoryOwner>
</configuration>
<executions>
  <execution>
    <goals>
      <goal>site</goal>
    </goals>
    <phase>deploy</phase>
  </execution>
</executions>
</plugin>

```

Enfin, on exécute la commande 'deploy' de nouveau et voilà ce qu'on voit dans Github :

```

[INFO] Creating commit with SHA-1: 0e75036d15ede66847a5b95f8ee1589ceae4310c
[INFO] Creating reference refs/tags/0.0.1-SNAPSHOT starting at commit 0e75036d15ede66847a5b95f8ee1589ceae4310c
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:21 min
[INFO] Finished at: 2017-02-19T22:24:06+02:00
[INFO] Final Memory: 12M/107M
[INFO] -----

```

The screenshot shows the GitHub interface for the repository 'elyzgheib / maven-repo'. The URL bar shows 'https://github.com/elyzgheib/maven-repo/tree/0.0.1-SNAPSHOT'. The repository has 1 commit, 1 branch, 1 release, and 1 contributor. The 'Code' tab is selected, showing a table of Maven artifacts for the 0.0.1-SNAPSHOT tag. The table has two rows: 'com/presentation/maven-demo' and '.nojekyll', both with a timestamp of '27 minutes ago'. A red box highlights these two rows.

File	Timestamp
com/presentation/maven-demo	27 minutes ago
.nojekyll	27 minutes ago

elyzgheib / maven-repo

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Tag: 0.0.1-SNAPSHOT maven-repo / com / presentation / maven-demo / Create new file Upload files Find file History

elyzgheib Maven artifacts for 0.0.1-SNAPSHOT Latest commit 0e75036 27 minutes ago

..		
0.0.1-SNAPSHOT	Maven artifacts for 0.0.1-SNAPSHOT	27 minutes ago
maven-metadata.xml	Maven artifacts for 0.0.1-SNAPSHOT	27 minutes ago
maven-metadata.xml.md5	Maven artifacts for 0.0.1-SNAPSHOT	27 minutes ago
maven-metadata.xml.sha1	Maven artifacts for 0.0.1-SNAPSHOT	27 minutes ago

elyzgheib / maven-repo

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Tag: 0.0.1-SNAPSHOT maven-repo / com / presentation / maven-demo / 0.0.1-SNAPSHOT / Create new file Upload files Find file History

elyzgheib Maven artifacts for 0.0.1-SNAPSHOT Latest commit 0e75036 30 minutes ago

..		
maven-demo-0.0.1-20170219.171745-1.jar	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago
maven-demo-0.0.1-20170219.171745-1.jar.md5	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago
maven-demo-0.0.1-20170219.171745-1.jar.sha1	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago
maven-demo-0.0.1-20170219.171745-1.pom	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago
maven-demo-0.0.1-20170219.171745-1.pom.md5	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago
maven-demo-0.0.1-20170219.171745-1.pom.sha1	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago
maven-demo-0.0.1-20170219.182049-2.jar	Maven artifacts for 0.0.1-SNAPSHOT	30 minutes ago