TP 1 Web Services SOAP avec Java (JAX-WS)

Safa Ferchichi – 12408918 – INFOA2

Introduction

L'objectif de ce TP est de mettre en œuvre un **service web SOAP** en Java en utilisant l'API **JAX-WS**.

1 RMI et Web Services SOAP

1.1 RMI (Remote Method Invocation)

Java RMI permet à une application Java d'appeler des méthodes d'un objet situé sur une autre machine (ou un autre processus), comme s'il s'agissait d'un appel local. Les caractéristiques principales de RMI sont :

— communication *Java vers Java*,
— utilisation d'objets distants, de *stubs* et de *skeletons*,
— sérialisation automatique des objets Java.

La limite importante de RMI est qu'il ne fonctionne qu'entre programmes Java. Pour communiquer avec des applications écrites dans d'autres langages (PHP, .NET, Python, etc.), on utilise des **web services**.

1.2 SOAP (Simple Object Access Protocol)

SOAP est un protocole de communication basé sur **XML** utilisé par certains web services qui définit un format standard pour échanger des messages entre un client et un serveur, généralement au-dessus de HTTP. Un message SOAP est structuré autour de trois éléments :

— `Envelope` : enveloppe du message,
— `Header` : informations supplémentaires (souvent optionnel),
— `Body` : contenu de la requête ou de la réponse.

2 JAX-WS et description du service

2.1 JAX-WS : API Java pour les services SOAP

JAX-WS (Java API for XML Web Services) est l'API standard de Java pour créer et consommer des web services SOAP. Elle permet :

— de déclarer un service avec l'annotation `@WebService`,
— d'exposer des méthodes comme opérations SOAP avec `@WebMethod`,
— de publier le service sur une URL via `Endpoint.publish(...)`.

JAX-WS se charge automatiquement de générer le **WSDL** et de convertir les objets Java en XML (via JAXB).

2.2 Classe de démarrage : **Application**

```
import javax.xml.ws.Endpoint;

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 ");
    }
}
```

Listing 1 – Classe de déploiement du service

Cette classe :

— La classe Endpoint fait partie de JAX-WS.

— La méthode publish() démarre un serveur HTTP embarqué sur le port 8888.

— Le service exposé est l'objet MonserviceWeb. Une fois lancé, le service est accessible via le navigateur ou SOAPUI à `http://localhost:8888/?wsdl`.

2.3 Classe de service : **MonserviceWeb**

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

@WebService(targetNamespace = "https://www.galille.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);
    }
}
```

Listing 2 – Service web SOAP

Les éléments importants sont :

— `@WebService` : indique que la classe est un service web.

— `targetNamespace` : identifiant logique unique du service. Ici `"https://www.galille.fr/"` sert d'espace de noms.

— `@WebMethod(operationName = "convertir")` : expose la méthode `conversion` sous le nom d'opération `convertir` dans le WSDL.

— `@WebParam` sert à donner un nom lisible au paramètre dans le WSDL et dans le message SOAP.

2.4 Classe **Etudiant** et sérialisation XML

```java
import javax.xml.bind.annotation.XmlRootElement;
import java.io.Serializable;

@XmlRootElement
public class Etudiant implements Serializable {
    private int id;
    private String nom;
    private double moyenne;

    public Etudiant() {}

    public Etudiant(int id, String nom, double moyenne) {
        this.id = id;
        this.nom = nom;
        this.moyenne = moyenne;
    }

    // Getters et setters...
}
```

Listing 3 – Classe Etudiant sérialisable en XML

L'annotation `@XmlRootElement` permet à JAXB de convertir l'objet `Etudiant` vers du XML dans les messages SOAP.

3 Messages SOAP échangés

3.1 Exemple de réponse SOAP pour `convertir`

Nous avons utilisé l'outil SOAPUI pour tester les différentes opérations de notre service web, Après un appel à l'opération `convertir`, le service renvoie une réponse SOAP de la forme :

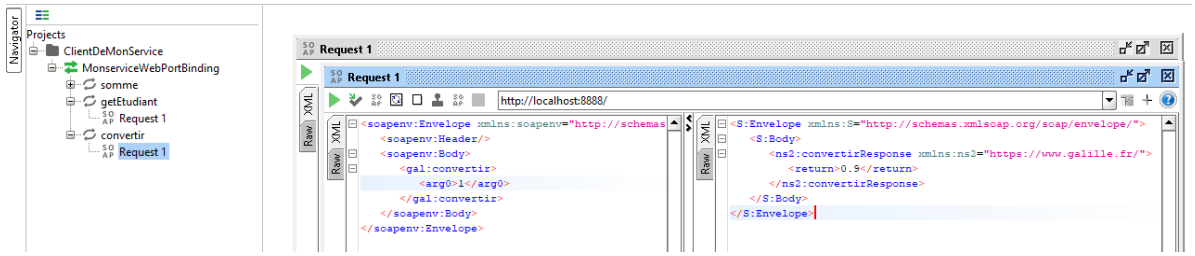

FIGURE 1 – Exemple de réponse SOAP affichée dans SOAPUI

— `S:Envelope` et `S:Body` appartiennent à la namespace SOAP standard.
— `ns2:convertirResponse` représente la réponse à l'opération `convertir`.
— `<return>0.9</return>` contient la valeur de retour de la méthode Java `conversion`.

Le rôle de la **targetNamespace** (`"https://www.galille.fr/"`) est visible dans le préfixe `ns2` : elle identifie le service et ses opérations de manière unique.

Conclusion

Ce TP m'a permis de découvrir les web services SOAP et de comprendre le rôle de JAX-WS pour exposer des méthodes Java à distance. J'ai développé un petit service avec des opérations comme convertir, somme et getEtudiant, dont les échanges sont décrits dans le WSDL et sérialisés en XML. Enfin, j'ai vérifié le bon fonctionnement du service en le testant avec SOAPUI et en observant les messages SOAP de requête et de réponse.