

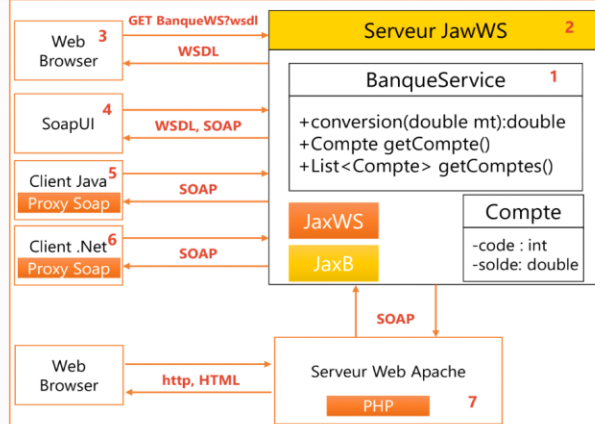
TP1 : Web Service SOAP WSDL

Application

1. Créer un Web service qui permet de :

- Convertir un montant de l'euro en DH
- Consulter un Compte
- Consulter une Liste de comptes

2. Déployer le Web service avec un simple Serveur JaxWS
3. Consulter et analyser le WSDL avec un Browser HTTP
4. Tester les opérations du web service avec un outil comme SoapUI ou Oxygen
5. Créer un Client SOAP Java
6. Créer un Client SOAP Dot Net
7. Créer un Client SOAP PHP
8. Déployer le Web Service dans un Projet Spring Boot



2 -Déploiement du web Service :

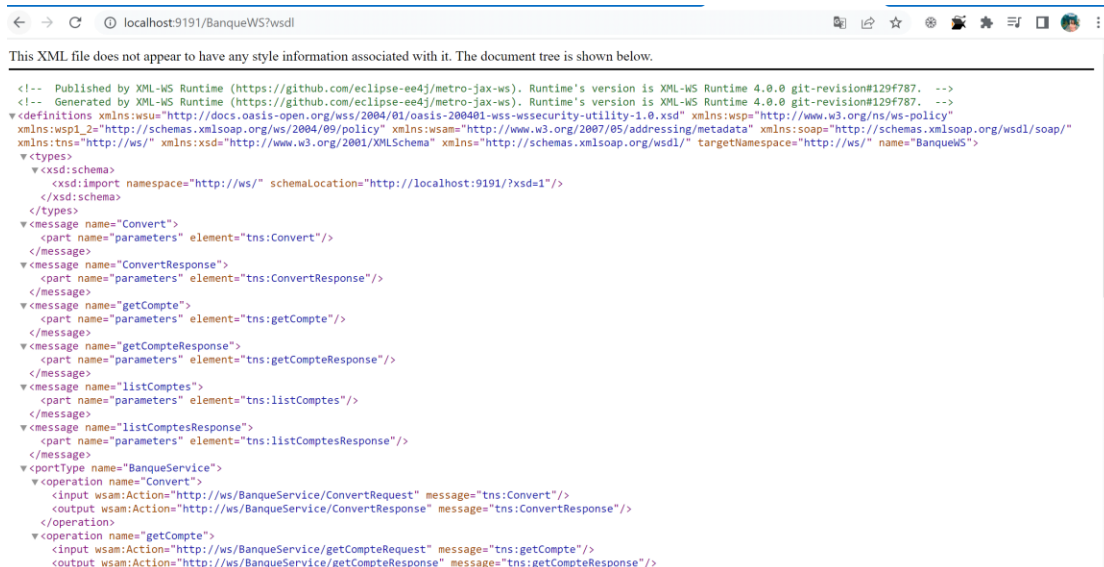
The screenshot shows an IDE with the following components:

- Project Explorer**: Shows the project structure with folders for `src`, `main`, `java`, `ws`, `resources`, `test`, and `target`. The `ws` folder contains `BanqueService` and `Compte`.
- Code Editor**: Displays the `ServerJWS.java` file with the following code:

```
1 import jakarta.xml.ws.Endpoint;
2 import ws.BanqueService;
3
4 public class ServerJWS {
5     public static void main(String[] args){
6         /*permet de démarrer un serveur http pour consulter uniquement ce webService*/
7         Endpoint.publish(address: "http://0.0.0.0:9191/", new BanqueService());
8         System.out.println("Web service déployé sur http://0.0.0.0:9191/");
9     }
10 }
11
```
- Run Console**: Shows the output of the `ServerJWS` application:

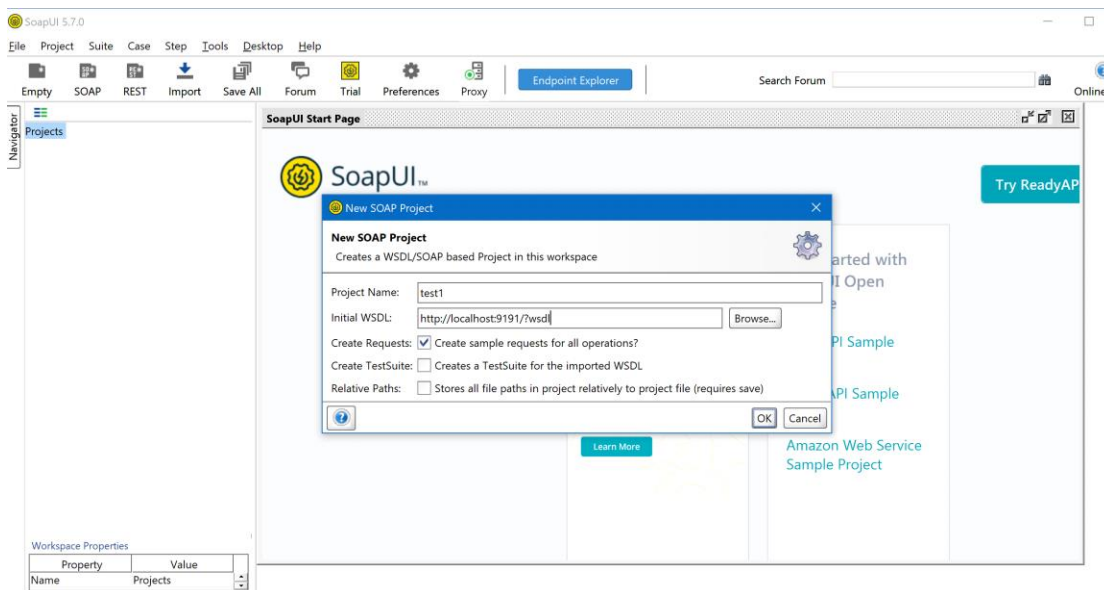
```
Run: ServerJWS X
C:\Users\lensy\.jdk\corretto-17.0.5\bin\java.exe ...
Web service déployé sur http://0.0.0.0:9191/
```

3-



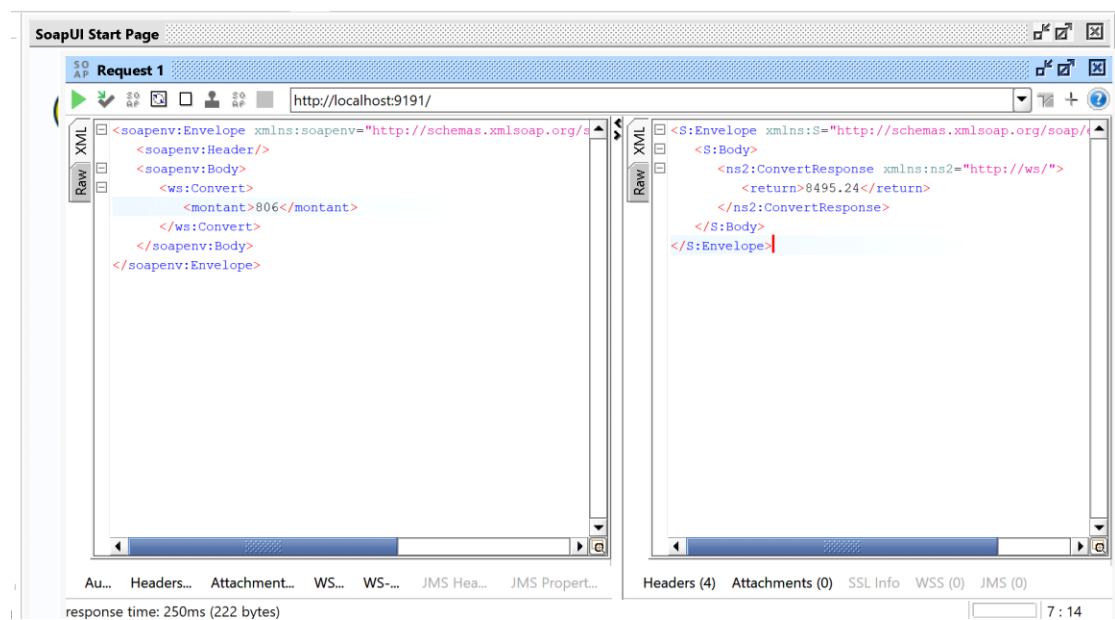
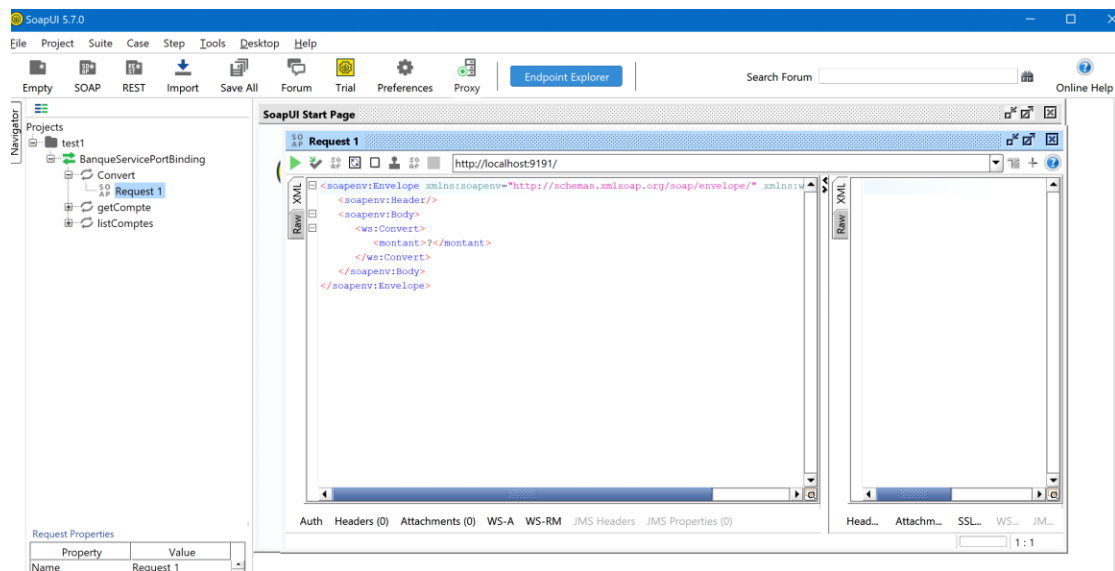
WSDL : Fichier xml qui permet de faire la définition du web service, pour chaque méthode, il déclare un msg pour input et un autre pour l'output. Les types de données sont déclarés en type xsd (schéma xml).

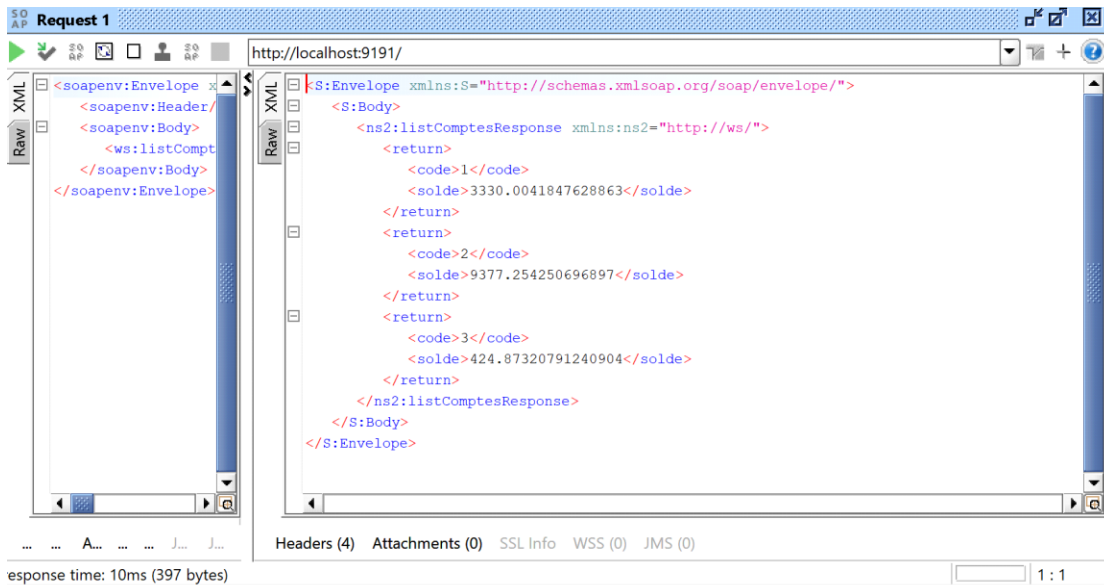
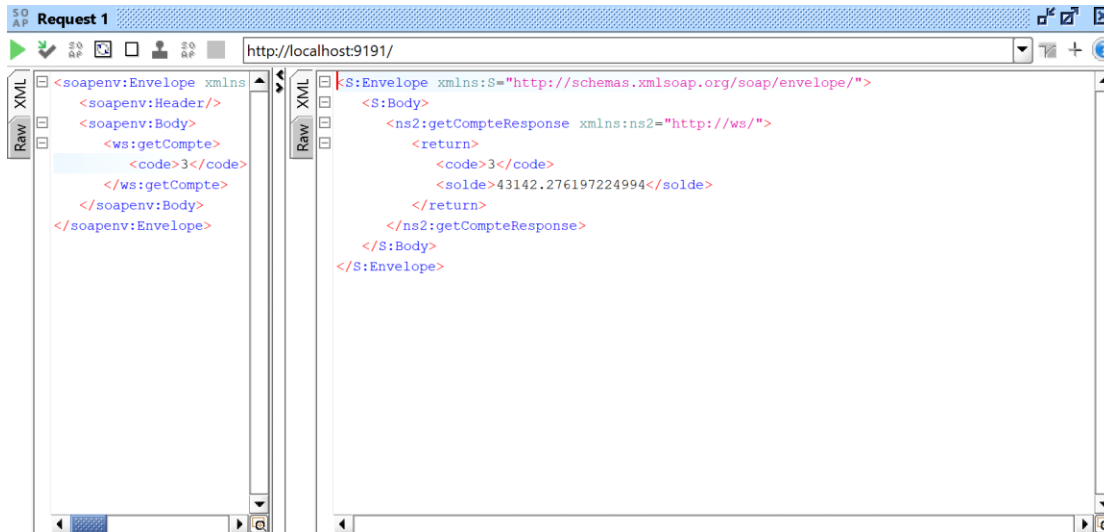
4- Tester le web service :



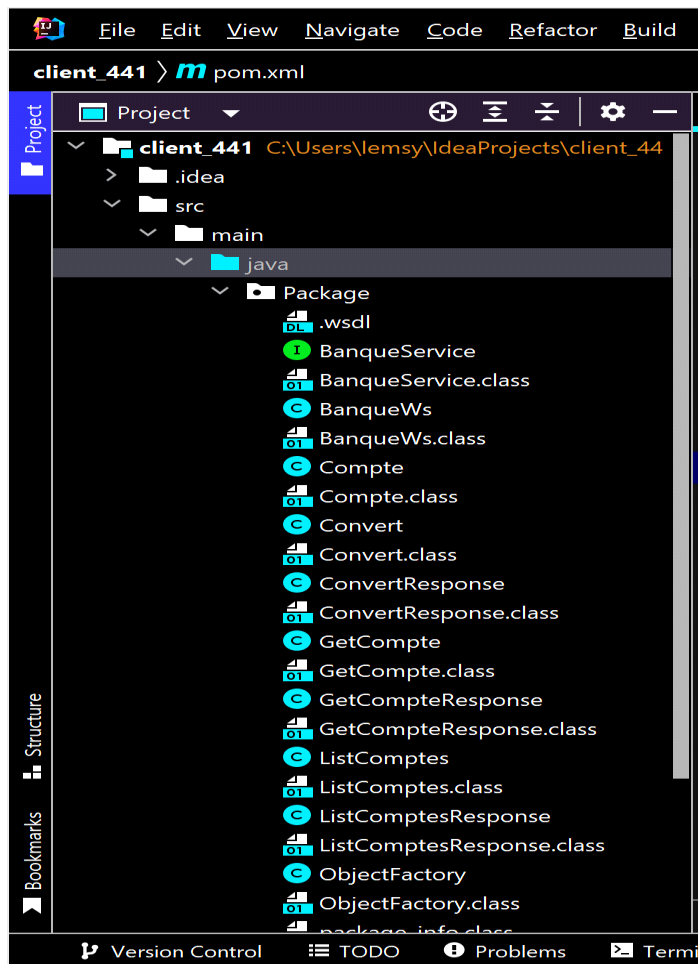
Test de la méthode convert :

-envoyer le montant en paramètre, le web service fait le traitement de conversion et renvoie la requête.





5 - SOAP Côté client :



Class ClientWS :

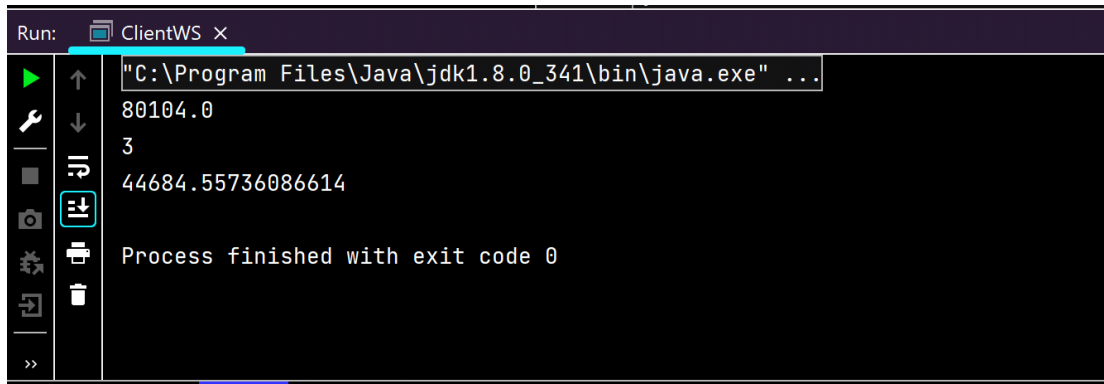
```

1  import Package.BanqueService;
2  import Package.BanqueWs;
3  import Package.Compte;
4
5  public class ClientWS {
6      public static void main(String[] args) {
7          BanqueService stub = new BanqueWs().getBanqueServicePort();
8          System.out.println(stub.convert(montant: 7600));
9          Compte cp = stub.getCompte(code: 3);
10         System.out.println(cp.getCode());
11         System.out.println(cp.getSolde());
12
13     }
14 }
15

```

Après consultation :

Client veut communiquer avec le webService -> on utilise Stub comme intermédiaire côté client, on appelle la méthode conversion et stub envoie une requête vers le serveur (skeleton aka proxy côté serveur), le skeleton renvoie la réponse soap (xml) grâce au wsdl.



The screenshot shows a 'Run' console window titled 'ClientWS x'. The command executed is `"C:\Program Files\Java\jdk1.8.0_341\bin\java.exe" ...`. The output consists of three lines of numbers: `80104.0`, `3`, and `44684.55736086614`. The final line of output is `Process finished with exit code 0`. The console window has a dark background and a toolbar on the left with various icons for running, debugging, and other IDE functions.

```
Run: ClientWS x
"C:\Program Files\Java\jdk1.8.0_341\bin\java.exe" ...
80104.0
3
44684.55736086614
Process finished with exit code 0
```

- Client demande au stub de faire appel à la méthode.
- Le Stub se connecte au skeleton et lui envoie une requête SOAP
- Le Skeleton fait appel à la méthode du web Service.
- Le web Service retourne le résultat au Skeleton.
- Le Skeleton envoie le résultat dans la réponse Soap
- Le Stub fournit le résultat au client