



# Atelier SOA - 01

## Configuration de l'environnement de développement

### Objectifs

Préparer l'environnement de création, de compilation et de déploiement d'une application web

#### A. Mettre en place l'environnement de développement

- Télécharger et installer l'EDI **IntelliJ IDEA**
- Configurer l'EDI
- Découvrir les caractéristiques de **IntelliJ IDEA**

#### B. Installer JDK et configurer les variables d'environnement

#### C. Installer l'outil Maven

#### D. Configurer un projet JAVA avec JDK en utilisant IntelliJ

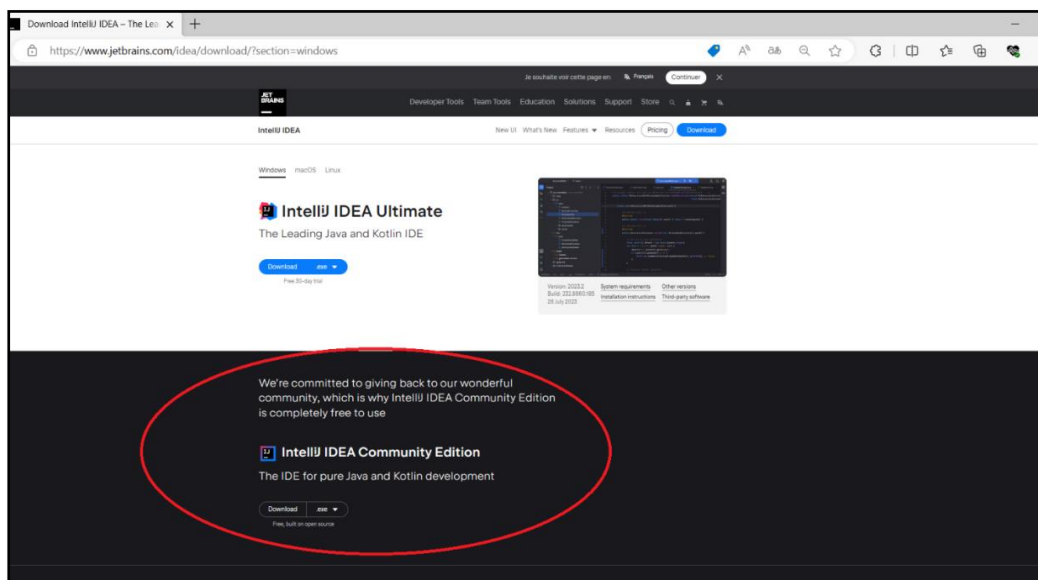
#### E. Utiliser les commandes de base de Maven (Les phases de cycle de vie)

#### F. Déployer une application web avec Tomcat en utilisant Maven

### A. Mettre en place l'environnement de développement

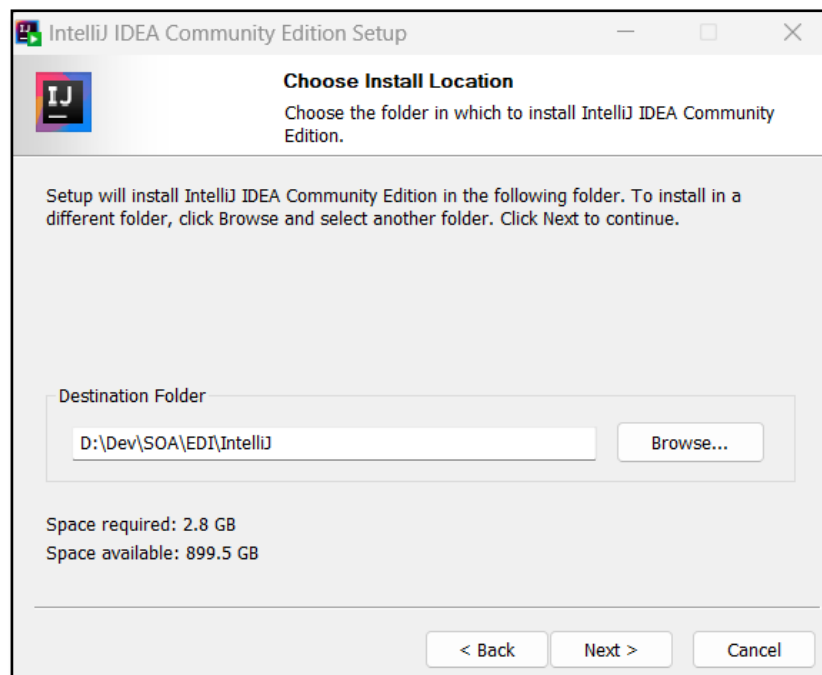
1. Télécharger la version (open source «.exe») de l'EDI «**IntelliJ IDEA**» (**Community Edition**) à travers le lien suivant :

<https://www.jetbrains.com/idea/download/#section=windows>

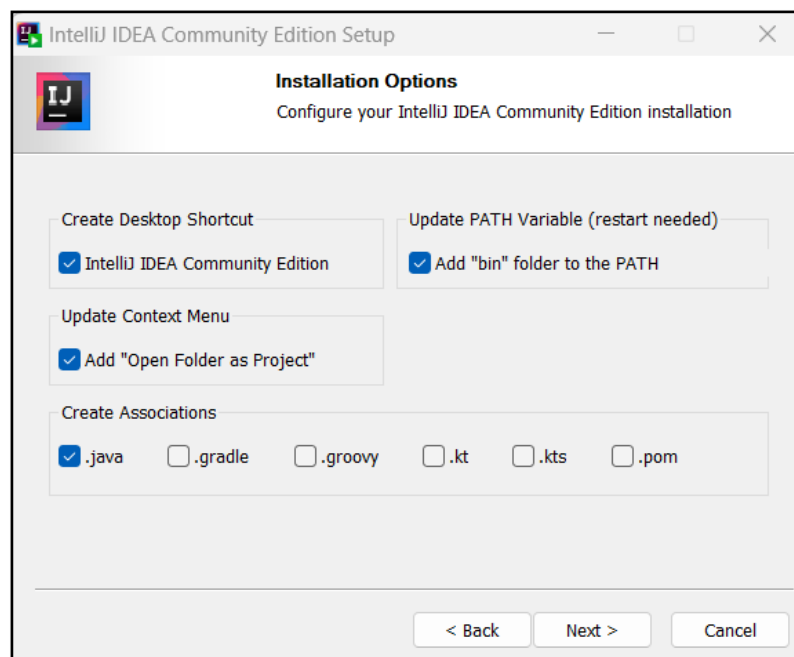


2. Double-cliquer sur l'exécutable téléchargé (**ideaIC-2023.2**) pour lancer l'installation. Choisir d'installer « **IntelliJ IDEA** » dans le dossier (**Destination Folder**) puis cliquer sur «**Next**» :

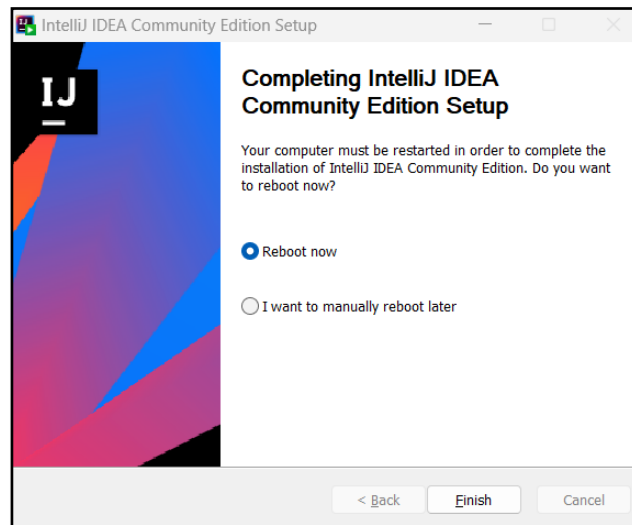
**D:\Dev\SOA\EDI\IntelliJ**



3. Cocher les options suivantes puis cliquer sur «**Next**» :



4. Terminer le processus d'installation, demander de redémarrer l'ordinateur et finaliser :

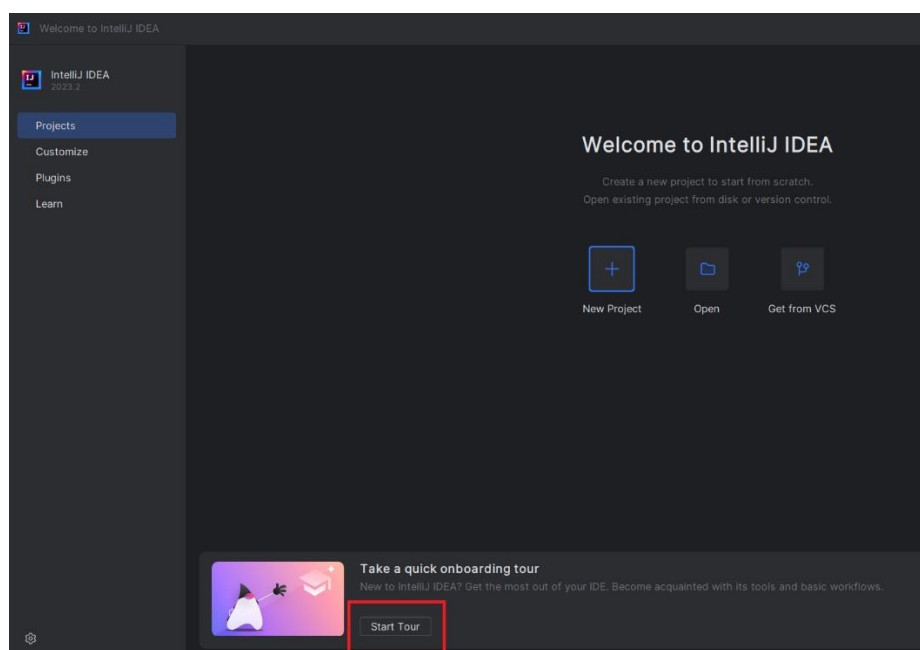


5. Après redémarrage, Ouvrir «**IntelliJ**» en double cliquant sur le raccourci dans le bureau **ou bien** sur l'application «**idea64**» située dans le dossier :

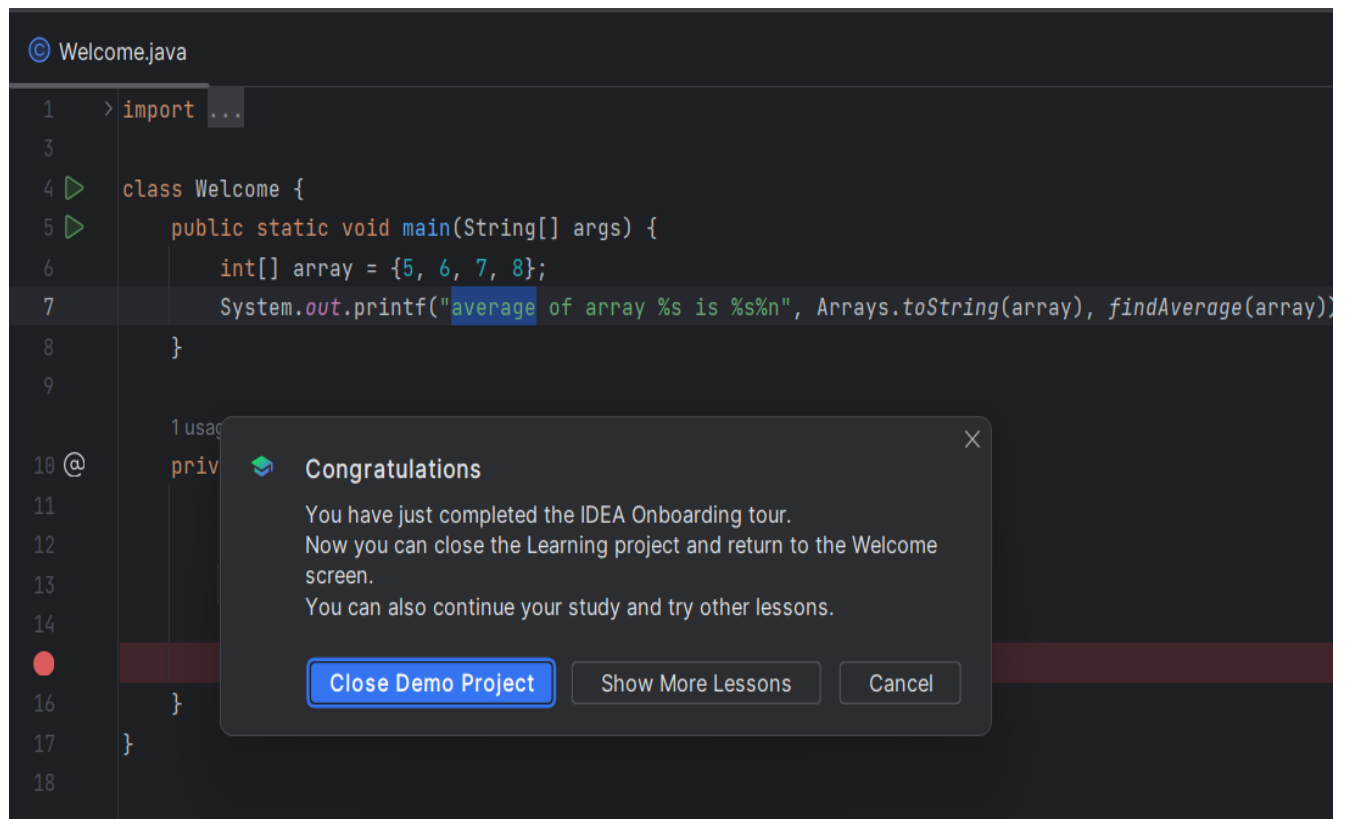
**D : \Dev\SOA\EDI\IntelliJ\bin**

6. Avant de commencer notre premier projet avec **IntelliJ**, aller pour une visite rapide pour découvrir l'EDI :

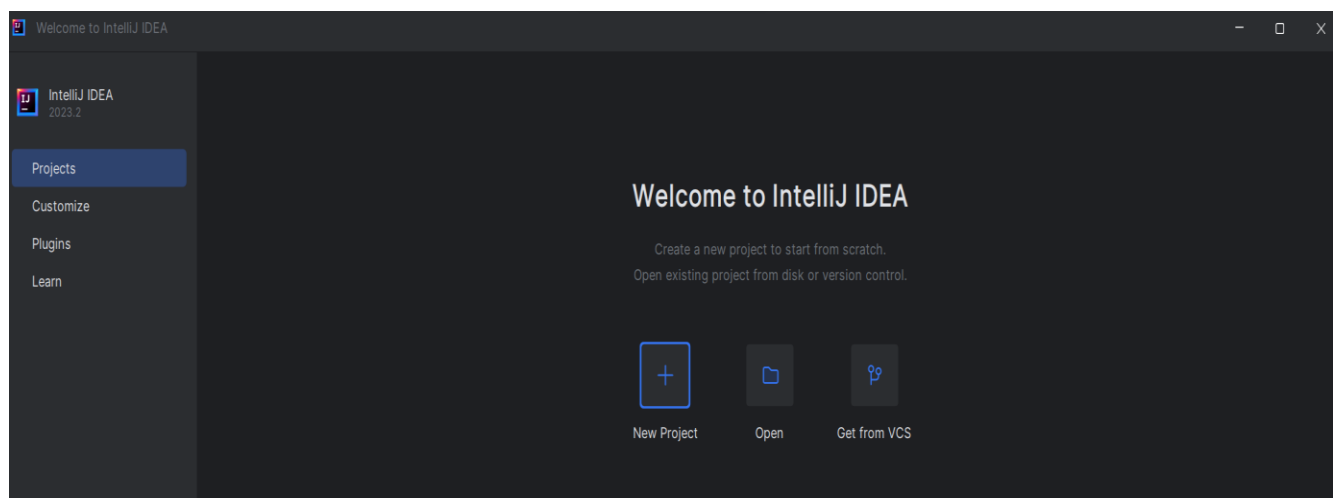
- a. Cliquer sur «**Start tour**» pour commencer cette leçon d'apprentissage d'utilisation de l'EDI :



- b. Suivre les consignes de la leçon d'apprentissage (étape par étape) jusqu'à arriver à la fin de cette leçon:



- c. Vous pouvez continuer suivre d'autres leçons pour mieux se former sur la bonne utilisation de cet EDI. Pour le moment, cliquer sur « **Close Demo Project** » pour mettre fin à cet apprentissage et aller à l'écran d'accueil :

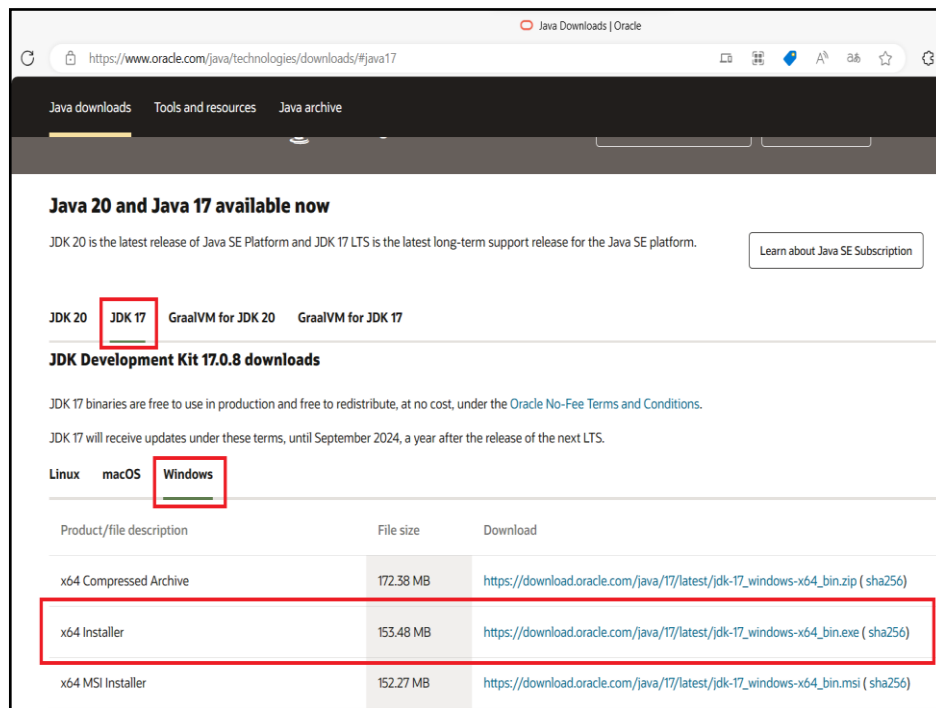


## B. Télécharger et installer «JDK17»

(Passer directement à section C. si le JDK17 est déjà installer et configuré)

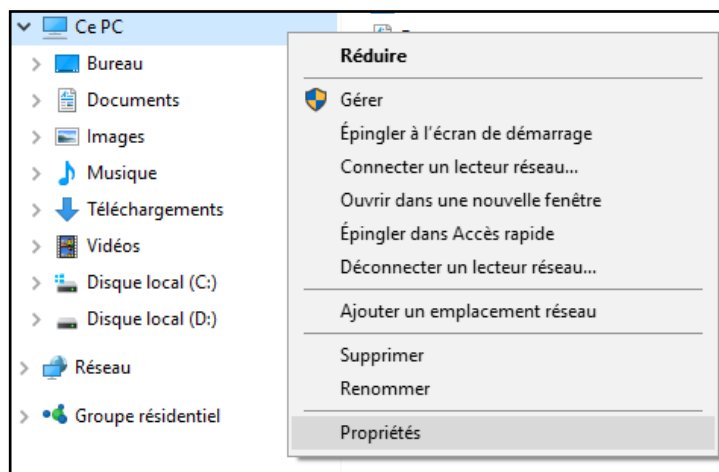
7. Tout d'abord, installer **JDK17**. Voici le lien de téléchargement :

**<https://www.oracle.com/java/technologies/downloads/#java17>**

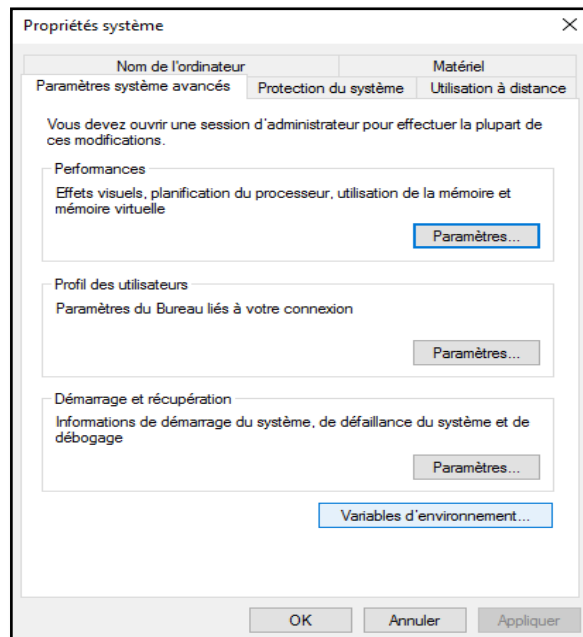


8. Une fois la JDK est installée, ajouter le chemin du dossier «**bin**» du **JDK17** à la variable d'environnement «**PATH**» (en premier lieu). Pour se faire, suivre les actions suivantes :

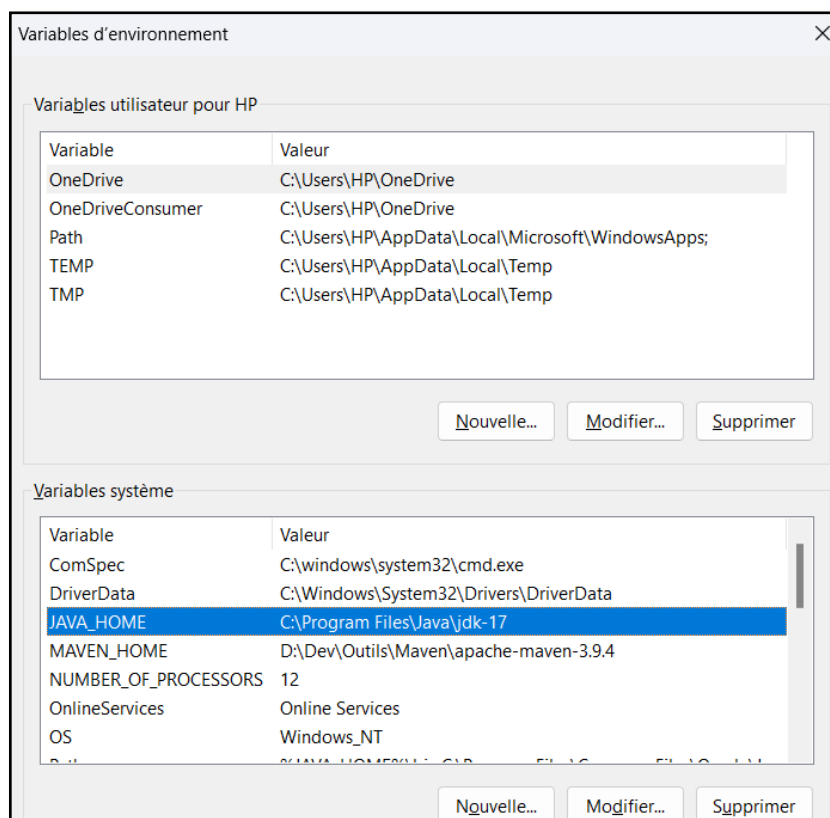
a. Cliquer avec le bouton droit de la souris sur l'icône « **Poste de travail** » puis sélectionner la commande «**Propriétés**» :



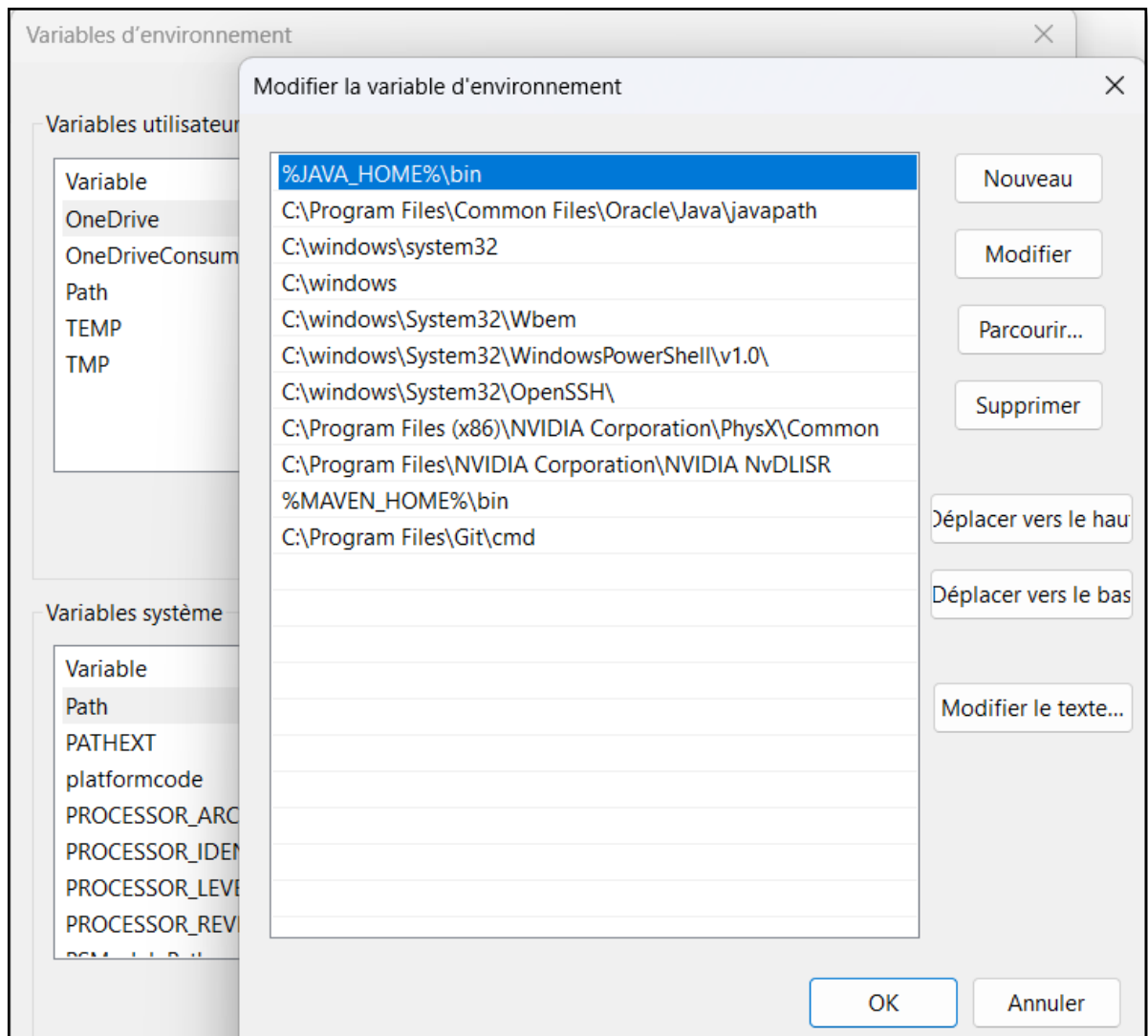
b. Appuyer sur le bouton «**Variables d'environnement**»:



c. Ajouter une variable d'environnement « **Système** » nommée **JAVA\_HOME** ayant comme valeur le chemin d'installation de JDK 17: «**C:\Program Files\Java\jdk-17**» : cliquer sur le bouton «**Nouvelle**» pour ajouter la nouvelle variable)



d. Double cliquer sur la variable système **Path** pour ajouter une nouvelle valeur «**%JAVA\_HOME%\bin**» (qui représente le chemin des exécutables de JDK 17) : Utiliser le bouton «**Déplacer vers le haut**» pour placer la nouvelle valeur en tête de liste. Enfin confirmer l'opération en appuyant sur le bouton «**OK**».



## C. Télécharger et installer l'outil « Maven »

9. Voici le lien de téléchargement de «**Maven**» :

<https://maven.apache.org/download.cgi>

The screenshot shows the Apache Maven Project download page for version 3.9.4. The page includes a sidebar with navigation links, a main heading 'Downloading Apache Maven 3.9.4', and a section for 'System Requirements'. Below this, there is a 'Files' section with a table of download links and checksums. The 'Binary zip archive' link is highlighted with a red box.

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.9.4-bin.tar.gz</a>	<a href="#">apache-maven-3.9.4-bin.tar.gz.sha512</a>	<a href="#">apache-maven-3.9.4-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.9.4-bin.zip</a>	<a href="#">apache-maven-3.9.4-bin.zip.sha512</a>	<a href="#">apache-maven-3.9.4-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.9.4-src.tar.gz</a>	<a href="#">apache-maven-3.9.4-src.tar.gz.sha512</a>	<a href="#">apache-maven-3.9.4-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.9.4-src.zip</a>	<a href="#">apache-maven-3.9.4-src.zip.sha512</a>	<a href="#">apache-maven-3.9.4-src.zip.asc</a>

10. Choisir le type «**Binary zip archive**» (version 3.9.4 : dernière version actuellement)

11. Télécharger le fichier sous :

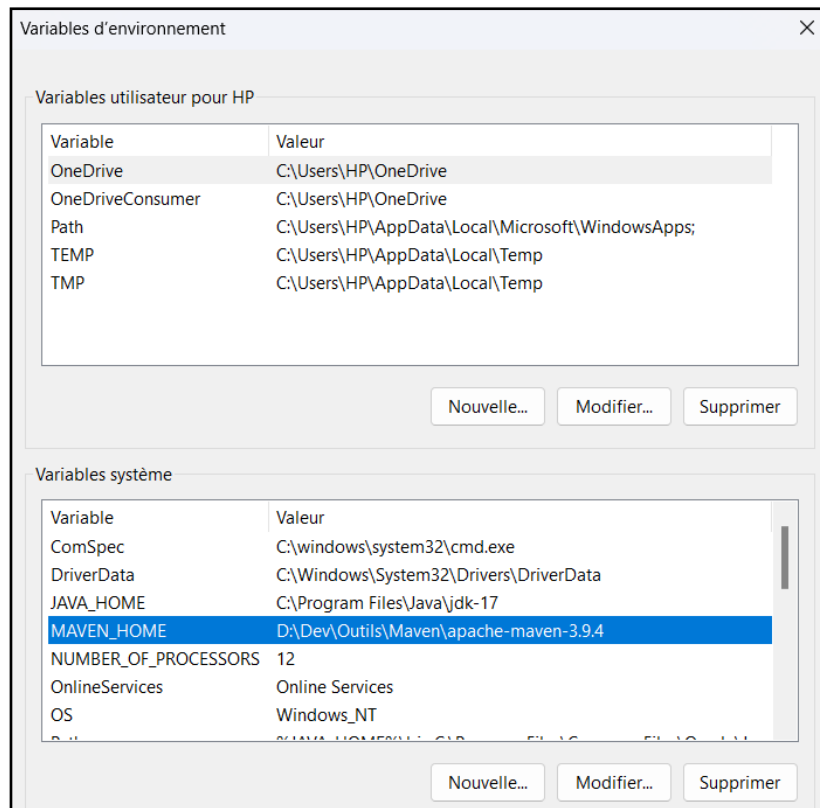
**D:\Dev\Outils\Maven**

12. Décompresser le fichier «**apache-maven-3.9.4-bin.zip**» dans le même dossier.

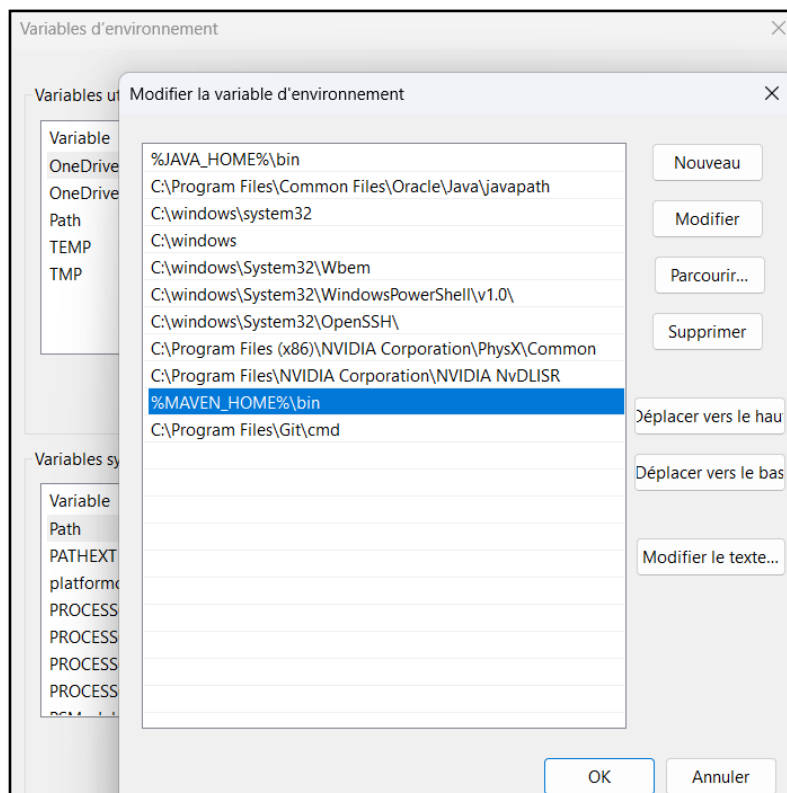
13. Ajouter une variable d'environnement système **MAVEN\_HOME** correspondant au chemin :

**D:\Dev\Outils\Maven\apache-maven-3.9.4**





14. Ajouter la valeur «**%MAVEN\_HOME%\bin**» à la variable d'environnement système **PATH**:



15. Ouvrir une fenêtre «**Invite de commandes**» et tester la configuration de «**Maven**» à travers la commande DOS `mvn -version` (Si tout va bien, les versions de **Maven**, de **JAVA** et de **Windows** seront affichées dans la console DOS).

```
C:\Users\HP>mvn -version
Apache Maven 3.9.4 (dfbb324ad4a7c8fb0bf182e6d91b0ae20e3d2dd9)
Maven home: D:\Dev\Outils\Maven\apache-maven-3.9.4
Java version: 17.0.8, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: fr_FR, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

16. Lancer maintenant la commande `mvn clean`. Cette commande génère une erreur à cause de l'absence d'un projet de type qui nécessite l'existence d'un fichier «**pom.xml**» ( voir la création d'un projet Maven dans la section suivante)

```
C:\Users\HP>mvn clean
[INFO] Scanning for projects...
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.111 s
[INFO] Finished at: 2023-08-29T12:33:44+01:00
[INFO] -----
[ERROR] The goal you specified requires a project to execute but there is no POM in this directory (C:\Users\HP). Please
verify you invoked Maven from the correct directory. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MissingProjectException
```

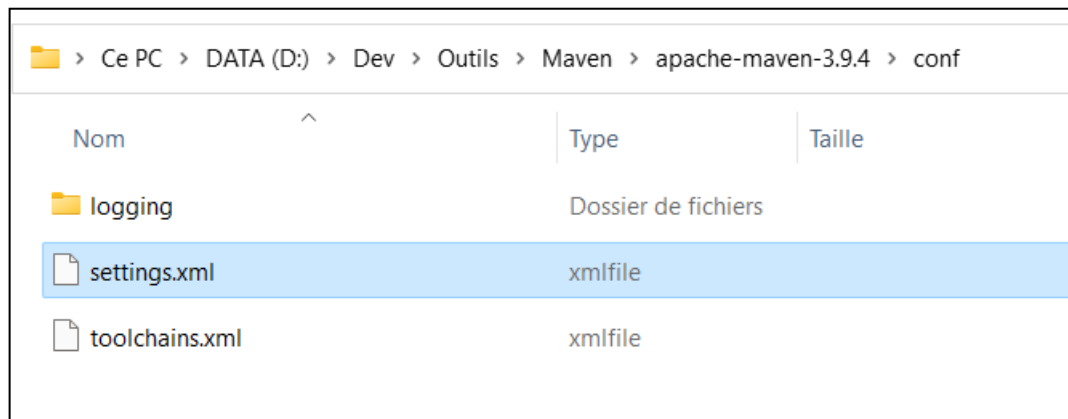
17. Remarquer la génération d'un dossier «**C:\Users\HP\.m2\repository**». Ce dossier consiste le dépôt local de **Maven**. («**C:\Users\HP\**» est le dossier racine de l'utilisateur courant):



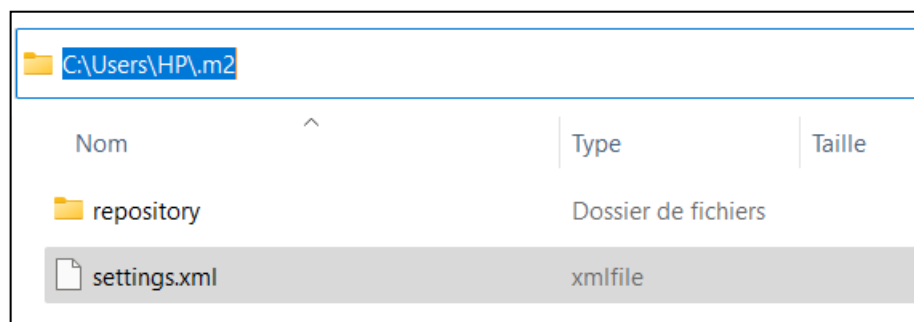
Nom	Type	Taille
antlr	Dossier de fichiers	
aopalliance	Dossier de fichiers	
asm	Dossier de fichiers	
backport-util-concurrent	Dossier de fichiers	
ch	Dossier de fichiers	
classworlds	Dossier de fichiers	
com	Dossier de fichiers	
commons-codec	Dossier de fichiers	
commons-io	Dossier de fichiers	
commons-lang	Dossier de fichiers	
io	Dossier de fichiers	
jakarta	Dossier de fichiers	
javax	Dossier de fichiers	
junit	Dossier de fichiers	
net	Dossier de fichiers	
org	Dossier de fichiers	

18. Il est possible de changer l'emplacement par défaut du dépôt local de **Maven** en suivant les instructions suivantes :

- a) Copier le fichier «**settings.xml**» qui existe dans le dossier «**D:\Dev\Outils\Maven\apache-maven-3.9.4\conf**».

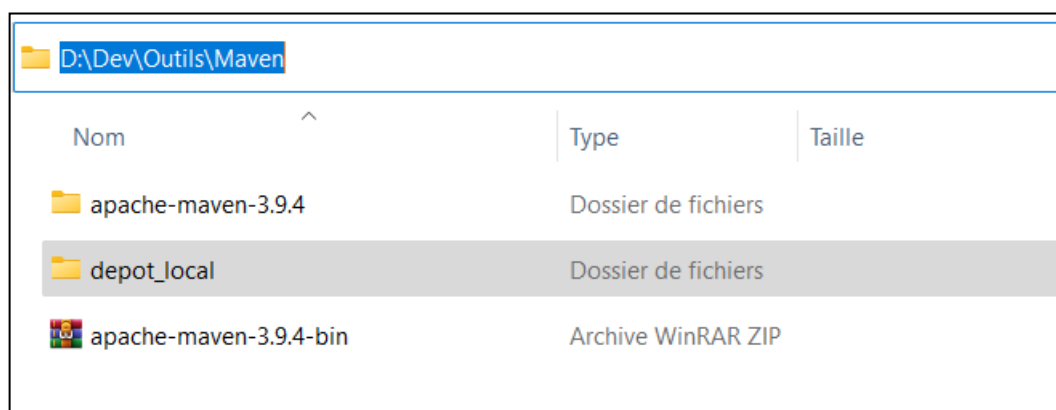


- b) Le coller dans le dossier «**C:\Users\HP\.m2**» :

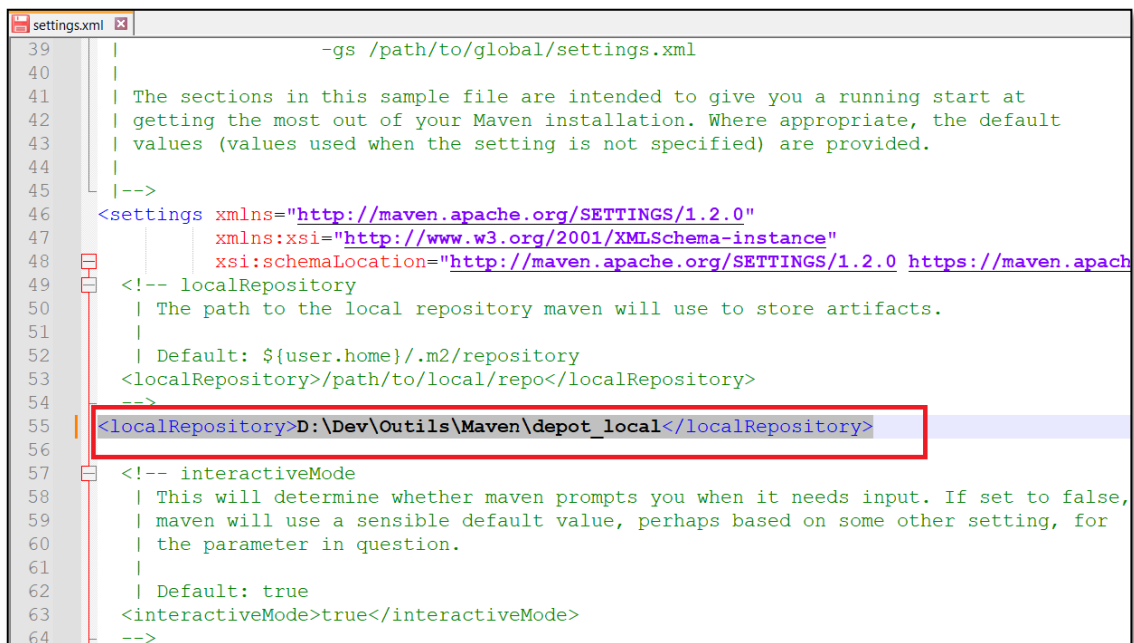


- c) Créer le sous dossier «**depot\_local**» sous :

**D:\Dev\Outils\Maven**



- d) Ouvrir le nouveau fichier «**settings.xml**» (copié sous «.m2 ») et ajouter la ligne sélectionnée :



```
39 | -gs /path/to/global/settings.xml
40 |
41 | The sections in this sample file are intended to give you a running start at
42 | getting the most out of your Maven installation. Where appropriate, the default
43 | values (values used when the setting is not specified) are provided.
44 |
45 | -->
46 | <settings xmlns="http://maven.apache.org/SETTINGS/1.2.0"
47 |   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
48 |   xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.2.0 https://maven.apache.org/SETTINGS/1.2.0.xsd">
49 |   <!-- localRepository
50 |    | The path to the local repository maven will use to store artifacts.
51 |    |
52 |    | Default: ${user.home}/.m2/repository
53 |    |>
54 |   <localRepository>/path/to/local/repo</localRepository>
55 |   <localRepository>D:\Dev\Outils\Maven\depot_local</localRepository>
56 |   <!-- interactiveMode
57 |    | This will determine whether maven prompts you when it needs input. If set to false,
58 |    | maven will use a sensible default value, perhaps based on some other setting, for
59 |    | the parameter in question.
60 |    |
61 |    | Default: true
62 |    |>
63 |   <interactiveMode>true</interactiveMode>
64 |   -->
```

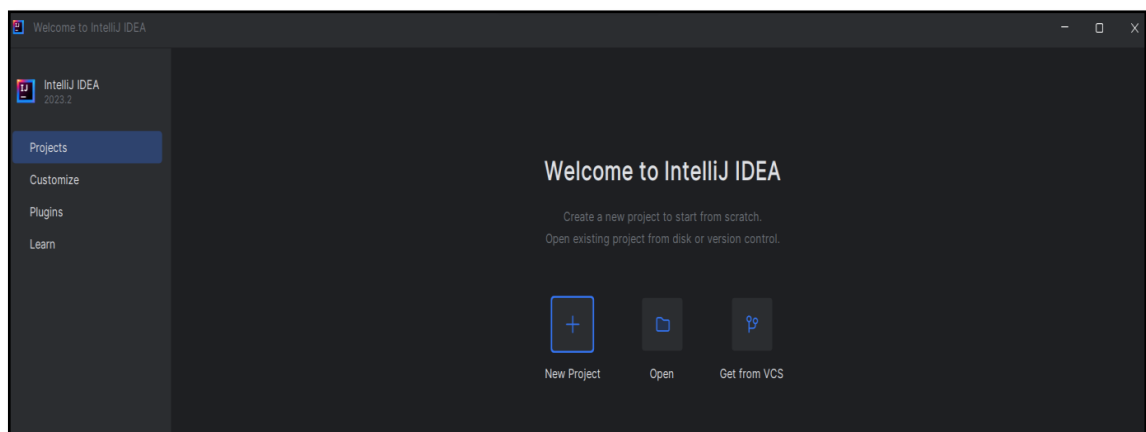
Ainsi le dépôt local de Maven est désormais :

**D:\Dev\Outils\Maven\depot\_local**

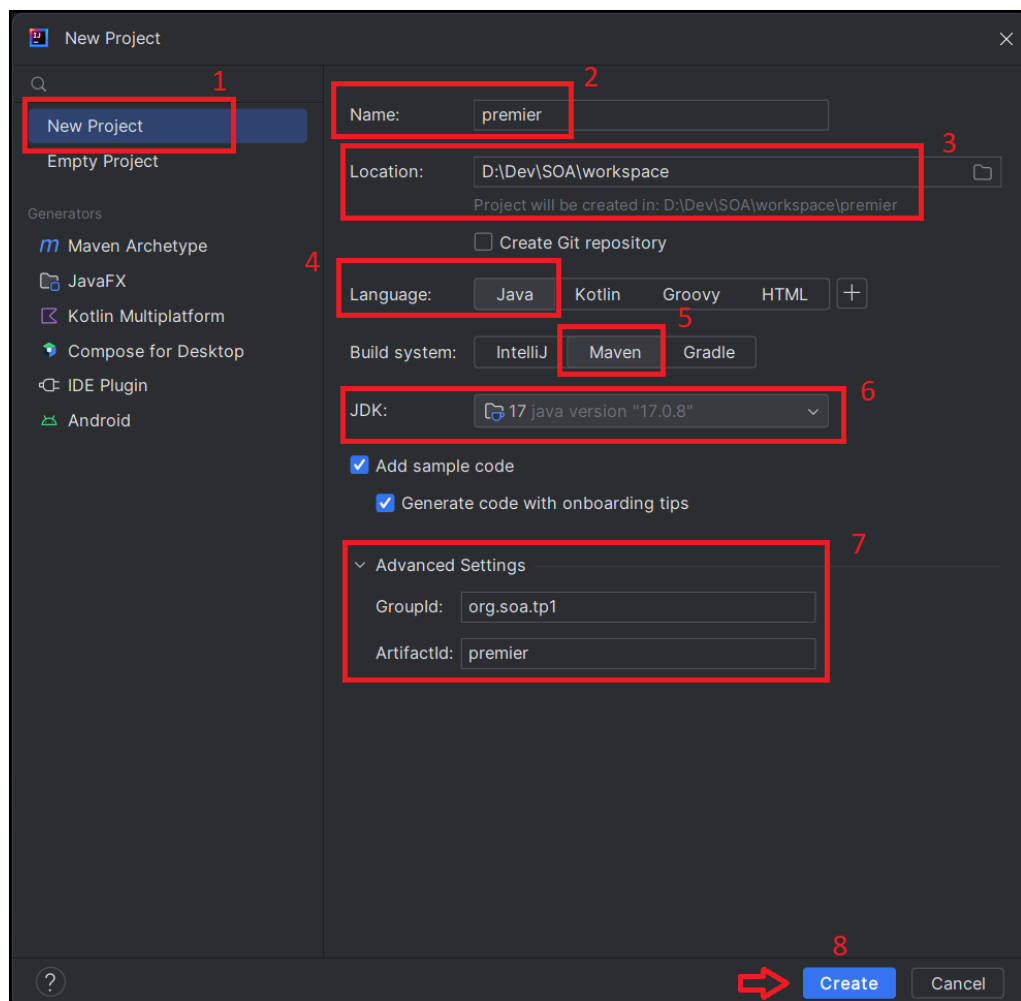
Désormais, tous les téléchargements à réaliser par «**Maven**» (dépendances, plugins..) seront stockés localement dans cet endroit.

## D. Créer un premier projet JAVA avec IntelliJ de type MAVEN

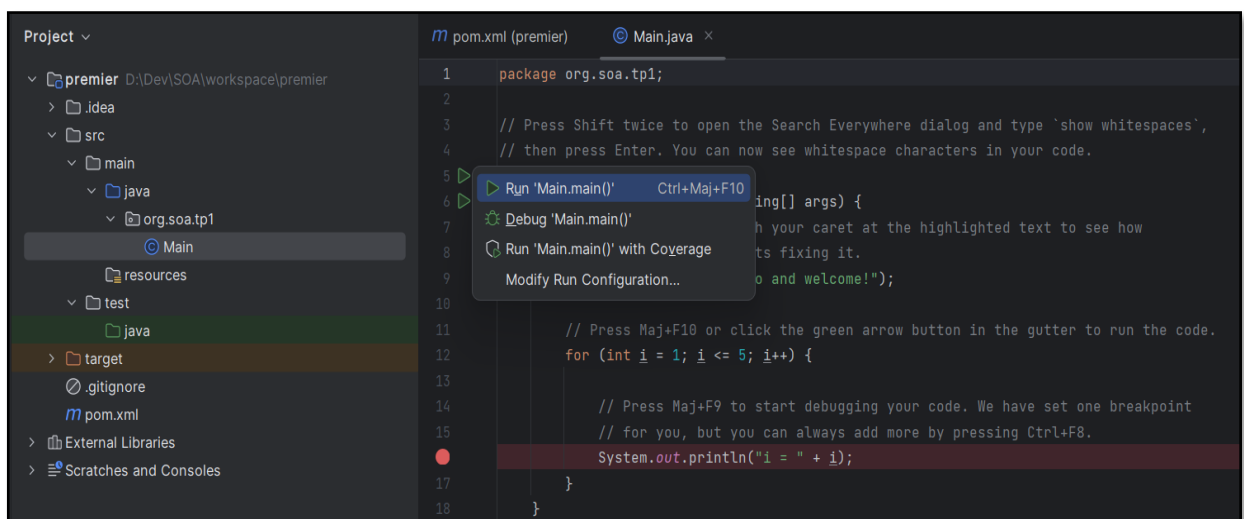
19. Revenons à IntelliJ pour créer un premier projet en choisissant l'option « **New Project** » :



20. Spécifier les paramètres du projet « **Maven** » comme suit et lancer la création :



21. Lancer l'exécution de la classe «**Main**» auto générée :

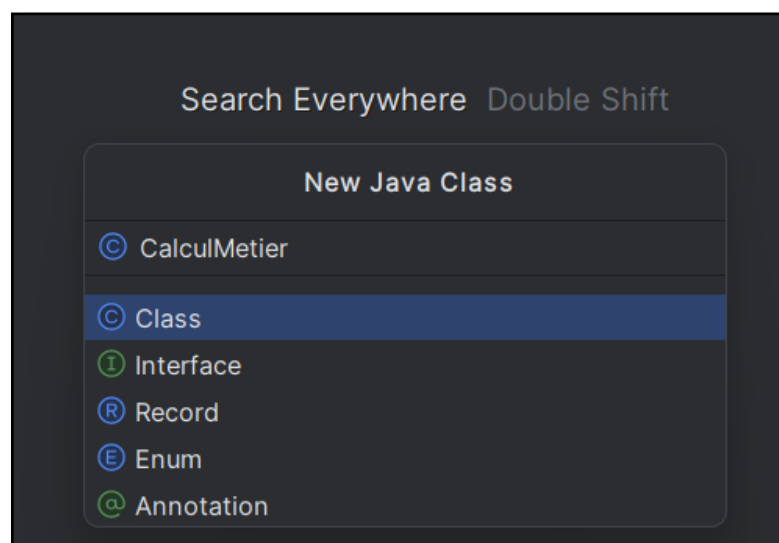
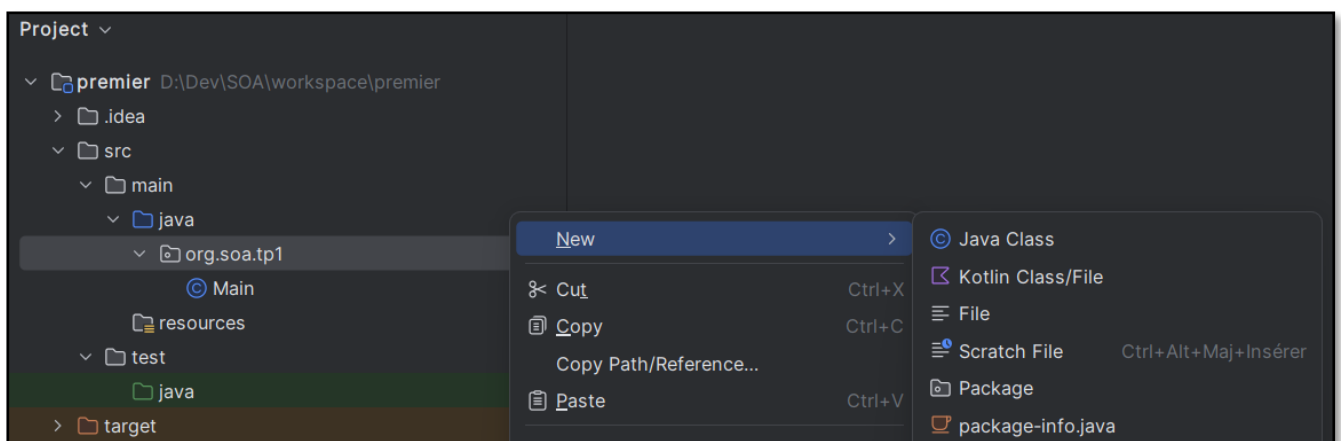


22. Visualiser le résultat dans le volet « **Run** » tout en bas :



```
Run Main x
:
"C:\Program Files\Java\jdk-17\bin\java.exe" -javaagent:D:\Dev\SOA\EDI\IntelliJ\lib\idea_rt.jar=5051:
Hello and welcome! i = 1
i = 2
i = 3
i = 4
i = 5
Process finished with exit code 0
```

23. Ajouter une classe «**CalculMetier**» dans le package «**org.soa.tp1**» :



24. Saisir le code suivant pour la classe «**CalculMetier**» :

```
package org.soa.tp1;

public class CalculMetier {
    public double somme(double a, double b)
    {
        return a + b;
    }

    public double produit(double a, double b)
    {
        return a * b;
    }
}
```

25. Changer le code de la classe de démarrage « **Main** » pour tester le fonctionnement des nouvelles méthodes «**somme**» et «**produit**» :

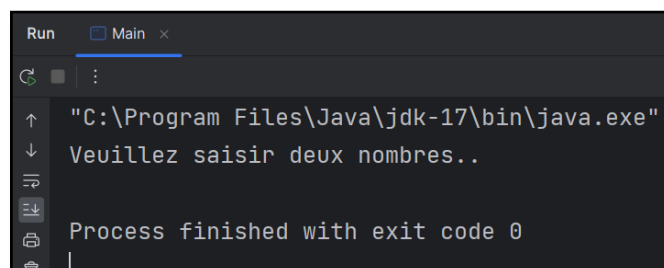
```
package org.soa.tp1;

public class Main {
    public static void main(String[] args) {

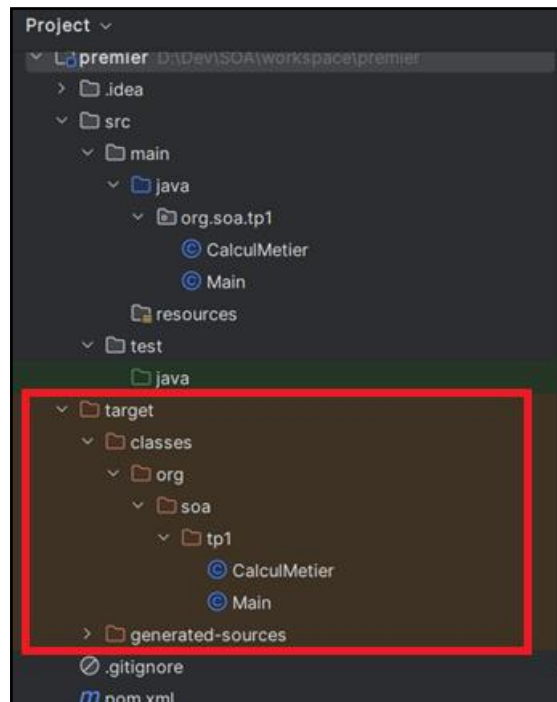
        if (args.length<2)
        {
            System.out.println("Veuillez saisir deux nombres..");
            System.exit(0);
        }
        double a = Double.parseDouble(args[0]);
        double b = Double.parseDouble(args[1]);
        CalculMetier cm =new CalculMetier();
        double s =cm.somme(a, b);
        double p = cm.produit(a, b);

        System.out.println("La somme de "+a+ "et "+b+ "est: "+s);
        System.out.println("Le produit de "+a+ "et "+b+ "est: "+p);
    }
}
```

26. Lancer l'exécution et remarquer que l'exécution n'est pas associée à des arguments (aucun argument n'est passé au programme «**Main**»):

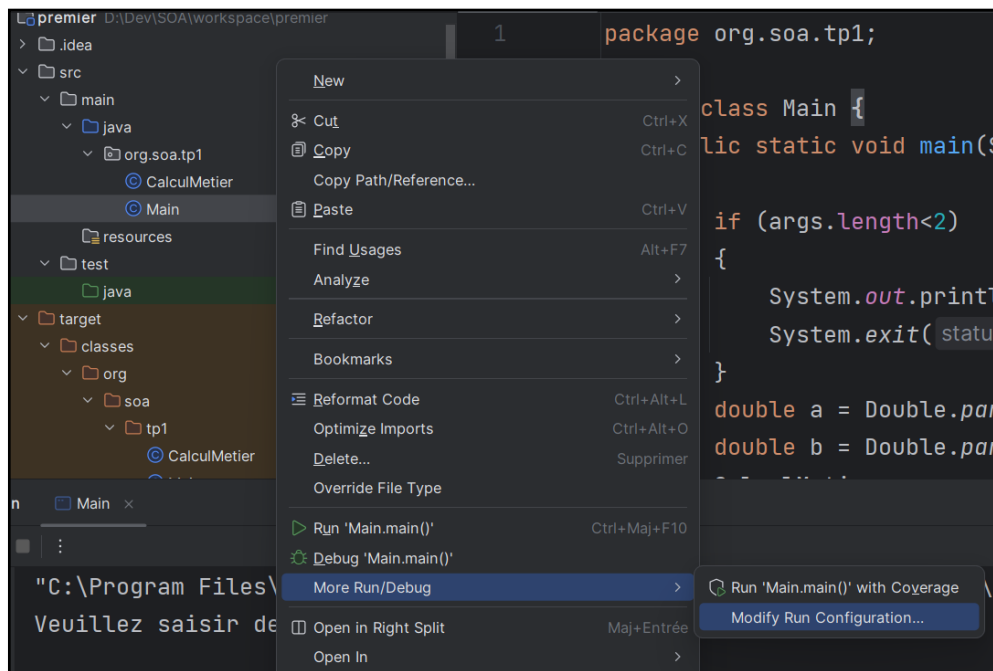


Remarquer, aussi, la génération d'un sous dossier «**target/classes**» qui regroupe les fichiers pré-compilés (.class) :



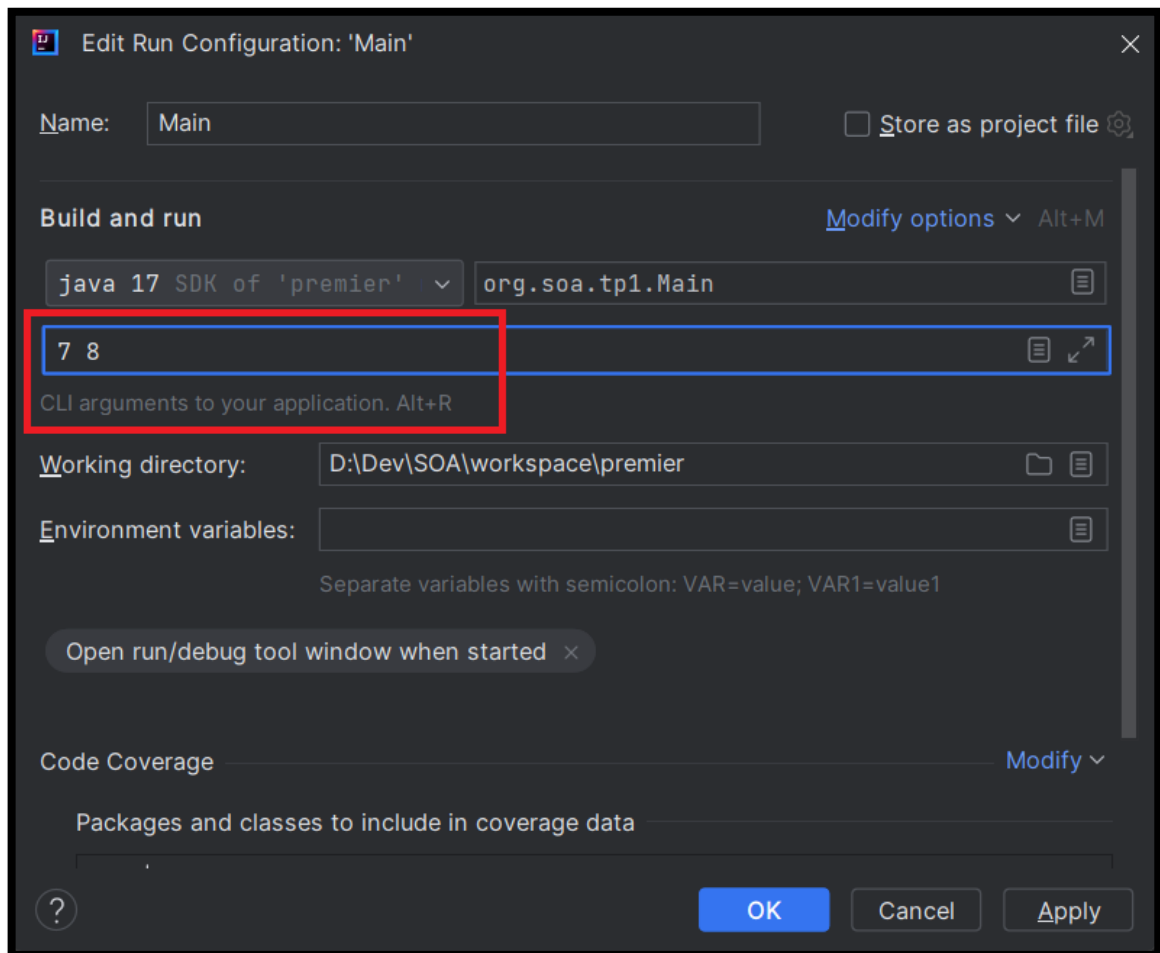
27. Pour passer des arguments pendant l'exécution :

- a. Sélectionner la classe « **Main** » et avec le bouton droit de la souris, choisir la commande « **Modify Run Configuration..** »

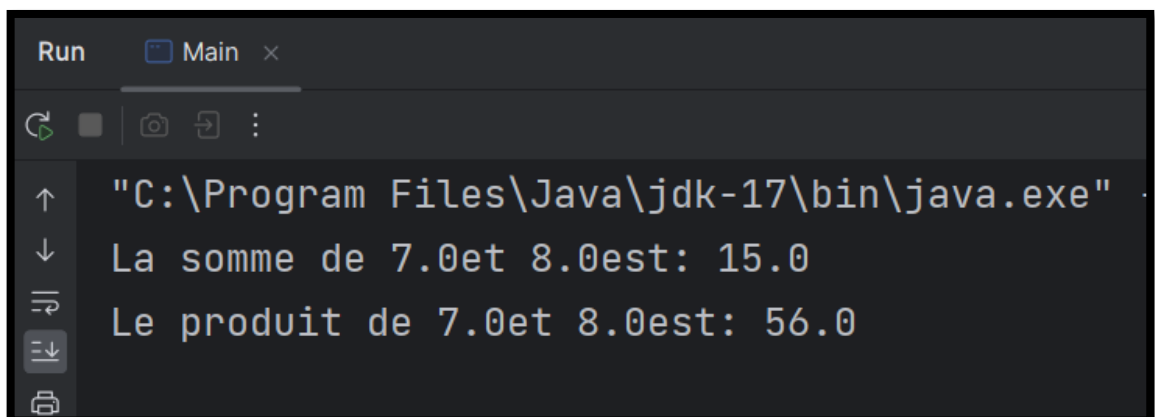




- b. Désigner, comme exemple, deux arguments: **7** et **8** séparés par espace comme suit :



- c. Relancer l'exécution et visualiser le résultat :



## E. Utiliser les commandes de base de Maven pour gérer le cycle de vie du projet

Dans le cycle de vie 'par défaut' d'un projet Maven, les phases les plus utilisées sont:

- **clean** : supprime tous les fichiers **Java .class** précédemment compilés
- **validate** : vérifie les prérequis d'un projet maven
- **compile** : compiler le code source
- **test** : compile et exécute des tests unitaires
- **package** : assemble le code compilé en un livrable
- **install** : partager le livrable pour d'autres projets sur le même ordinateur
- **deploy** : publier le livrable pour d'autres projets dans le « repository » distant
- Les phases s'exécutent de façon **séquentielle** de façon à ce qu'une phase dépende de la phase précédente.
- Par exemple, le lancement par l'utilisateur de la phase test (**mvn test**) impliquera le lancement préalable par maven des phases « **validate** » et « **compile** ».

28. Il est à noter qu'il est possible d'automatiser la procédure de test avec des tests unitaires en utilisant la bibliothèque **JUnit** et **Maven**. Pour se faire :
- a. Ajouter sous «**src/test/java**» un package «**org.soa.tp1**».
  - b. Ajouter dans le package «**org.soa.tp1**» une classe de test «**ClaculMetierTest**» ayant le code suivant :

```

package org.soa.tp1;

import static org.junit.Assert.*;

import org.junit.BeforeClass;
import org.junit.Test;

public class CalculMetierTest {
    //Déclarer une reference vers la classe CalculMetier
    private static CalculMetier metier;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        //Instancier la classe CalculMetier
        metier = new CalculMetier();
    }

    @Test
    public void testSomme ()
    {
        assertTrue(metier.somme(5, 2) == 7);
    }

    @Test
    public void testProduit ()
    {
        assertTrue(metier.produit(5, 2) == 10);
    }
}

```

- La méthode « **setUp()BeforeClass** » est exécutée avant tous les tests de la classe : Elle est annotée avec **@BeforeClass**
- Toutes les méthodes de test sont annotées par **@Test**
- La méthode **assertTrue( Boolean condition )** est une méthode statique de la classe « **org.junit.Assert** » et indique que le test est correct si la condition est correcte.

c. Remarquer que le code présente un ensemble **d'erreurs** dues à l'absence de la bibliothèque **JUnit** :

- Il est donc nécessaire de déclarer dans le fichier «**pom.xml**» la dépendance de **JUnit** que Maven l'apporte à notre projet.
- Pour cela ajouter le code **en gras** ci-dessous au fichier « **pom.xml** » à l'intérieur de la balise <project>:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

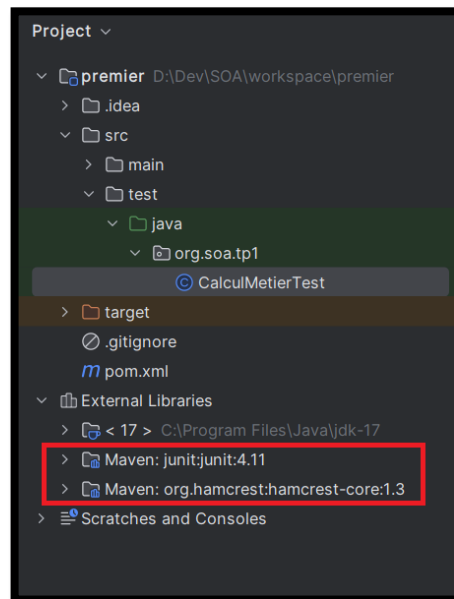
  <groupId>org.soa.tp1</groupId>
  <artifactId>premier</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <b><dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- d. Enregistrer** les modifications et **charger** les changements MAVEN en appuyant sur le bouton situé en haut à droite dans le fichier « **pom.xml** » :

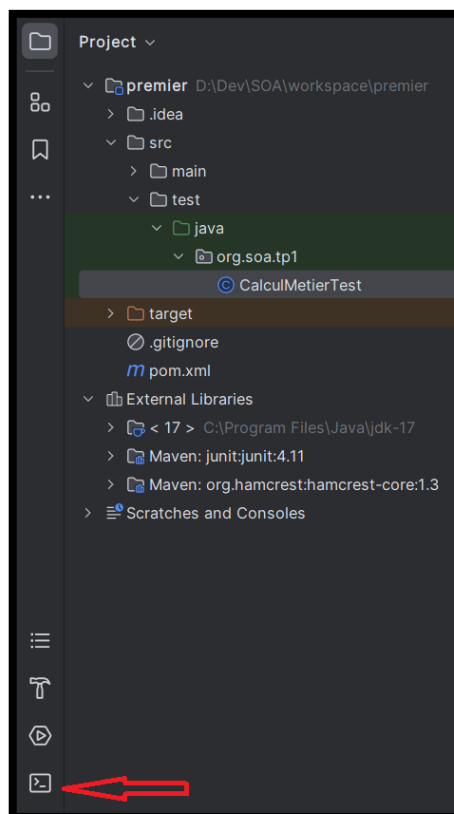


- e. Remarquer la correction des erreurs dans la classe «**CalculMetierTest**» et l'ajout de deux bibliothèques relatives à la dépendance MAVEN de JUnit :

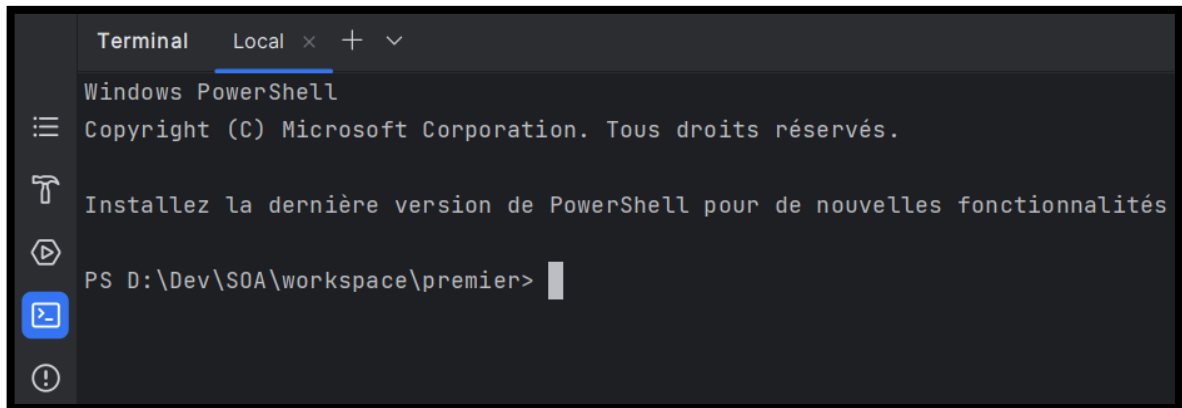


- Gérer le cycle de vie du projet dans le « Terminal »

29. Ouvrir le volet de «Terminal» dans IntelliJ :



Une fenêtre d'invite de commandes s'ouvre en se positionnant sous le dossier de notre projet «**premier**» :



```
Terminal Local x + v
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités

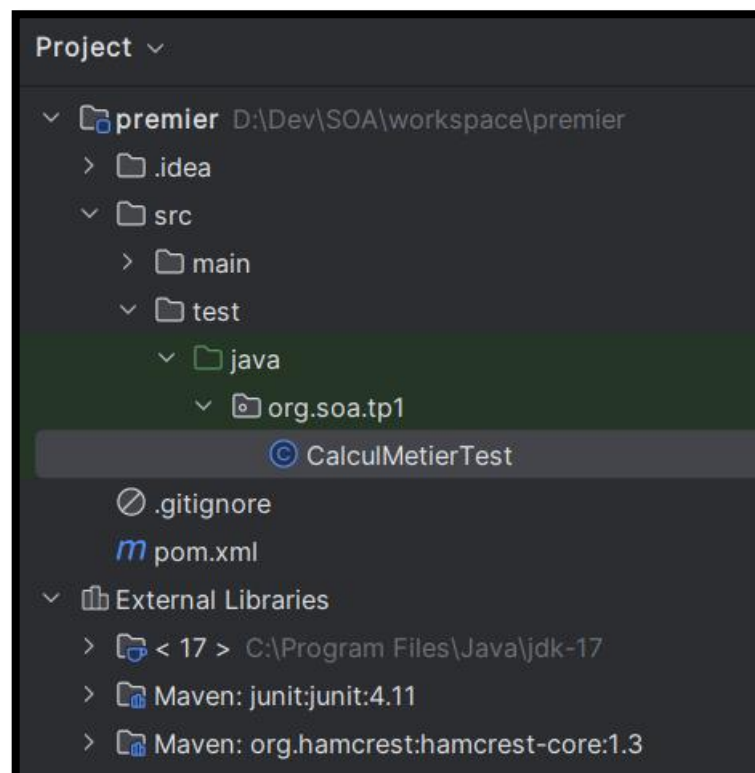
PS D:\Dev\SOA\workspace\premier> |
```

### 30. Commande «**clean**» :

Taper la commande **clean** pour supprimer tous les fichiers **.class** récemment compilés (nettoyer le projet) :

**mvn clean**

Ceci permet de supprimer le sous-dossier « **target** » :



### 31. Commande «**validate**» :

Taper la commande **validate** pour vérifier que la configuration projet est correcte (pas d'éléments manquants...)

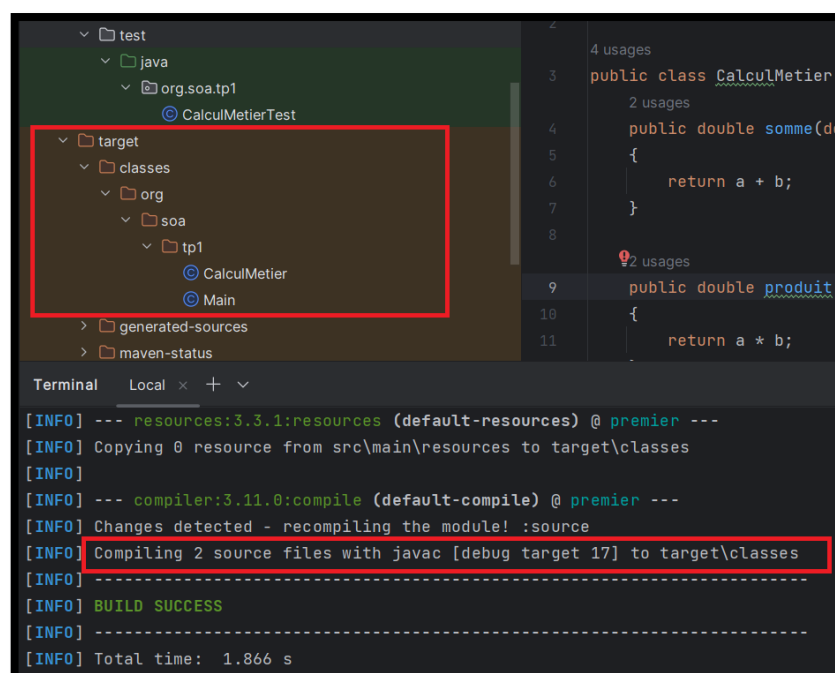
**mvn validate**

```
PS D:\Dev\SOA\workspace\premier> mvn validate
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.soa.tp1:premier >-----
[INFO] Building premier 1.0-SNAPSHOT
[INFO]    from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time:  0.110 s
```

### 32. Commande «**compile**» :

Taper la commande **compile** pour compiler les fichiers sources (et non les fichiers de test) puis générer les fichiers **.class** sous un sous dossier «**/target/classes**» :

**mvn compile**



The screenshot shows an IDE with a project structure on the left and a terminal window at the bottom. The project structure includes a `test` directory with `java` and `org.soa.tp1` packages, and a `target` directory with `classes`, `org`, `soa`, and `tp1` packages. The `target/classes` directory is highlighted with a red box. The terminal window shows the output of the `mvn compile` command, which includes the following lines:

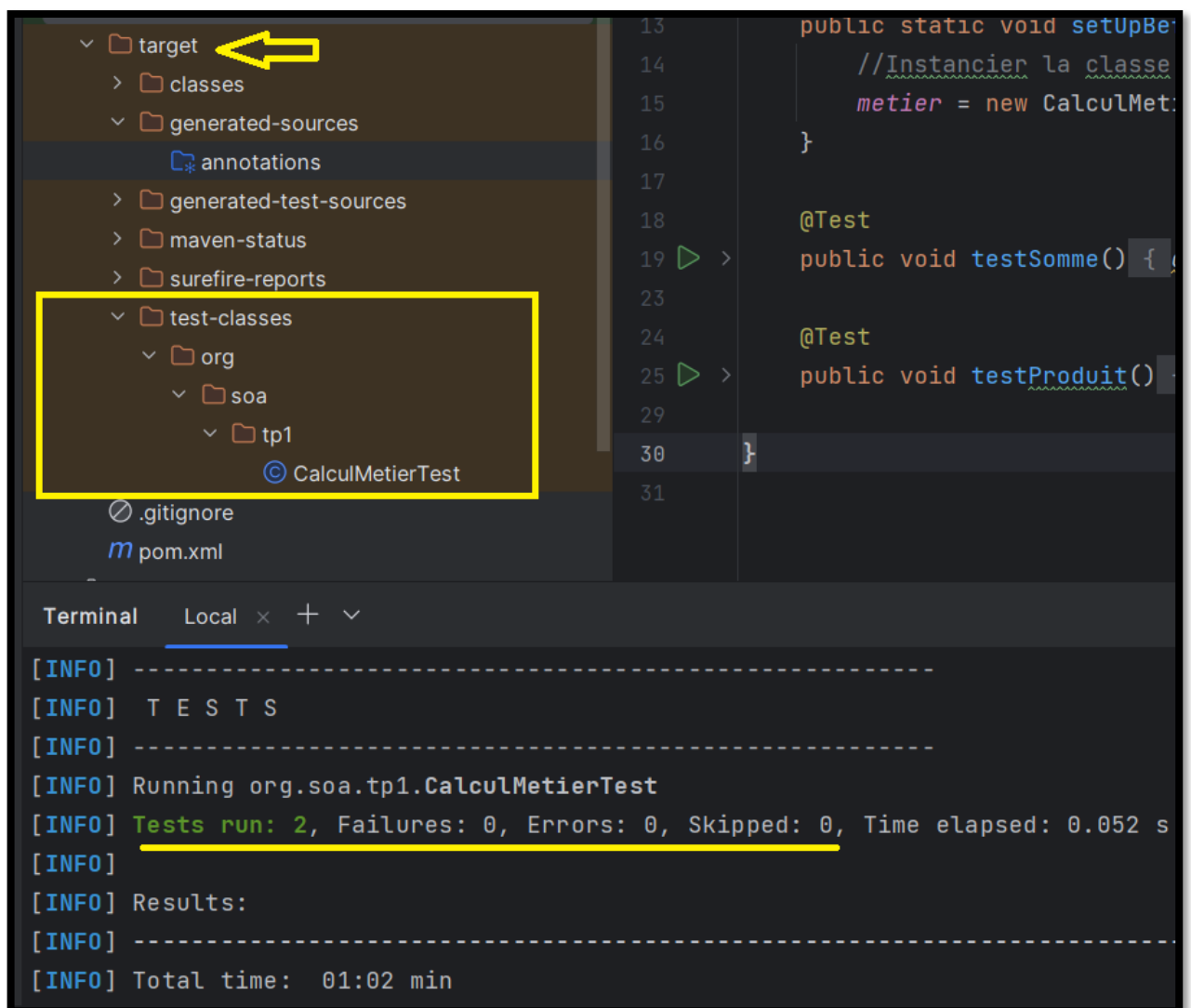
```
[INFO] --- resources:3.3.1:resources (default-resources) @ premier ---
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ premier ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 2 source files with javac [debug target 17] to target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time:  1.866 s
```

### 33. Commande «test» :

- a. Taper la commande **compile** pour compiler et exécuter la classe de test unitaire :

**mvn test**

- b. Remarquer que le test est réalisé avec succès (sans erreur) et une classe précompilée est générée pour la classe de test sous «**target/test-classes**» dans un package « **org.soa.tp1** » :



- c. Modifier la classe «**CalculMetier**» pour avoir un fonctionnement erroné des méthodes «**somme**» et «**produit**». Prendre l'exemple suivant :



```

package org.soa.tp1;

public class CalculMetier
{
    public double somme(double a, double b)
    {
        return a - b; // au lieu de a+b
    }

    public double produit(double a, double b)
    {
        return a / b; // au lieu de a*b
    }
}

```

- d. Relancer la compilation et le test : Pas de problème au niveau de compilation. Mais, au niveau de la phase de test, deux erreurs ont été déclenchées :

```

[INFO]
[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR]   CalculMetierTest.testProduit:27
[ERROR]   CalculMetierTest.testSomme:21
[INFO]
[ERROR] Tests run: 2, Failures: 2, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----

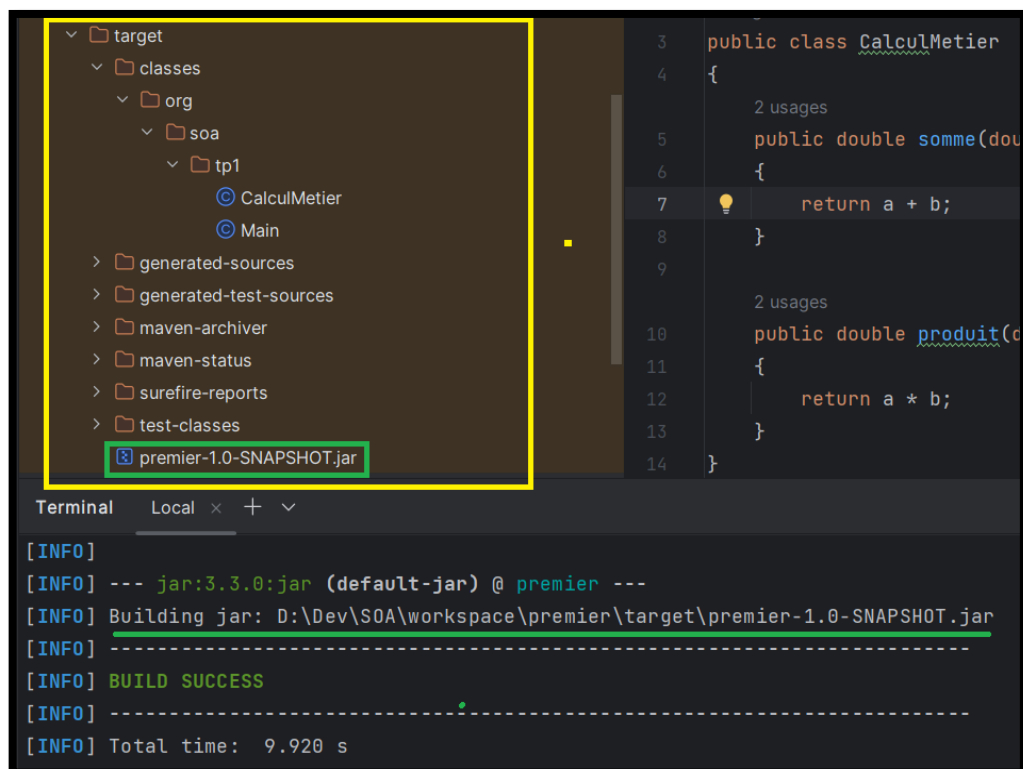
```

- e. Reprendre la version correcte de la classe « **CalculMetier** » et refaire la compilation et le test.

### 34. Commande « **package** » :

- a. Taper la commande **package** pour archiver le projet et le packager dans un format distribuable (JAR, WAR, ...). Remarquer la création d'un package d'archive «**premier-1.0-SNAPSHOT.jar**» directement dans le sous-dossier « **target** » :

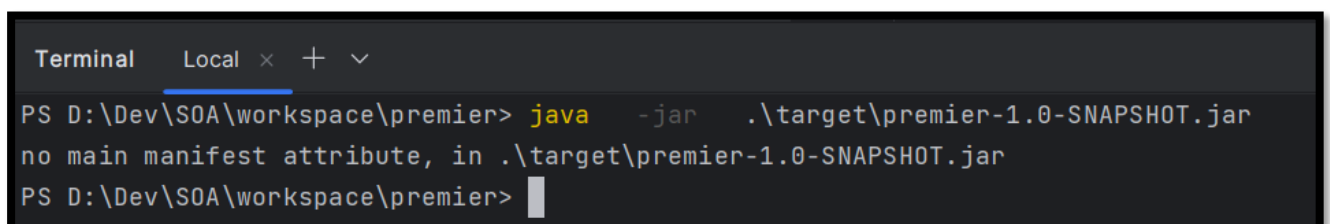
**mvn package**



- b. Essayer le lancer l'exécution du programme à travers le fichier d'archive nouvellement créé à travers la commande suivante :

**java -jar .\target\premier-1.0-SNAPSHOT.jar**

Une erreur est déclenchée pour signaler qu'il est impossible de détecter une classe de démarrage ayant une méthode main :



- c. Pour remédier à ce problème, éditer le fichier «**pom.xml**» et ajouter une configuration pour le plugin «**jar**» pour spécifier la classe main (classe de démarrage) comme suit avec le caractère en **gras** :

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
      <configuration>
        <archive>
          <manifest>
            <mainClass>org.soa.tp1.Main</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

- d. Réaliser successivement les actions suivantes :

- Enregistrer les modifications
- Lancer le chargement des changements de Maven
- Refaire la phase d'archivage (**mvn package**)
- saisir la commande suivante:

**java -jar .\target\premier-1.0-SNAPSHOT.jar**

```
PS D:\Dev\SOA\workspace\premier> java -jar .\target\premier-1.0-SNAPSHOT.jar
Veuillez saisir deux nombres..
```

- e. Passer, comme exemple, les deux arguments **5** et **6** :

```
PS D:\Dev\SOA\workspace\premier> java -jar .\target\premier-1.0-SNAPSHOT.jar 5 6
La somme de 5.0et 6.0est: 11.0
Le produit de 5.0et 6.0est: 30.0
```

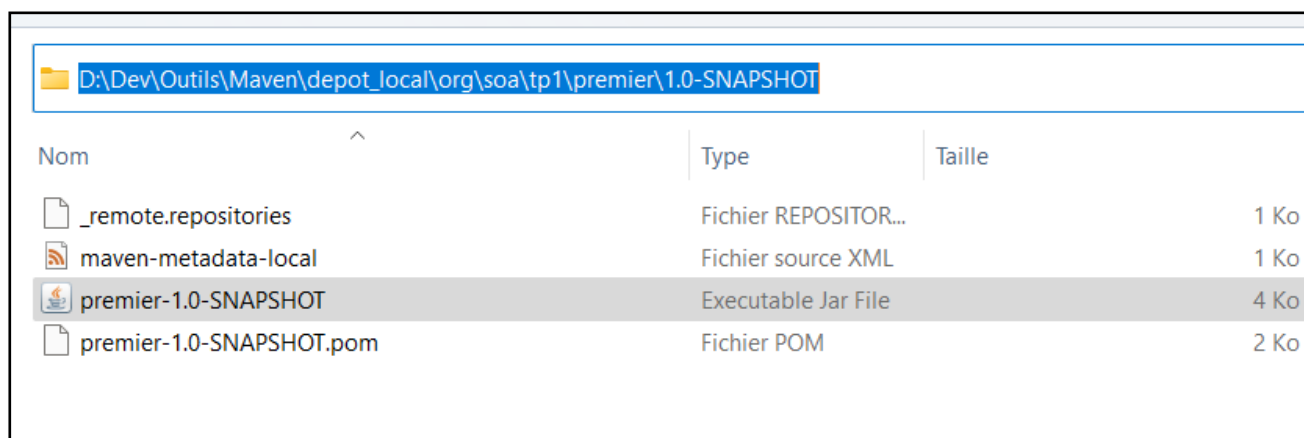
### 35. Commande «**install**» :

Taper la commande **install** pour livrer l'archive du projet dans le dossier local du Maven :

**mvn install**

```
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ premier ---
[INFO] Installing D:\Dev\SOA\workspace\premier\pom.xml to D:\Dev\Outils\Maven\depot_local\org\soa\tp1\premier\1.0-SNAPSHOT\premier-1.0-SNAPSHOT.pom
[INFO] Installing D:\Dev\SOA\workspace\premier\target\premier-1.0-SNAPSHOT.jar to D:\Dev\Outils\Maven\depot_local\org\soa\tp1\premier\1.0-SNAPSHOT\premier-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.205 s
```

Remarquer la livraison de l'archive du projet dans le repository local de Maven :



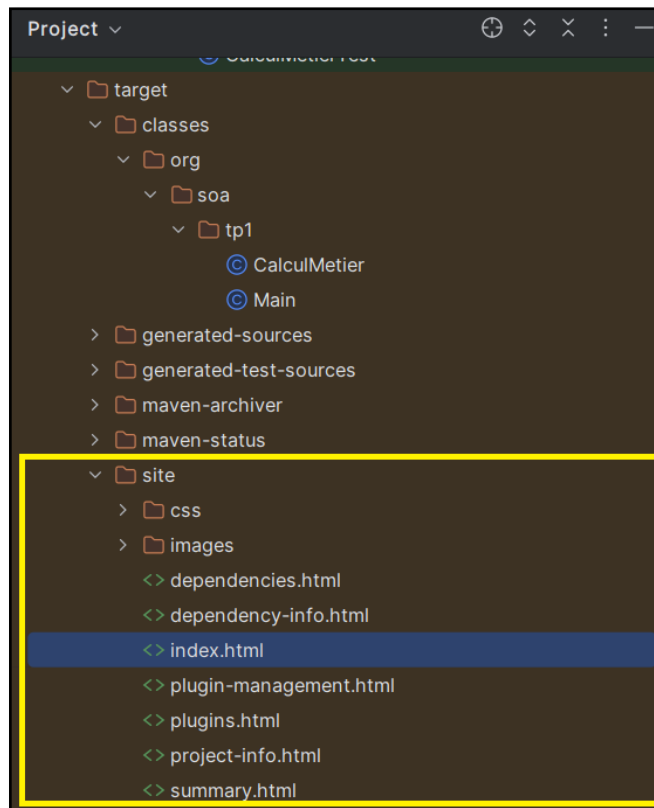
Nom	Type	Taille
_remote.repositories	Fichier REPOSITOR...	1 Ko
maven-metadata-local	Fichier source XML	1 Ko
premier-1.0-SNAPSHOT	Executable Jar File	4 Ko
premier-1.0-SNAPSHOT.pom	Fichier POM	2 Ko

### 36. Commande «**site**» :

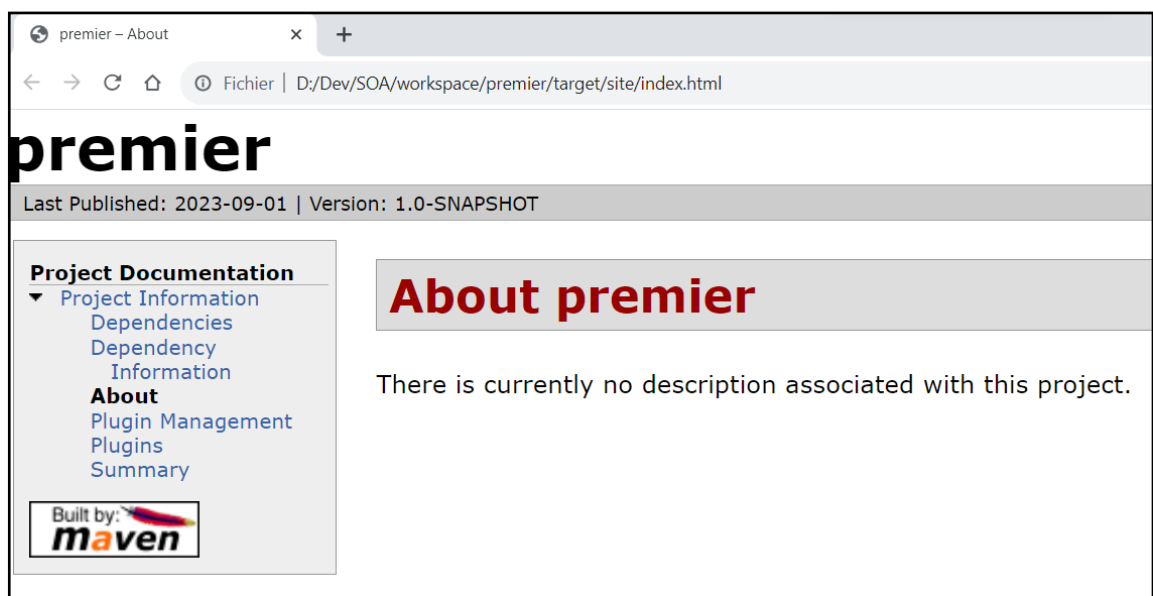
Taper la commande **site** pour générer un site pour le projet. Ce site inclut également les rapports du projet qui ont été configurés dans le fichier « **pom.xml** »

**mvn site**

Remarquer la création d'un sous-dossier « **/target/site** » qui regroupe des pages HTML décrivant le projet :



Ouvrir le fichier `index.html` avec un navigateur web pour consulter la page d'accueil du site du projet « **premier** » :

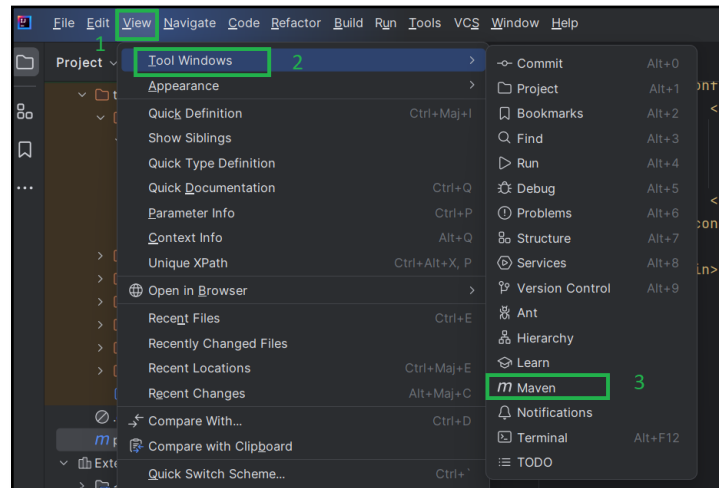


Le volet de navigation présente des liens pour décrire les dépendances, les plugins et les informations relatives au projet.

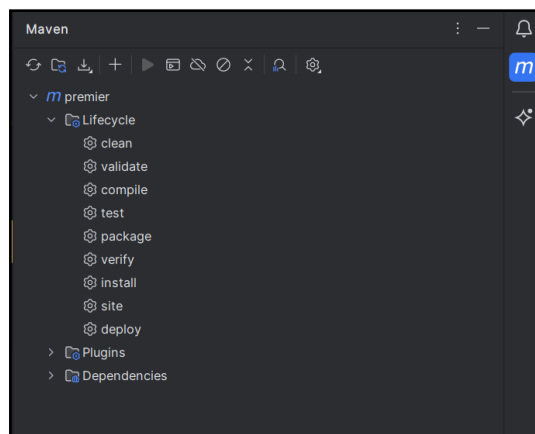
## •Gérer le cycle de vie du projet dans Graphiquement

37. Toutes ces commandes Maven sont accessibles graphiquement par l'EDI **IntelliJ** :

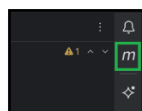
- Accéder au menu « **View** »
- Choisir « **Tool Windows** »
- Choisir « **Maven** » :



Ainsi une boîte à outils s'affiche pour utiliser les commandes Maven désirées :



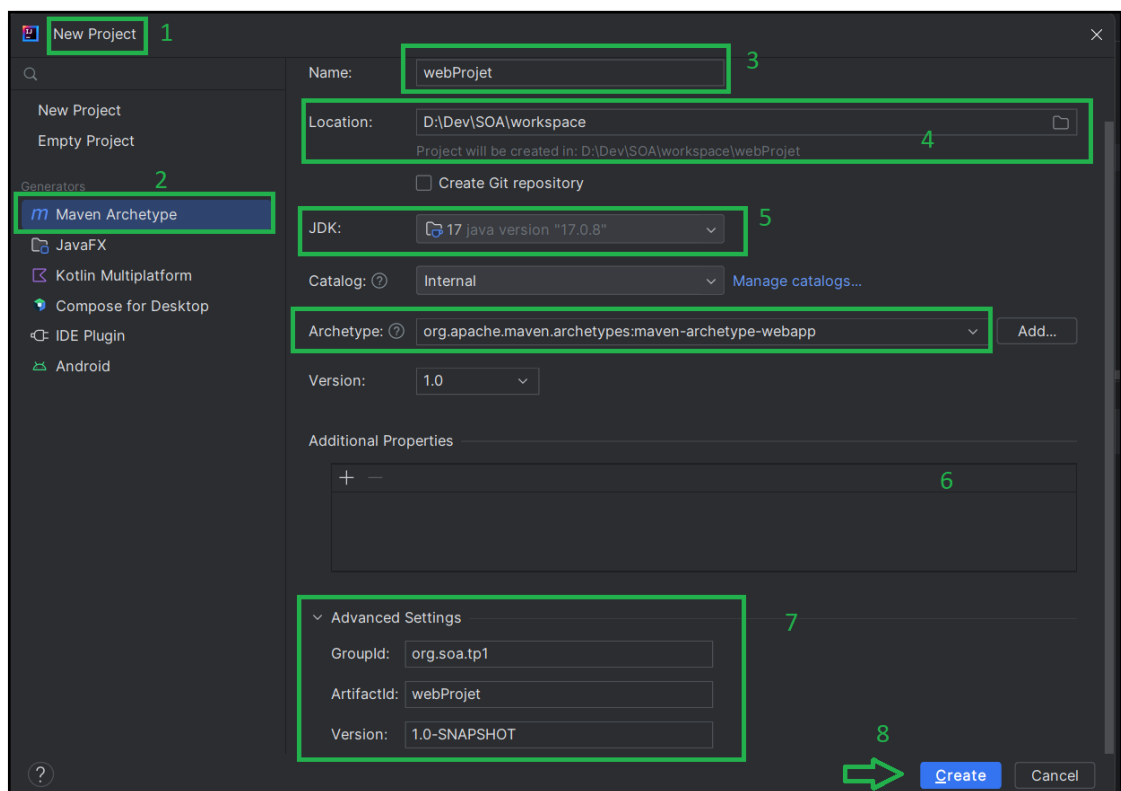
Ces commandes sont aussi accessibles à travers le bouton à droit en haut :



## F. Déployer une application web avec Tomcat en utilisant Maven

38. Créer une application web avec un modèle (archetype) Maven et spécifier les paramètres du projet :

- **groupId** : **org.soa.tp1**
- **artifactId** : **webProjet**
- **version** : **0.0.1-SNAPSHOT**
- **package** : **org.soa.tp1** (même valeur que « **groupId** »)

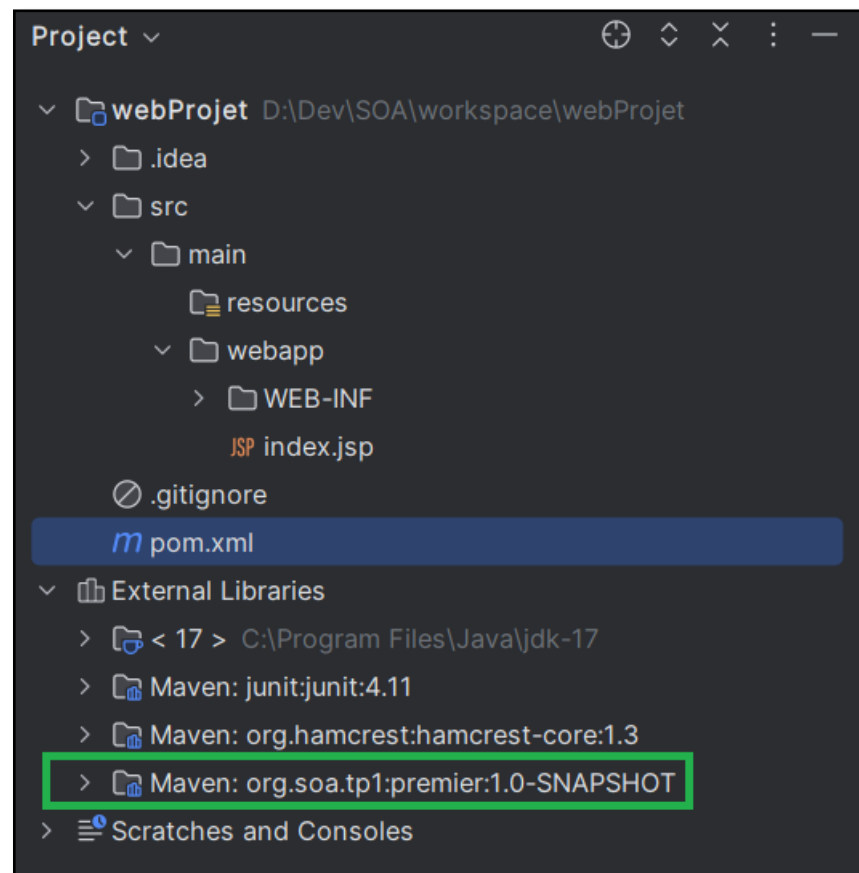


39. Ce nouveau projet dépend de notre premier projet «**premierProjet**» (càd utilise ses classes). Pour cela, ouvrir le fichier « **pom.xml** » du projet « **webProjet** » pour ajouter la dépendance suivante à l'intérieur des deux balises **<dependencies>** et **</dependencies>** suite à la dépendance de « **junit** »:

```
<dependency>
  <groupId>org.soa.tp1</groupId>
  <artifactId>premier</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

**NB : Ne pas oublier d'enregistrer et synchroniser Maven après chaque modification du pom.xml**

40. Remarquer l'ajout de la nouvelle bibliothèque :



41. Modifier le fichier «**index .jsp**» du projet et qui existe sous «**src/main/webapp**» comme suit :

```
<%@page import="org.soa.tp1.CalculMetier"
    contentType="text/html; charset=UTF-8" %>
<% double a= 0.0; double b= 0.0; double res= 0.0;
String action = request.getParameter("action");
if (action!= null) {
    a = Double.parseDouble(request.getParameter("a"));
    b = Double.parseDouble(request.getParameter("b"));
    CalculMetier metier = new CalculMetier();
    if (action.equals ("somme")){
        res= metier.somme(a,b);
    }
    else {
        res= metier.produit(a,b);
    }
}
%>
<html>
<body>
<form action ="index.jsp" method ="post">
<table>
```



```

<tr>
  <td> A: </td> <td> <input type ="text" name ="a" value ="<%=a%>" /> </td>
  <td> B: </td> <td> <input type ="text" name ="b" value ="<%=b%>" /></td>
</tr>
<tr>
  <td> <input type ="submit" name ="action" value ="somme" /></td>
  <td> <input type ="submit" name ="action" value ="produit"/> </td>
</tr>
<tr> <td> Résultat : </td> <td> <%=res%></td> </tr>
</table>
</form>
</body>
</html>

```

42. Lancer la commande Maven « **install** » :

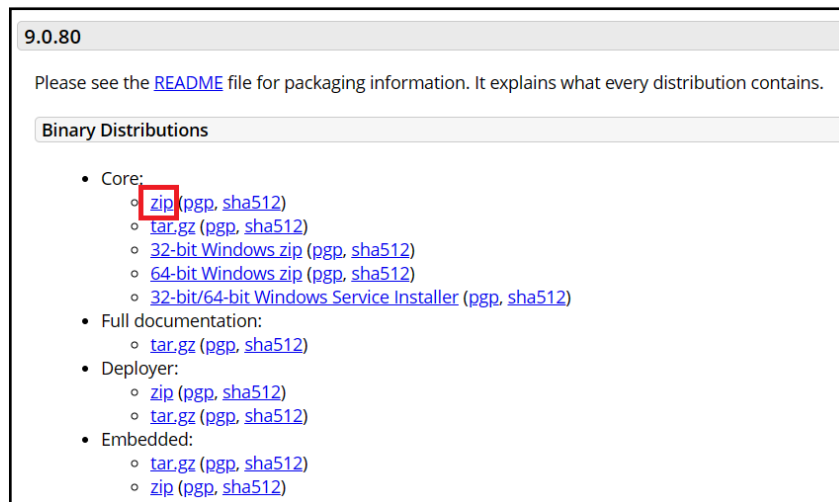


Remarquer la livraison de l'archive du projet (au format WAR) dans le dépôt local de Maven :

D:\Dev\Outils\Maven\depot_local\org\soa\tp1\webProjet\1.0-SNAPSHOT		
Nom	Type	Taille
_remote.repositories	Fichier REPOSITOR...	
maven-metadata-local	Fichier source XML	
webProjet-1.0-SNAPSHOT.pom	Fichier POM	
webProjet-1.0-SNAPSHOT.war	Fichier WAR	

43. Passons maintenant à configurer le serveur web «**Tomcat**» (version 9). Accéder à l'url suivante :

**<https://tomcat.apache.org/download-90.cgi>**



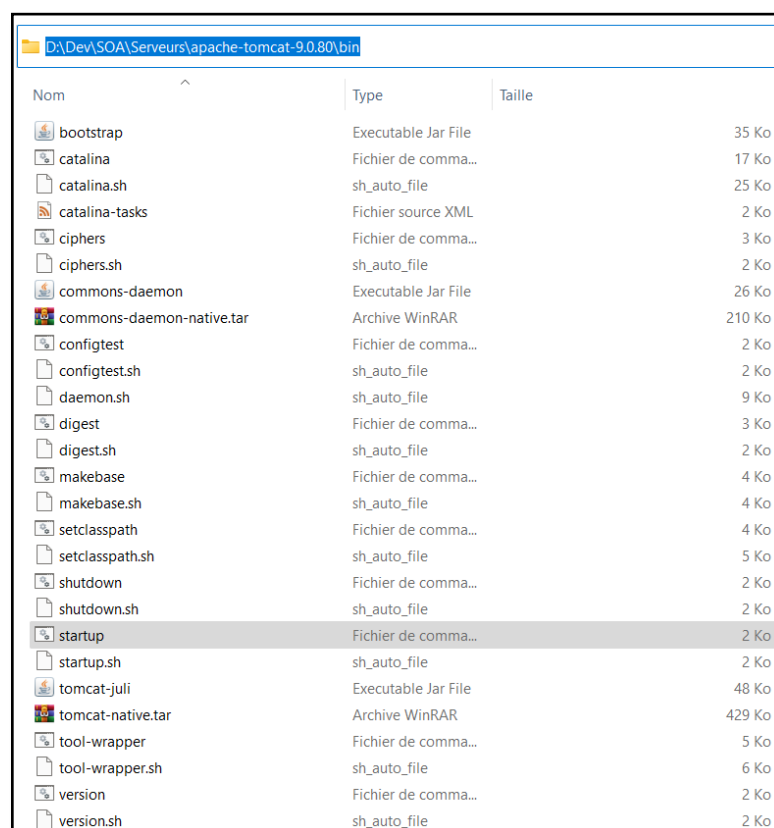
44. Télécharger l'archive (zip) du « **Tomcat** » dans le dossier :

**D:\Dev\SOA\Outils**

45. Extraire l'archive du « **Tomcat** » dans le dossier :

**D:\Dev\SOA\Serveurs**

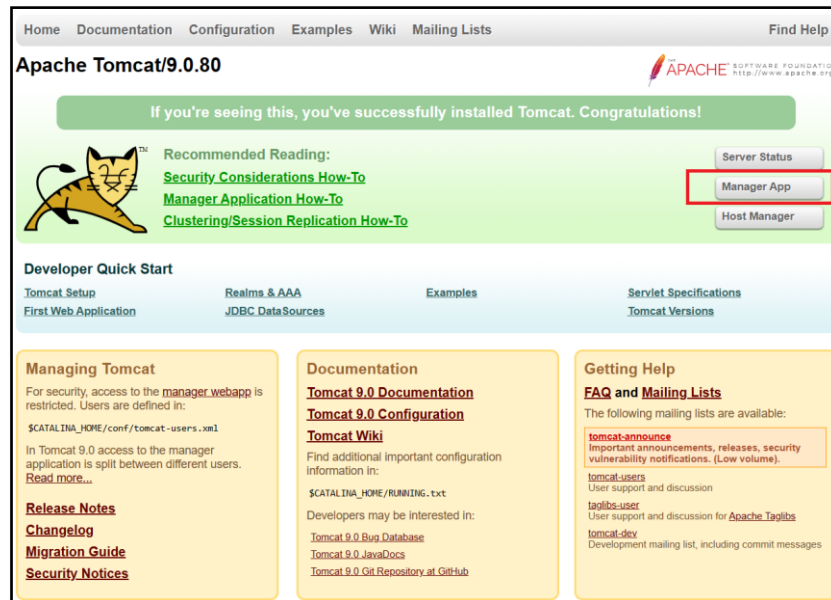
46. Accéder au dossier « **bin** » de Tomcat et lancer programme « **startup.bat** » :



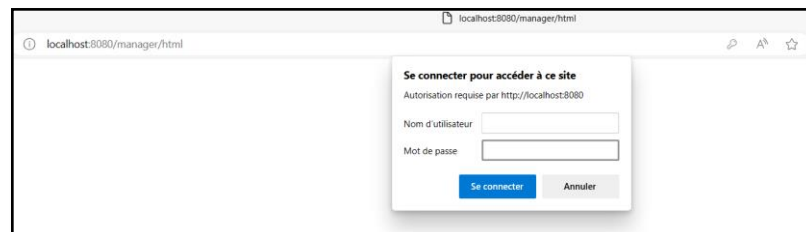
47. Accéder à travers le navigateur web au serveur par le lien :

**http://localhost:8080**

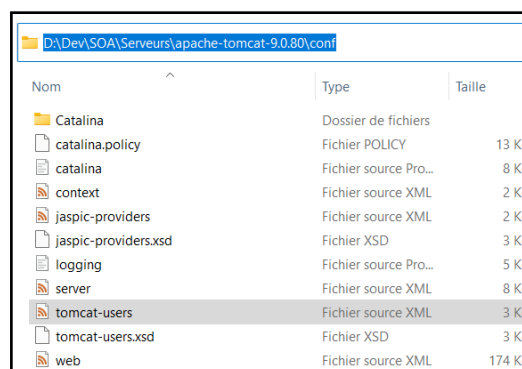
48. Voici la page d'accueil de Tomcat :



49. Accéder au volet d'administration « **Manager App** ». Tomcat nécessite une authentification avec un nom d'utilisateur et un mot de passe :



50. Pour se faire, accéder au sous dossier « **conf** » de Tomcat et éditer le fichier « **tomcat-users.xml** » :



51. Ajouter, entre les deux balises `<tomcat-users>` et `</tomcat-users>`, les instructions suivantes pour définir un utilisateur (**admin**, **admin**) ayant les droits nécessaires pour gérer les applications web :

```
<role rolename="manager-gui"/>

<role rolename="admin-gui"/>

<role rolename="manager-script"/>

<user username="admin" password="admin" roles="manager-gui,admin-gui, manager-script"/>
```

52. Enregistrer le fichier puis Arrêter le serveur « **Tomcat** » en cliquant sur le fichier « **shutdown.bat** » (dans le sous-dossier « **bin** »).

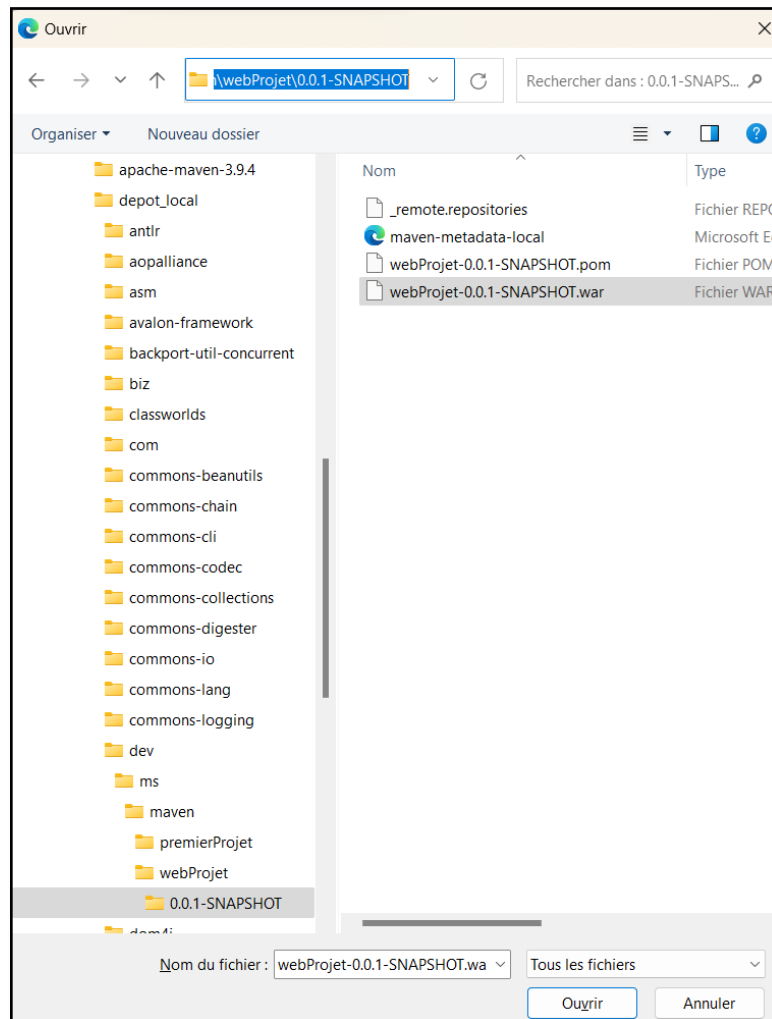
53. Redémarrer « **Tomcat** » (en cliquant sur le fichier **startup.bat**) et réaccéder à l'interface web d'administration de tomcat pour s'authentifier avec les deux paramètres (« **admin** » , « **admin** ») et gérer les applications déployées :

The screenshot shows the Tomcat Manager web interface at localhost:8080/manager/html. It features a 'Gestionnaire' (Manager) section with links for 'Lister les applications', 'Aide HTML Gestionnaire', and 'Aide Gestionnaire'. Below this is a table of 'Applications' with columns for 'Chemin', 'Version', 'Nom d'affichage', 'Fonctionnelle', 'Sessions', and 'Commandes'. The table lists several applications including 'Welcome to Tomcat', 'Tomcat Documentation', 'Servlet and JSP Examples', 'Tomcat Host Manager Application', and 'Tomcat Manager Application'. Each application has a set of control buttons (Démarrer, Arrêter, Recharger, Retirer) and a link to 'Explorer les sessions'. At the bottom, there is a 'Deployer' section with input fields for 'Chemin de contexte (requis)', 'Version (pour les déploiements en parallèle)', 'URL du fichier XML de configuration', and 'URL vers WAR ou répertoire', followed by a 'Deployer' button.

54. Accéder dans l'interface web ainsi ouverte au volet « **Déployer \Fichier WAR à déployer** » pour déployer l'application web qui est déjà livrée dans le repository local.

This screenshot is a closer view of the 'Deployer' section of the Tomcat Manager interface. It shows the 'Emplacement du répertoire ou fichier WAR de déploiement sur le serveur' section with input fields for 'Chemin de contexte (requis)', 'Version (pour les déploiements en parallèle)', 'URL du fichier XML de configuration', and 'URL vers WAR ou répertoire', along with a 'Deployer' button. Below this, the 'Fichier WAR à déployer' section is highlighted with a red box. It contains a label 'Choisir le fichier WAR à téléverser', a 'Choisir un fichier' button, and a 'Deployer' button. The text 'Aucun fichier choisi' is also visible.

55. Sélectionner le fichier WAR du projet « **webProjet** » déjà installé dans le dépôt local de Maven :

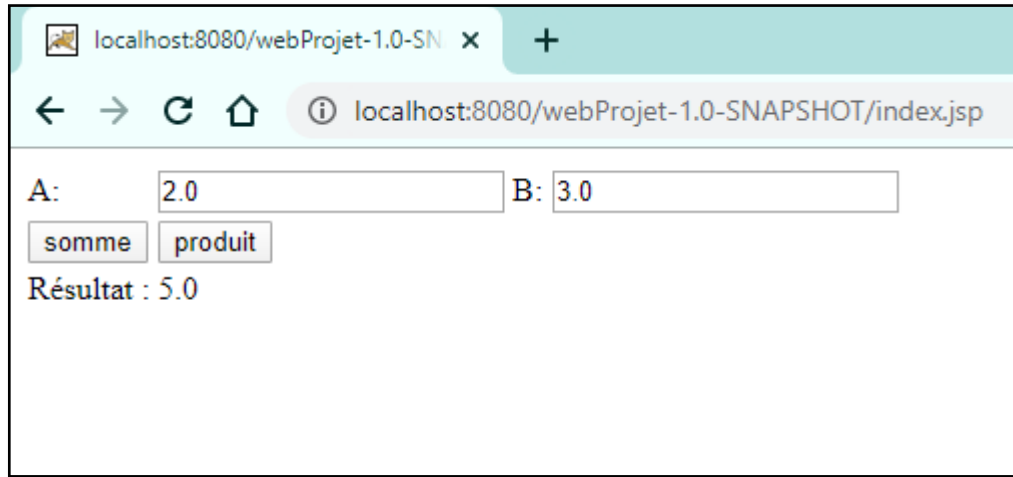


56. Cliquer ensuite sur « **Déployer** » pour avoir le projet web déployé dans la liste des applications prises en compte par le serveur Tomcat :

Applications					
Chemin	Version	Nom d'affichage	Fonctionnelle	Sessions	Commandes
/	Aucun spécifié	Welcome to Tomcat	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/docs	Aucun spécifié	Tomcat Documentation	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/examples	Aucun spécifié	Servlet and JSP Examples	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/host-manager	Aucun spécifié	Tomcat Host Manager Application	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/manager	Aucun spécifié	Tomcat Manager Application	true	1	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes
/webProjet-0.0.1-SNAPSHOT	Aucun spécifié	Archetype Created Web Application	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes

57. Ainsi le projet WEB est déployé il suffit de cliquer sur le lien correspondant pour lancer son exécution.

Voici le résultat :



#### 58. Exercice

Réaliser les modifications nécessaires pour que la page **index.jsp** affiche aussi un troisième bouton «**Division**» pour réalisation une opération **a / b**