Rapport TP - Services Web SOAP/WSDL

1 Architecture et Concepts Fondamentaux

1.1 Les Technologies Utilisées

JAX-WS et JAXB : JAX-WS (Java API for XML Web Services) fournit les annotations nécessaires pour transformer une classe Java en service web. Ces annotations reposent sur JAXB (Java Architecture for XML Binding) qui assure la **sérialisation et désérialisation automatique** entre objets Java et XML. Cela permet de créer des objets Java depuis des fichiers XML et vice-versa.

SOAP (Simple Object Access Protocol) : Protocole de communication basé sur XML permettant l'échange de messages structurés entre applications distribuées. Un message SOAP est encapsulé dans une requête HTTP dont le corps contient du XML.

WSDL (Web Services Description Language) : Fichier XML décrivant complètement le service web :

— Le nom du service et son URL d'accès (endpoint)
— Les opérations disponibles avec leurs paramètres d'entrée et de sortie
— Les types de données manipulés (consultables via le schéma XSD associé)

Point important : Le WSDL expose uniquement l'interface du service, jamais l'implémentation ou la logique métier (les méthodes de calcul ou les détails des algorithmes ne sont jamais visibles).

1.2 URI, URL et URN

— **URL (Uniform Resource Locator) :** Chaîne de caractères indiquant l'emplacement d'une ressource, permettant d'accéder à un emplacement ou une ressource en utilisant son adresse
— **URN (Uniform Resource Name) :** Identifiant fournissant des informations sur une ressource (son nom)
— **URI (Uniform Resource Identifier) :** Concept englobant URL et URN, fournissant l'emplacement et/ou le nom d'une ressource (URI = URL + URN)

1.3 Interopérabilité

L'un des avantages majeurs de SOAP est l'**interopérabilité** : la capacité à faire communiquer des applications développées dans des langages différents (pas forcément du même type) grâce à l'utilisation du standard XML.

2 Implémentation du Service Web

2.1 Annotations Clés

Pour transformer une classe Java en service web, on utilise plusieurs annotations essentielles :

— **@WebService(targetNamespace = "...")** : Transforme la classe en service web. Le `targetNamespace` permet de différencier les services et d'éviter les conflits de noms entre différents services web.
— **@WebMethod(operationName = "...")** : Personnalise le nom de l'opération exposée dans le WSDL. Si cette annotation est utilisée, c'est ce nom qui sera affiché dans le WSDL et dans SoapUI, et non le nom original de la méthode Java.
— **@WebParam(name = "...")** : Permet de nommer explicitement les paramètres des méthodes dans le WSDL. Sans cette annotation, les noms des paramètres peuvent être génériques (arg0, arg1, etc.).

2.2 Gestion des Objets Complexes

Pour permettre la manipulation d'objets Java personnalisés dans un service web SOAP, plusieurs éléments sont nécessaires :
— **@XmlRootElement :** Cette annotation JAXB permet la sérialisation automatique de l'objet en XML. Elle indique que la classe peut être transformée en élément XML racine.
— **implements Serializable :** L'interface Serializable est nécessaire pour permettre la transmission de l'objet à travers le réseau. Elle marque l'objet comme étant convertible en flux d'octets.
— **Constructeur par défaut :** Un constructeur sans paramètres est obligatoire pour la désérialisation JAXB. Sans lui, JAXB ne peut pas recréer l'objet à partir du XML reçu.

2.3 Déploiement du Service

Le déploiement du service web s'effectue grâce à la méthode `Endpoint.publish()`. Cette méthode prend deux paramètres :
— L'URL où le service sera accessible (l'endpoint)
— Une instance de la classe du service web
Cette méthode déploie le service à distance et le rend immédiatement accessible aux clients distants.

3 Tests et Validation

3.1 Accès au WSDL

Une fois le service déployé, le WSDL est automatiquement généré et accessible à l'adresse : `url/?wsdl` (où `url` est l'endpoint du service).
Ce fichier contient toutes les informations nécessaires pour utiliser le service :
— La description complète du service (nom, location)
— Les opérations disponibles avec leurs entrées et sorties
— Le schéma XSD des types de données : accessible via `url/?xsd=1`
Remarque importante : Le WSDL est sauvegardé dans le JNDI (Java Naming and Directory Interface) de Java pour permettre de tester le service web. Il est régénéré à chaque exécution/compilation. Pour prendre en compte des modifications dans l'application, il ne suffit pas d'actualiser la page, il faut arrêter le déploiement et relancer l'application pour que les changements soient pris en compte.

3.2 Utilisation de SoapUI

SoapUI est un outil qui joue le rôle du client et nous permet de tester notre serveur SOAP. En important le WSDL du service, SoapUI génère automatiquement les requêtes SOAP pour chaque opération disponible. Cela permet de valider le bon fonctionnement du service web sans avoir à développer un client complet.

3.3 Structure d'un Message SOAP

Un message SOAP observé dans SoapUI suit une structure XML standardisée. Voici un exemple concret d'une requête et de sa réponse :

Exemple de requête SOAP (appel de l'opération convertir avec le paramètre 100) :

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:poly="http://www.polytech.fr">
   <soapenv:Header/>
   <soapenv:Body>
      <poly:convertir>
         <mt>100</mt>
      </poly:convertir>
   </soapenv:Body>
</soapenv:Envelope>
```

Listing 1 – Requête SOAP

Exemple de réponse SOAP :

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:convertirResponse xmlns:ns2="http://www.polytech.fr">
         <return>90.0</return>
      </ns2:convertirResponse>
   </S:Body>
</S:Envelope>
```

Listing 2 – Réponse SOAP

On observe qu'un message SOAP est en réalité une **requête HTTP dont le corps est en XML**. La structure comporte :

- **Enveloppe SOAP** (`Envelope`) : Conteneur principal du message
- **En-tête** (`Header`) : Section optionnelle pour les métadonnées
- **Corps** (`Body`) : Contient l'opération appelée avec ses paramètres, et dans la réponse, le résultat dans la balise `<return>`

Les résultats des opérations sont retournés dans des balises spécifiques (comme `<return>` dans cet exemple), permettant une identification claire des données de réponse.