Mise en place d’un service web SOAP en Java

Dans ce compte-rendu, je présente les étapes que j’ai suivies pour créer et tester un service web SOAP en Java. Le but était de comprendre comment exposer une classe comme service web, générer automatiquement un fichier WSDL, et tester les méthodes via un client SOAP comme SoapUI.
Le tout a été réalisé en Java 8, car cette version intègre directement les bibliothèques nécessaires pour SOAP (JAX-WS et JAXB).

1. Création du projet Java et configuration

J’ai commencé par créer un projet Java simple dans IntelliJ IDEA.
Le choix de **Java 8** est essentiel, car les versions plus récentes ne contiennent plus les packages liés aux services web SOAP.

Une fois le projet créé, toutes les classes ont été placées dans le dossier src.

2. Mise en place de la classe Application

La première classe créée est la classe qui contient la méthode principale.
Elle a servi à :

- démarrer le programme ;
- définir l’URL du service web ;
- publier le service grâce à un composant Java capable d’exposer un service sans serveur externe.

Cette classe joue le rôle de “mini serveur” pour tester le service en local.

3. Création du service SOAP (MonServiceWeb)

La deuxième classe contient la logique du service web : ce sont les méthodes que le client pourra appeler.
Pour que cette classe soit reconnue comme un service SOAP, elle doit être annotée de manière spécifique.

Une fois annotée, Java est capable de :

- l’exposer automatiquement comme service web ;
- générer un fichier WSDL décrivant son interface ;
- gérer les échanges XML entre le client et le serveur.

Le service contenait au début une seule méthode simple de type “conversion”.

4. Comprendre JAX-WS et JAXB

JAX-WS sert à créer le service web SOAP.
Il analyse les classes, les annotations, et génère automatiquement l'interface WSDL utilisée par les clients.

JAXB, lui, gère la transformation des données :

- objets Java → XML (sérialisation)
- XML → objets Java (désérialisation)

Ces deux technologies fonctionnent ensemble dans un service SOAP.

5. Déploiement du service et accès au WSDL

En lançant l'application, le service web est accessible localement sur une URL précise.
En ajoutant ?wsdl à la fin de cette adresse, on obtient le WSDL généré automatiquement.

Ce fichier contient :

- les opérations disponibles ;
- leurs paramètres ;
- les types de données utilisés ;
- l'adresse du service.

Le WSDL ne décrit pas le fonctionnement interne des méthodes, uniquement leur interface.

6. Test du service avec SoapUI

Pour tester le service, j'ai utilisé SoapUI.
J'ai créé un projet SOAP, entré l'URL du WSDL, et l'outil a généré automatiquement les requêtes nécessaires.

Il suffit ensuite de :

- modifier les arguments dans le message XML ;
- envoyer la requête ;
- consulter la réponse, elle aussi en XML.

Cela permet de vérifier facilement que le service fonctionne correctement.

7. Ajout de nouvelles méthodes

Pour ajouter une méthode supplémentaire dans le service, il suffit de l'écrire dans la classe du service.
Après cela, il faut :

1. arrêter l'application,
2. la relancer pour republier le service,
3. mettre à jour le WSDL dans SoapUI.

La nouvelle méthode apparaît alors automatiquement comme opération SOAP.

8. Manipulation d’objets Java via JAXB

La manipulation d’objets Java a été introduite afin de permettre au service web de renvoyer autre chose que de simples valeurs numériques. Une classe représentant un étudiant, contenant un identifiant, un nom et une moyenne, a été définie pour servir de structure de données. Les annotations appropriées ont été ajoutées pour que l’objet puisse être sérialisé en XML par JAXB. Une opération supplémentaire a ensuite été intégrée au service afin de retourner une instance de cette classe. Après redéploiement, la requête envoyée depuis le client SOAP reçoit en réponse un document XML structuré contenant l’ensemble des champs de l’objet étudiant.