Université de Bouira
Département d'Informatique
Module : Les Architectures Orientées Services
Niveau M1-ISIL

Série TP1

Les services web avec JAX-WS

I- Prérequis Logiciels

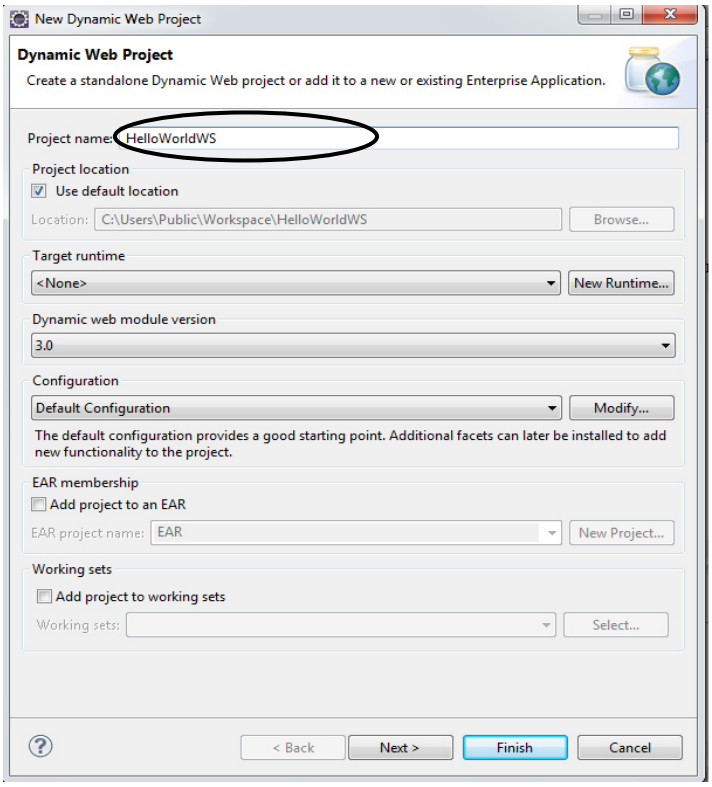
Pour les besoins de ce TP, les outils suivants seront utilisés :

- ***JDK 1.7*** + *Variables d'environnement (JAVA_HOME + Path).*
- *eclipse-jee-kepler-SR2-win32*: C'est l'environnement de développement Eclipse, pour Java EE.
- *soapUI-x32-5.2.1.exe* : outil de test complet pour les services web.

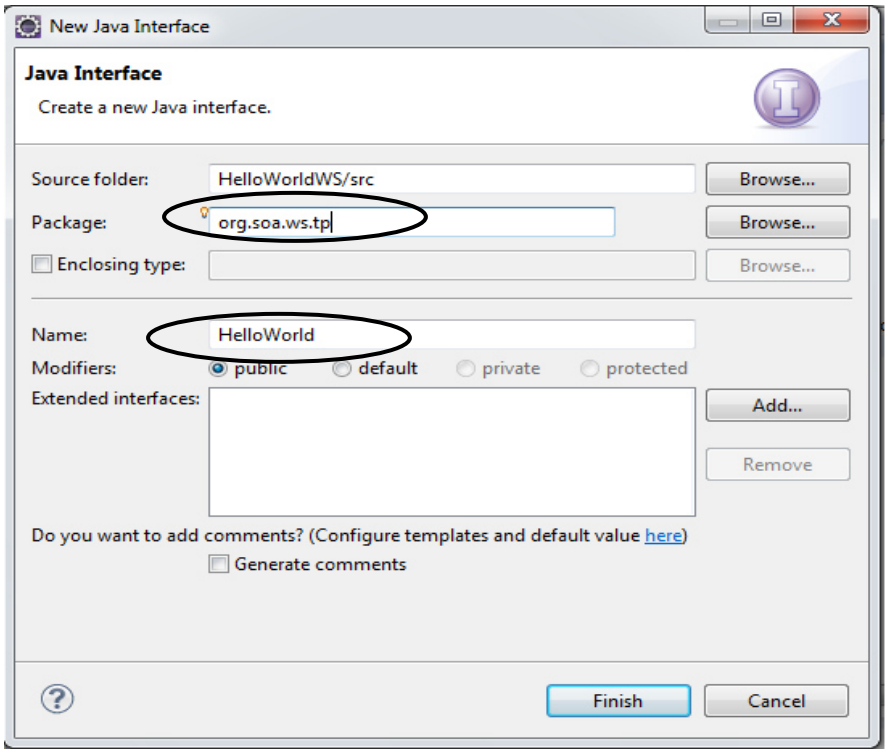
1 Implementation d'un Service Web SOAP – Hello World

Suivre les étapes dans l'ordre.

- Démarrer l'environnement de développement Eclipse.
- Créer un nouveau projet de type **Dynamic Web Project**. Appeler votre projet HelloWorldWS. *Finish*.



- Ajouter à votre projet une **interface** représentant la description du service web (**Clic droit sur le projet -> New puis choisir Interface**). Définir comme nom de l'interface créée *HelloWorld* et *org.soa.ws.tp* comme nom de package.



- Taper le code suivant :

```java
package org.soa.ws.tp;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public interface HelloWorld {

    @WebMethod
    public String simpleHello();

    @WebMethod
    public String makeHello(String nom);

}
```

- Créer dans le package org.soa.ws.tp (**clic droit sur le package -> New puis choisir Class**) une nouvelle classe appelée *HelloWorldImpl* qui implémente le traitement de l'interface HelloWorld.
- Taper le code suivant :

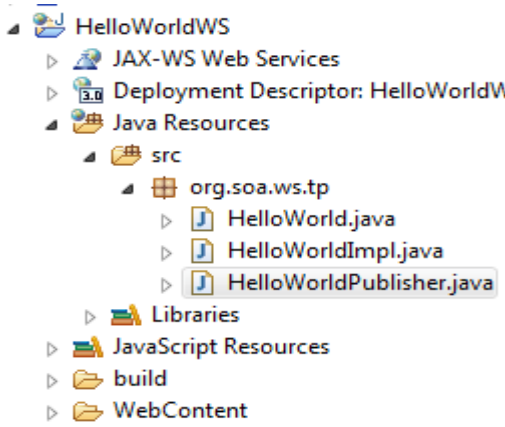
```java
package org.soa.ws.tp;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService(endpointInterface="org.soa.ws.tp.HelloWorld")
public class HelloWorldImpl implements HelloWorld{

    @Override
    @WebMethod
    public String simpleHello() {
        return "Hello Wolrd!";
    }
```

```
    @Override
    public String makeHello(String nom) {
        return "Hello Wolrd, " + nom;
    }

}
```

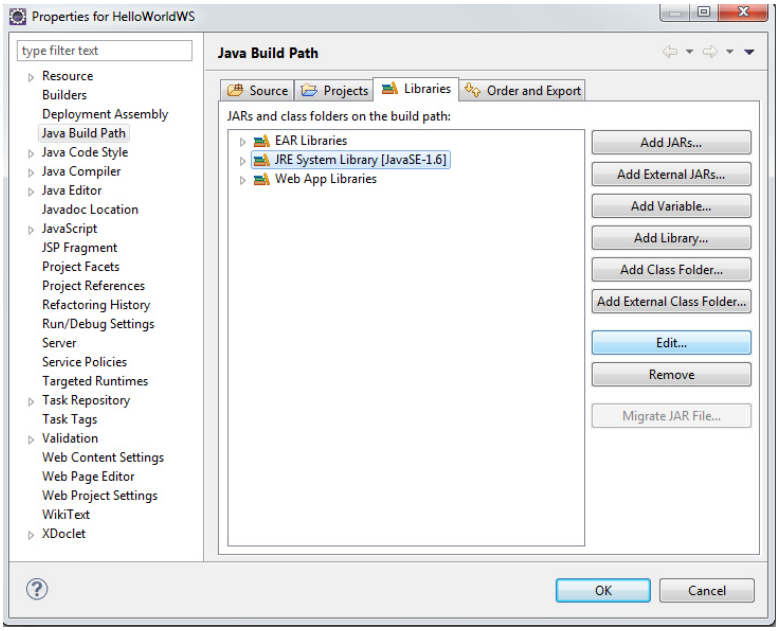
- Afin de déployer localement ce service web HelloWorld, ajouter une troisième classe au package org.soa.ws.tp appelée *HelloWorldPublisher* et saisir le code suivant :

```
package org.soa.ws.tp;

import javax.xml.ws.Endpoint;

public class HelloWorldPublisher {

    public static void main(String[] args) {
        Endpoint.publish("http://localhost:4848/helloworldws", new HelloWorldImpl());
    }

}
```

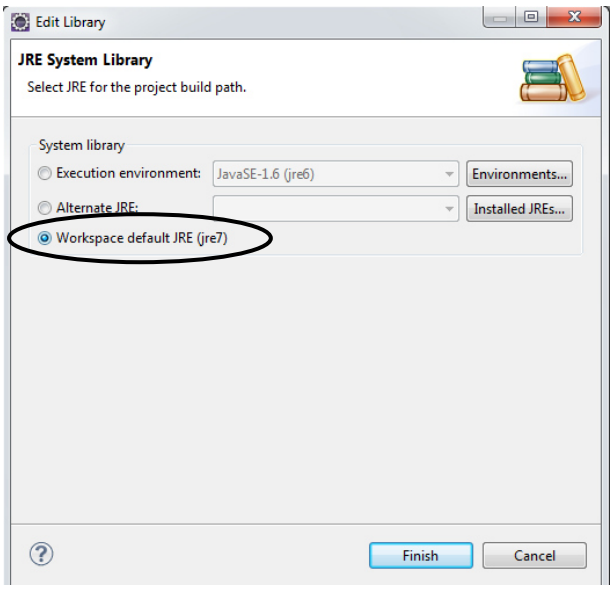
- L'arborescence de votre projet devra ressembler à ci-dessous :



- Cliquer droit sur le projet *HelloWorldWS* puis choisir *Properties*. Dans l'onglet *Libraries,* cliquer sur *JRE System Library* puis sur *Edit.*



- Choisir JRE 7. Finish.



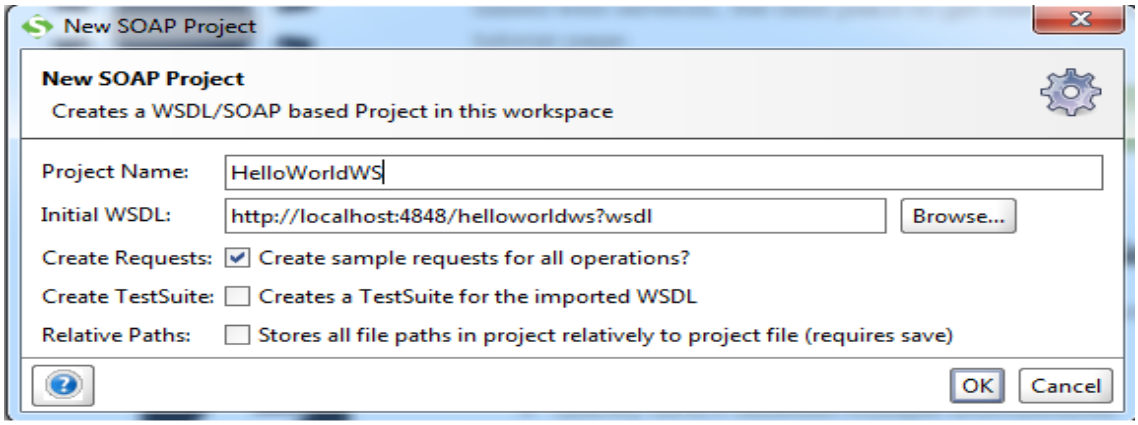
- Exécuter la classe HelloWorldPublisher pour démarrer votre service web : clic droit sur la classe -> Run As -> Java Application.
- Ouvrir un navigateur web (Chrome par exemple) puis entrer l'url du service web.

- Afficher la description WSDL de votre service web - générée automatiquement – en cliquant sur **http://localhost:4848/helloworldws?wsdl**

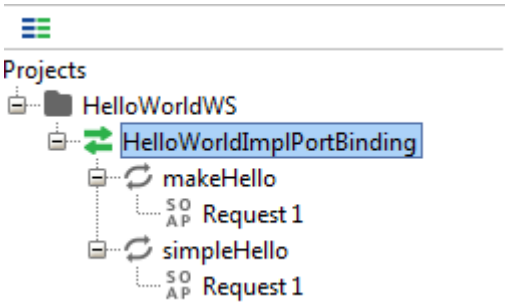
2 Test du Service Web SOAP implémenté – Hello WorldWS

a. Tester avec soapUI

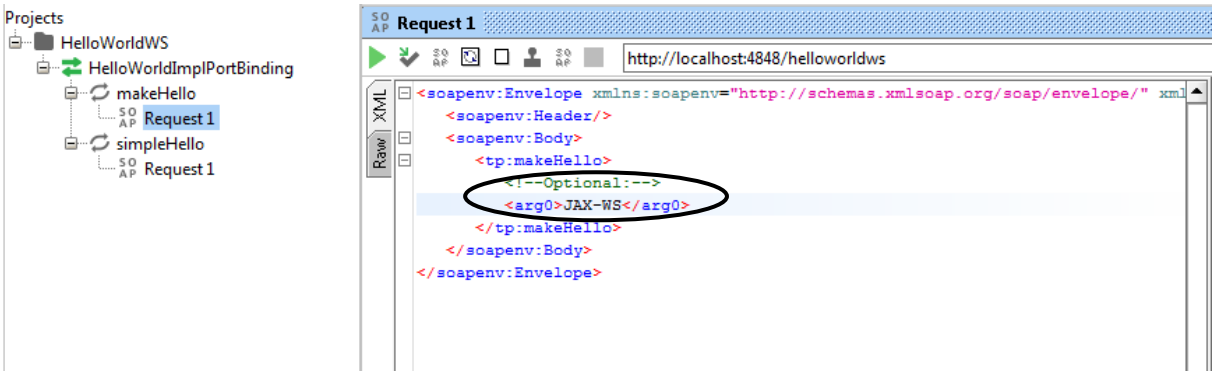
- Installer l'outil soapUI.
- Pour tester votre service, créer un nouveau projet en cliquant : *File -> New SOAP Project*
- Taper comme nom de projet *HelloWorldWS* et donner l'URL du fichier WSDL du service *HelloWorld*. Cliquer ensuite sur OK.



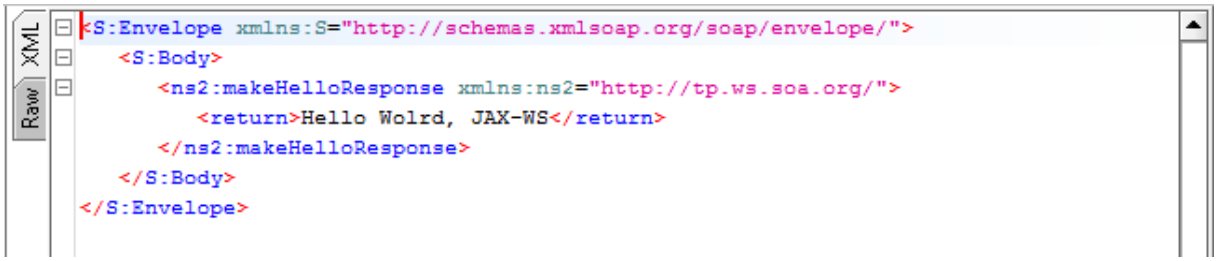
- L'arborescence de votre projet devient comme suit :



- Pour tester l'opération 2, double-cliquer sur *Request1* puis taper **JAX-WS** à la place du « ? » dans la balise *<arg0> ?</arg0>*



- Cliquer sur la flèche verte (Run) et observer le résultat (i.e. message réponse SOAP).



b. Tester avec le Tester de Glassfish Server

- Démarrer le serveur Glassfish depuis Eclipse.
- Pour déployer votre service web sur le serveur Glassfish, faire un clic-droit sur le projet et choisir *Run As* puis *Run on Server*.
- Ouvrir un navigateur web, et accéder à la console d'administration de Glassfish en entrant l'url : htt://localhost:4848
- Dans le menu *Common Tasks* qui se trouve à gauche, cliquer sur *Applications*.
- Dans le tableau des applications déployées, cliquer sur *HelloWorldWebService*.
- Choisir l'action *View Endpoint*, puis cliquer sur le lien du *Tester*. Choisir le premier des deux.
- La page suivante sera affichée :

- Pour tester l'opération 1 du service web, cliquer sur *simpleHello*( ). Pour tester l'opération 2, taper votre nom dans la case qui vous est fournie, et cliquer sur *nameHello*( ).
- Observer le résultat, ainsi que les **requêtes** et les **réponses SOAP** générées.

c. Tester via une application cliente

- Télécharger le fichier WSDL du service web. Appeler ce fichier *HelloWorldImplService*.wsdl
- Copier ce fichier dans le dossier Web HelloWorldWebService/WebContent.
- Sélectionner le fichier WSDL copié, puis clic-droit et faire *New -> Other ... -> Web Services -> Web Service Client*. Cliquer sur *Next*.
- Déplacer le slider du client à la position Test Client.
- Cliquer sur le lien *Client Project* sous *Configuration* et entrer dans la fenêtre qui apparait *HelloWorldWebServiceClient* comme nom de projet client. Cliquer sur *Finish*.
- Attendre quelques secondes. Une application JSP nommée *TestClient* (dans le dossier WebConetnt/sampleHelloWorldProxy) est générée et affichée dans le browser view.
- Cliquer sur l'une des opérations de *TestClient*, puis sur Invoke pour l'exécuter.