

MAVEN

esprit 
Se former autrement



Bureau E204

PLAN DU COURS

- C'est Quoi **Maven**?
- Création d'un Projet Maven
- Balises du POM.XML
- Arborescence Standard
- Buts (Goals)
- TP : Projet avec Maven (**JAR**)
- TP : Projet Web Avec Maven (**WAR**)

C'EST QUOI MAVEN?

- Maven est un outil de construction de projets (build) open source développé par la fondation Apache.
- Il permet de faciliter et d'automatiser certaines tâches de la gestion d'un projet JavaEE.

FONCTIONNALITÉS

- Fonctionnalités
 - Automatisation de tâches récurrentes
 - Construction, Compilation des projets
 - Gestion des dépendances
 - Génération des livrables
 - Génération de la documentation et de rapports
 - Déploiement d'applications
- Modèle de projet basé sur des conventions (POM)
 - Configuration basée sur le format XML

INSTALLATION DE MAVEN

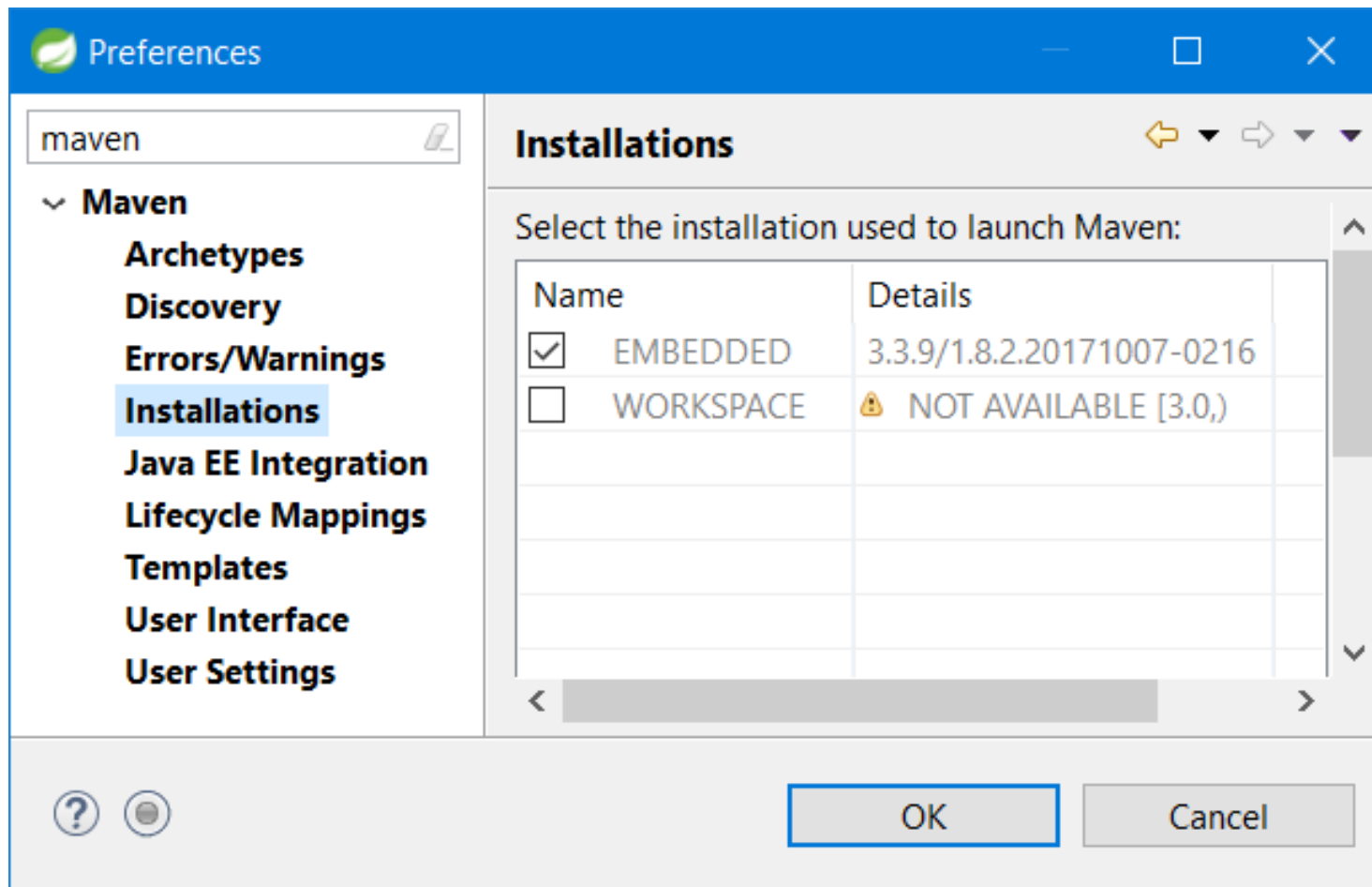
- Maven en tant que Plugin : Un plugin Maven est déjà installé par défaut avec Eclipse.
- Maven en standalone (non traité dans notre cours).

EXERCICE

- Comment savoir si Maven est bien installé, et quelle version nous utilisons (Dans le cas où Maven est installé en tant que Plugin dans STS)?

EXERCICE

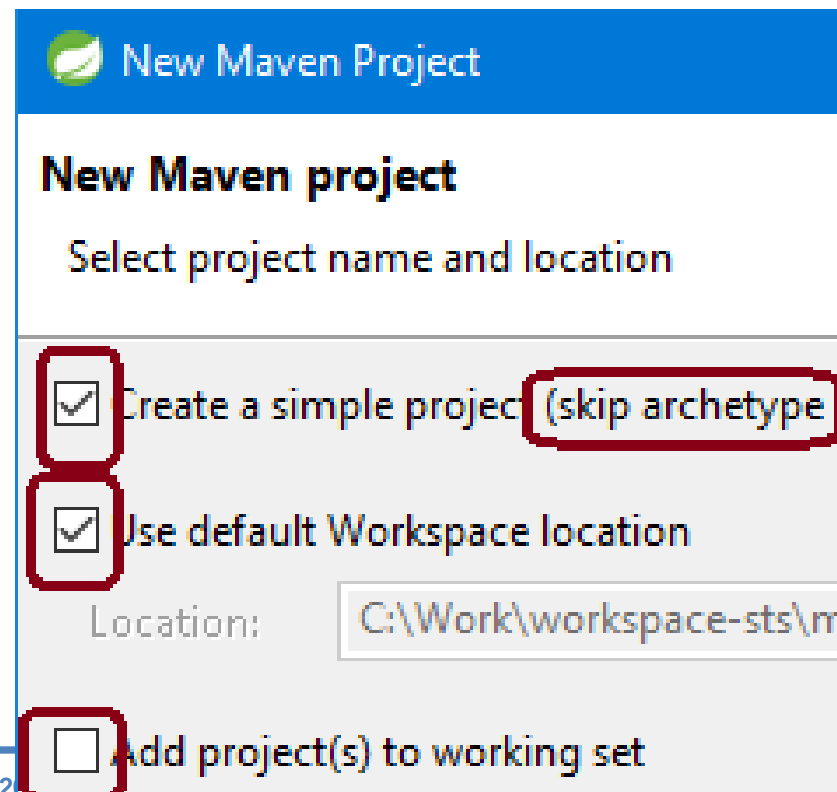
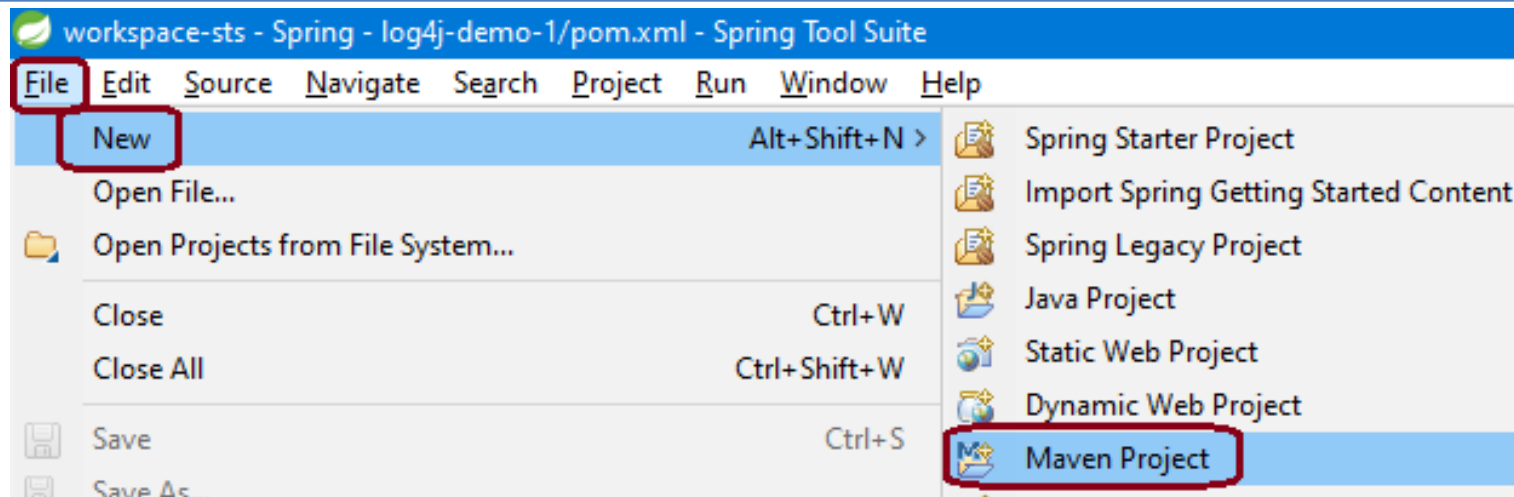
- En mode plugin :



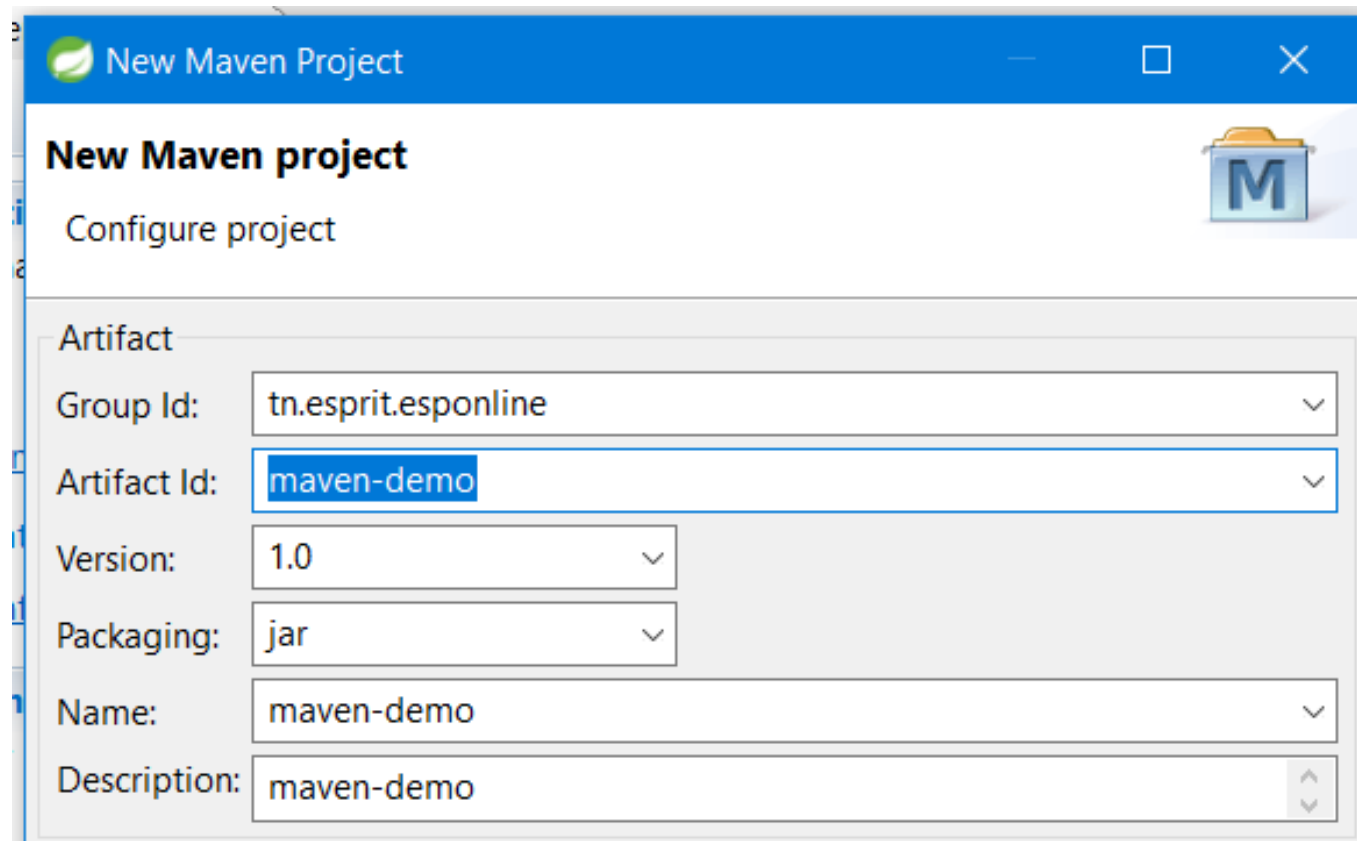
Suite du Cours

- Créer un projet MAVEN simple via le plugin Maven
- Maîtriser l'arborescence standard du code et de ses ressources
- Maîtriser les différents buts (Goals) du cycle de vie d'un projet Maven (la compilation, le test, le packaging d'une application, ...)
- Installer une application dans un Repository local
- Gérer les dépendances (bibliothèques) d'un projet donné
- Exécuter les tests unitaires automatiquement

CRÉATION D'UN PROJET MAVEN



CRÉATION D'UN PROJET MAVEN



New Maven project
Configure project

Artifact

Group Id: tn.esprit.esponline

Artifact Id: maven-demo

Version: 1.0

Packaging: jar

Name: maven-demo

Description: maven-demo

Configuration pom.xml

- Correction du warning « Java 1.5 », Pointer sur Java 8 :
- Correction de l'erreur dans le pom.xml (pointer sur le plugin Maven 3.1.1) :
- Ajouter dans pom.xml les propriétés suivantes :

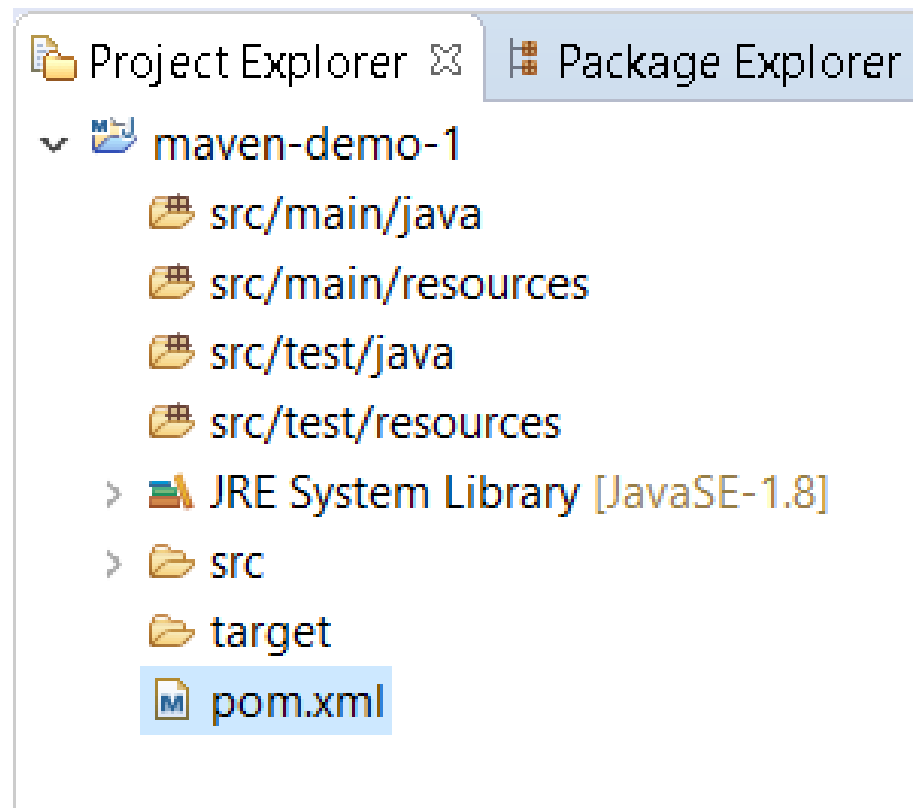
```
<properties>  
<maven.compiler.target>1.8</maven.compiler.target>  
<maven.compiler.source>1.8</maven.compiler.source>  
<maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>  
</properties>
```

- Puis, Faites un Maven Update.

BALISES DU POM.XML

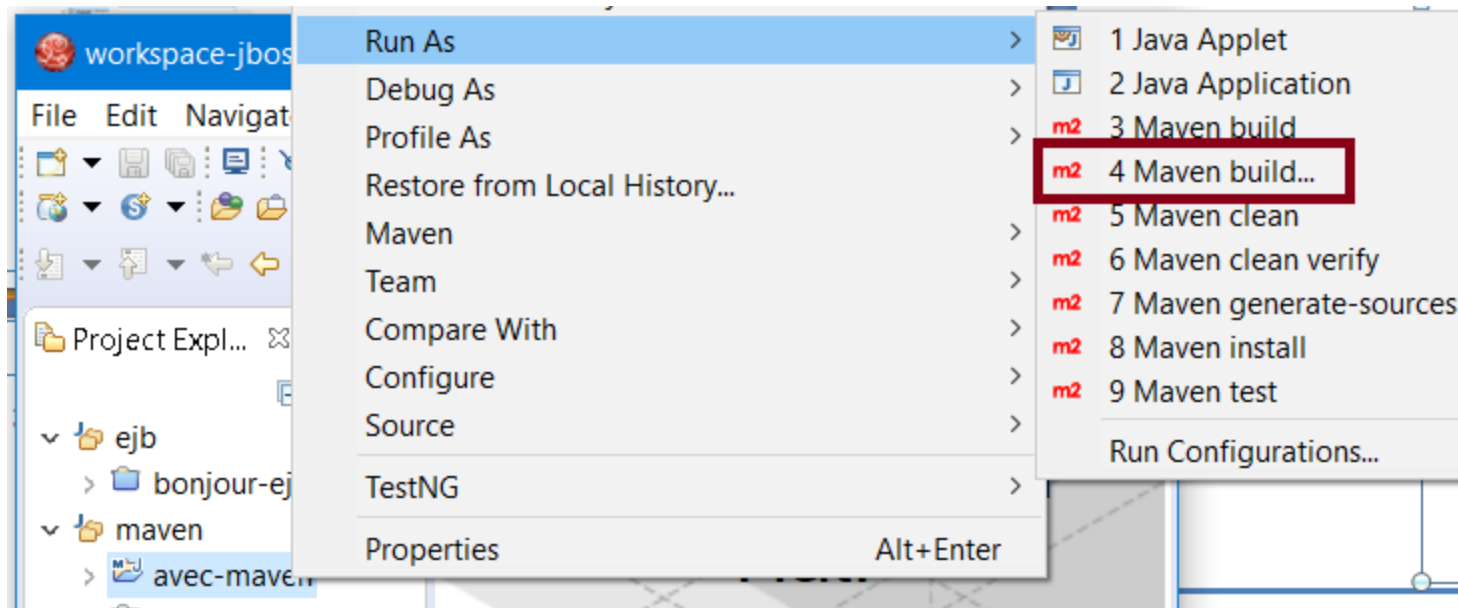
- **pom.xml** : Project Model Object
- **project** : Balise racine de tous les fichiers pom.xml.
- **modelVersion** : Version de POM utilisée.
- **groupId** : Identifier un groupe qui a créé le projet. Ex: org.apache.
- **artifactId** : Nom unique utilisé pour nommer l'artifacts à construire.
- **packaging** : Type de packaging du projet (ex. : JAR, WAR, EAR...).
- **version** : Version de l'artifact généré par le projet.
- **name** : Nom du projet.
- **description** : Description du projet.
- **dependencies** : balise permettant de gérer les dépendances.
- **archetype** : Template de Projet.

ARBORESCENCE STANDARD



PREMIÈRES COMMANDES

- Assurez vous que vous accés à Internet, Puis lancer les commandes suivantes **clean / install** avec JBoss Dev Studio :



- Ceci va mettre à jour votre Repository local avec l'ensemble des plugin et dépendances nécessaires pour que le bon fonctionnement de Maven.
- Partager votre Repository avec vos collègues qui n'ont pas accès à internet.

ARBORESCENCE STANDARD

- **pom.xml** : le fichier de configuration du projet
- **/src** : code source et fichiers source principaux
- **/src/main/java** : code source java
- **/src/main/resources** : fichiers de ressources (images, fichiers config...)
- **/src/main/webapp** : webapp du projet
- **/src/test** : fichiers de test
- **/src/test/java** : code source Java de test
- **/src/test/resources** : fichiers de ressources de test
- **/target** : fichiers résultat, les binaires (du code et des tests), les packages générés et les résultats des tests

BUTS (GOALS)

- **mvn compile** : Créer les .class
- **mvn test** : Jouer les tests unitaires
- **mvn package** : Creation du livrable dans target.
- **mvn install** : Copie du livrable dans le Repository local :
~\.m2\repository\...
- **mvn deploy** : Copie du livrable sur le repository distant
- **mvn clean** : Supprime le contenu du dossier target.

BUTS (GOALS)

- Emplacement du livrable :
{emplacement Repository}/groupId/artifactId/version
- Nom du package (jar en général) : {artifactId}-{version}.{package}

```
<dependency>  
  <groupId>log4j</groupId>  
  <artifactId>log4j</artifactId>  
  <version>1.2.17</version>  
</dependency>
```

Dépôts (Repository) Maven

- Dépôt Local : .m2
- Dépôt Distant : propre à une entreprise donnée (Nexus)
- Dépôt Central : <https://mvnrepository.com>

TP1 – Projet avec Maven (JAR)

- Créer un Projet **Maven** :
 - Simple : sans archetype, type Jar, groupId : tn.esprit
 - artefactId / nom / description : avec-maven
 - package : tn.esprit
- Mettre à jour le pom.xml pour utiliser Java 1.8
- Créer le package : tn.esprit
- Créer la Classe : CallRestWebService (Voir Code Source pages suivantes).
- Ajouter dans le pom.xml les dépendances JSON et HTTPCLIENT (voir dépendances pages suivantes).
- Créer le livrable avec Maven
- Exécuter la méthode main.

TP1 – Projet avec Maven (JAR)

```
package tn.esprit;
import java.io.IOException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

/** @author Walid-YAICH */

public class CallRestWebService {
    public static final String endpoint = "https://httpbin.org/get";
    public static void main(String[] args) {
        HttpClient client = new DefaultHttpClient();
        HttpGet request = new HttpGet(endpoint);
        String ip = "not found";
    }
}
```

TP1 – Projet avec Maven (JAR)

```
try {  
    HttpResponse response = client.execute(request);  
    String jsonResponse = EntityUtils.toString(response.getEntity());  
    System.out.println("Response as String : " + jsonResponse);  
    JSONObject responseObj = new JSONObject(jsonResponse);  
  
    ip = responseObj.getString("origin");  
    System.out.println("ip : " + ip);  
  
} catch (IOException e) { e.printStackTrace(); }  
}  
}
```

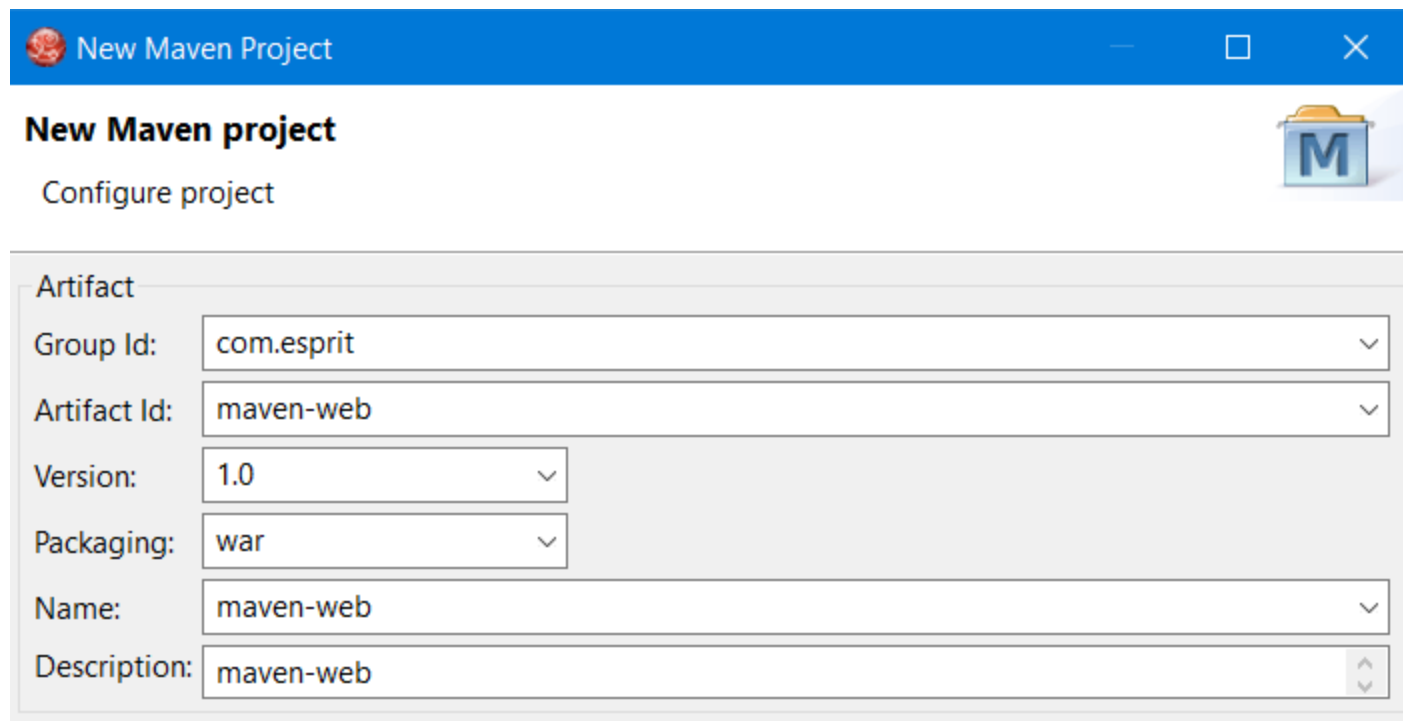
TP1 – Projet avec Maven (JAR)

```
<project ...>
  ...
  <properties>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.json</groupId>
      <artifactId>json</artifactId>
      <version>20160810</version>
    </dependency>
    <dependency>
      <groupId>org.apache.httpcomponents</groupId>
      <artifactId>httpclient</artifactId>
      <version>4.1.1</version>
    </dependency>
  </dependencies>
</project>
```

TP2 – Projet Web avec Maven (WAR)

- Créer un nouveau projet de type Maven, simple (sans archetype)
- Projet de type **WAR**



New Maven Project

New Maven project

Configure project

Artifact

Group Id: com.esprit

Artifact Id: maven-web

Version: 1.0

Packaging: war

Name: maven-web

Description: maven-web

TP2 – Projet Web avec Maven (WAR)

- Corriger l'erreur et le warning ci-dessous :

The screenshot shows the Red Hat JBoss Developer Studio interface. The Project Explorer on the left shows a project named 'maven-web' with a 'pom.xml' file selected. The main editor displays the content of 'pom.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/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.esprit</groupId>
  <artifactId>maven-web</artifactId>
  <version>1.0</version>
  <packaging>war</packaging>
  <name>maven-web</name>
  <description>maven-web</description>
</project>
```

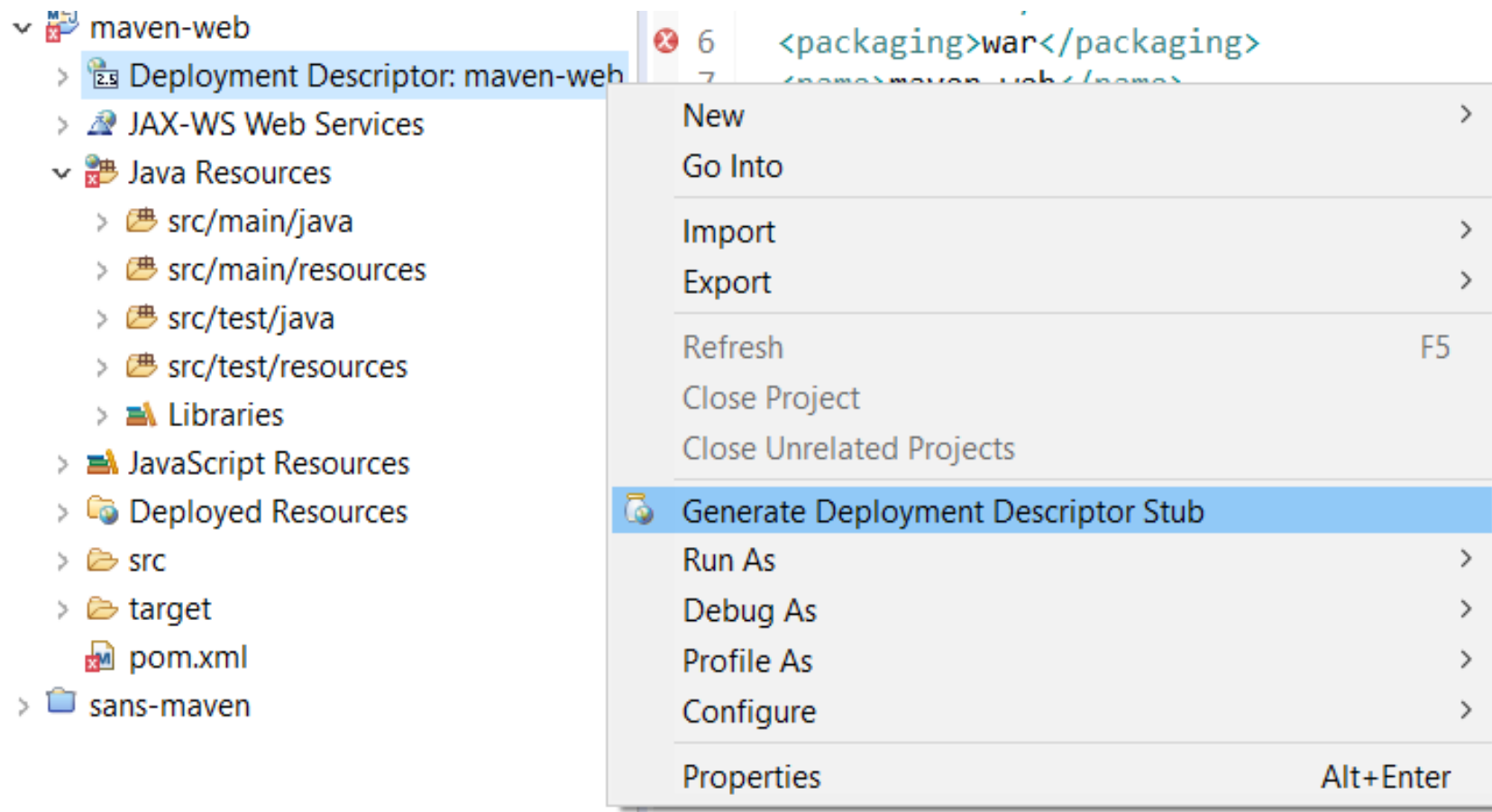
Below the editor, the 'Problems' tab is active, showing 1 error and 1 warning:

Description	Resource
web.xml is missing and <failOnMissingWebXml> is set to true	pom.xml
Build path specifies execution environment J2SE-1.5. There are no JREs installed	maven-web

At the bottom, it states: 2 items selected: 1 error, 1 warning, 0 others

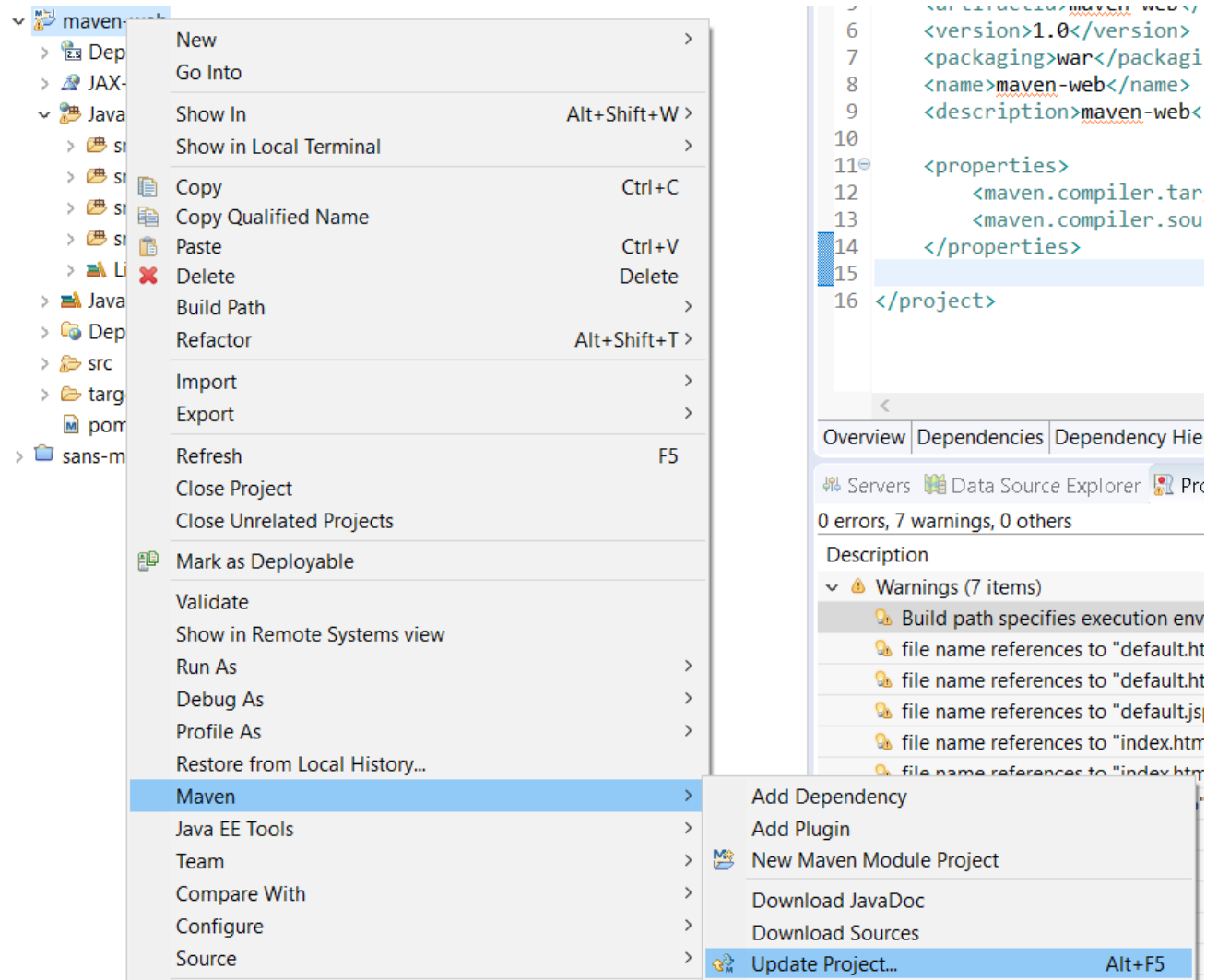
TP2 – Projet Web avec Maven (WAR)

- Correction de l'erreur « web.xml missing », tout projet web doit contenir un fichier web.xml, cliquer sur « Generate ... » pour le générer :



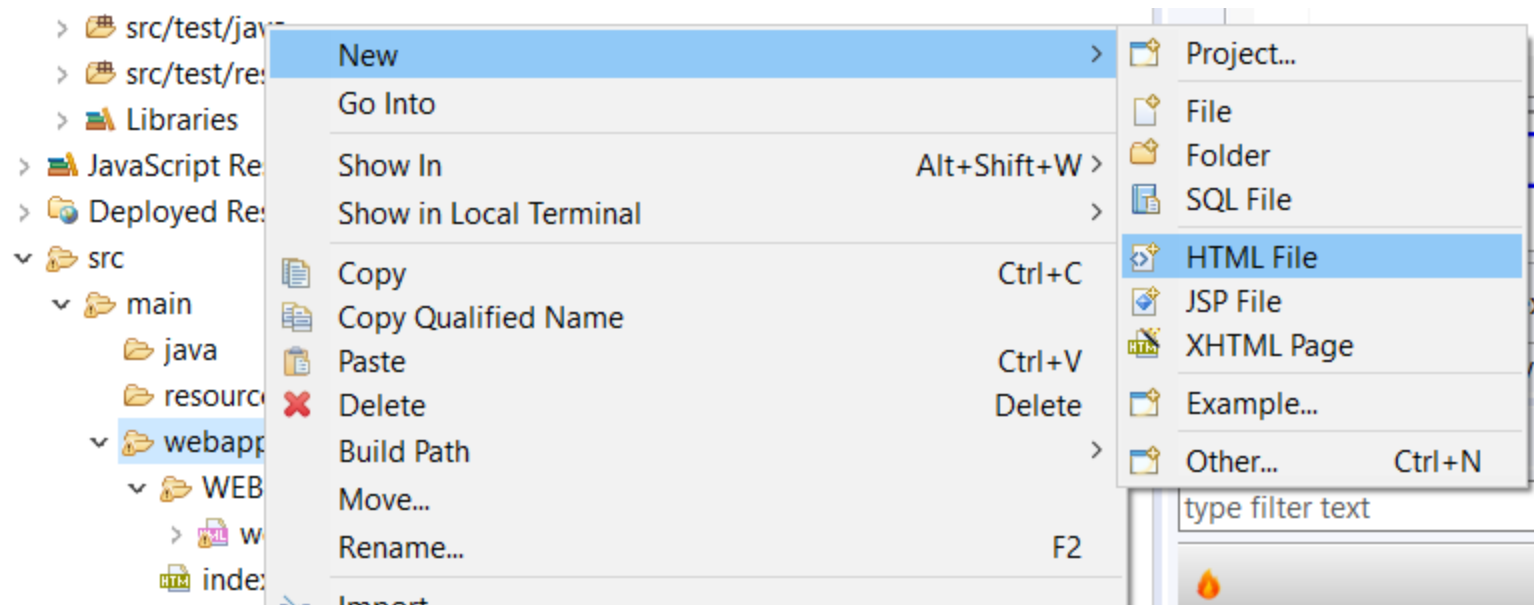
TP2 – Projet Web avec Maven (WAR)

- Faites un Maven Update.



TP2 – Projet Web avec Maven (WAR)

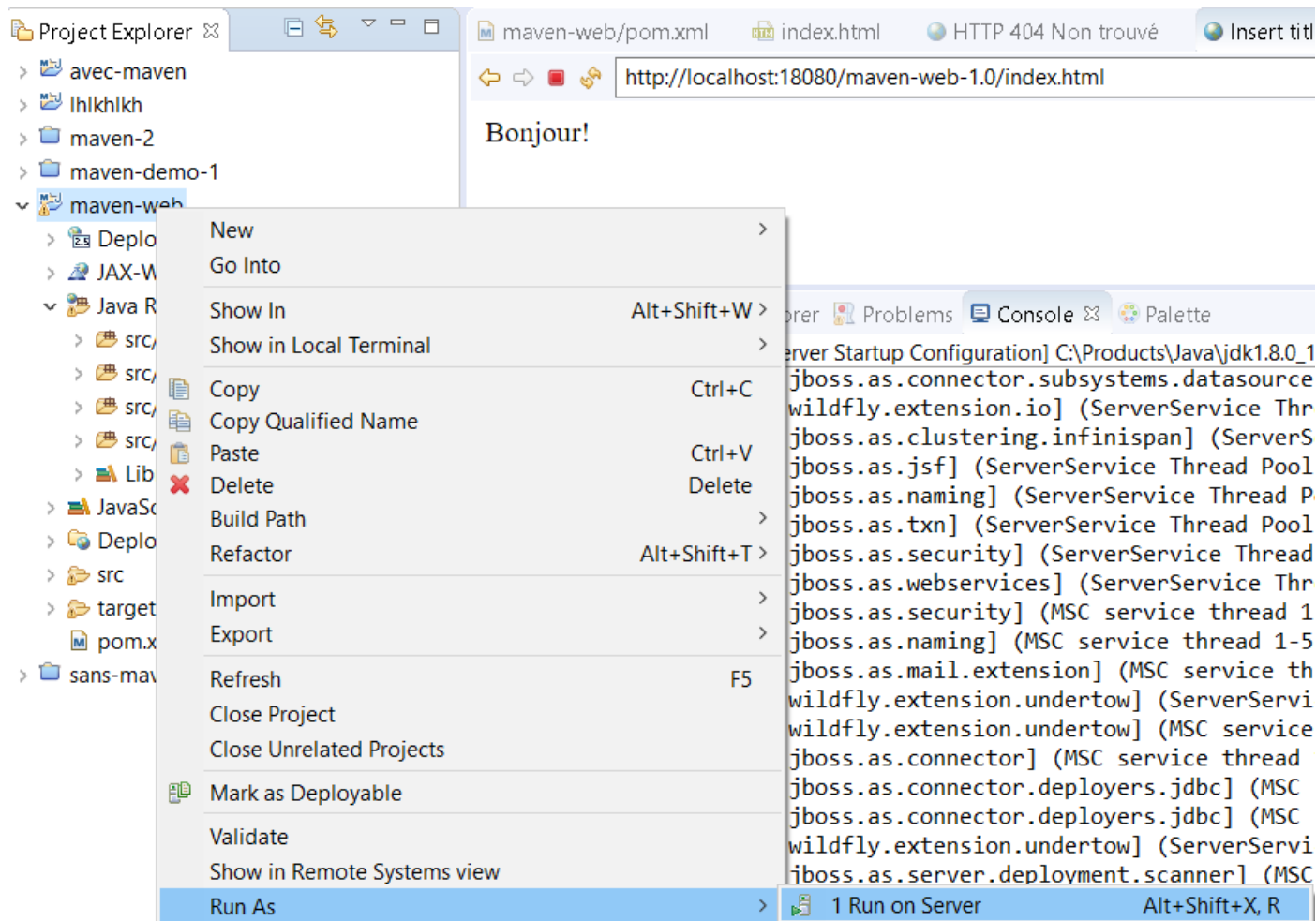
- Ajouter une page web basique index.html :



TP2 – Projet Web avec Maven (WAR)

- Déployer l'application sur Tomcat, et lancer l'URL :

<http://localhost:8080/maven-web>



MAVEN

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP ASI
(Architectures des Systèmes d'Information)

Bureau E204