esprit Se former autrement	Année Universitaire : 2018-2019 **Ateliers SOA** **UP_WEB**

Atelier n°2
Création d'un service web HelloWorld avec JAX-WS

Objectifs

Le but de cet atelier est:

- La création d'un service web HelloWorld.
- Le test du service web et la visualisation des messages SOAP avec SOAP UI.

Méthodologie de développement du service web

Deux approches s'opposent : l'approche bottom-up (à partir du code source Java du service web) et l'approche top-down (à partir du WSDL du service web).

Vous allez ici mettre en œuvre l'approche bottom-up:

- Création du service web HelloWorld.
- Déploiement du service web.

Création de l'implémentation du service

1- Dans Eclipse, créez un simple projet **maven** de packaging **war**.

2- Ajoutez ces lignes au niveau fichier pom.xml comme suit:

```
<properties>
    <failOnMissingWebXml>false</failOnMissingWebXml>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.source>1.8</maven.compiler.source>
</properties>
<build>
    <plugins>
        <plugin>
             <groupId>org.wildfly</groupId>
            <artifactId>wildfly-server</artifactId>
            <version>11.0.Final</version>
        </plugin>
    </plugins>
</build>
```

3. Cliquez sur Maven Update Project.
4. Créez une classe nommée « HelloWS» contenant les méthodes publiques suivantes:

```
package tn.esprit.soa.ws.etendu;
public class HelloWS {
	public String sayHello()
	{
		return "Hello from JAX-WS";
	}
	public String sayHelloTo(String nom)
	{
		return "Hello from JAX-WS "+nom;
	}
}
```

5. Ajoutez l'annotation **@WebService** au-dessus de la déclaration de la classe:

```
package tn.esprit.soa.ws.etendu;
import javax.jws.WebService;
@WebService
public class HelloWS {
	public String sayHello()
	{
		return "Hello from JAX-WS";
	}
	public String sayHelloTo(String nom)
	{
		return "Hello from JAX-WS "+nom;
	}
}
```

Déploiement du service web

Après avoir créé le service web, il faut le publier sur le serveur d'application pour pouvoir l'utiliser.

1- Déployez le projet sur le serveur.
2- Depuis la console, un service Web doit avoir été découvert, un message similaire doit être présent :

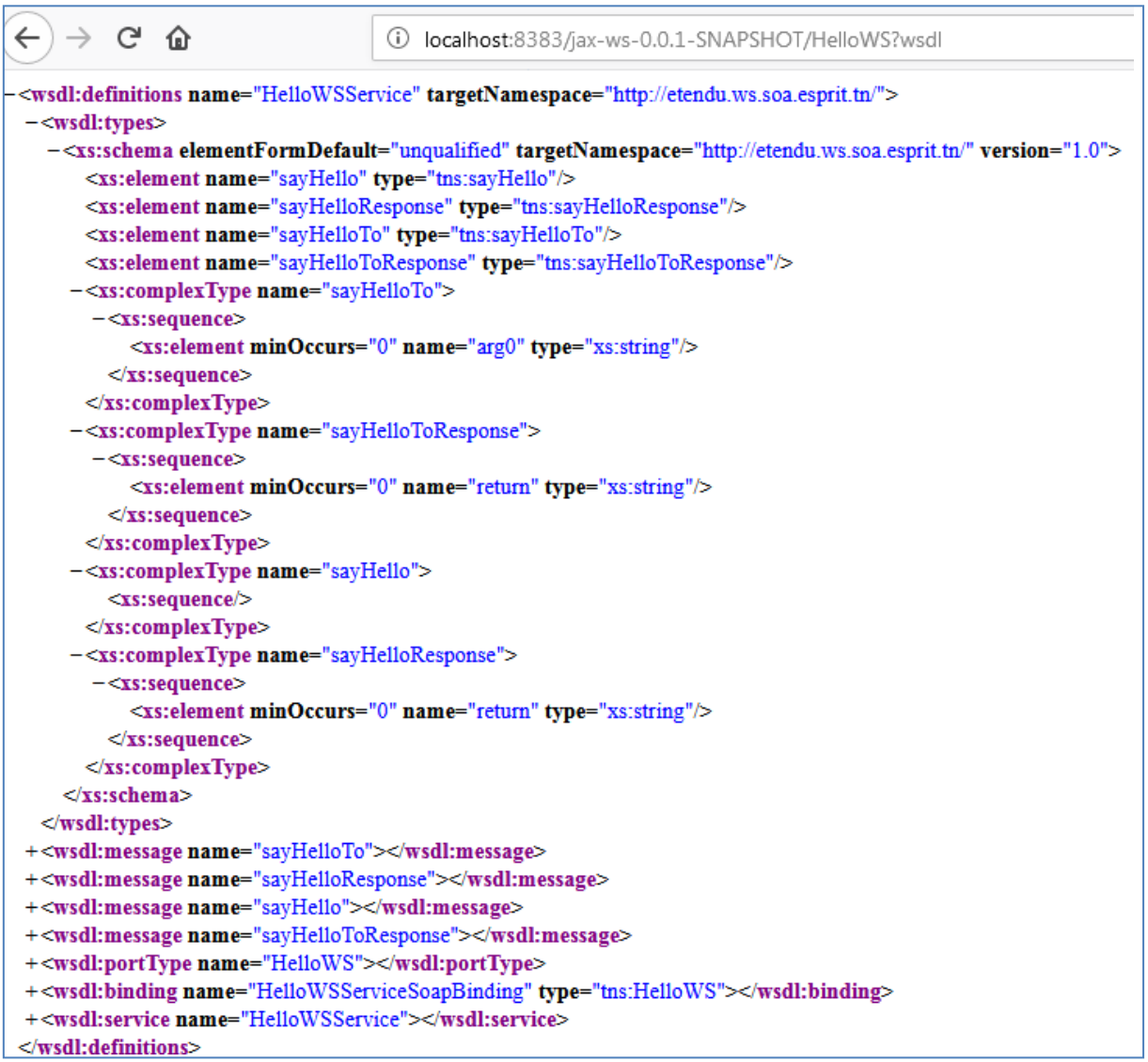
```
address=http://localhost:8383/jax-ws-0.0.1-SNAPSHOT/HelloWS
implementor=tn.esprit.soa.ws.etendu.HelloWS
serviceName={http://etendu.ws.soa.esprit.tn/}HelloWSService
portName={http://etendu.ws.soa.esprit.tn/}HelloWSPort
annotationWsdlLocation=null
wsdlLocationOverride=null
mtomEnabled=false
```

Test du service web

Récupération du contrat WSDL

Par convention, vous pouvez récupérer le contrat WSDL d'un service web en ajoutant la chaine « **?wsdl** » à la fin de l'URL du service web.

Allez sur « *http://localhost:8383/jax-ws-0.0.1-SNAPSHOT/HelloWS?wsdl* »



Service web personnalisé

1- Vous allez personnaliser, grâce aux annotations JAX-WS, le contrat WSDL généré. Ajoutez les annotations JAX-WS suivantes avec les attributs spécifiés:

- **@WebService** : qui déclare un service web.
    - **targetNamespace:** Indique l'espace de nom XML du WSDL et des éléments XML générés à partir du service Web. La valeur par défaut est l'espace de nom mappé à partir du nom de package contenant le service Web.
    - **name:** Nom de wsdl:portType. La valeur par défaut est le nom non qualifié de la classe ou de l'interface Java.
    - **serviceName:** Indique le nom du service Web :wsdl:service. La valeur par défaut est le nom simple de la classe Java + *Service*.
    - **portName:** Correspond à l'attribut wsdl:portName. La valeur par défaut est WebService.name +*Port* .

- **@WebMethod** : qui indique que c'est une opération du service web.
    - **operationName:** permet de définir un nom de l'opération différent de celui défini dans la classe.
    - **exclude:** permet d'exclure la méthode java du contrat WSDL généré. Sa valeur par défaut est **false**.
- **@WebResult** : qui définit le nom du paramètre de sortie de la méthode. Par défaut il prend la valeur **return**.
- **@WebParam**: qui définit le nom du paramètre que l'on veut avoir. Par défaut il prend la valeur **arg0**, **arg1**…

```
package tn.esprit.soa.ws.etendu;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebResult;
import javax.jws.WebService;
@WebService(name="greetingPortype",portName="greetingPort",
            serviceName="greetingService",targetNamespace="http://greeting.tn")
public class HelloWS {
    @WebMethod(operationName="greetingOperation")
    @WebResult(name="greeting")
    public String sayHello()
    {
        return "Hello from JAX-WS";
    }
    @WebMethod(operationName="greetingToOperation")
    @WebResult(name="greetingTo")
    public String sayHelloTo(@WebParam(name="LastName")String nom)
    {
        return "Hello from JAX-WS "+nom;
    }
}
```

2- Republiez votre service web et détectez les changements au niveau de votre contrat WSDL.

Test du Service Web avec SOAP UI

Consommez le service web avec SOAP UI et analysez les messages SOAP échangés entre le consommateur et le fournisseur du service web.