Destinataires : Tout le personnel du CATI SICPA

Date de révision : Février 2021

Utiliser Maven pour déployer ses webservices

Avant de commencer

Cette documentation va vous présenter comment installer maven, le configurer dans Eclipse et l'utiliser pour générer/déployer vos war. Pour cela, laissez-vous guider dans la documentation à la façon d'un livre dont vous êtes le héros :

- C'est votre première lecture de cette documentation ? Lisez attentivement les points n°1, n°2 et n°3
- Vous voulez migrez un webservice existant ? Rendez-vous au point n°4
- Vous voulez créer un nouveau webservice ? Rendez-vous au point n°5

Table des matières

Avant de commencer 1
1. Installation de Maven 3
2. Intégrer Maven dans Eclipse 5
3. Préambule sur la structure des fichiers pom.xml 6
4. Convertir un projet java de webservices existant en projet Maven 7
5. Créer un nouveau projet java de webservices à l'aide de Maven 10
6. Compiler et packager un projet à l'aide de Maven 15
7. Gérer les configurations production et recette à l'aide de Maven 16
8. Déployer une archive à l'aide de Maven 20
9. Principales dépendances MAVEN pour nos webservices 24

Annexe 1 : Le fichier **pom.xml** dans le cadre de la migration d'un webservice existant 26

Annexe 2 : Le fichier **pom.xml** dans le cadre de la création d'un nouveau webservice 29

Au risque de me répéter, je ne saurais que trop vous conseiller d'installer, avant de commencer cette documentation, cet excellent émulateur de console qu'est CMDer : https://cmder.net/. Il vous sera utile tout au long de la documentation et essentiel lors de la partie 7 pour la mise en place de l'authentification par clés SSH

A quoi sert Maven ?

Maven est un outil permettant d'automatiser la gestion de projets Java. Il offre les fonctionnalités suivantes :

- Compilation et déploiement des applications Java *(JAR, WAR)*
- Gestion des librairies requises par l'application
- Exécution des tests unitaires
- Intégration dans différents IDE *(Eclipse, IntelliJ, Netbeans, Jbuilder, …)*

Problématique

Le déploiement d'applications est devenu aujourd'hui un vrai casse-tête. En effet, à chaque phase du projet, les développeurs doivent gérer un environnement différent : intégration, recette, pré-production, production. Chaque environnement possède ses propres caractéristiques : adresses IP, serveurs de bases de données, etc. Maven permet donc de s'affranchir de ces contraintes et d'uniformiser le déploiement des applications via un fichier de configuration de la compilation, des tests et du déploiement *(le fichier **pom.xml**)* et l'accès à des dépendances/plugins/extensions distantes *(les repository maven)*

Project Object Model (POM)

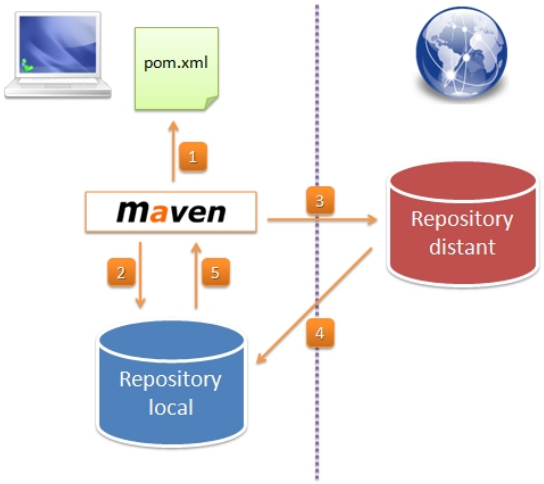
Maven utilise un paradigme connu sous le nom de Project Object Model *(POM)* afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches prédéfinies, comme la compilation de code Java ou encore sa modularisation. Les plugins les représentants sont automatiquement téléchargés lors de la première exécution de maven, il n'est donc nul besoin de les inclure dans chacun de vos fichiers pom.xml

Exemples : maven-archetype-plugin, maven-clean-plugin, maven-compiler-plugin, maven-deploy-plugin, maven-install-plugin, maven-jar-plugin, maven-war-plugin, … et tant d'autres !!!

Repository Maven

Afin de bien gérer les dépendances, Maven s'appuie sur des repositories de librairies *(jar)*. Ces repositories peuvent être locaux à la machine ou distants, accessibles via HTTP/HTTPS. A sa première exécution, Maven télécharge les différents plugins dont il a besoin et les installe dans le répertoire maven situé dans le répertoire de travail de l'utilisateur : c'est le repository local *(par défaut, il se trouve dans C:\Users\<username>\.m2\repository)*. Ainsi, ces mêmes librairies peuvent être réutilisées entre les différents projets et Maven ne téléchargera que les nouvelles librairies pour un nouveau projet.

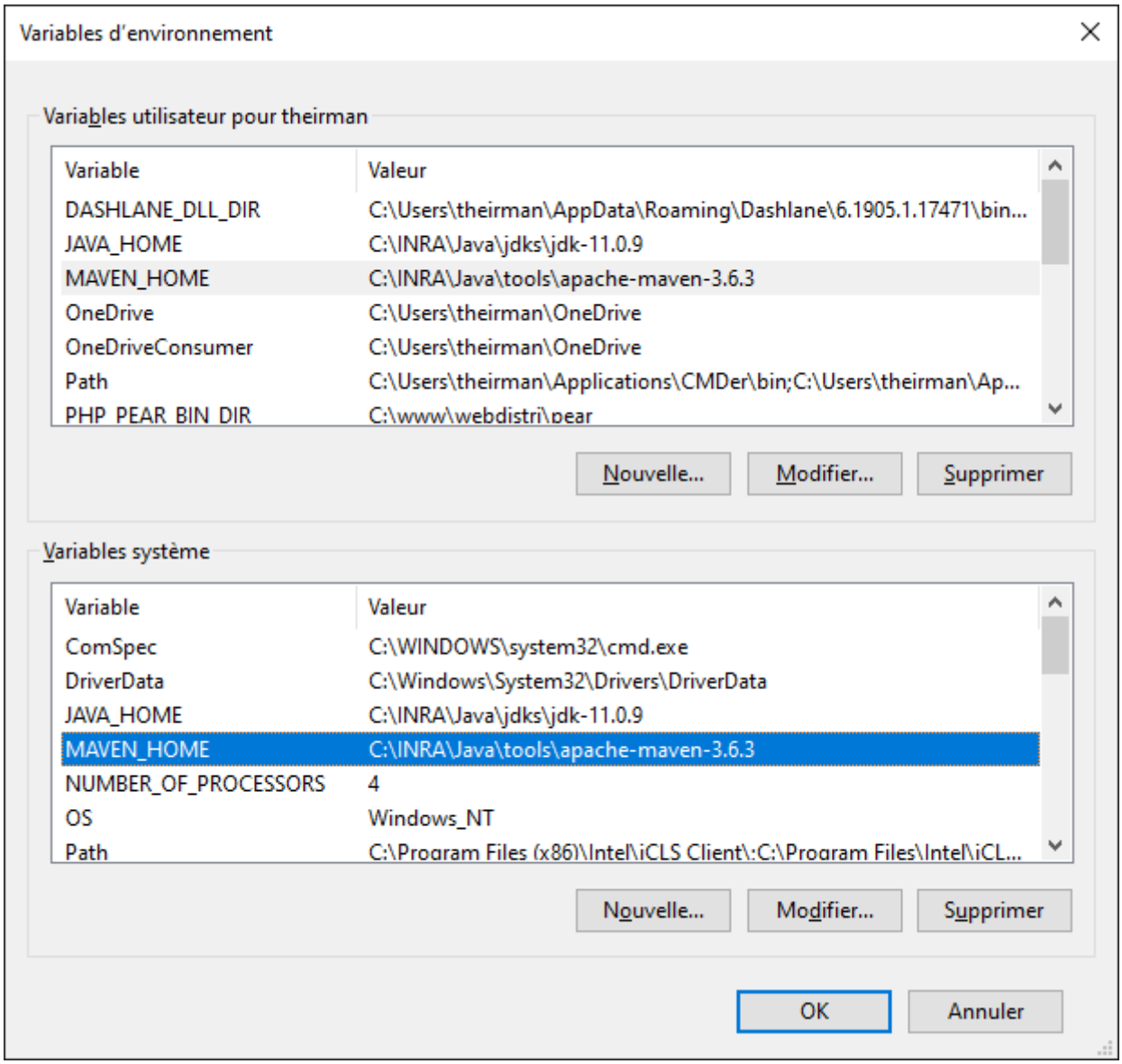
Et donc, comment fonctionne Maven ?



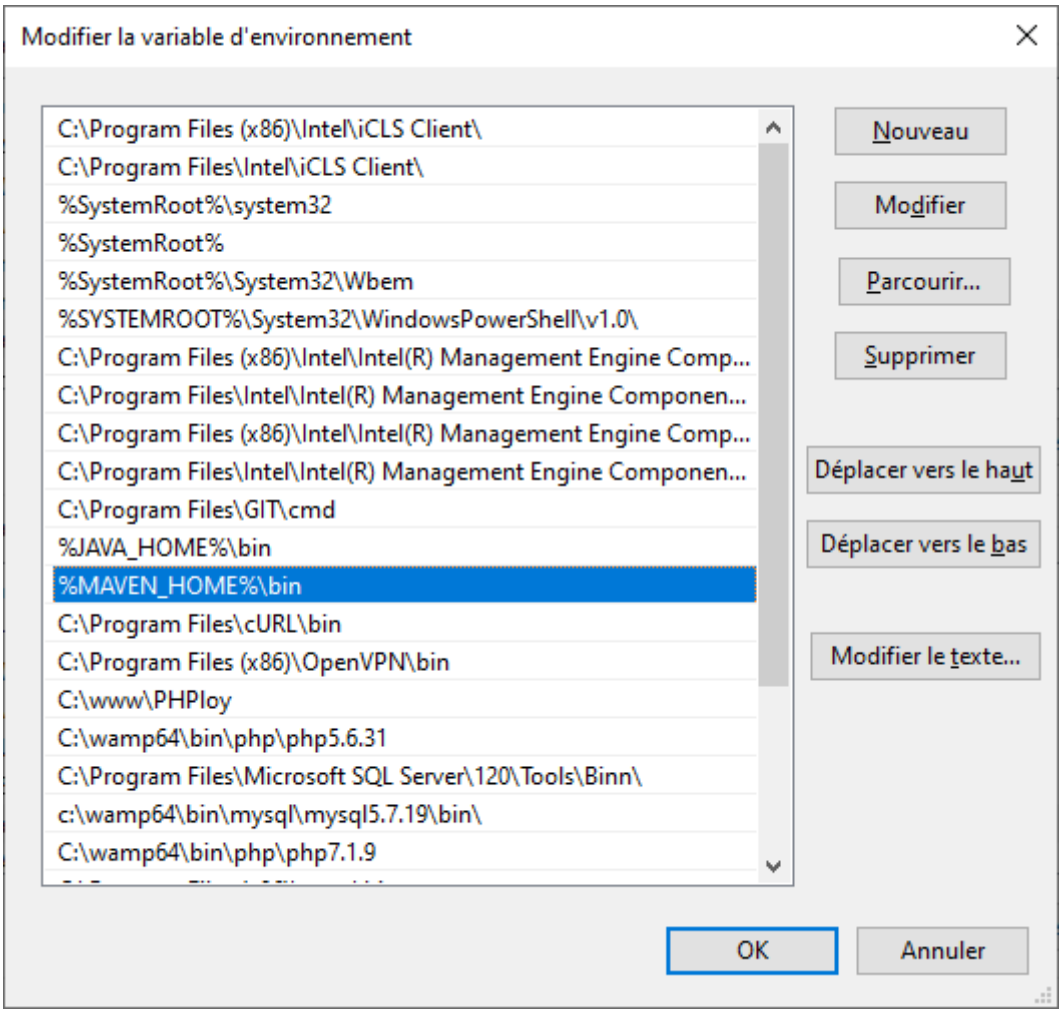
1. Maven récupère la liste des dépendances nécessaires au projet en lisant le fichier pom.xml
2. Maven interroge le repository local afin de trouver les dépendances utilisées.
3. Si la dépendance n'est pas trouvée, alors Maven va interroger les repositories distants.
4. Les dépendances absentes du repository local sont alors téléchargées pour qu'elles soient disponibles lors des prochains builds.
5. Maven peut maintenant utiliser la dépendance pour la construction du projet.

1. Installation de Maven

- Télécharger Maven ici : https://maven.apache.org/download.cgi
- Dézipper l'archive dans votre dossier C:\...\java\tools *(ici, C:\inra\java\tools\apache-maven-3.6.3)*
- Ouvrir la page des paramètres et aller dans le menu « Système »
- Sélectionner l'onglet « A propos de », puis cliquer sur le lien « Paramètres avancés du système »
- Cliquer sur « Variables d'environnement »
- Si besoin, ajouter JAVA_HOME avec le chemin vers votre JDK *(ici, C:\INRA\Java\jdks\jdk-11.0.9)*
- Ajouter la variable MAVEN_HOME avec le chemin vers Maven *(ici, C:\INRA\Java\tools\apache-maven-3.6.3)*



- Modifier la variable PATH en ajoutant la valeur Maven et, si besoin, la valeur java comme suit :



Redémarrer la session Windows afin de prendre en compte les modifications ci-dessus.

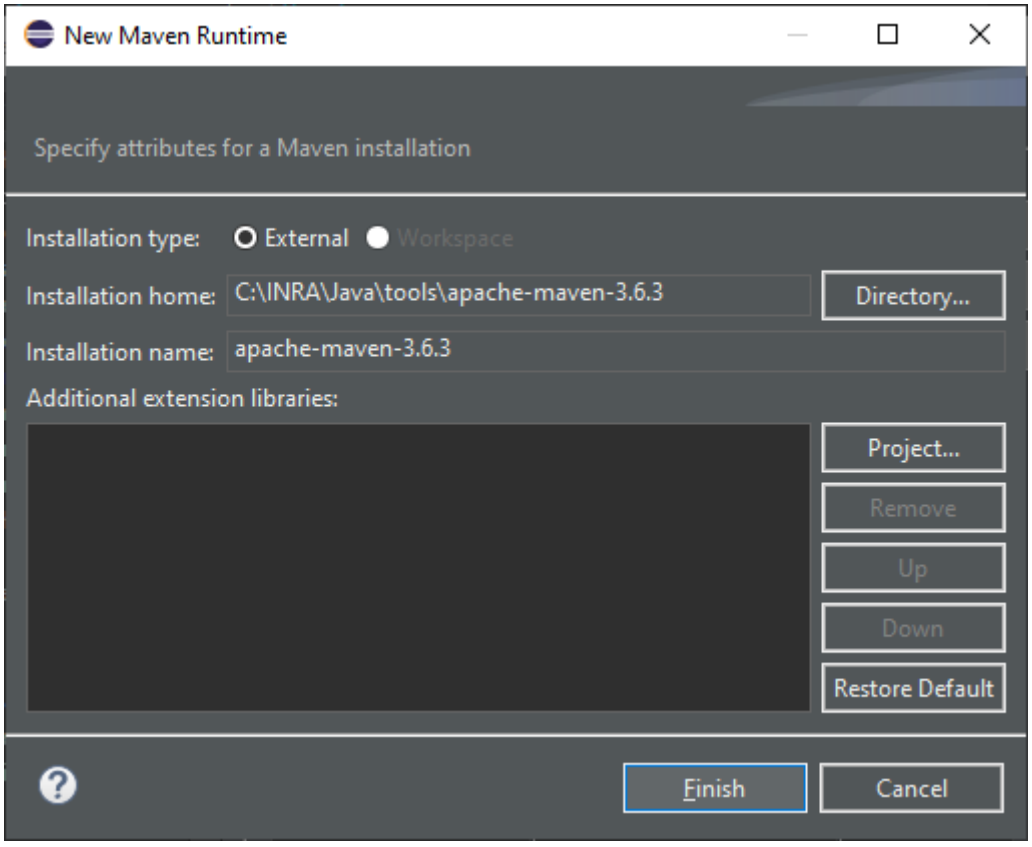
Lancer une invite de commande et tester que vos utilitaires sont bien reconnus :

```
C:\Users\theirman
λ java -version
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (Zulu 8.40.0.25-CA-win64) (build 1.8.0_222-b10)
OpenJDK 64-Bit Server VM (Zulu 8.40.0.25-CA-win64) (build 25.222-b10, mixed mode)

C:\Users\theirman
λ mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\INRA\Java\tools\apache-maven-3.6.3\bin\..
Java version: 1.8.0_222, vendor: Azul Systems, Inc., runtime: C:\INRA\Java\jdks\jdk-1.8.0_222\jre
Default locale: fr_FR, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

2. Intégrer Maven dans Eclipse

Note : Pour poursuivre confortablement à travers cette documentation, il est préférable que vous installiez une version d'eclipse 2020 (cette documentation a été écrite à partir d'eclipse-2020-09, une version plus récente nommée eclipse-2020-12 existe)

- Lancer Eclipse
- Aller dans **Window** > **Preferences**
- Cliquer sur **Maven** > **Installations**
- Cliquer sur **Add**
- Saisir votre home maven *(C:\INRA\Java\tools\apache-maven-3.6.3)* dans **Installation home**
- Saisir un nom pour votre home maven *(apache-maven-3.6.3)* dans **Installation name**
- Cliquer sur **Finish**
- Sélectionner le home maven que vous venez de déclarer



Pour savoir comment migrer un projet existant, rendez-vous au point n°4 *(page 7)*
Pour savoir comment créer un nouveau projet, rendez-vous au point n°5 *(page 10)*
Dans tous les cas, prenez le temps de lire attentivement le point n°3 *(page 6)*

3. Préambule sur la structure des fichiers pom.xml

La structure d'un fichier maven pom.xml est la suivante :

- La balise racine s'appelle **<project>**
- Elle doit obligatoirement contenir :
  - Une balise de version du fichier pom.xml : <modelVersion>
  - Des balises d'identification du projet maven : <groupId>, <articfactId> et <version>
  - Une balise pour définir le type de déploiement : <packaging>
  - Une balise pour gérer les dépendances du projet : <dependencies>
  - Une balise pour gérer les paramètres de compilation du projet : <build>

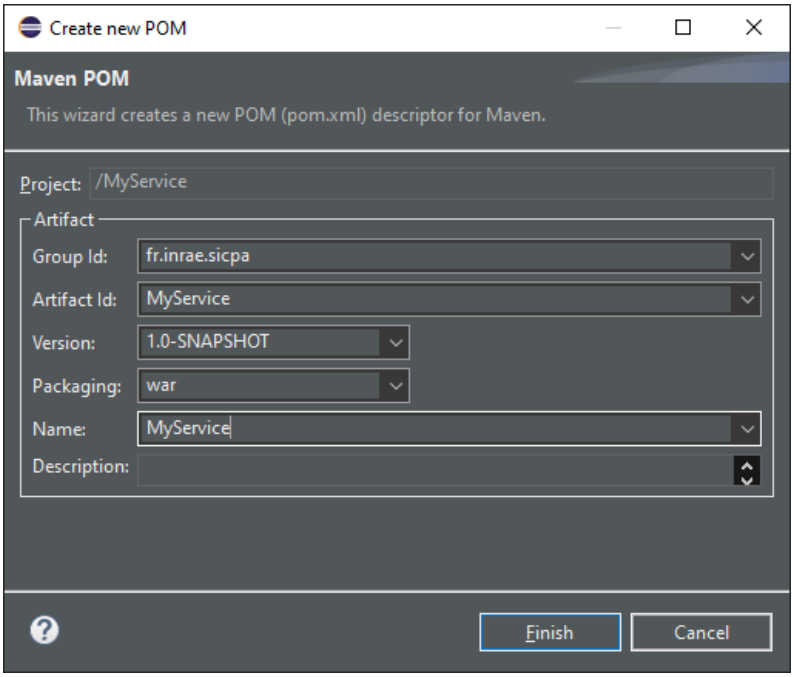
- Elle peut également contenir :
  - Une balise pour définir le nom du projet : <name>
  - Une balise pour définir les propriétés du projet : <properties>
  - Une balise pour gérer différents types de déploiement : <profiles>

Que vous migriez un projet existant ou créiez un nouveau projet, je vous conseille de **toujours** adopter la structure XML ci-dessous, quitte à laisser des balises vides.

```xml
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>fr.inrae.sicpa</groupId>
  <artifactId>MyService</artifactId>
  <version>1.0-SNAPSHOT</version>

  <packaging>war</packaging>
  <name>MyService</name>

  <properties>
  </properties>

  <dependencies>
  </dependencies>

  <profiles>
  </profiles>

  <build>
    <resources>
    </resources>

    <plugins>
    </plugins>

    <extensions>
    </extensions>
  </build>

</project>
```

4. Convertir un projet java de webservices existant en projet Maven

- Cliquer droit sur votre projet
- Cliquer sur **Configure** > **Convert to Maven Project**
- Group Id : fr.inrae.sicpa
- Artifact Id : le nom de votre service
- Version : 1.0-SNAPSHOT
- Packaging : war
- Cliquer sur **Finish**



- Editer le fichier **pom.xml**
- Si ce n'est pas encore fait, mettre en place la structure XML décrite dans le point 3
- Dans la balise **<properties>**, ajouter les propriétés suivantes

```xml
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.version>3.1</maven.compiler.version>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>
```

- Dans la balise **<dependencies>**, ajouter les dépendances suivantes

REST

```xml
<!-- https://mvnrepository.com/artifact/jakarta.ws.rs/jakarta.ws.rs-api -->
<dependency>
  <groupId>jakarta.ws.rs</groupId>
  <artifactId>jakarta.ws.rs-api</artifactId>
  <version>2.1.2</version>
</dependency>
```

SOAP

```
<!-- https://mvnrepository.com/artifact/jakarta.jws/jakarta.jws-api -->
<dependency>
  <groupId>jakarta.jws</groupId>
  <artifactId>jakarta.jws-api</artifactId>
  <version>2.1.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api -->
<dependency>
  <groupId>jakarta.xml.bind</groupId>
  <artifactId>jakarta.xml.bind-api</artifactId>
  <version>2.3.3</version>
</dependency>
<!-- https://mvnrepository.com/artifact/jakarta.xml.soap/jakarta.xml.soap-api -->
<dependency>
  <groupId>jakarta.xml.soap</groupId>
  <artifactId>jakarta.xml.soap-api</artifactId>
  <version>1.4.2</version>
</dependency>
```

- Dans la balise **<build>**, ajouter :

```
<build>
  <finalName>TraceLBSoapWS</finalName>
  <sourceDirectory>${basedir}/src</sourceDirectory>
  <testSourceDirectory>${basedir}/test</testSourceDirectory>
```

- Encore dans la balise **<build>**, ajouter ou mettre à jour les ressources comme suit :

```
<resources>
  <resource>
    <directory>${basedir}/resources</directory>
    <filtering>true</filtering>
    <includes>
      <include>*.properties</include>
      <include>*.xml</include>
    </includes>
    <excludes>
      <exclude>**/*.java</exclude>
    </excludes>
  </resource>
</resources>
```

- Toujours dans la balise **<build>**, configurer le plugin **maven-war-plugin** comme suit :

```
<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-war-plugin -->
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <version>3.2.3</version>
  <configuration>
    <failOnMissingWebXml>false</failOnMissingWebXml>
    <warSourceDirectory>${basedir}/WebContent</warSourceDirectory>
    <warSourceExcludes>${basedir}/WebContent/WEB-INF/web.xml</warSourceExcludes>
  </configuration>
</plugin>
```

- Enfin dans la balise **<build>**, configurer le plugin **maven-compiler-plugin** comme suit :

```
<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-compiler-plugin -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>3.8.1</version>
</plugin>
```

- Dans **/WebContent** > **WEB-INF**
  - Cliquer-droit sur le fichier **glassfish-web.xml**
  - Sélectionner **Refactor**
  - Sélectionner **Rename**
  - Renommer le fichier en **payara-web.xml**

- Editer le fichier **/WebContent** > **WEB-INF** > **payara-web.xml**
- Remplacer son contenu comme suit :

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

<!DOCTYPE payara-web-app PUBLIC
          "-//Payara.fish//DTD Payara Server 4 Servlet 3.0//EN"
          "https://raw.githubusercontent.com/payara/Payara-Server-Documentation/master/schemas/payara-web-app_4.dtd">
<payara-web-app>
    <context-root>/MyService</context-root>
</payara-web-app>
```

- Les ressources telles que les fichiers d'options, de configuration, etc doivent se trouver dans le dossier **/resources**. Si ce n'est pas le cas, les-y copier.
- Supprimer, s'il existe, le dossier **lib** situé dans le dossier **/WebContent** > **WEB-INF**

Pour finir :

- Cliquer droit sur le projet
- Cliquer sur **Maven** > **Update project**
- Cliquer sur **OK** pour recompiler votre projet à l'aide de maven et du nouveau pom.xml
- Normalement, tout est OK à ce niveau. Sinon, pensez à débuguer vos scripts

Pour savoir comment compiler votre projet et générer votre archive war, rendez-vous au point n°6 *(page 15)*

5. Créer un nouveau projet java de webservices à l'aide de Maven

Pour les nouveaux webservices, nous allons utiliser un archétype nommé **maven-archetype-quickstart** et nous conformer aux bonnes pratiques en matière d'emplacement des sources, soit :

- Le dossier des sources dans **/src/main/java**
- Le dossier des ressources dans **/src/main/resources**
- Le dossier des tests dans **/src/test/java**

- Ouvrir un terminal et se placer dans le workspace
- Lancer la commande ci-dessous pour générer votre projet :

```
mvn archetype:generate --batch-mode -DarchetypeGroupId=org.apache.maven.archetypes
                                    -DarchetypeArtifactId=maven-archetype-quickstart
                                    -DarchetypeVersion=1.1
                                    -DgroupId=fr.inrae.sicpa
                                    -DartifactId=MyServiceWS
                                    -Dversion=1.0-SNAPSHOT
                                    -Dpackage=fr.inrae.sicpa
```

- o `-DgroupId` : la structure à laquelle appartient le projet
- o `-DartifactId` : le nom du projet
- o `-Dversion` : la version du projet
- o `-Dpackage` : par convention, identique au groupId

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace
λ mvn archetype:generate --batch-mode -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-quickstar
t -DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=fr.inrae.sicpa -DartifactId=MyServiceWS -Dversion=1.0-SNAPSHOT -Dpackag
e=fr.inrae.sicpa
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] Archetype [org.apache.maven.archetypes:maven-archetype-quickstart:1.4] found in catalog remote
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: fr.inrae.sicpa
[INFO] Parameter: artifactId, Value: MyServiceWS
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: fr.inrae.sicpa
[INFO] Parameter: packageInPathFormat, Value: fr/inrae/sicpa
[INFO] Parameter: package, Value: fr.inrae.sicpa
[INFO] Parameter: groupId, Value: fr.inrae.sicpa
[INFO] Parameter: artifactId, Value: MyServiceWS
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.010 s
[INFO] Finished at: 2021-01-15T16:17:27+01:00
[INFO] ------------------------------------------------------------------------
```

- Supprimer le fichier **MyServiceWS/src/main/java/fr/inrae/sicpa/App.java**
- Supprimer le fichier **MyServiceWS/src/test/main/java/fr/inrae/sicpa/AppTest.java**
- Dans le dossier **MyServiceWS/src/main**, créer le dossier **resources**
- Dans le dossier **MyServiceWS/src/main/java/fr/inrae/sicpa**, créer les dossiers **dao**, **exceptions**, **helper**, **model** et **services**

- Vérifier ensuite que l'arborescence soit correcte, comme ci-dessous :

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace
λ tree MyServiceWS\
Structure du dossier pour le volume OS
Le numéro de série du volume est 701A-0796
C:\INRA\JAVA\IDES\ECLIPSE-JAVA-2020-09\WORKSPACE\MYSERVICEWS
└───src
    ├───main
    │   ├───java
    │   │   └───fr
    │   │       └───inrae
    │   │           └───sicpa
    │   │               ├───dao
    │   │               ├───exceptions
    │   │               ├───helper
    │   │               ├───model
    │   │               └───services
    │   └───resources
    └───test
        ├───java
        │   └───fr
        │       └───inrae
        │           └───sicpa
        └───resources
```

- Lancer Eclipse
- Cliquer sur **File** > **Import**
- Sélectionner **Maven** > **Existing Maven Project**
- Cliquer sur **Next**
- Si besoin, se repositionner dans le dossier du workspace
- Sélectionner le projet dans la liste
- Cliquer sur **Finish**

- Editer le fichier **pom.xml**
- Si ce n'est pas encore fait, mettre en place la structure XML décrite dans le point 3
- Changer le packaging en le passant en **war**

```xml
<groupId>fr.inrae.sicpa</groupId>
<artifactId>MyServiceWS</artifactId>
<version>1.0</version>
<packaging>war</packaging>
<name>MyServiceWS</name>
```

- Dans la balise **<properties>**, ajouter les propriétés suivantes :

```xml
<properties>
  <encoding>UTF-8</encoding>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>
```

- Dans la balise **<dependencies>**, ajouter la dépendance suivante pour ajouter le support de REST :

```xml
<!-- https://mvnrepository.com/artifact/javax.ws.rs/javax.ws.rs-api -->
<dependency>
  <groupId>javax.ws.rs</groupId>
  <artifactId>javax.ws.rs-api</artifactId>
  <version>2.1.1</version>
</dependency>
```

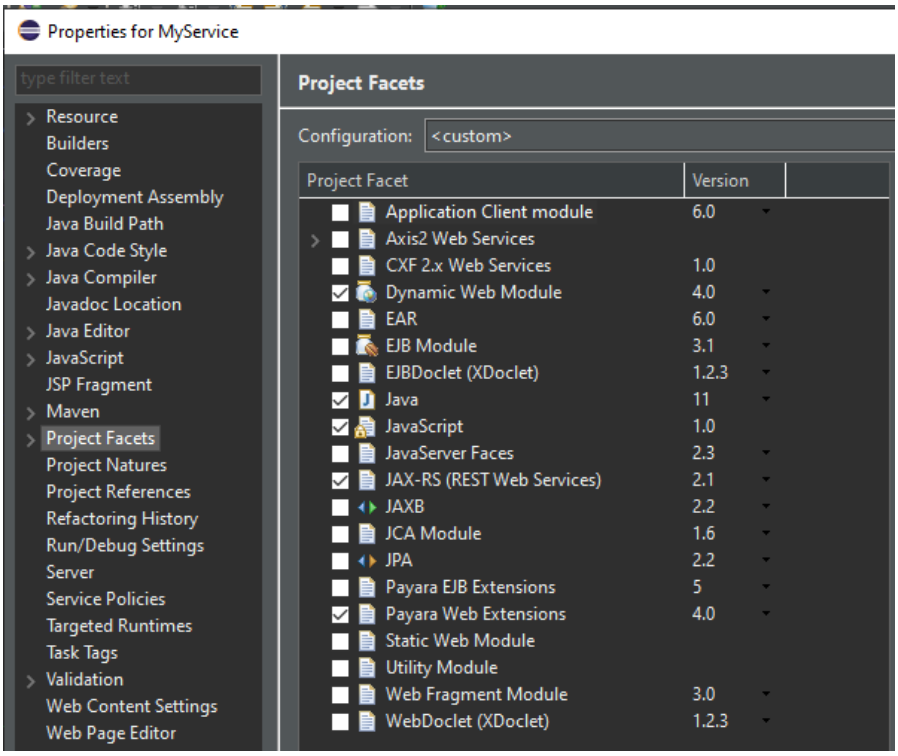
- Dans la balise **<dependencies>**, ajouter les dépendances suivantes pour ajouter le support de SOAP :

```
<!-- https://mvnrepository.com/artifact/jakarta.jws/jakarta.jws-api -->
<dependency>
  <groupId>jakarta.jws</groupId>
  <artifactId>jakarta.jws-api</artifactId>
  <version>2.1.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api -->
<dependency>
  <groupId>jakarta.xml.bind</groupId>
  <artifactId>jakarta.xml.bind-api</artifactId>
  <version>2.3.3</version>
</dependency>
<!-- https://mvnrepository.com/artifact/jakarta.xml.soap/jakarta.xml.soap-api -->
<dependency>
  <groupId>jakarta.xml.soap</groupId>
  <artifactId>jakarta.xml.soap-api</artifactId>
  <version>1.4.2</version>
</dependency>
```

- Modifier les informations de build comme suit :

```
<finalName>WebDistriWS</finalName>
<directory>${basedir}/target</directory>
<!-- main -->
<sourceDirectory>${basedir}/src/main/java</sourceDirectory>
<outputDirectory>${basedir}/target/classes</outputDirectory>
<!-- test -->
<testSourceDirectory>${basedir}/src/test/java</testSourceDirectory>
<testOutputDirectory>${basedir}/target/test-classes</testOutputDirectory>
```

- Cliquer-droit sur le projet et sélectionner **Properties**
- Cliquer sur **Project Facets**
- Cocher **Dynamic Web Module** et lui attribuer la valeur **4.0**
- Cocher **Java** et lui attribuer, si besoin, la valeur **11**
- Cocher **Payara Web Extensions** et lui attribuer, si besoin, la valeur **4.0**
- Cliquer sur **Apply and Close**



Pour la suite de ce tuto, je vais utiliser l'interface **IMyService** et la classe **MyServiceImpl** *(oui, je sais, c'est très original)* telles que :

```java
package fr.inrae.sicpa.services;

public interface IMyService
{
    public double getPrixTTC( double ht );
    public double getPrixHT( double ttc );
    public double getTVA( double prix, boolean isTTC );
}
```

```java
package fr.inrae.sicpa.services;

import javax.jws.WebMethod;

@WebService(serviceName="MyServiceWS")
@SOAPBinding(style=Style.DOCUMENT, use=Use.LITERAL)
public class MyServiceImpl implements IMyService
{
    private final static double TAUX = 0.196;

    @WebMethod(operationName="getPrixTTC")
    @WebResult(name = "ttc")
    @Override
    public double getPrixTTC(double ht)
    {
        return ht * (1 + TAUX);
    }

    @WebMethod(operationName="getPrixHT")
    @WebResult(name = "ht")
    @Override
    public double getPrixHT(double ttc)
    {
        return ttc / (1 + TAUX);
    }

    @WebMethod(operationName="getTVA")
    @WebResult(name = "tva")
    @Override
    public double getTVA(double prix, boolean isTTC)
    {
        if(isTTC)
            return prix - (prix/(1 + TAUX));
        else
            return prix * TAUX;
    }
}
```

- Pour générer le fichier web.xml dans le dossier **/src/main/webapp/WEB-INF**
  - Cliquer-droit sur le projet
  - Sélectionner **Java EE Tools**
  - Sélectionner **Generate Deployment Descriptor Stub**

- Dans le dossier **/WebContent/WEB-INF**, renommer le fichier **glassfish-web.xml** en **payara-web.xml**
- Modifier le fichier tel que :

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

<!DOCTYPE payara-web-app PUBLIC
         "-//Payara.fish//DTD Payara Server 4 Servlet 3.0//EN"
         "https://raw.githubusercontent.com/payara/Payara-Server-Documentation/master/schemas/payara-web-app_4.dtd">
<payara-web-app>
    <context-root>/MyService</context-root>
</payara-web-app>
```

- Editer le fichier **pom.xml**
- Dans la balise **<build>**, ajouter ou mettre à jour les ressources comme suit :

```
<resources>
  <resource>
    <directory>${basedir}/src/main/resources</directory>
    <filtering>true</filtering>
    <includes>
      <include>*.properties</include>
      <include>*.xml</include>
    </includes>
    <excludes>
      <exclude>**/*.java</exclude>
    </excludes>
  </resource>
</resources>
```

- Dans la balise **<pluginManagement>**, configurer le plugin **maven-war-plugin** comme suit :

```
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <version>3.2.3</version>
  <configuration>
    <failOnMissingWebXml>false</failOnMissingWebXml>
    <warSourceDirectory>${basedir}/src/main/webapp/WebContent</warSourceDirectory>
    <warSourceExcludes>${basedir}/src/main/webapp/WEB-INF/web.xml</warSourceExcludes>
  </configuration>
</plugin>
```

- Cliquer droit sur le projet
- Cliquer sur **Maven** > **Update project**
- Cliquer sur **OK** pour recompiler votre projet à l'aide de maven et du nouveau pom.xml

Pour savoir comment compiler votre projet et générer votre archive war, rendez-vous au point n°6 *(page 15)*

6. Compiler et packager un projet à l'aide de Maven

- Ouvrir un terminal
- Se rendre dans le dossier du projet *(C:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS)*
- Lancer la commande **mvn clean package** pour générer votre nouveau fichier WAR

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< fr.inrae.sicpa:MyServiceWS >----------------------
[INFO] Building MyServiceWS 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ MyServiceWS ---
[INFO] Deleting c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ MyServiceWS ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ MyServiceWS ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\classes
[INFO]
[INFO] --- cxf-java2ws-plugin:3.4.0:java2ws (generate-wsdl) @ MyServiceWS ---
[INFO] Running java2ws in fork mode...
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ MyServiceWS ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ MyServiceWS ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ MyServiceWS ---
[INFO]
[INFO] --- maven-war-plugin:3.2.3:war (default-war) @ MyServiceWS ---
[INFO] Packaging webapp
[INFO] Assembling webapp [MyServiceWS] in [c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS]
[INFO] Processing war project
[INFO] Copying webapp resources [c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\src\main\webapp\WebContent]
[INFO] Webapp assembled in [396 msecs]
[INFO] Building war: c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  15.350 s
[INFO] Finished at: 2021-01-19T11:21:29+01:00
[INFO] ------------------------------------------------------------------------

c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ dir target
 Le volume dans le lecteur C s'appelle OS
 Le numéro de série du volume est 701A-0796

 Répertoire de c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target

19/01/2021  11:21    <DIR>          .
19/01/2021  11:21    <DIR>          ..
19/01/2021  11:21    <DIR>          classes
19/01/2021  11:21    <DIR>          generated-sources
19/01/2021  11:21    <DIR>          maven-archiver
19/01/2021  11:21    <DIR>          maven-status
19/01/2021  11:21    <DIR>          MyServiceWS
19/01/2021  11:21        11 862 128 MyServiceWS.war
19/01/2021  11:21    <DIR>          test-classes
               1 fichier(s)       11 862 128 octets
               8 Rép(s)  332 921 298 944 octets libres
```

Note : les parties 7 et 8 sont optionnelles mais vont permettre de gérer les configurations du projet d'une part, leur déploiement d'autre part.

Pour savoir comment gérer les configurations de votre projet, rendez-vous au point n°7 *(page 16)*
Pour savoir comment déployer votre projet, rendez-vous au point n°8 *(page 20)*

7. Gérer les configurations production et recette à l'aide de Maven

On va ici créer deux profils de déploiements de notre application :

- rec : pour tester notre webservice sur les instances recette de Payara sur sicpa-interop.inra.fr
- prod : pour publier notre webservice sur les instances production de Payara sur sicpa-interop.inra.fr

- Dans le dossier **/resources**, créer les dossiers **rec** et **prod**
- Copier votre (vos) fichier(s) de configuration dans chacun des répertoires
- Adapter les valeurs de chaque fichier en fonction de l'environnement cible *(exemples ci-dessous)*

```
# Version appli
version=MyServiceWS 1.0
```

```
# Version appli
version=MyServiceWS 1.0-SNAPSHOT
```

- Editer le fichier **pom.xml**
- Créer la section **<profiles>** dans la section **<project>**
- Ajouter une section **<profile>** pour chaque environnement *(exemple avec l'environnement recette)*
    - Dans **<id>**, mettre l'identifiant de l'environnement
    - Dans **<activation>**, créer une balise **<property>** avec **<name>** et **<value>**
    - Dans **<property>,** mettre les informations permettant l'activation de l'environnement
    - Créer les balises **<build>**, **<resources>**, **<resource>**
    - Ajouter le chemin vers le dossier de configuration de l'environnement dans la balise **<directory>**

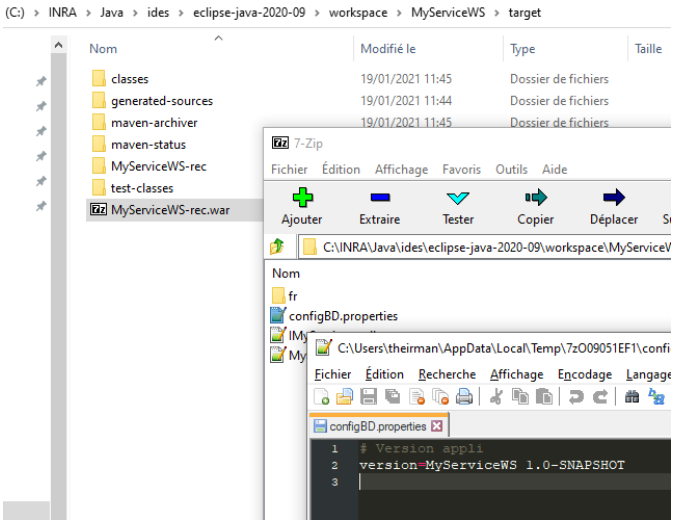
```
<profile>
  <id>rec</id>
  <activation>
    <property>
      <name>env</name>
      <value>rec</value>
    </property>
  </activation>
  <build>
    <finalName>MyServiceWS-rec</finalName>
    <resources>
      <resource>
        <directory>${basedir}/resources/rec</directory>
        <filtering>true</filtering>
        <includes>
          <include>*.properties</include>
        </includes>
      </resource>
    </resources>
  </build>
</profile>
```

- Procéder de même pour l'environnement **production**

```xml
<profile>
  <id>prod</id>
  <activation>
    <property>
      <name>env</name>
      <value>prod</value>
    </property>
  </activation>
  <build>
    <finalName>MyServiceWS-prod</finalName>
    <resources>
      <resource>
        <directory>${basedir}/resources/prod</directory>
        <filtering>true</filtering>
        <includes>
          <include>*.properties</include>
        </includes>
      </resource>
    </resources>
  </build>
</profile>
```

- Compiler le projet pour l'environnement cible : **mvn clean package -Denv=rec**

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ mvn clean package -Denv=rec
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------< fr.inrae.sicpa:MyServiceWS >---------------------
[INFO] Building MyServiceWS 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ MyServiceWS ---
[INFO] Deleting c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target
[INFO]
[INFO] --- maven-enforcer-plugin:1.4.1:enforce (enforce-profile-target) @ MyServiceWS ---
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ MyServiceWS ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO] skip non existing resourceDirectory c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\resources\rec
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ MyServiceWS ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\classes
[INFO]
[INFO] --- cxf-java2ws-plugin:3.4.0:java2ws (generate-wsdl) @ MyServiceWS ---
[INFO] Running java2ws in fork mode...
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ MyServiceWS ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ MyServiceWS ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ MyServiceWS ---
[INFO]
[INFO] --- maven-war-plugin:3.2.3:war (default-war) @ MyServiceWS ---
[INFO] Packaging webapp
[INFO] Assembling webapp [MyServiceWS] in [c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS-rec]
[INFO] Processing war project
[INFO] Copying webapp resources [c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\src\main\webapp\WebContent]
[INFO] Webapp assembled in [331 msecs]
[INFO] Building war: c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS-rec.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  13.601 s
[INFO] Finished at: 2021-01-19T11:33:30+01:00
[INFO] ------------------------------------------------------------------------

c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ dir target
 Le volume dans le lecteur C s'appelle OS
 Le numéro de série du volume est 701A-0796

 Répertoire de c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target

19/01/2021  11:33    <DIR>          .
19/01/2021  11:33    <DIR>          ..
19/01/2021  11:33    <DIR>          classes
19/01/2021  11:33    <DIR>          generated-sources
19/01/2021  11:33    <DIR>          maven-archiver
19/01/2021  11:33    <DIR>          maven-status
19/01/2021  11:33    <DIR>          MyServiceWS-rec
19/01/2021  11:33        11 862 133 MyServiceWS-rec.war
19/01/2021  11:33    <DIR>          test-classes
               1 fichier(s)       11 862 133 octets
               8 Rép(s)  332 917 977 088 octets libres
```



- **(optionnel)** Pour s'assurer qu'au moins un profile est actif à la compilation, on va ajouter le plugin **maven-enforcer-plugin** dans la section **<build>** du projet et le configurer comme suit :

```xml
<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-enforcer-plugin -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.4.1</version>
  <executions>
    <execution>
      <id>enforce-profile-target</id>
      <phase>validate</phase>
      <goals>
        <goal>enforce</goal>
      </goals>
      <configuration>
        <rules>
          <requireActiveProfile>
            <profiles>rec,prod</profiles>
            <all>false</all>
          </requireActiveProfile>
        </rules>
      </configuration>
    </execution>
  </executions>
</plugin>
```

- Ci-dessous, la compilation sans préciser l'environnement dans la commande **mvn**. Un warning nous prévient qu'il manque le profil dans la commande puis la compilation plante.

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< fr.inrae.sicpa:MyServiceWS >----------------------
[INFO] Building MyServiceWS 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ MyServiceWS ---
[INFO]
[INFO]     maven-enforcer-plugin:1.4.1:enforce (enforce-profile-target) @ MyServiceWS
[WARNING] Rule 0: org.apache.maven.plugins.enforcer.RequireActiveProfile failed with message:
Profile "rec" is not activated.
Profile "prod" is not activated.

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.996 s
[INFO] Finished at: 2021-01-19T11:49:28+01:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-enforcer-plugin:1.4.1:enforce (enforce-profile-target) on project MySe
rviceWS: Some Enforcer rules have failed. Look above for specific messages explaining why the rule failed. -> [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/MojoExecutionException
```

- **(optionnel)** Toujours dans le cadre de carrosser notre projet et éviter des déconvenues, on va mettre en place un garde-fou pour éviter de déployer malencontreusement une version snapshot en production. Et pour cela, on va utiliser le plugin **maven-enforcer-plugin** dans le **<build>** du profil **prod** :

```xml
<profile>
  <id>prod</id>
  <activation>
    <property>
      <name>env</name>
      <value>prod</value>
    </property>
  </activation>
  <build>
    <finalName>MyServiceWS-prod</finalName>
    <resources>
      <resource>
        <directory>${basedir}/src/main/resources/prod</directory>
      </resource>
    </resources>
    <plugins>
      <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-enforcer-plugin -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-enforcer-plugin</artifactId>
        <version>1.4.1</version>
        <executions>
          <execution>
            <id>enforce-prod-no-snapshot</id>
            <phase>validate</phase>
            <goals>
              <goal>enforce</goal>
            </goals>
            <configuration>
              <rules>
                <requireReleaseVersion/>
                <requireReleaseDeps/>
              </rules>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</profile>
```

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ mvn clean package -Denv=prod
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< fr.inrae.sicpa:MyServiceWS >----------------------
[INFO] Building MyServiceWS 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ MyServiceWS ---
[INFO] Deleting c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target
[INFO]
[INFO] --- maven-enforcer-plugin:1.4.1:enforce (enforce-profile-target) @ MyServiceWS ---
[INFO]
[WARNING] Rule 0: org.apache.maven.plugins.enforcer.RequireReleaseVersion failed with message:
This project cannot be a snapshot:fr.inrae.sicpa:MyServiceWS:war:1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.494 s
[INFO] Finished at: 2021-01-19T12:02:33+01:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-enforcer-plugin:1.4.1:enforce (enforce-prod-no-snapshot) on project My
ServiceWS: Some Enforcer rules have failed. Look above for specific messages explaining why the rule failed. -> [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/MojoExecutionException
```

Pour savoir comment déployer votre projet, rendez-vous au point n°8 *(page 20)*

8. Déployer une archive à l'aide de Maven

Dans cette partie, on va déployer notre archive dans le répertoire **autodeploy** de Payara à l'aide de Maven. On pourrait directement déployer notre archive dans Payara mais alors on perdrait le filtrage des droits de déploiement par projet que j'ai mis en place grâce au script de déploiement.

Avant de commencer le déploiement, on va préparer l'authentification automatique au serveur à l'aide de clés SSH. Cette opération n'est à faire que la première fois

*Attention : Pour cette partie, vous allez devoir lancer CMDer, la commande **ssh-copy-id** n'étant pas disponible dans la console Windows.*

- Créer la clé SSH :
  - `ssh-keygen -t rsa -b 4096 -f C:\Users\theirman\.ssh\id_rsa`
- Copier la clé publique vers le serveur, grâce à **ssh-copy-id**, pour lui faire connaitre notre identité
  - `sh ssh-copy-id -i c:\users\theirman\.ssh\id_rsa.pub theirman@sicpa-interop.inra.fr`
- Tester la connexion SSH : l'authentification sécurisée se fait désormais par vos clés privée/publique
  - `ssh theirman@sicpa-interop.inra.fr`

```
C:\Users\theirman\.ssh
λ ssh-keygen -t rsa -b 4096 -f C:\Users\theirman\.ssh\id_rsa
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\theirman\.ssh\id_rsa.
Your public key has been saved in C:\Users\theirman\.ssh\id_rsa.pub.
The key fingerprint is:
SHA256:DDPHsf7+hLWWxJ5F3ARszCqb3/65MzNz8331/uXy9hc theirman@TLS-GPS-THEIRMA
The key's randomart image is:
+---[RSA 4096]----+
|        .    +...|
|       . o   .=o |
|      + +    oo .|
|       B  ....   |
|        S  ++ .  |
|         .o= = E.|
|          o.B.  =|
|         . o. oOX|
|          .....O^|
+----[SHA256]-----+

C:\Users\theirman\.ssh
λ sh ssh-copy-id -i c:\users\theirman\.ssh\id_rsa.pub theirman@sicpa-interop.inra.fr
ssh-copy-id: INFO: Source of key(s) to be installed: "c:\users\theirman\.ssh\id_rsa.pub"
ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
theirman@sicpa-interop.inra.fr's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'theirman@sicpa-interop.inra.fr'"
and check to make sure that only the key(s) you wanted were added.


C:\Users\theirman\.ssh
λ ssh theirman@sicpa-interop.inra.fr
Last login: Mon Feb 15 11:19:22 2021 from tls-gps-theirma.inra.local
Last login: Mon Feb 15 11:19:22 2021 from tls-gps-theirma.inra.local
[theirman@sicpa-interop ~]$
```

Pour des raisons qui m'échappent, sur certains postes windows dans le domaine INRA, il arrive que la commande **ssh-copy-id** ne fonctionne pas. Voici une méthode alternative :

- Sur l'ordinateur, afficher le contenu du fichier **id_rsa.pub**
  - `more c:\users\theirman\.ssh\id_rsa.pub`
- Copiez ce contenu dans votre presse-papier
- Se connecter en SSH sur le serveur
- Ajouter sa clé publique dans le fichier **authorized_keys** comme suit :
  - `echo 'ssh-rsa AAAA…F1w== theirman@TLS-GPS-THEIRMA' >> /home/theirman/.ssh/authorized_keys`

```
C:\Users\theirman\.ssh
λ more id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDIo/WBV04XBX5KZ2ZkMiG53Fo+boqoa4eNEMrZ/kFTQR//+BZPzO+VJQIwR9hA1fzb8Zonq8rYr5K
d3T5Kz9/A3LZ14A0TY0IKp5HClCCPS+ynQLPx0bh+MM8jokprWEllzFuUqfSvljsXdnmEOSJmCbq0g2D8vSr7XOHNAz11K12hffgpslBO0tKJ5Keevc
rwwCqQaJM0r2DZRW3UgBvVw4yAauZOT6wWLtHzMeexvLDjOUw5aeUrUTBR2iQGVjki+SQsbRIGmwiFR8qcKCFDWbVDCwFpz63LFcFgl2j/8R8y+MUss
5hjM3A7F9kv68IV4eK6s7JuGoH9xrtBpRyrJsyD6/FC06f81JY1emu6YzqzccpU/1HIk63DyHKV+bplQ6teN8eE//1DbBN82DbK/LeqK4LvCAd09yn4
EUzTUj9EI5VIyPLutH6tFUgz5V9/aKXF45HeKH2+Y0cOVJlFNbm0IP+4C8WLtELDCiNSRiZ/5IWPWyelnjQV33Ecl/r55S7Q5vXgiUxvWsKSSnHTv3Z
PKF7fRdZLD0MZSyvUSDCTT6tD9n5Och4WJTh48SFGqvadmRr9017+HeNkTAtSvzI9/grCFPCpsmn4tg2o6Ypbn/8PEMl3n9WFIxkDxVybnotDftsSHJ
8nBREqHk+4uWQLsKsMrHJcb1kGB+7F1w== theirman@TLS-GPS-THEIRMA

C:\Users\theirman\.ssh
λ ssh theirman@sicpa-interop.inra.fr
theirman@sicpa-interop.inra.fr's password:
Last login: Wed Feb 17 15:06:51 2021 from tls-gps-theirma.inra.local
Last login: Wed Feb 17 15:06:51 2021 from tls-gps-theirma.inra.local
[theirman@sicpa-interop ~]$
[theirman@sicpa-interop ~]$
[theirman@sicpa-interop ~]$ echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDIo/WBV04XBX5KZ2ZkMiG53Fo+boqoa4eNEMrZ/kFT
QR//+BZPzO+VJQIwR9hA1fzb8Zonq8rYr5Kd3T5Kz9/A3LZ14A0TY0IKp5HClCCPS+ynQLPx0bh+MM8jokprWEllzFuUqfSvljsXdnmEOSJmCbq0g2
D8vSr7XOHNAz11K12hffgpslBO0tKJ5KeevcrwwCqQaJM0r2DZRW3UgBvVw4yAauZOT6wWLtHzMeexvLDjOUw5aeUrUTBR2iQGVjki+SQsbRIGmwiF
R8qcKCFDWbVDCwFpz63LFcFgl2j/8R8y+MUss5hjM3A7F9kv68IV4eK6s7JuGoH9xrtBpRyrJsyD6/FC06f81JY1emu6YzqzccpU/1HIk63DyHKV+b
plQ6teN8eE//1DbBN82DbK/LeqK4LvCAd09yn4EUzTUj9EI5VIyPLutH6tFUgz5V9/aKXF45HeKH2+Y0cOVJlFNbm0IP+4C8WLtELDCiNSRiZ/5IWP
WyelnjQV33Ecl/r55S7Q5vXgiUxvWsKSSnHTv3ZPKF7fRdZLD0MZSyvUSDCTT6tD9n5Och4WJTh48SFGqvadmRr9017+HeNkTAtSvzI9/grCFPCpsm
n4tg2o6Ypbn/8PEMl3n9WFIxkDxVybnotDftsSHJ8nBREqHk+4uWQLsKsMrHJcb1kGB+7F1w== theirman@TLS-GPS-THEIRMA' >> /home/thei
rman/.ssh/authorized_keys
[theirman@sicpa-interop ~]$
```

Reste à positionner convenablement les droits …

```
[theirman@sicpa-interop ~]$ chmod 700 /home/theirman/.ssh/
[theirman@sicpa-interop ~]$ chmod 600 /home/theirman/.ssh/authorized_keys
```

… et à retester la connexion sans mot de passe

```
C:\Users\theirman\.ssh
λ ssh theirman@sicpa-interop.inra.fr
Last login: Wed Feb 17 15:08:00 2021 from tls-gps-theirma.inra.local
[theirman@sicpa-interop ~]$
```

C'est OK ! Fermons maintenant la parenthèse SSH et passons au déploiement de nos archives

- Dans le <**build**>, ajouter l'extension **wagon-ssh** pour activer les connexions sécurisées aux serveurs

```
<extensions>
  <!-- https://mvnrepository.com/artifact/org.apache.maven.wagon/wagon-ssh -->
  <extension>
    <groupId>org.apache.maven.wagon</groupId>
    <artifactId>wagon-ssh</artifactId>
    <version>3.4.0</version>
  </extension>
```

- Toujours dans le **<build>**, ajouter le plugin **wagon-maven-plugin** dans la section **<plugins>** :

```
<!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>wagon-maven-plugin</artifactId>
  <version>2.0.0</version>
</plugin>
```

- Dans la section du profil **recette**, créer si besoin la section **<plugins>** dans le **<build>** du profil
- Ajouter et configurer le plugin **wagon-maven-plugin** comme suit en prenant soin de modifier votre nom d'utilisateur et le dossier de déploiement dans l'instruction **<url>**

```
<!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>wagon-maven-plugin</artifactId>
  <version>2.0.0</version>
  <executions>
    <execution>
      <id>rec</id>
      <phase>deploy</phase>
      <goals>
        <goal>upload-single</goal>
      </goals>
      <configuration>
        <fromFile>${project.build.directory}/${project.build.finalName}.${project.packaging}</fromFile>
        <url>scp://theirman@sicpa-interop.inra.fr:/data/apps/payara/autodeploy/group-recette/webdistri</url>
      </configuration>
    </execution>
  </executions>
</plugin>
```

- Procéder de la même façon avec le profil **production :**

```
<!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>wagon-maven-plugin</artifactId>
  <version>2.0.0</version>
  <executions>
    <execution>
      <id>prod</id>
      <phase>deploy</phase>
      <goals>
        <goal>upload-single</goal>
      </goals>
      <configuration>
        <fromFile>${project.build.directory}/${project.build.finalName}.${project.packaging}</fromFile>
        <url>scp://theirman@sicpa-interop.inra.fr:/data/apps/payara/autodeploy/group-production/webdistri</url>
      </configuration>
    </execution>
  </executions>
</plugin>
```

- Compiler, packager et déployer votre archive en recette avec la commande :
  mvn clean package -Denv=rec wagon:upload-single@rec

```
c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS
λ mvn clean package -Denv=rec wagon:upload-single@rec
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------< fr.inrae.sicpa:MyServiceWS >---------------------
[INFO] Building MyServiceWS 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ MyServiceWS ---
[INFO]
[INFO] --- maven-enforcer-plugin:1.4.1:enforce (enforce-profile-target) @ MyServiceWS ---
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ MyServiceWS ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ MyServiceWS ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 2 source files to c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\classes
[INFO]
[INFO] --- cxf-java2ws-plugin:3.4.0:java2ws (generate-wsdl) @ MyServiceWS ---
[INFO] Running java2ws in fork mode...
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ MyServiceWS ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ MyServiceWS ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ MyServiceWS ---
[INFO]
[INFO] --- maven-war-plugin:3.2.3:war (default-war) @ MyServiceWS ---
[INFO] Packaging webapp
[INFO] Assembling webapp [MyServiceWS] in [c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS-rec]
[INFO] Processing war project
[INFO] Copying webapp resources [c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\src\main\webapp\WebContent]
[INFO] Webapp assembled in [362 msecs]
[INFO] Building war: c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS-rec.war
[INFO]
[INFO] ---------------------< fr.inrae.sicpa:MyServiceWS >---------------------
[INFO] Building MyServiceWS 1.0-SNAPSHOT
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- wagon-maven-plugin:2.0.0:upload-single (rec) @ MyServiceWS ---
[INFO] Uploading: c:\INRA\Java\ides\eclipse-java-2020-09\workspace\MyServiceWS\target\MyServiceWS-rec.war scp://sicpa-interop.inra.f
r:/data/apps/payara/autodeploy/group-recette/webdistri/MyServiceWS-rec.war
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  45.058 s
[INFO] Finished at: 2021-01-19T12:06:44+01:00
[INFO] ------------------------------------------------------------------------
```

- De la même manière, compiler, packager et déployer votre archive en production avec la commande :
  mvn clean package -Denv=prod wagon:upload-single@prod

Ensuite, c'est le script d'auto déploiement sur sicpa-interop qui fera le reste comme c'est le cas actuellement

Voilà, c'en est fini de cette documentation autour de maven. Maintenant à vous de créer/migrer vos propres services et surtout de les étoffer avec, par exemple, de la sécurité (JWT), avec un ORM (hibernate), …

Dans la partie 9, je vais citer les dépendances principales à inclure dans nos projets. *(page 24)*

9. Principales dépendances MAVEN pour nos webservices

Les dépendances ci-dessous sont les dépendances les plus communes à ajouter à nos projets, dans la balise **<dependencies>** du fichier **pom.xml**. Elles ne sont pas toutes nécessaires à tel ou tel projet, il vous faudra choisir les plus pertinentes pour votre projet.

Par exemple, pour SOAP, j'ai listé trois dépendances ci-dessous. Les deux premières sont indispensables, la troisième *(`jakarta.xml.soap`)* est nécessaire uniquement si l'on veut lever des exceptions SOAP personnalisées.

- Hibernate : https://mvnrepository.com/artifact/org.hibernate/hibernate-core

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.28.Final</version>
</dependency>
```

- MySQL : https://mvnrepository.com/artifact/mysql/mysql-connector-java

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.23</version>
</dependency>
```

- Oracle : https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc8

```
<!-- https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc8 -->
<dependency>
    <groupId>com.oracle.database.jdbc</groupId>
    <artifactId>ojdbc8</artifactId>
    <version>21.1.0.0</version>
</dependency>
```

- PostGreSQL : https://mvnrepository.com/artifact/org.postgresql/postgresql

```
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.2.18</version>
</dependency>
```

- REST : https://mvnrepository.com/artifact/javax.ws.rs/javax.ws.rs-api

```
<!-- https://mvnrepository.com/artifact/javax.ws.rs/javax.ws.rs-api -->
<dependency>
  <groupId>javax.ws.rs</groupId>
  <artifactId>javax.ws.rs-api</artifactId>
  <version>2.1.1</version>
</dependency>
```

- SOAP : https://mvnrepository.com/artifact/jakarta.jws/jakarta.jws-api

```
<!-- https://mvnrepository.com/artifact/jakarta.jws/jakarta.jws-api -->
<dependency>
  <groupId>jakarta.jws</groupId>
  <artifactId>jakarta.jws-api</artifactId>
  <version>2.1.0</version>
</dependency>
```

- SOAP : https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api

```
<!-- https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api -->
<dependency>
  <groupId>jakarta.xml.bind</groupId>
  <artifactId>jakarta.xml.bind-api</artifactId>
  <version>2.3.3</version>
</dependency>
```

- SOAP : https://mvnrepository.com/artifact/jakarta.xml.soap/jakarta.xml.soap-api

```
<!-- https://mvnrepository.com/artifact/jakarta.xml.soap/jakarta.xml.soap-api -->
<dependency>
  <groupId>jakarta.xml.soap</groupId>
  <artifactId>jakarta.xml.soap-api</artifactId>
  <version>1.4.2</version>
</dependency>
```

Annexe 1 : Le fichier **pom.xml** dans le cadre de la migration d'un webservice existant

```xml
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>fr.inrae.sicpa</groupId>
  <artifactId>MyService</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>MyService</name>

  <properties>
    <encoding>UTF-8</encoding>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.version>3.1</maven.compiler.version>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/jakarta.jws/jakarta.jws-api -->
    <dependency>
      <groupId>jakarta.jws</groupId>
      <artifactId>jakarta.jws-api</artifactId>
      <version>2.1.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api -->
    <dependency>
      <groupId>jakarta.xml.bind</groupId>
      <artifactId>jakarta.xml.bind-api</artifactId>
      <version>2.3.3</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/jakarta.xml.soap/jakarta.xml.soap-api -->
    <dependency>
      <groupId>jakarta.xml.soap</groupId>
      <artifactId>jakarta.xml.soap-api</artifactId>
      <version>1.4.2</version>
    </dependency>
  </dependencies>

  <profiles>
    <profile>
      <id>rec</id>
      <activation>
        <property>
          <name>env</name>
          <value>rec</value>
        </property>
      </activation>
      <build>
        <finalName>MyService-rec</finalName>
        <resources>
          <resource>
            <directory>${basedir}/resources/rec</directory>
          </resource>
        </resources>
        <plugins>
          <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
          <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>wagon-maven-plugin</artifactId>
            <version>2.0.0</version>
            <executions>
              <execution>
                <id>rec</id>
                <phase>deploy</phase>
                <goals>
                  <goal>upload-single</goal>
                </goals>
                <configuration>
                  <fromFile>${project.build.directory}/${project.build.finalName}.${project.packaging}</fromFile>
```

```
                <url>scp://theirman@sicpa-interop.inra.fr:/data/apps/payara/autodeploy/group-
recette/webdistri</url>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
  <profile>
    <id>prod</id>
    <activation>
      <property>
        <name>env</name>
        <value>prod</value>
      </property>
    </activation>
    <build>
      <finalName>MyService-prod</finalName>
      <resources>
        <resource>
          <directory>${basedir}/resources/prod</directory>
        </resource>
      </resources>
      <plugins>
        <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-enforcer-plugin -->
          <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-enforcer-plugin</artifactId>
          <version>1.4.1</version>
          <executions>
            <execution>
              <id>enforce-prod-no-snapshot</id>
              <phase>validate</phase>
              <goals>
                <goal>enforce</goal>
              </goals>
              <configuration>
                <rules>
                  <requireReleaseVersion/>
                  <requireReleaseDeps/>
                </rules>
              </configuration>
            </execution>
          </executions>
        </plugin>
        <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>wagon-maven-plugin</artifactId>
          <version>2.0.0</version>
          <executions>
            <execution>
              <id>prod</id>
              <phase>deploy</phase>
              <goals>
                <goal>upload-single</goal>
              </goals>
              <configuration>
                <fromFile>${project.build.directory}/${project.build.finalName}.${project.packaging}</fromFile>
                <url>scp://theirman@sicpa-interop.inra.fr:/data/apps/payara/autodeploy/group-
production/webdistri</url>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
 </profiles>

 <build>
   <finalName>MyService</finalName>
   <sourceDirectory>${basedir}/src</sourceDirectory>
   <testSourceDirectory>${basedir}/test</testSourceDirectory>
   <resources>
     <resource>
       <directory>${basedir}/resources</directory>
```

```
      <includes>
        <include>*.properties</include>
        <include>*.xml</include>
      </includes>
      <excludes>
        <exclude>**/*.java</exclude>
      </excludes>
    </resource>
  </resources>
  <plugins>
    <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-compiler-plugin -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
    </plugin>
    <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-war-plugin -->
    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.2.3</version>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
        <warSourceDirectory>${basedir}/WebContent</warSourceDirectory>
        <warSourceExcludes>${basedir}/WebContent/WEB-INF/web.xml</warSourceExcludes>
      </configuration>
    </plugin>
    <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-enforcer-plugin -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-enforcer-plugin</artifactId>
      <version>1.4.1</version>
      <executions>
        <execution>
          <id>enforce-profile-target</id>
          <phase>validate</phase>
          <goals>
            <goal>enforce</goal>
          </goals>
          <configuration>
            <rules>
              <requireActiveProfile>
                <profiles>rec,prod</profiles>
                <all>false</all>
              </requireActiveProfile>
            </rules>
          </configuration>
        </execution>
      </executions>
    </plugin>
    <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>wagon-maven-plugin</artifactId>
      <version>2.0.0</version>
    </plugin>
  </plugins>
  <extensions>
    <!-- https://mvnrepository.com/artifact/org.apache.maven.wagon/wagon-ssh -->
    <extension>
      <groupId>org.apache.maven.wagon</groupId>
      <artifactId>wagon-ssh</artifactId>
      <version>3.4.0</version>
    </extension>
  </extensions>
 </build>
</project>
```

Annexe 2 : Le fichier **pom.xml** dans le cadre de la création d'un nouveau webservice

```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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>fr.inrae.sicpa</groupId>
  <artifactId>MyServiceWS</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>MyServiceWS</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/jakarta.jws/jakarta.jws-api -->
    <dependency>
      <groupId>jakarta.jws</groupId>
      <artifactId>jakarta.jws-api</artifactId>
      <version>2.1.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api -->
    <dependency>
      <groupId>jakarta.xml.bind</groupId>
      <artifactId>jakarta.xml.bind-api</artifactId>
      <version>2.3.3</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/jakarta.xml.soap/jakarta.xml.soap-api -->
    <dependency>
      <groupId>jakarta.xml.soap</groupId>
      <artifactId>jakarta.xml.soap-api</artifactId>
      <version>1.4.2</version>
    </dependency>
  </dependencies>

  <profiles>
    <profile>
      <id>rec</id>
      <activation>
        <property>
          <name>env</name>
          <value>rec</value>
        </property>
      </activation>
      <build>
        <finalName>MyServiceWS-rec</finalName>
        <resources>
          <resource>
            <directory>${basedir}/src/main/resources/rec</directory>
          </resource>
        </resources>
        <plugins>
          <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
          <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>wagon-maven-plugin</artifactId>
            <version>2.0.0</version>
            <executions>
              <execution>
                <id>rec</id>
                <phase>deploy</phase>
                <goals>
                  <goal>upload-single</goal>
                </goals>
                <configuration>
                  <fromFile>${project.build.directory}/${project.build.finalName}.${project.packaging}</fromFile>
```

```
                <url>scp://theirman@sicpa-interop.inra.fr:/data/apps/payara/autodeploy/group-
recette/webdistri</url>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
  <profile>
    <id>prod</id>
    <activation>
      <property>
        <name>env</name>
        <value>prod</value>
      </property>
    </activation>
    <build>
      <finalName>MyServiceWS-prod</finalName>
      <resources>
        <resource>
          <directory>${basedir}/src/main/resources/prod</directory>
        </resource>
      </resources>
      <plugins>
        <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-enforcer-plugin -->
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-enforcer-plugin</artifactId>
          <version>1.4.1</version>
          <executions>
            <execution>
              <id>enforce-prod-no-snapshot</id>
              <phase>validate</phase>
              <goals>
                <goal>enforce</goal>
              </goals>
              <configuration>
                <rules>
                  <requireReleaseVersion/>            <!-- The project must not be in SNAPSHOT -->
                  <requireReleaseDeps/>               <!--  No dependency must be in SNAPSHOT  -->
                </rules>
              </configuration>
            </execution>
          </executions>
        </plugin>
        <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>wagon-maven-plugin</artifactId>
          <version>2.0.0</version>
          <executions>
            <execution>
              <id>prod</id>
              <phase>deploy</phase>
              <goals>
                <goal>upload-single</goal>
              </goals>
              <configuration>
                <fromFile>${project.build.directory}/${project.build.finalName}.${project.packaging}</fromFile>
                <url>scp://theirman@sicpa-interop.inra.fr:/data/apps/payara/autodeploy/group-
production/webdistri</url>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
  </profile>
</profiles>

<build>
  <finalName>MyServiceWS</finalName>
  <directory>${basedir}/target</directory>
  <!-- main -->
  <sourceDirectory>${basedir}/src/main/java</sourceDirectory>
  <outputDirectory>${basedir}/target/classes</outputDirectory>
  <!-- test -->
```

```
<testSourceDirectory>${basedir}/src/test/java</testSourceDirectory>
<testOutputDirectory>${basedir}/target/test-classes</testOutputDirectory>
<resources>
  <resource>
    <directory>${basedir}/src/main/resources</directory>
    <filtering>true</filtering>
    <includes>
      <include>*.properties</include>
      <include>*.xml</include>
    </includes>
    <excludes>
      <exclude>**/*.java</exclude>
    </excludes>
  </resource>
</resources>
<plugins>
  <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-enforcer-plugin -->
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-enforcer-plugin</artifactId>
    <version>1.4.1</version>
    <executions>
      <execution>
        <id>enforce-profile-target</id>
        <phase>validate</phase>
        <goals>
          <goal>enforce</goal>
        </goals>
        <configuration>
          <rules>
            <requireActiveProfile>
              <profiles>rec,prod</profiles>
              <all>false</all>
            </requireActiveProfile>
          </rules>
        </configuration>
      </execution>
    </executions>
  </plugin>
  <!-- https://mvnrepository.com/artifact/org.codehaus.mojo/wagon-maven-plugin -->
  <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>wagon-maven-plugin</artifactId>
    <version>2.0.0</version>
  </plugin>
</plugins>
<extensions>
  <!-- https://mvnrepository.com/artifact/org.apache.maven.wagon/wagon-ssh -->
  <extension>
    <groupId>org.apache.maven.wagon</groupId>
    <artifactId>wagon-ssh</artifactId>
    <version>3.4.0</version>
  </extension>
</extensions>

<pluginManagement>
  <plugins>
    <plugin>
      <artifactId>maven-clean-plugin</artifactId>
      <version>3.1.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>3.0.2</version>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.0</version>
    </plugin>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.1</version>
    </plugin>
    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.2.3</version>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
```

```
            <warSourceDirectory>${basedir}/src/main/webapp/WebContent</warSourceDirectory>
            <warSourceExcludes>${basedir}/src/main/webapp/WEB-INF/web.xml</warSourceExcludes>
          </configuration>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```