TP0 – Services Web SOAP et RMI

1 Introduction

L'objectif de ce TP est de reproduire un service Web SOAP similaire à celui présenté en cours. Le travail consiste à déployer un service SOAP en Java, à l'exposer via un serveur RMI, puis à le tester à l'aide de l'outil SOAPUI. Ce rapport synthétise la démarche suivie, les mécanismes techniques utilisés ainsi que les différentes étapes de test.

Un service Web SOAP permet à une application d'en appeler une autre à distance via des messages XML standardisés. Le WSDL (Web Services Description Language) décrit formellement toutes les opérations exposées par le service.

2 Architecture générale du projet

Le projet repose sur trois composants principaux :

- **La classe Application** : point d'entrée du programme permettant de publier le service.
- **La classe MonServiceWeb** : contient l'ensemble des opérations accessibles à distance.
- **La classe Etudiant** : représente un objet métier échangé sous forme XML.

Les annotations Java, notamment `@WebService`, `@WebMethod` et `@XmlRootElement`, permettent la génération automatique du WSDL et l'exposition du service.

3 Description du code

3.1 Classe Application

Cette classe publie le service sur l'URL : `http://localhost:8888/`. L'instruction `Endpoint.publish()` démarre le serveur SOAP.

```
public class Application {
    public static void main(String[] args) {
        System.out.println("D but de d ploiement de mon service");
        String url = "http://localhost:8888/";
        Endpoint.publish(url, new MonServiceWeb());
        System.out.println("le service web est d ploy ");
    }
}
```

3.2 Classe MonServiceWeb

Cette classe regroupe les trois opérations exposées :

- **conversion** : multiplie un nombre par 0.9.
- **somme** : additionne deux valeurs.
- **getEtudiant** : renvoie un objet `Etudiant` sérialisé.

```
@WebService(targetNamespace = "http://www.polytech.fr")
public class MonServiceWeb {
    @WebMethod(operationName = "convertir")
    public double conversion(double mt){
        return mt * 0.9;
    }

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

    public Etudiant getEtudiant(int identifiant) {
        return new Etudiant(1, "mario", 19);
    }
}
```

3.3 Classe Etudiant

Cette classe constitue une structure de données qui peut être envoyée via SOAP. Grâce à `@XmlRootElement`, Java peut automatiquement transformer l'objet en XML.

```
@XmlRootElement
public class Etudiant implements Serializable {
    private int identifiant;
    private String nom;
    private double moyenne;
    // Constructeurs, getters et setters
}
```

4 Génération du WSDL et tests SOAPUI

Lors du lancement du service, un fichier WSDL est automatiquement mis à disposition. Il décrit :

- les opérations disponibles (convertir, somme, getEtudiant),
- les messages d'entrée et de sortie en XML,
- les types utilisés (double, Etudiant),
- la structure du service.

SOAPUI permet d'importer le WSDL et de générer automatiquement les requêtes SOAP. On peut alors tester chacune des fonctionnalités :

- **convertir** : envoie un nombre et récupère la valeur multipliée par 0.9.
- **somme** : vérifie le bon fonctionnement de l'addition.
- **getEtudiant** : reçoit un XML contenant un étudiant.

5 Conclusion

Ce TP a permis de mettre en place un service SOAP complet : définition, publication, sérialisation d'objets, génération d'un WSDL et tests réalisés avec SOAPUI. Cette manipulation illustre les concepts fondamentaux des services Web basés sur XML et prépare la transition vers des technologies plus modernes telles que les API REST.