Rapport de TP : Guide de mise en œuvre d'un Service Web SOAP

Auteur : RANDRIANARISOA Ambinintsoa

Sujet : Développement, déploiement et consommation d'un Service Web SOAP avec JAX-WS.
Prérequis : JDK 1.8 (Java 8), un IDE (Eclipse/IntelliJ), SoapUI.

1. Objectif

Ce document détaille la procédure pour créer un service web basé sur le protocole SOAP sans utiliser de conteneur de servlets lourd (comme Tomcat), en s'appuyant uniquement sur l'API JAX-WS native de Java SE. Il couvre la manipulation de types simples puis d'objets complexes.

2. Partie I : Création et déploiement du Service Simple

Étape 1 : Création du Projet

Créez un simple **Projet Java** (Console Application). Aucune bibliothèque externe n'est nécessaire sous Java 8.

Étape 2 : Développement du POJO (Service)

Créez une classe Java classique (ex: MonServiceWeb). Pour la transformer en service web, appliquez les règles suivantes :

1. **Annotation de classe :** Ajoutez @WebService juste avant la déclaration de la classe.
    - *Optionnel :* Spécifiez targetNamespace="http://www.votre-domaine.com/" pour définir l'espace de nommage XML.
2. **Méthodes métier :** Définissez vos méthodes publiques.
    - Exemple : public double conversion(double montant) { return montant * 10.6; }
3. **Personnalisation (Facultatif) :**
    - Utilisez @WebMethod(operationName="nomOperation") pour renommer la méthode dans le WSDL.
    - Utilisez @WebParam(name="nomParametre") pour remplacer les noms par défaut (arg0, arg1) dans les arguments.

Étape 3 : Publication du Service (Serveur)

Pour déployer le service, inutile d'installer un serveur externe. Créez une classe lanceur (ex: ServeurJWS) contenant une méthode main :

1. Définissez l'URL de publication (ex: String url = "http://localhost:8888/";).
2. Utilisez la méthode statique : Endpoint.publish(url, new MonServiceWeb());.
3. Ajoutez un message console (ex: "Service déployé") pour confirmer le lancement.
4. **Exécutez l'application** en tant qu'application Java. Le service reste actif tant que la console tourne.

Étape 4 : Vérification du WSDL

Ouvrez un navigateur web et accédez à l'URL définie suivie de ?wsdl (ex: http://localhost:8888/?wsdl).
Action : Vérifiez que le document XML s'affiche. Il décrit le contrat du service (méthodes, types, adresses). Si le WSDL s'affiche, le serveur fonctionne
.

3. Partie II : Test avec un Client (SoapUI)

L'utilisation d'un outil tiers garantit l'interopérabilité du service.

1. **Lancement :** Ouvrez **SoapUI**.
2. **Nouveau Projet :** Créez un projet SOAP (New SOAP Project).
    - Nommez le projet.
    - Dans le champ "Initial WSDL", collez l'URL complète (http://localhost:8888/?wsdl).
3. **Exécution de la requête :**
    - Dépliez l'arborescence du projet pour trouver la méthode conversion.
    - Ouvrez la requête Request 1.
    - Remplacez le ? dans la balise XML (ex: <arg0>?</arg0>) par une valeur numérique.
    - Cliquez sur le bouton "Play" (flèche verte).
4. **Observation :** La réponse du serveur apparaît dans le volet droit sous forme d'une enveloppe SOAP contenant le résultat calculé.

4. Partie III : Manipulation d'Objets Complexes

Cette section explique comment échanger des objets structurés plutôt que de simples types primitifs.

Étape 1 : Création du Bean (Modèle)

Créez une classe représentant l'objet métier (ex: Etudiant). Pour que JAXB (Java Architecture for XML Binding) puisse transformer cet objet en XML, respectez impérativement ces contraintes :

1. **Sérialisation :** La classe doit implémenter l'interface Serializable.
2. **Constructeur :** La classe **doit** posséder un constructeur sans paramètres (constructeur par défaut public).

3. **Encapsulation :** Mettez les attributs en private et générez les Getters et Setters.
4. **Annotation XML :** Ajoutez @XmlRootElement au-dessus de la classe pour définir la racine de l'arbre XML.

Étape 2 : Mise à jour du Service

Dans la classe MonServiceWeb, ajoutez une méthode qui retourne cet objet complexe :

- Exemple : public Etudiant getEtudiant(int id) { ... }
- Instanciez un étudiant, définissez ses valeurs (setters) et retournez l'objet.

Étape 3 : Redéploiement et Test

1. **Arrêtez et relancez** le serveur Java (nécessaire pour prendre en compte les changements de code).
2. Dans SoapUI, faites un clic droit sur le WSDL du projet > **Update Definition**.
3. Testez la nouvelle méthode getEtudiant.
4. **Résultat :** Observez que la réponse XML est structurée hiérarchiquement, représentant les attributs de l'objet (ex: <nom>...</nom>, <moyenne>...</moyenne>).

5. Points de vigilance lors de la reproduction

- **Version Java :** Utilisez impérativement **Java 8** (ou ajoutez les bibliothèques JAX-WS manuellement pour les versions ultérieures).
- **Port occupé :** Si le port 8888 est déjà utilisé, changez-le dans l'URL de publication (ex: 8080 ou 9000).
- **Constructeur vide :** L'oubli du constructeur sans paramètres dans la classe Etudiant est la cause principale d'erreur ("JAXBException") lors de la manipulation d'objets complexes.