TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr

Maven : création d'un projet Web

a. Compétences visées :

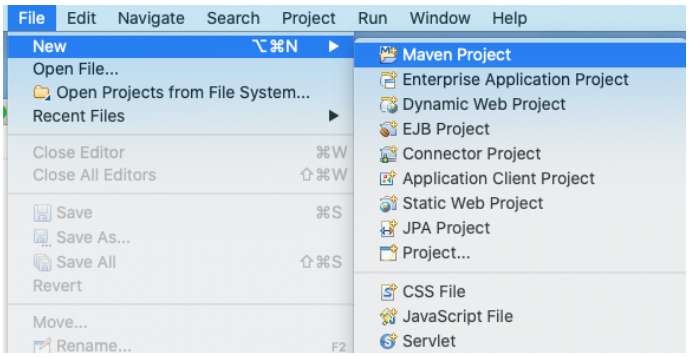
En suivant les étapes indiquées dans ce tutoriel, vous devriez être capable de créer une application simple JEE en utilisant Maven, d'explorer et de comprendre l'arborescence du projet, ajouter des dépendances, et finalement exécuter votre application.

Maven :

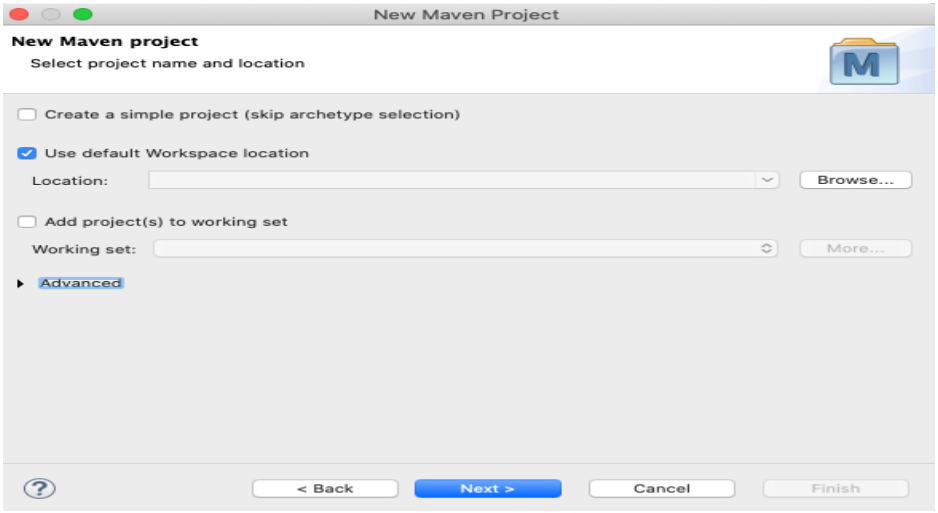
Maven est un outil open source de Apache écris en Java. Il est utilisé pour automatiser la gestion et la construction (i.e., build) des projets Java.
Il peut être utilisé en ligne de commande ou via un IDE (Eclipse, Intellij IDEA, …).

b. Création d'un projet Maven

Créez un projet Maven : File/New/Maven Project (ou File/New/others/Maven/Maven Project).



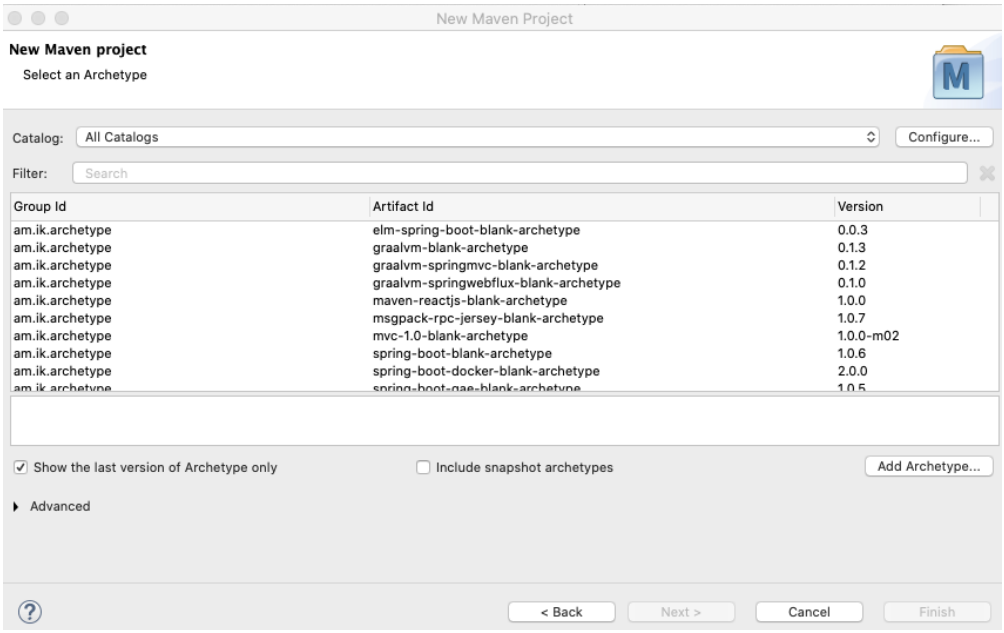
Vous obtenez la fenêtre suivante.



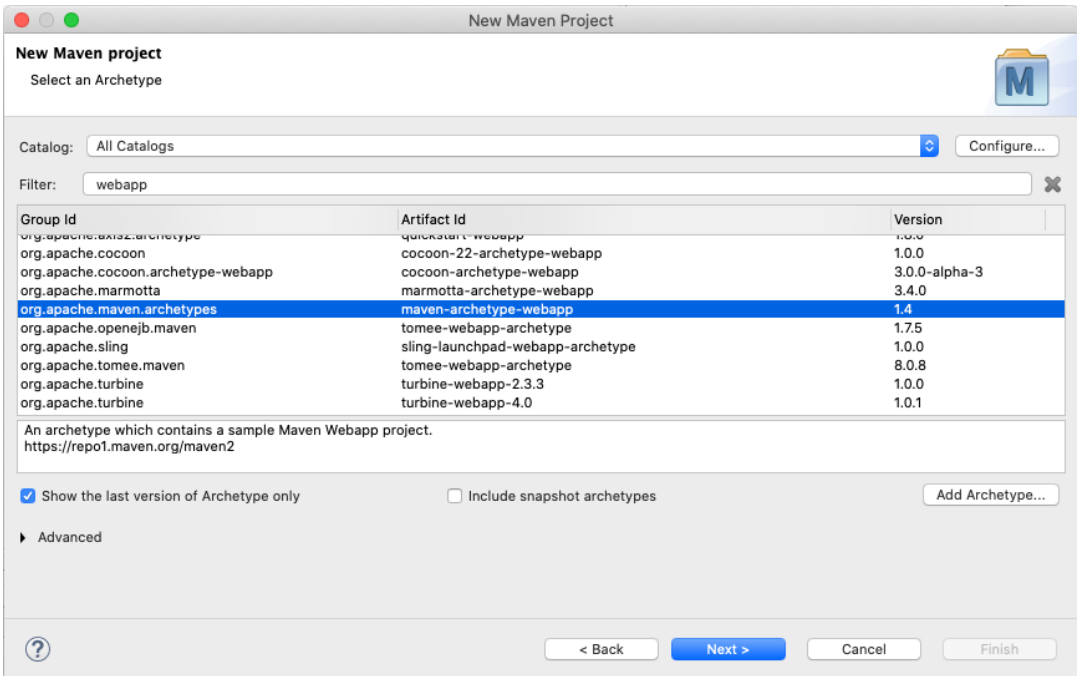
guermouc@insa-toulouse.fr

Cliquez directement sur Next. Vous remarquez une liste d'éléments, appelés archetype.

Un archetype peut être vu comme un moyen de créer un modèle de projet (i.e., template). Par exemple, on peut créer un projet Web à l'aide de son archetype correspondant. On obtient un projet Web prêt à être compilé, à être exécuté, et à être exécuté.



Justement, ce qui nous intéresse ici c'est la création d'un projet Web. Donc dans le champs « Filter », tappez *webapp* pour pré-filtrer les archetype en relation avec les applications Web, puis sélectionner le comme suit :

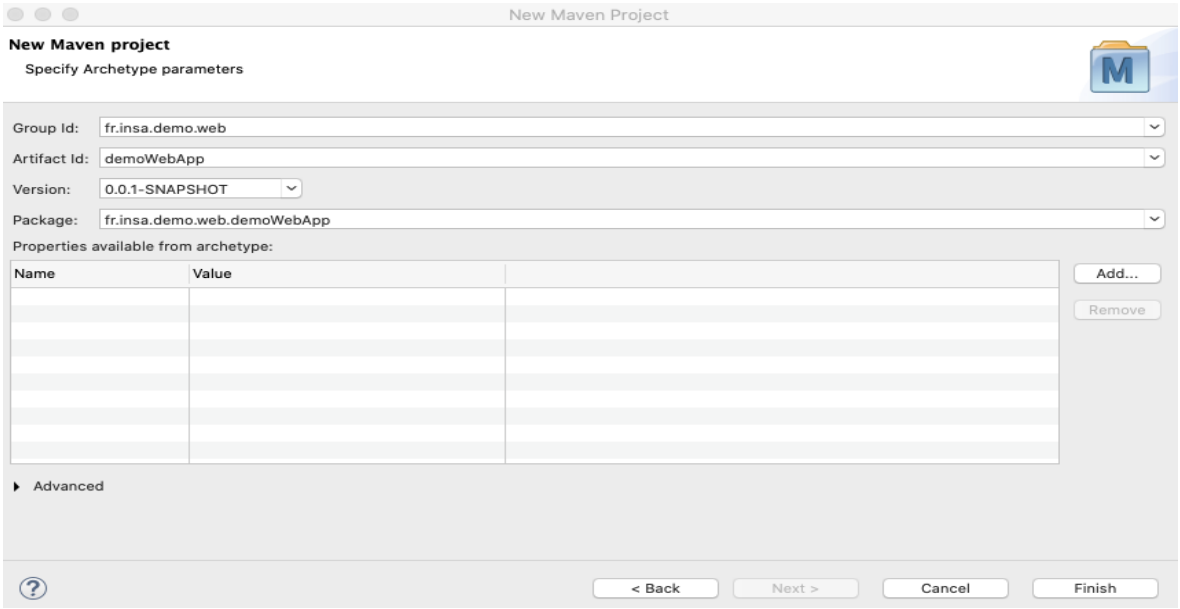


guermouc@insa-toulouse.fr

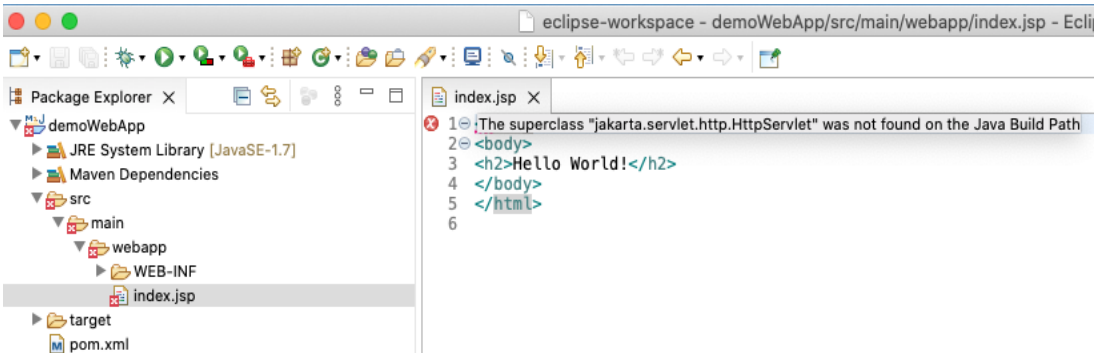
Cliquez sur Next. Vous devez renseigner les éléments group Id, Artifcat ID.

- *groupId* : correspond à la racine qui va englober tous les projets d'une entreprise par exemple. Il peut être vu comme un nom de domaine. Une entreprise crée plusieurs projets.
- *artifactId* : correspond au nom du projet
- *Version* : correspond au numéro de version du projet. Tout au long de la vie d'un logiciel, le numéro de version peut évoluer (e.g., numéro des releases).

Pour la version, soit vous gardez tel quel soit mettre une version par exemple1.0. Cliquez sur finish.



Un projet Maven est crée. Avant de passer à la suite, prenez quelques minutes pour examiner la structure du projet créé. Vous remarquerez par exemple qu'une page index.jsp a été créée.

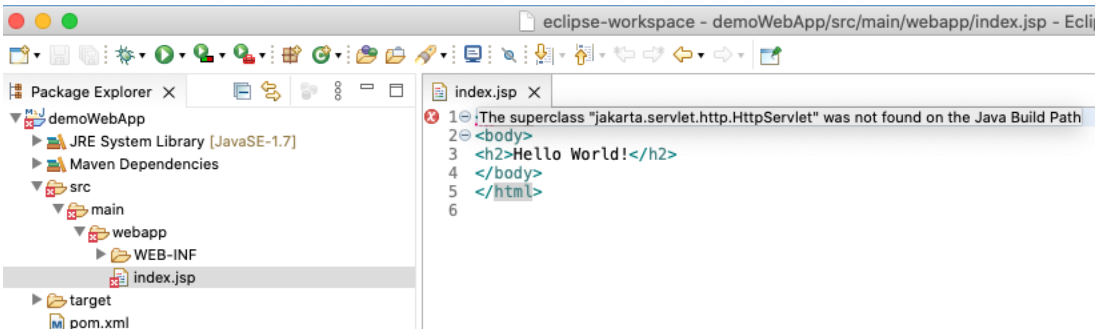


TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr

D'une manière générale, un projet Web Maven a la structure suivante :

- **Le répertoire src/main/java** : qui va contenir les codes sources Java
- **Le répertoire src/test/java** : dédié aux tests unitaires
- **Le répertoire target** : les fichiers générés du projet vont être stockés sous target. Cela concerne par exemple les fichiers des classes compilées, les rapports des tests, les fichiers Jar …etc
- **Sous src/main/webapp** : contient les fichiers Web comme les JSP (Java Server Page). Vous allez voir plus tard ce que c'est une JSP
- **Le fichier pom.xml** (POM pour Project Object Model): est le fichier central de chaque projet Maven. Il contient toutes les informations nécessaires pour que Maven gère le projet. On y trouve par exemple les informations du projet lui-même (groupId, ArtifacId,…), la version de Java utilisée, les librairies nécessaires (appelées dépendances). Examinez le.

On reprend maintenant la suite de la création du projet. Si vous remarquez une erreur dans votre projet, dépliez l'arborescence du projet. Vous remarquerez qu'il est indiqué qu'une librairie (jakarta.servlet.http.HttpServlet) manque. Autrement dit, l'API jakarta.sevlet manque (jakarta.servlet-api)

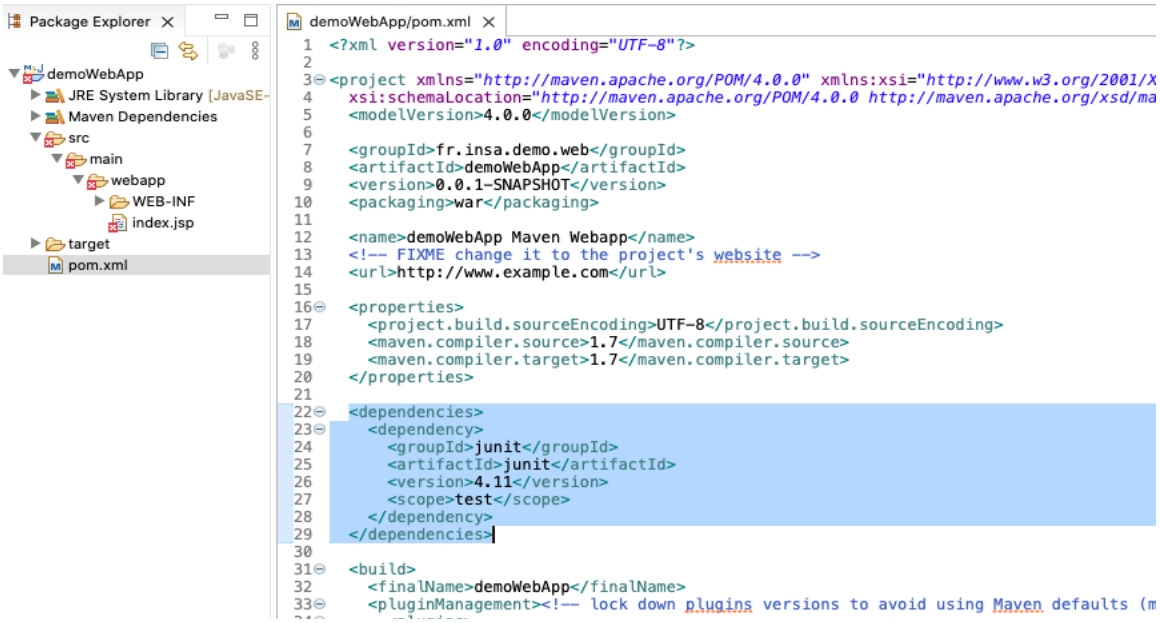


Classiquement, pour résoudre cela, on doit chercher la librairie manquante manuellement et l'intégrer au projet. Ce qui peut être lourd, notamment dans le cadre de projets conséquents où plusieurs librairies sont nécessaires. Pour éviter

de chercher et de télécharger les dépendances une à une, on va demander à Maven de s'en occuper.

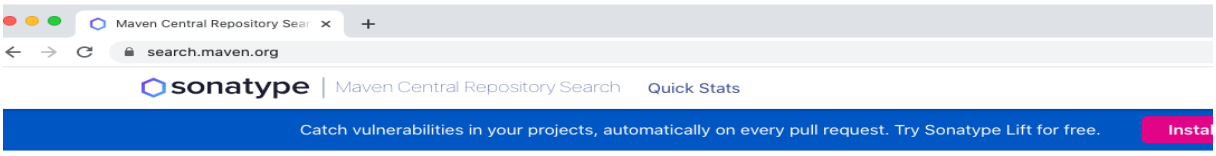
Avec Maven, il suffit d'indiquer dans le fichier pom.xml dédié aux dépendances de récupérer cette librairie si elle n'est pas déjà mentionnée. Ainsi, Maven les télécharge depuis un dépôt central (Maven Central Repository).

Vous allez d'abord vérifier si cette librairie (i.e., dépendance) existe ou pas. Pour cela, ouvrez le fichier pom.xml se trouvant à la fin de l'arborescence. Examinez l'élément <dependencies>. Vous remarquez que seule la dépendance junit est présente. Il faudra étendre les dépendances pour ajouter la dépendance manquante.

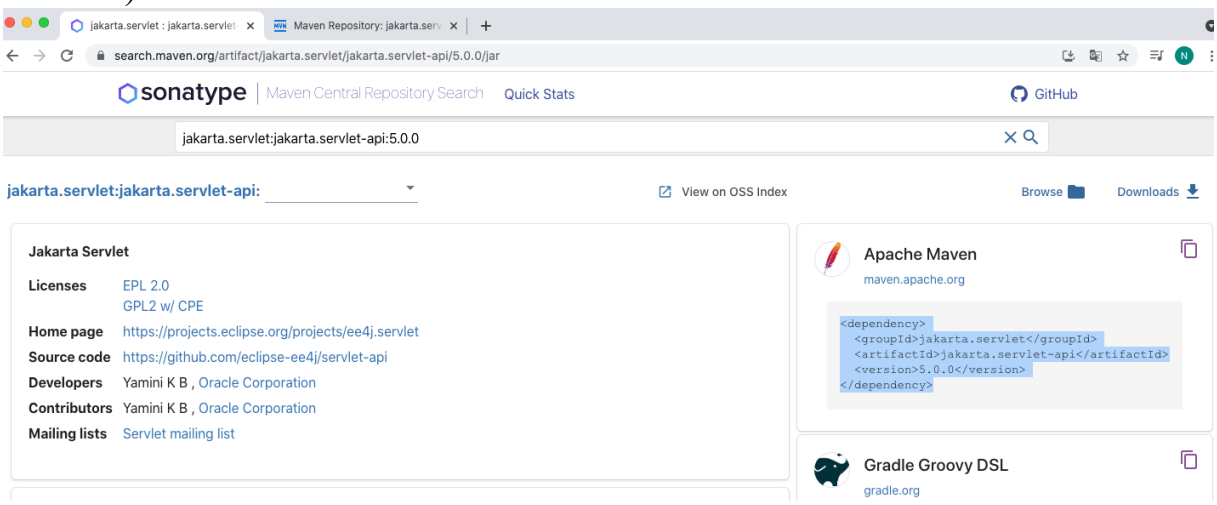


Il faut ajouter la dépendance *jakarta.servlet-api*. Pour cela, vous allez sur le site search.maven.org et cherchez cette API :

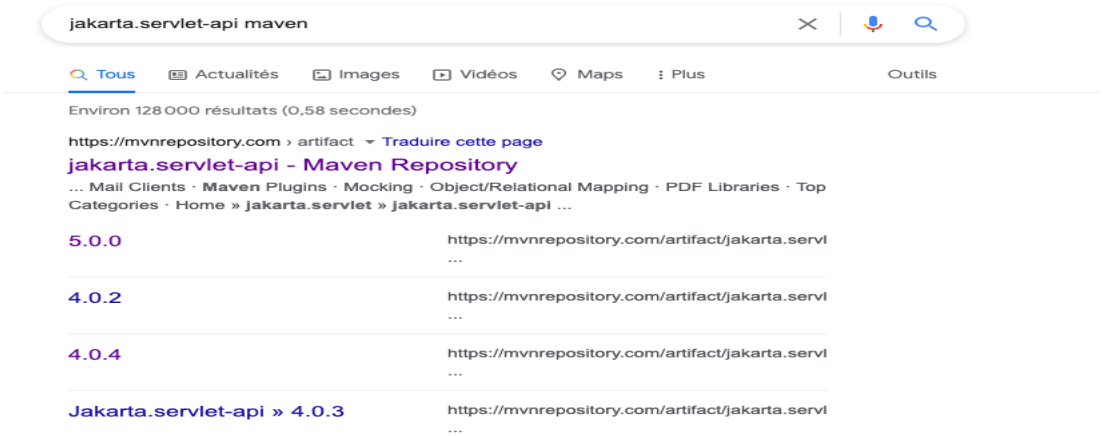
TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr



Cliquez sur l'API et copiez la dépendance Maven (la fenêtre gauche sur la figure suivante).

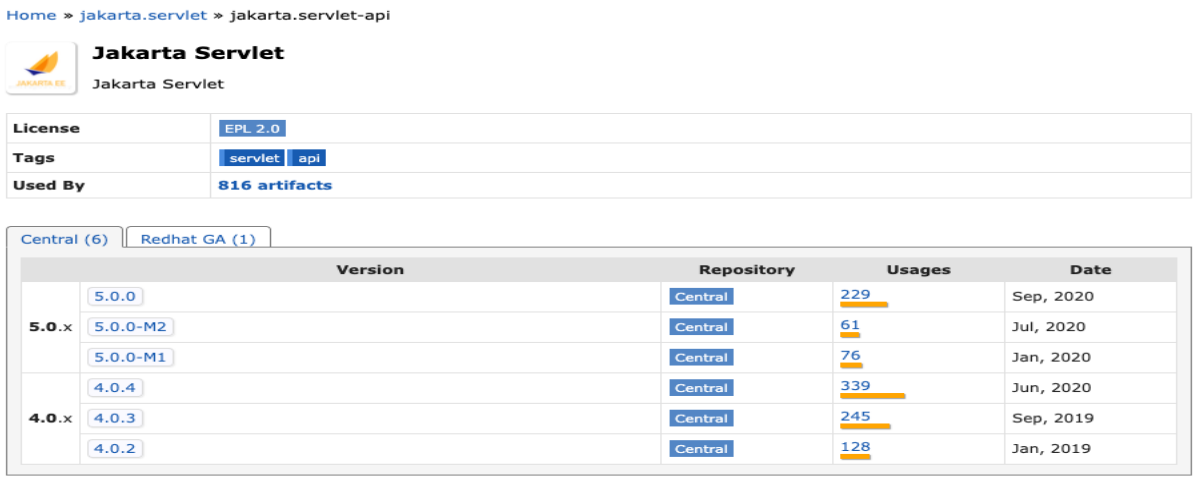


Remarque : vous pouvez tout simplement chercher la dépendance via un moteur de recherche en tapant *jakarta.servlet-api maven* comme le montre la figure suivante. Ensuite, il suffit de choisir la version que vous souhaitez utiliser.



TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr

Ici on a choisi la version 5.0.0



Il suffit de copier la dépendance et de la coller dans le fichier pom.xml :

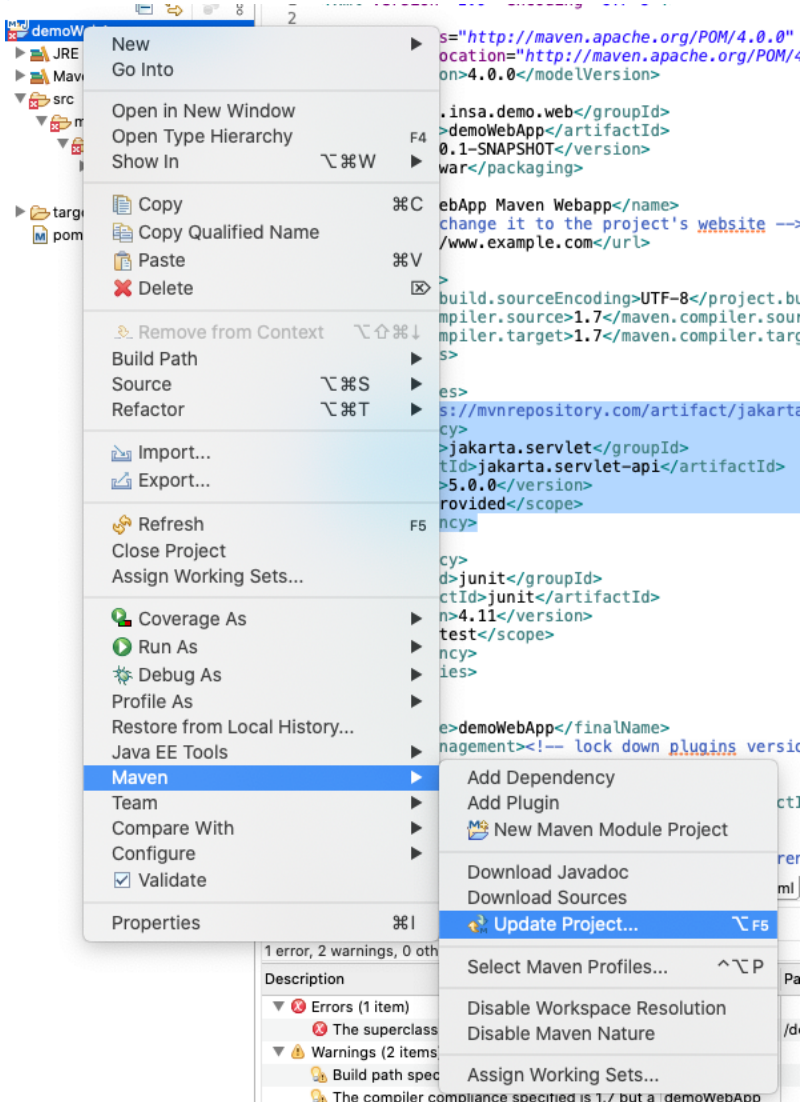
TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr

*demoWebApp/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XM
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
  <modelVersion>4.0.0</modelVersion>

  <groupId>fr.insa.demo.web</groupId>
  <artifactId>demoWebApp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>demoWebApp Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/jakarta.servlet/jakarta.servlet-api -->
    <dependency>
      <groupId>jakarta.servlet</groupId>
      <artifactId>jakarta.servlet-api</artifactId>
      <version>5.0.0</version>
      <scope>provided</scope>
    </dependency>

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

  <build>
    <finalName>demoWebApp</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (ma
      <plugins>
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
        <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#F
```

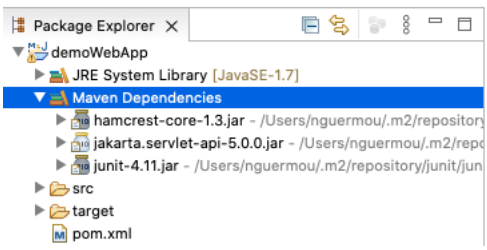
Sauvegardez le document. Si l'erreur persiste, il faut indiquer à Maven qu'il faut mettre à jour le projet. Pour cela, clic droit sur le projet/Maven/Update comme suit :

TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr



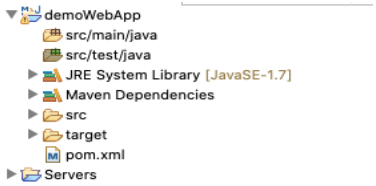
Ainsi, Maven va importer la librairie manquante et votre projet ne devrait plus signaler l'erreur.

Les dépendances importées peuvent être vérifiées dans l'arborescence du projet comme suit :



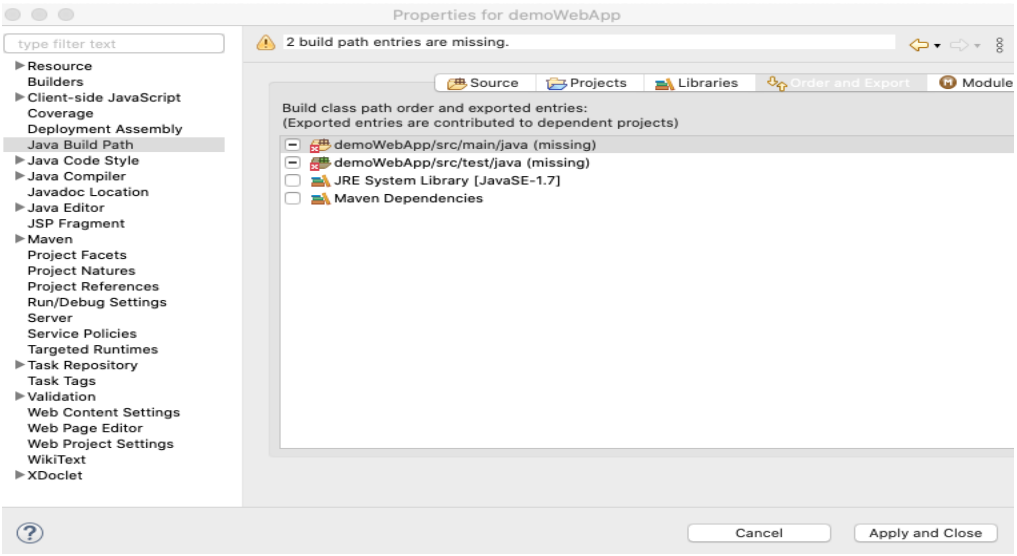
guermouc@insa-toulouse.fr

Vous devriez avoir l'arborescence suivante.

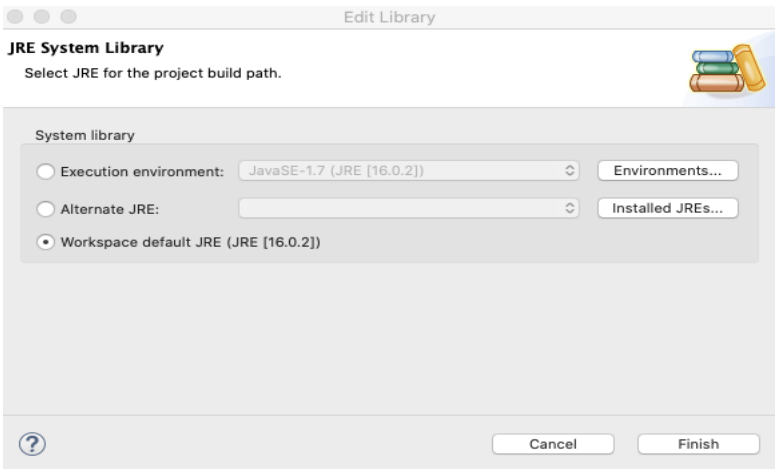


Si dans l'arborescence de votre projet, vous ne voyez pas *src/main/java* et *src/test.java*, c'est qu'il y a un problème de configuration avec la version de java utilisée dans Maven et celle de Eclipse.

Clic droit sur le projet/properties, puis Java Build Path.



Cliquez sur *Librairies*, puis sélectionner *JRE System Library* puis cliquez sur Edit (ou double clic). Sélectionnez Worspace default JRE.



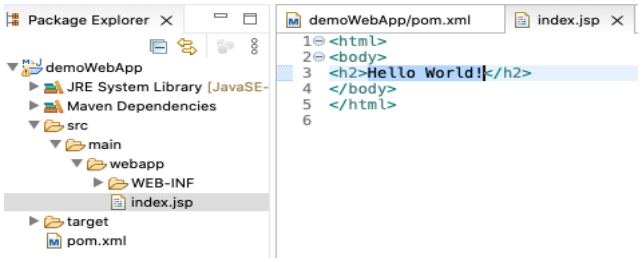
En retournant à votre projet, vous devriez retrouver les répertoires manquants.

Maintenant vous connaissez les bases de Maven, vous savez créer un projet, cherchez des dépendances et de les ajouter.

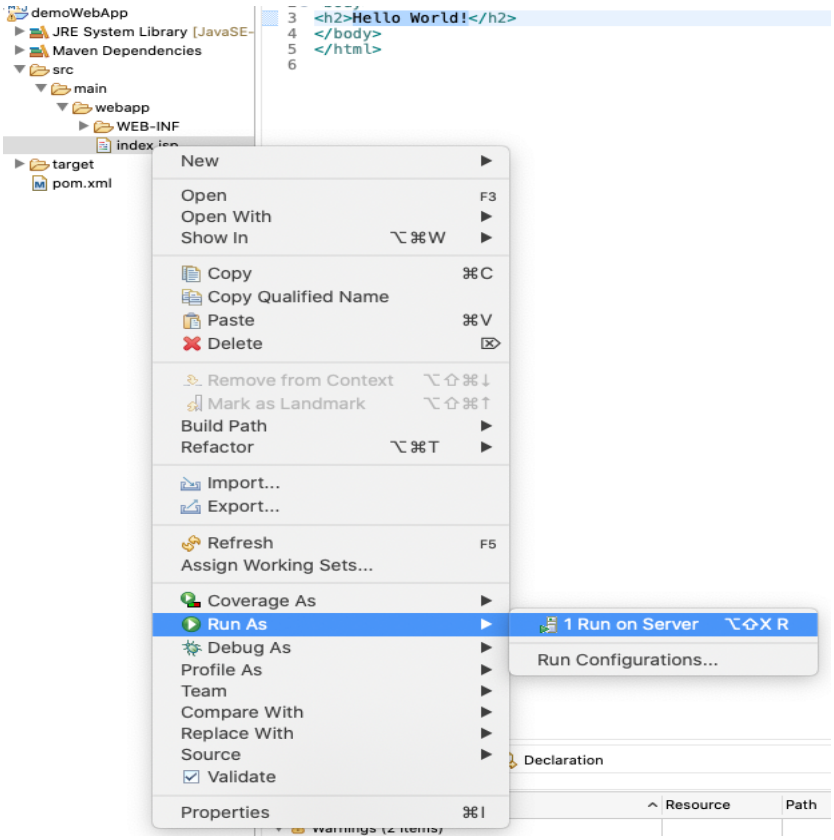
Exécution de votre projet :

Vous remarquerez sous src/main/webapp la page index.jsp. C'est une page JSP (Java Server Page). Elle permet de créer des pages HTML dynamiques qui intègrent du code Java.

En créant ce projet, une page JSP par défaut est générée, qui permet d'afficher Hello World.



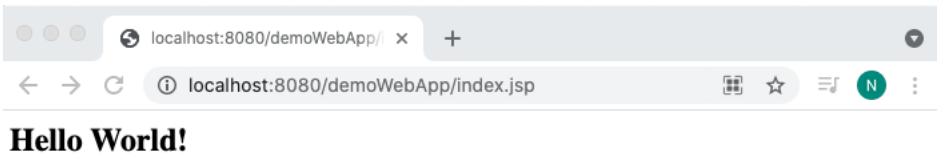
Vous pouvez l'exécuter pour vérifier que le projet s'exécute correctement.



TD PDLA
4IR
INSA de Toulouse
guermouc@insa-toulouse.fr

Via un navigateur, ouvrez l'URL suivante, qui est de la forme *serveur/projet/page_JSP* : http://localhost:8080/demoWebApp/index.jsp
Le déploiement de votre projet est fait localement sur la machine (i.e., serveur localhost :8080)

En ouvrant avec un navigateur l'URL précédente, ça permet d'envoyer une requête http au serveur.



Le projet s'exécute sans problème.

Bilan :

- C'est quoi Maven ?
- Qu'est ce qu'il permet de faire ?