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

Série TD 1

Création de services web SOAP avec JAX-WS

L'objectif de cette deuxième série est d'apprendre à manipuler l'API JAX-WS pour *implémenter*, *déployer* et *tester* des services web étendus à partir de la plateforme de développement Java.

I. Prérequis Logiciels

- Java 7
- Eclipse JEE Kepler – Glassfish4 Server
- Navigateur Web
- SoapUI

II. Exercice 1 – Approche Code First (Bottom/Up)

Ce premier exercice consiste en la création d'un service web HelloWorld. Ce service fournit deux opérations :

- ✓ Une opération simpleHello sans paramètre en entrée et retourne la chaîne de caractères *Hello World*.

- ✓ Une opération nameHello qui prend en paramètre une chaîne de caractères *name* et retourne la chaîne de caractères *Hello World, name*.

1. Etapes à suivre

- ❖ Etape 1 - Création de l'interface : HelloWorld
- ❖ Etape 2 - Création de la classe implémentant l'interface créée : HelloWorldImpl
- ❖ Etape 3 - Création de la classe endpoint déployant le service web localement : HelloWorldPublisher

Etape 1

- Démarrer l'environnement de développement Eclipse.
- Créer un nouveau projet de type Dynamic Web Project. Appeler votre projet HelloWorldWebService.
- Ajouter une interface représentant la description du service web (File -> New puis choisir Interface). Définir comme nom de l'interface créée *HelloWorld* et *org.soa.ws.tp2* comme nom de package.
- Ajouter les méthodes suivantes dans l'interface HelloWorld :
  - ✓ String simpleHello( ) : représentant l'opération 1.
  - ✓ String nameHello(String name) : représentant l'opération 2.
- Pour indiquer que l'interface créée est un service web, ajouter une annotation **@WebService** à son niveau et des annotations @WebMethod au niveau de chaque méthode.

Etape 2

- Créer une nouvelle classe appelée *HelloWorldImpl* qui implémente le traitement de l'interface HelloWorld.
- Compléter l'implémentation des deux méthodes simpleHello et nameHello.
- Ajouter une annotation @WebService au niveau de la classe, puis modifier l'attribut endpointInterface de l'annotation comme suit :

  @WebService(endpointInterface = "org.soa.ws.tp2.HelloWorld")

Etape 3

- Afin de déployer localement ce service web HelloWorld, définir une troisième classe appelée HelloWorldPublisher à déposer dans le package déjà créé et saisir le code suivant :

```
package org.soa.ws.tp2 ;

import javax.xml.ws.Endpoint;

public class HelloWorldPublisher {

         public static void main(String[] args) {
            Endpoint.publish("http://localhost:1234/ws/hello", new HelloWorldImpl());
    }
}
```

- Exécuter la classe HelloWorldPublisher pour démarrer votre service web. Ce dernier sera déployé via l'url : **http://localhost:1234/ws/hello**
- Afficher la description WSDL de votre service web - générée automatiquement - **http://localhost:1234/ws/hello?wsdl** et comparer le résultat par rapport à ce qui a été défini dans l'interface Java.

2. Tester le Service Web

Pour tester ou utiliser votre service web, plusieurs méthodes sont disponibles. Nous allons en montrer trois d'entres elles.

a. Tester avec le Tester de Glassfish Server

Le serveur Glassfish fournit un outil pour tester votre service web directement sur le serveur sans recourir à la classe endpoint Publisher. Pour cela, suivre les instructions suivantes :

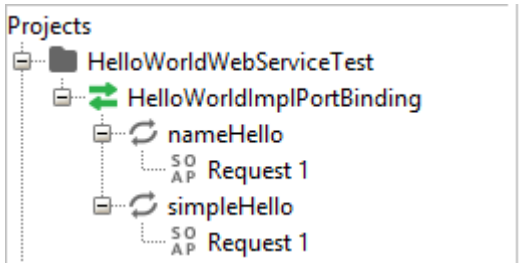
- Démarrer le serveur Glassfish depuis Eclipse.
- Pour déployer votre service web sur le serveur Glassfish, faire un clic-droit sur le projet et choisir *Run As* puis *Run on Server*.
- Ouvrir un navigateur web, et accéder à la console d'administration de Glassfish en entrant l'url : htt://localhost/4848
- 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.

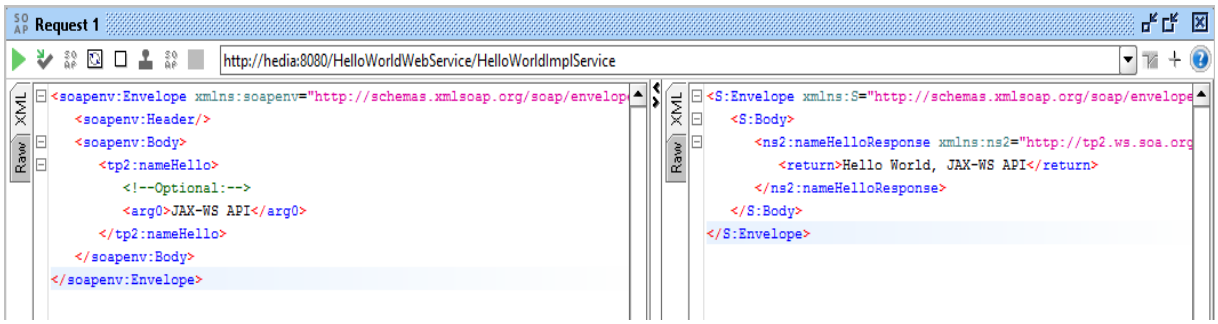
b. Tester avec soapUI

Le serveur d'application qui héberge le service peut parfois être inaccessible à l'utilisateur. On dispose donc d'autres outils pour tester notre service web. On cite par exemple l'outil *soapUI5* qui est une solution de test open source et multi-plateforme, permettant de créer rapidement et facilement des tests fonctionnels, grâce à une interface graphique intuitive.

- Installer l'outil soapUI. (Téléchargeable depuis : https://www.soapui.org/)
- Pour tester votre service, créer un nouveau projet en cliquant : *File -> New SOAP Project*
- Taper comme nom de projet *HelloWorldWebServiceTest* 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*. La fenêtre suivante va s'ouvrir :



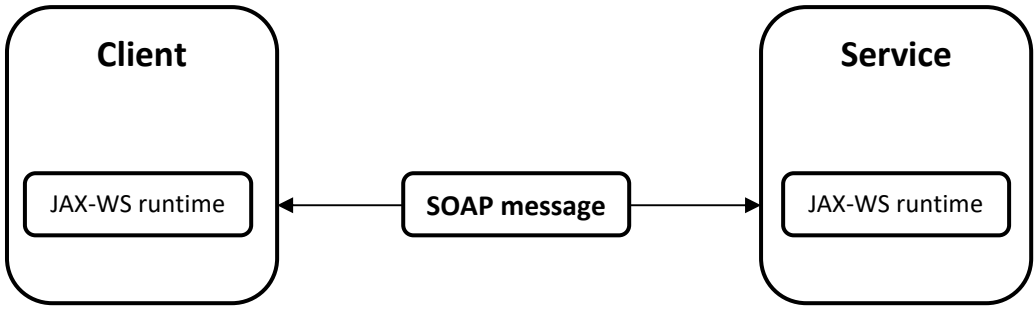
- Le premier cadre représente la requête *soap*, et le second représentera la réponse. Pour tester le service, taper votre nom à la place du « ? » dans la balise *<arg0> ?</arg0>*
- Cliquer sur la flèche verte (Run) et observer le résultat.

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.

Le service web HelloWorld peut être testé notamment en créant un Client consommateur comme indiquer dans le TP 1.

III. Schéma général et structure d'un message SOAP



Les services web étendus communiquent via le protocole SOAP (Simple Object Access Protocol). C'est une spécification du W3C et est basé sur XML. Il est par conséquent léger et standard ayant pour but d'échanger des messages structurés dans un environnement décentralisé, hétérogène, et distribué.
Un message SOAP est véhiculé vers le récepteur en utilisant un protocole de transport (http,).
Un message SOAP est constitué d'une enveloppe SOAP composée de deux parties : un en-tête (header, qui peut être facultatif) et un corps (body).

Exemple : message SOAP d'une requête pour appeler *nameHello*( ) du service *HelloWorld*.

```
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <ns2:nameHello xmlns:ns2="http://tp2.ws.soa.org/">
            <arg0>JAX-WS API</arg0>
        </ns2:nameHello>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

QCM/QCU - Les Services Web SOAP

1) Cochez les technologies permettant de développer des AOS ?

- ☐ CORBA
- ☐ DPWS
- ☐ Services Web
- ☐ Javascript
- ☐ UPnP
- ☐ Azure

2) Un service est une entité logicielle qui expose un certain nombre de :

- ☐ Instructions
- ☐ Opérations
- ☐ Messages
- ☐ Processus

3) Cochez les familles des services web qui existent actuellement :

- ☐ Etendus
- ☐ REST
- ☐ XML
- ☐ W3C

4) Cochez les acteurs essentiels opérant dans une architecture Service Web :

- ☐ Fournisseur de service
- ☐ Traiteur de service
- ☐ Consommateur de service
- ☐ Annuaire de services
- ☐ Orchestrateur de services

5) Quelle est l'API Java permettant d'implémenter les services web étendus SOAP ?

- ☐ JAX-RS
- ☐ Jersey
- ☐ RESTeasy
- ☐ JAX-WS

6) En quel langage est décrit un service web SOAP ?

- ☐ SOAP

- ☐ Java
- ☐ RDF
- ☐ WSDL

7) Quelles sont les deux approches permettant d'implémenter un service web SOAP ?

- ☐ Integration First
- ☐ Mobile First
- ☐ Contract First
- ☐ Code First
- ☐ Test First

8) Que représente l'interface ?

- ☐ Message
- ☐ Classe
- ☐ Service
- ☐ Contrat

9) Cocher les annotations de JAX-WS :

- ☐ @ServiceWeb
- ☐ @WebMethod
- ☐ @SOAPService
- ☐ @EndpointInterface

10) Un service web SOAP implémenté sous JAX-WS a :

- ☐ Une interface
- ☐ Une classe d'implémentation
- ☐ Une classe endpoint
- ☐ Un framework

11) soapUI permet de :

- ☐ Implémenter un service web
- ☐ Développer un service web
- ☐ Tester un service web
- ☐ Générer un service web

12) Une opération possède des paramètres et un résultat return appelés :

- ☐ Requête
- ☐ Transaction
- ☐ Message
- ☐ Signature

13) Que retrouve-t-on dans une interface ?

- □ Déclaration des opérations
- □ Implémentation des opérations
- □ Découverte des opérations
- □ Souscription aux opérations

14) Qu'est ce qui permet l'indépendance technologique et l'interopérabilité des services web ?

- □ Annotations
- □ Contrat de service
- □ XML
- □ Annuaire

Exercice 1:

Ecrire un service web SOAP Convertisseur fournissant deux opérations pour convertir un montant donné en paramètre :

- Une opération ***getDinarFromEuro*** : pour convertir de l'euro au dinar.
- Une opération ***getEuroFromDinar*** : pour convertir du dinar à l'euro.

1. Ecrire les classes de ce service web, à savoir, Convertisseur (interface), ConvertisseurImpl, et ConvertisseurPublisher.

Exercice 2:

Ecrire un service web SOAP Random fournissant deux opérations :

- Une opération ***getRandomValue*** : pour récupérer une valeur aléatoire entre 0 et 1.
- Une opération ***getSinusValue*** : pour récupérer le sinus d'une valeur passée en paramètre.

2. Ecrire les classes de ce service web, à savoir, RandomWS (interface), RandomWSImpl, et RandomWSPublisher.