

Atelier Framework Spring-01

Maven-Partie-04

Objectifs

Gestion de dépendances avec Maven

1. Déclarer une dépendance

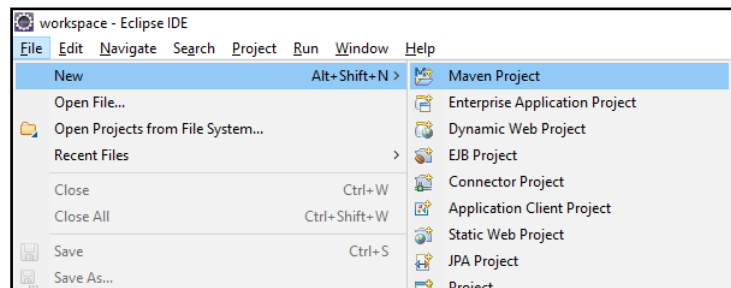
- Créer un projet Maven Avec « eclipse »
- Editer le fichier « pom.xml »
- Réaliser une dépendance à partir du dépôt central (distant)
- Réaliser une dépendance avec un projet livré dans le dépôt local

2. Exécuter et déployer une application WEB

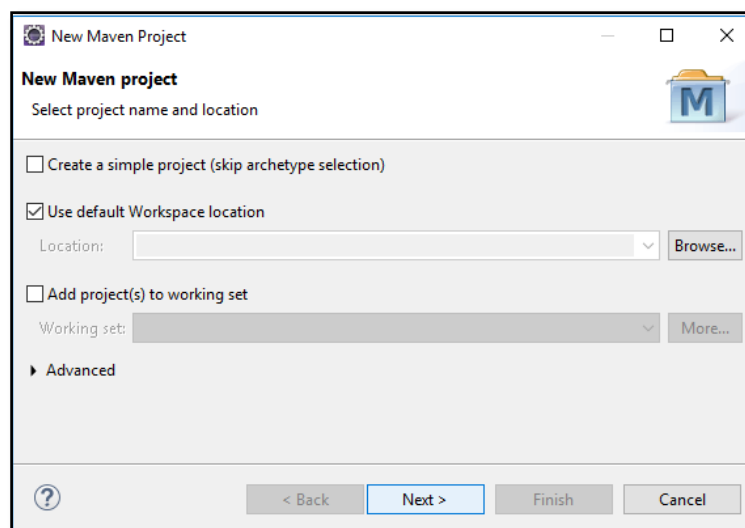
- Exécuter une application WEB
- Déployer une application WEB dans un serveur externe

A. Utiliser le fichier jar livré par un autre projet web

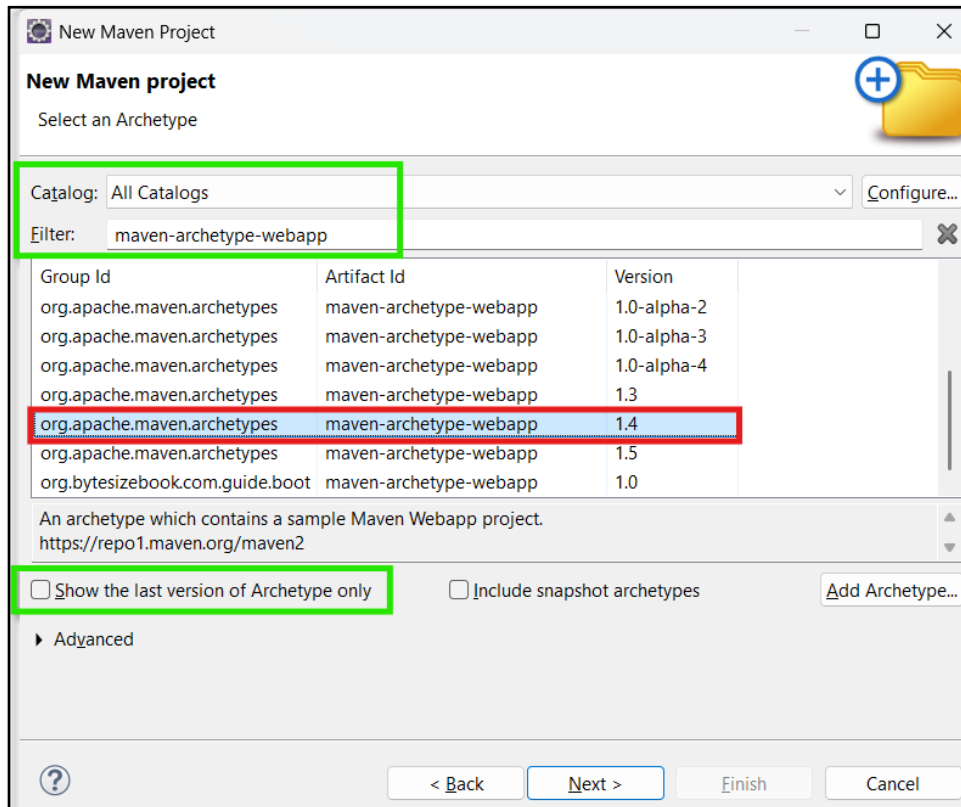
1. Créer un projet de type « **Maven Project** » en utilisant l'EDI « **eclipse** » :



2. Un assistant de construction de projet « **Maven** » s'affiche :



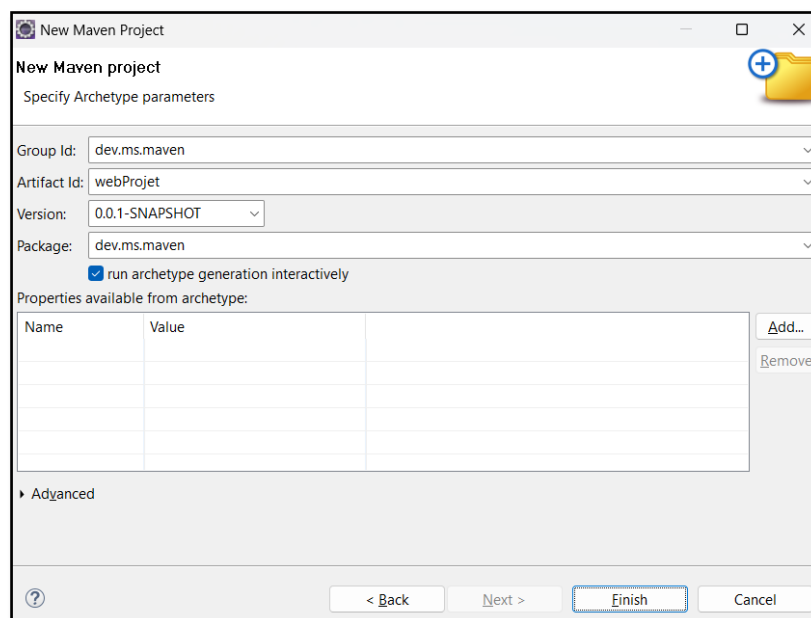
3. Appuyer sur « **Next** » pour choisir un archetype (**modèle**). Cette fois, choisir un archetype « **webapp** » pour créer une application web : **maven-archetype-webapp (version 1.4)**:



Remarquer le téléchargement des plugins par Maven

4. Appuyer sur « **Next** » pour spécifier les paramètres du projet :

- **groupId** : **dev.ms.maven**
- **artifactId** : **webProjet**
- **version** : **0.0.1-SNAPSHOT**
- **package** : **dev.ms.maven** (même valeur que « **groupId** »)



- Appuyer sur « **Finish** » pour lancer la construction du projet (**nouvelle application web**)
- Dans le volet « **Console** », taper « **ENTREE** » pour confirmer les paramètres de construction et achever le processus de création du projet « **webProjet** » :

```
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from [org.apache.maven.archetypes:maven-archetype-webapp:1.4] found in catalog remote
[INFO] Using property: groupId = dev.ms.maven
[INFO] Using property: artifactId = webProjet
[INFO] Using property: version = 0.0.1-SNAPSHOT
[INFO] Using property: package = dev.ms.maven
Confirm properties configuration:
groupId: dev.ms.maven
artifactId: webProjet
version: 0.0.1-SNAPSHOT
package: dev.ms.maven
Y: : |
```

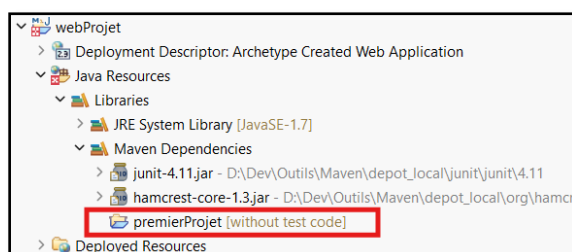
- Pour indiquer que ce projet dépend de notre premier projet « **premierProjet** » (**càd utilise des fonctionnalités définies dans une classe existante dans le projet «premierProjet** »), ouvrir le fichier « **pom.xml** » du projet « **webProjet** » pour ajouter la dépendance suivante à l'intérieur des deux balises **<dependencies>** et **</dependencies>** suite à la dépendance de « **junit** » :

```
<dependency>
  <groupId>dev.ms.maven</groupId>
  <artifactId>premierProjet</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

Voici la nouvelle liste de dépendances dans le fichier « **pom.xml** » :

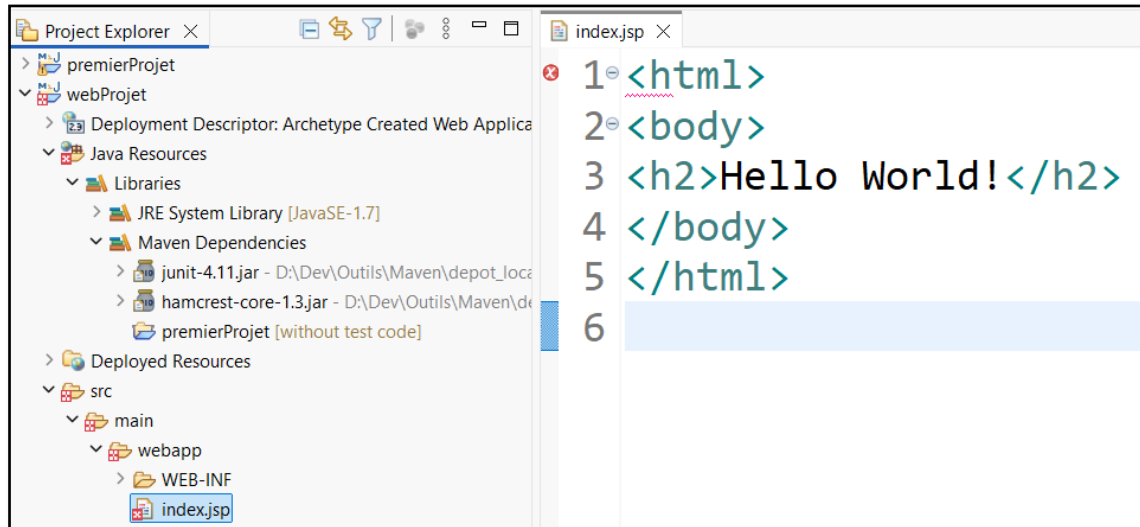
```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>dev.ms.maven</groupId>
    <artifactId>premierProjet</artifactId>
    <version>1.0-SNAPSHOT</version>
  </dependency>
</dependencies>
```

- Enregistrer et remarquer l'existence du jar « **premierProjet** » dans la liste des dépendances MAVEN :



B. Configurer le projet web

9. Remarquer qu'une erreur est signalée au niveau du projet et plus particulièrement au niveau de la page « **index.jsp** » due à la non existence de la classe « **javax.servlet.http.HttpServlet** » :

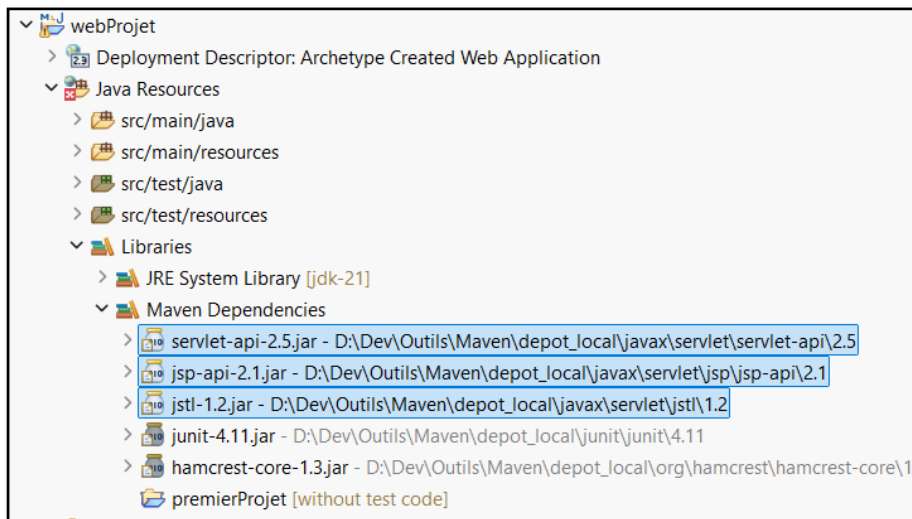


10. Ouvrir le fichier « **pom.xml** » pour ajouter les dépendances suivantes à l'intérieur de la balise **<dependencies>** :

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>servlet-api</artifactId>
  <version>2.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>jsp-api</artifactId>
  <version>2.1</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
  <scope>compile</scope>
</dependency>
```

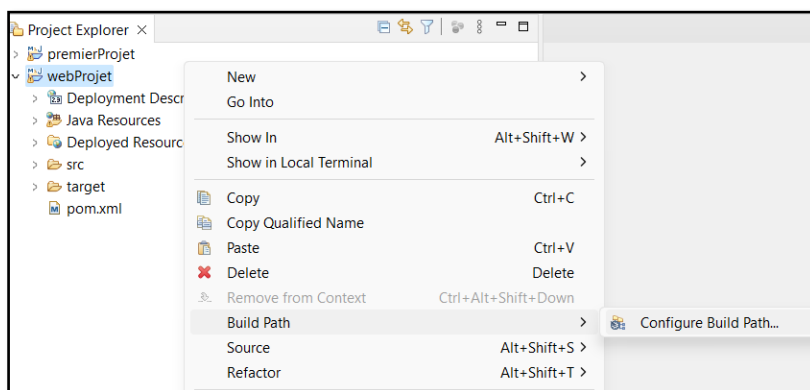
NB : Ne pas oublier d'enregistrer le fichier « **pom.xml »**

Remarquer que l'outil « **Maven** » lance la mise à jour des dépendances et se met à télécharger les nouvelles bibliothèques signalées pour les ajouter aux dépendances de Maven :

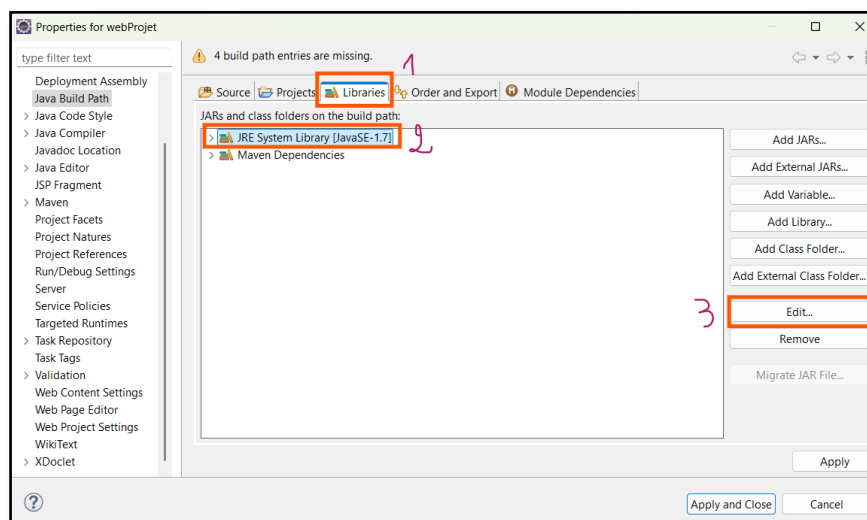


Dans le cas où l'erreur persiste, vérifier la version du JDK :

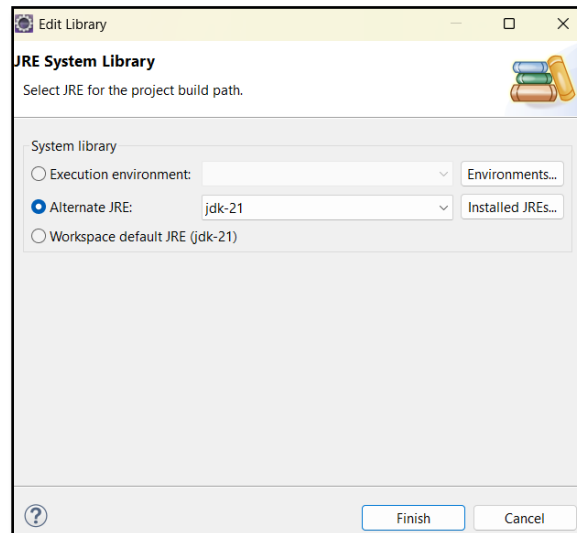
a. Choisir la commande « **Java Build Path** » :



b. Aller à l'onglet « **Librairies** » et éditer le JDK :



c. Spécifier JDK 21 :

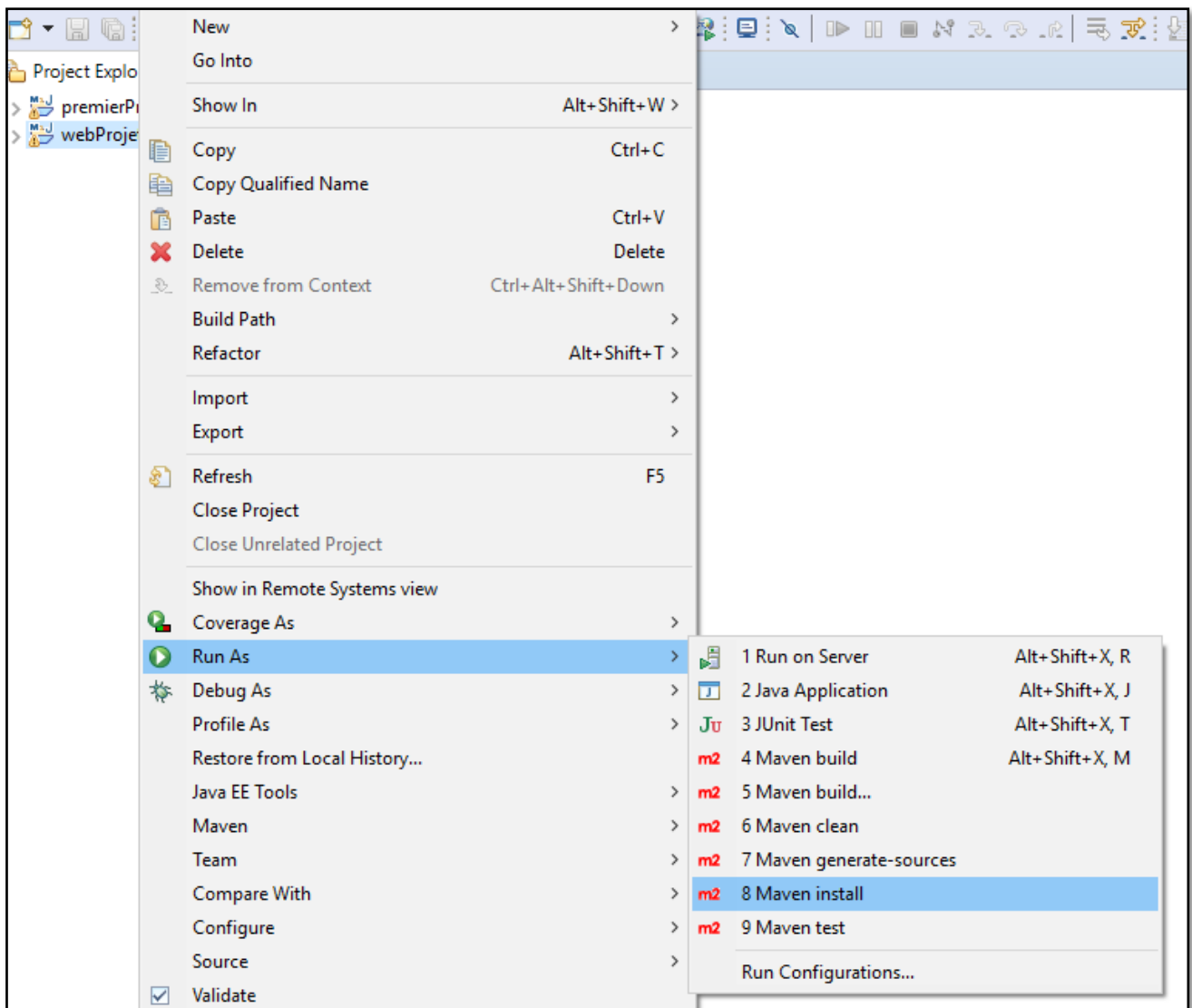


C. Exécuter et déployer l'application WEB

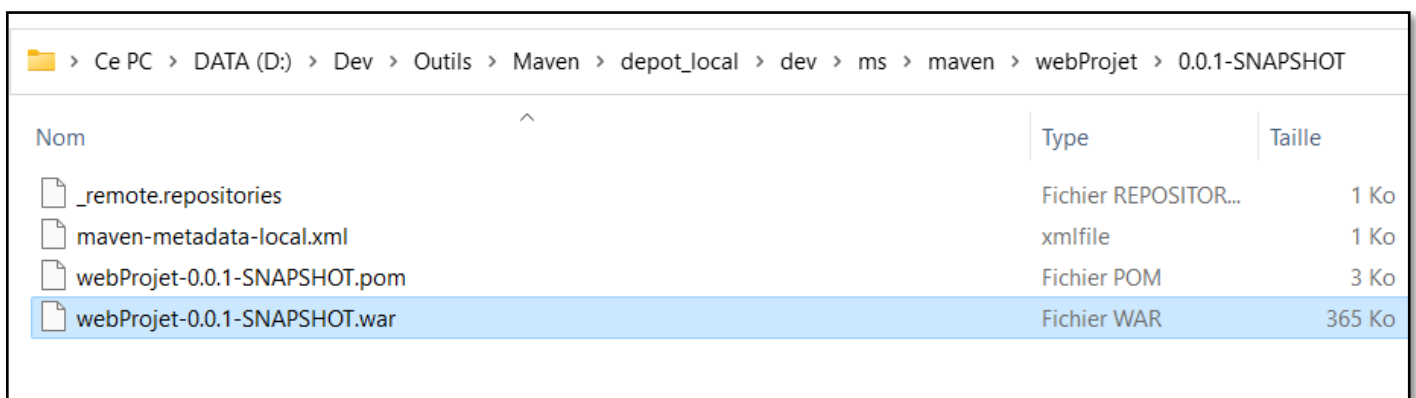
11. Changer le code du fichier «index.jsp» qui existe sous «src/main/webapp» comme suit :

```
<%@page import="dev.ms.maven.CalculMetier" %>
<% 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>
```

12. Maintenant, utiliser l'assistant « **eclipse** », pour installer (livrer) le projet « **webProjet** » en sélectionnant le projet puis choisissant la commande :

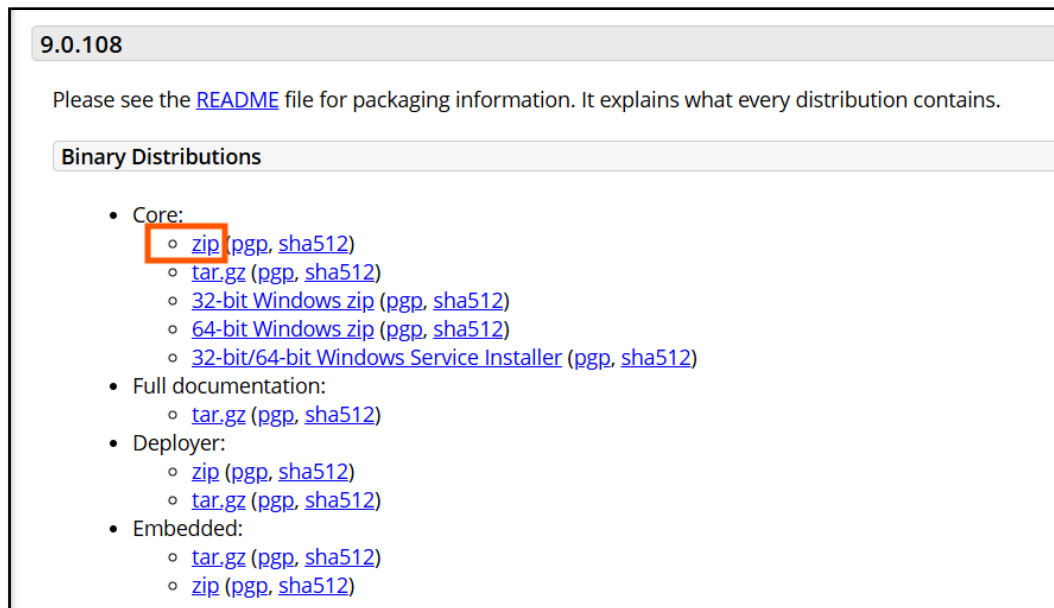


13. Remarquer la génération d'un fichier « **webProjet.war** » dans le dossier « **target** » à être déployer avec le serveur web « **Tomcat** ».



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

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



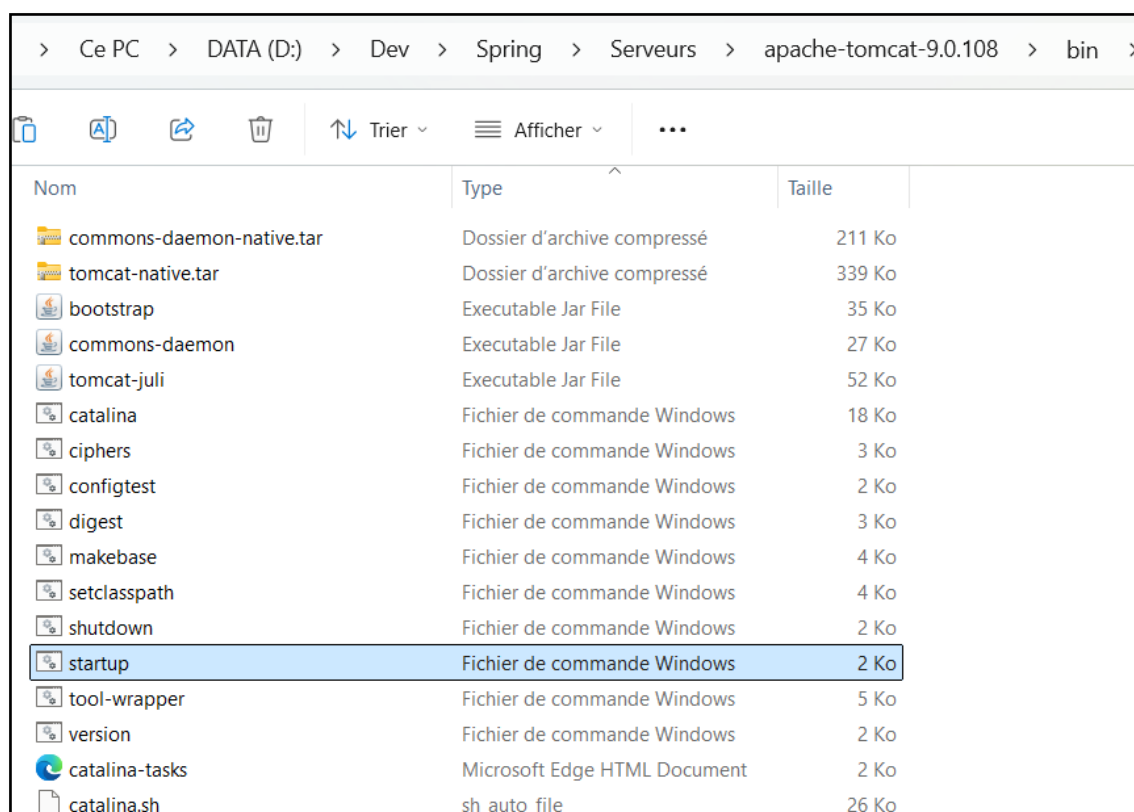
15. Télécharger l'archive (zip) du «**Tomcat**» dans le dossier :

D:\Dev\Spring\Outils

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

D:\Dev\Spring\Serveurs

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



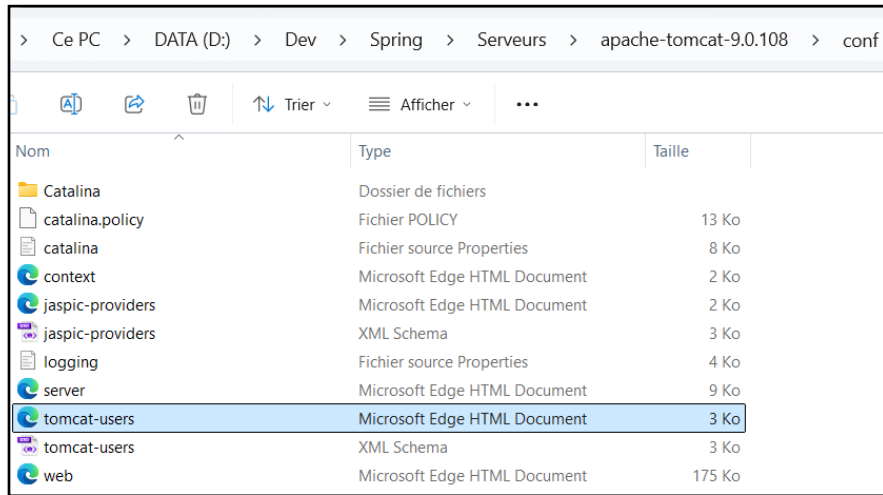
18. Accéder à travers le navigateur web au serveur par le lien :

http://localhost:8080

Voici la page d'accueil de Tomcat :

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

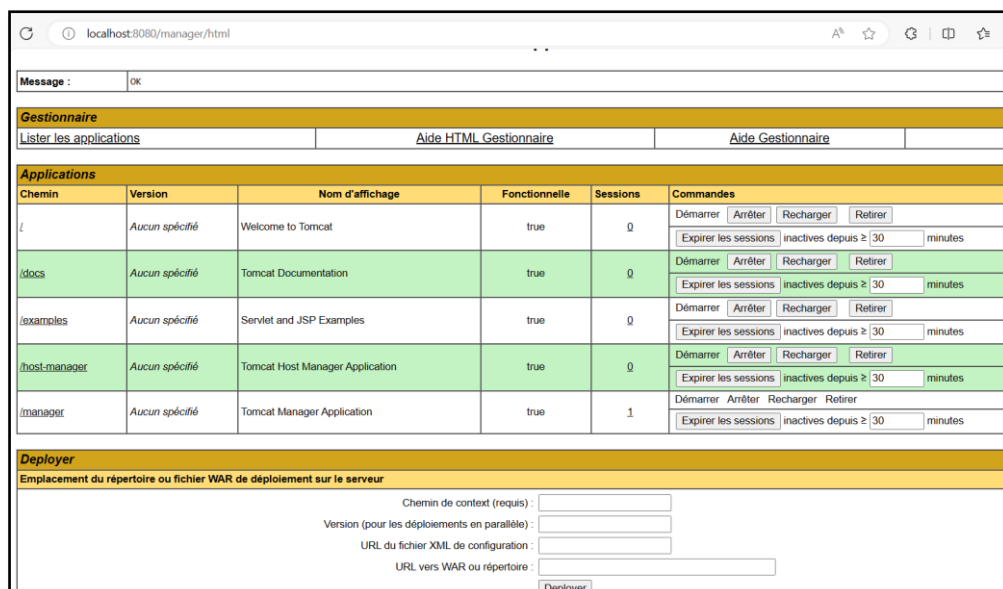
20. Pour se faire, accéder au sous dossier « **conf** » de Tomcat et éditer le fichier « **tomcat-users.xml** » :



21. 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"/>
```

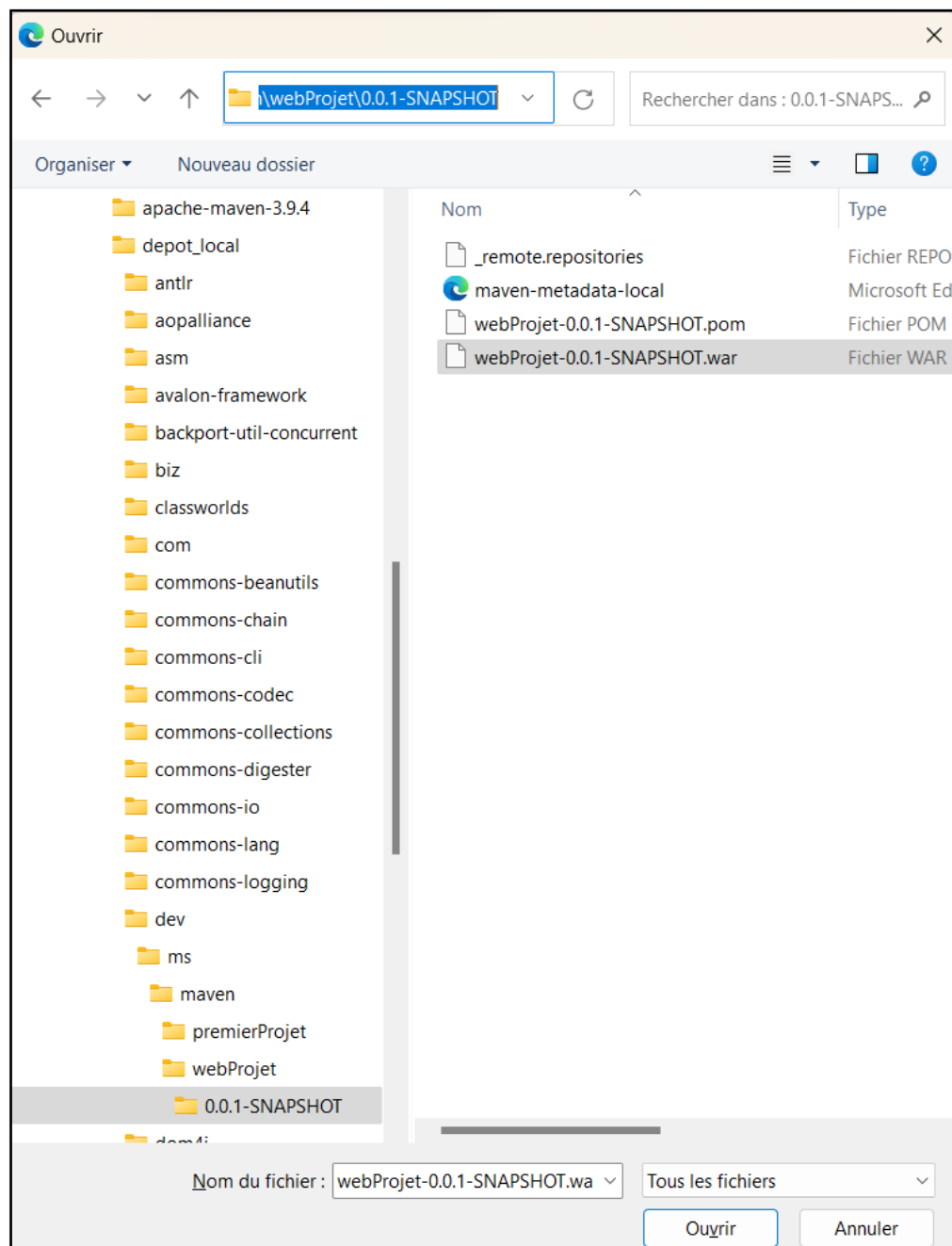
- ✓ Enregistrer le fichier puis Arrêter le serveur « **Tomcat** » en cliquant sur le fichier « **shutdown.bat** » (dans le sous-dossier « **bin** »)
- ✓ Redémarrer « **Tomcat** » (en cliquant sur le fichier « **startup.bat** »)
- ✓ Ré-accéder à l'interface web d'administration de tomcat pour s'authentifier avec les deux paramètres (« **admin** », « **admin** ») et gérer les applications déployées :



- ✓ 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.

Deployer	
Emplacement du répertoire ou fichier WAR de déploiement sur le serveur	
Chemin de contexte (requis):	<input type="text"/>
Version (pour les déploiements en parallèle):	<input type="text"/>
URL du fichier XML de configuration:	<input type="text"/>
URL vers WAR ou répertoire:	<input type="text"/>
<input type="button" value="Deployer"/>	
Fichier WAR à déployer	
Choisir le fichier WAR à téléverser	<input type="button" value="Choisir un fichier"/> <input type="button" value="Aucun fichier choisi"/>
<input type="button" value="Deployer"/>	

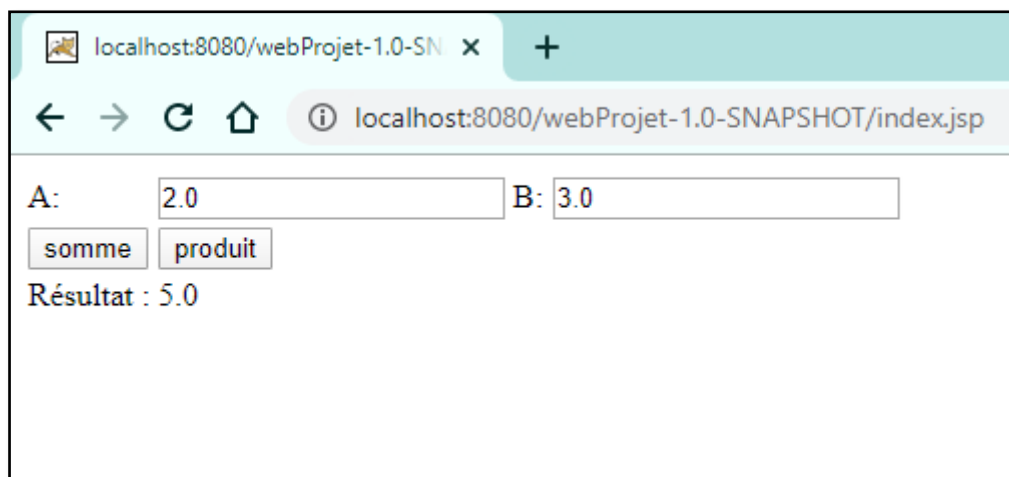
- ✓ Sélectionner le fichier WAR du projet « **webProjet** » :



- ✓ 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
/webProject-0.0.1-SNAPSHOT	Aucun spécifié	Archetype Created Web Application	true	0	Démarrer Arrêter Recharger Retirer Expirer les sessions inactives depuis ≥ 30 minutes

- ✓ Ainsi le projet WEB est déployé il suffit de cliquer sur le lien correspondant pour lancer son exécution.
- ✓ Voici le résultat :



Exercice

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