Ingénieur 2ème année
Spécialité Informatique - Apprentissage

QUIZON Rosselle

TP n°0 :
RMI et Services Web Soap

1. Objectifs du TP

L'objectif de ce TP est de découvrir concrètement les services web avec une approche SOAP en Java.

Un service SOAP est décrit par un fichier WSDL (*Web Service Description Language*), qui contient :
- la localisation du service (URL),
- les opérations disponibles (méthodes),
- les types d'entrée et de sortie (décrits en XSD),
- les messages et bindings utilisés.

Un message SOAP est une requête HTTP dont le corps est un document XML structuré (<Envelope>, <Body>, etc.). SoapUI permet de générer ces messages et d'afficher la réponse XML renvoyée par le serveur.

2. Service Web SOAP en Java

Dans l'IDE IntelliJ, on crée un projet Java simple.
Une première classe DemoApplication est définie avec une méthode main qui affiche un message pour vérifier que tout fonctionne.

On crée ensuite une classe Java, MonServiceWeb, qui devient un service web SOAP grâce à l'annotation :

```
@WebService(targetNamespace = "http://www.polytech.fr/")
public class MonServiceWeb { ... }
```

3. Publication du service avec Endpoint.publish

Au lieu de déployer sur un serveur type Tomcat ou Glassfish, on utilise ici la méthode statique Endpoint.publish :

```
String url = "http://localhost:8888/";
Endpoint.publish(url, new MonServiceWeb());
System.out.println("Service web déployé");
```

Cette méthode démarre un mini-serveur HTTP intégré à la JVM et publie le service à l'URL choisie.

En ouvrant dans le navigateur l'URL http://localhost:8888/?wsdl, on obtient le fichier WSDL généré automatiquement. On y retrouve :
- le nom du service,
- l'URL réelle d'accès,
- l'opération conversion,
- les types d'entrée (un double) et de sortie (double).

4. **Test du service avec SoapUI**

Le message de requête SOAP est un XML dans lequel on renseigne la valeur du paramètre (par exemple 1). Après envoi de la requête, le serveur répond par un message SOAP dont le résultat se trouve dans la balise <return>.

Lors d'ajout de nouvelles méthodes, comme la méthode somme, si l'on souhaite qu'elle apparaît dans le WSDL, il faut d'abord arrêter le service (arrêter l'exécution de la classe Application), puis redéployer (relancer Endpoint.publish).

En rechargeant ?wsdl dans le navigateur, une nouvelle opération somme apparaît.
Dans SoapUI, on utilise "Update Definition" pour mettre à jour le projet et voir la nouvelle opération, que l'on peut ensuite tester (par exemple 2 et 2, on obtient 4).

5. **Personnalisation via @WebParam et @WebMethod**

Pour ne plus avoir des noms de paramètres génériques du type arg0, on ajoute une annotation :

```
public double somme(
   @WebParam(name = "param1") double a,
   @WebParam(name = "param2") double b) {
   return a + b;
}
```

Après redéploiement, les paramètres du WSDL sont nommés param1 et param2.
De même, pour renommer une opération dans le WSDL, on utilise :

```
@WebMethod(operationName = "convertir")
public double conversion(double mt) { ... }
```

L'opération ne s'appelle plus conversion dans le WSDL mais convertir.

6. Manipulation d’un objet Etudiant

On crée la classe Etudiant avec les attributs privés :
- int identifiant;
- String nom;
- double moyenne;

Et on ajoute :
- un constructeur sans paramètre (obligatoire pour la désérialisation),
- un constructeur avec paramètres,
- des getters et setters pour chaque attribut.

On annote la classe Etudiant avec @XmlRootElement (JAXB) pour permettre la sérialisation en XML.

Après arrêt et redéploiement du service, le WSDL est mis à jour : on voit un type complexe correspondant à Etudiant dans la partie XSD. Dans SoapUI, une nouvelle opération getEtudiant apparaît, et la réponse SOAP contient un XML représentant l’étudiant (avec les balises <identifiant>, <nom>, <moyenne>).

Cette partie montre comment SOAP peut transporter des objets structurés en XML.

Bilan

Ce TP m’a permis de comprendre concrètement le fonctionnement d’un service web SOAP, depuis sa création jusqu’à son déploiement et son test. J’ai appris à utiliser l’annotation @WebService, à publier un service avec Endpoint.publish, et à analyser le fichier WSDL généré automatiquement. L’utilisation de SoapUI m’a aidée à visualiser la structure des messages SOAP et à tester les différentes opérations du service. Enfin, l’ajout d’annotations comme @WebParam ou @WebMethod, ainsi que la manipulation d’un objet Etudiant, m’a montré comment personnaliser et enrichir un service SOAP. Ce TP constitue donc une bonne introduction aux services web et pose des bases solides pour les TP suivants.