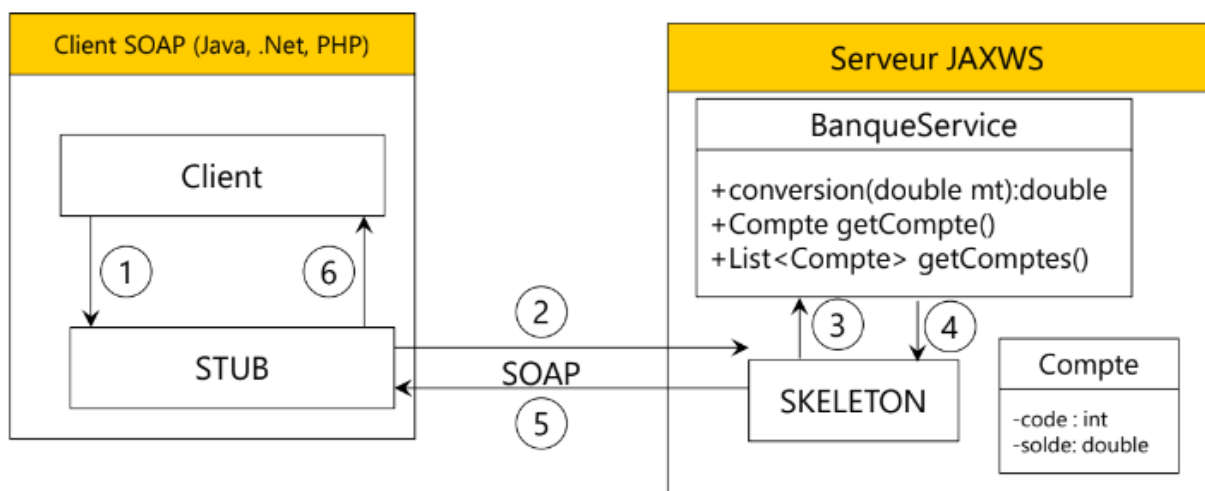


## COMPTE-RENDU DE L'ACTIVITE PRATIQUE N° 3 : WEB SERVICES SOAP, WSDL, UDDI AVEC JAXWS

Filière : « Ingénierie Informatique : Big Data et Cloud  
Computing » II-BDCC



Réalisé par :

Khadija BENJILALI

Encadré par :

Pr. Mohamed YOUSSEFI

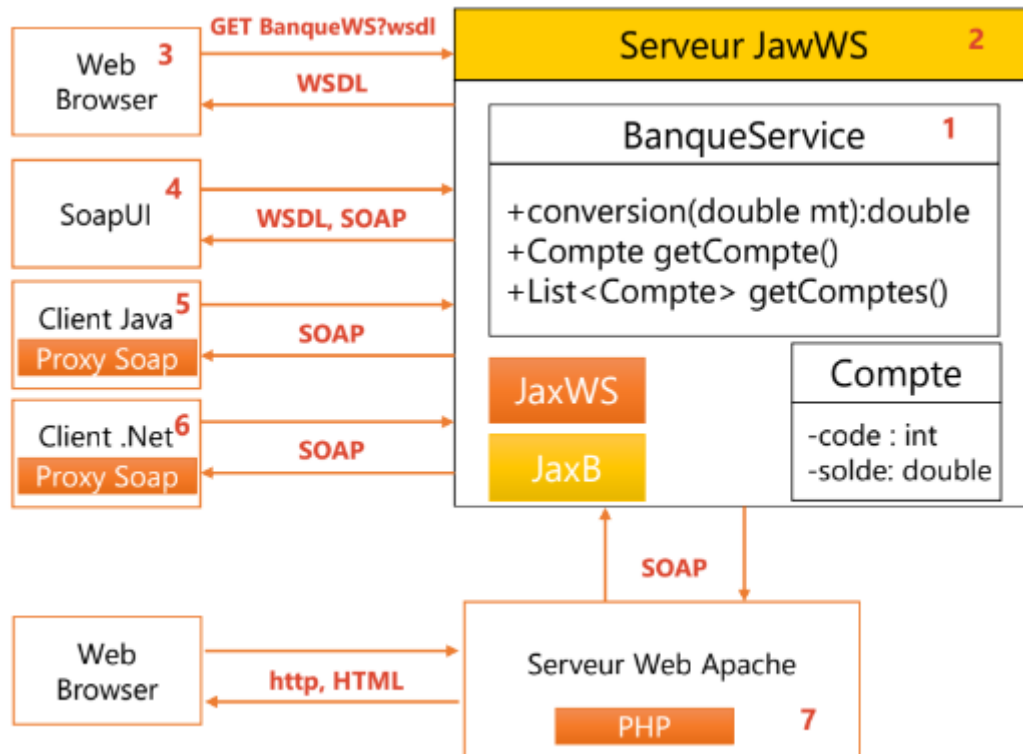
Année Universitaire : 2022-2023

# Sommaire

<b>Travail à faire.....</b>	<b>1</b>
1.   Création du Web service « BanqueWS » : .....	2
2.   Déployer le Web service « BanqueWS» : .....	2
3.   Visualiser le WSDL avec un Browser HTTP :.....	2
4.   Tester le Web Service de type SOAP en utilisant SoapUI : .....	4
5.   Création d'un Client SOAP Java :.....	5
<b>Résumé .....</b>	<b>6</b>

# Travail à faire

Créer un Microservice Spring boot Multi Connecteurs SOAP, REST, GRAPHQL qui permet de gérer des comptes:



1. Créer un Web service qui permet de :
  - Convertir un montant de l'auro 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

## 1. Création du Web service « BanqueWS » :

```
10 //POJO Plain Old Java Object
11 @WebService(serviceName = "BanqueWS")
12 public class BanqueService
13 {
14     @WebMethod(operationName = "convert")
15     public double conversion(@WebParam(name = "montant") double mt){
16         return mt*11.54;
17     }
18
19     @WebMethod
20     public Compte getCompte(@WebParam(name = "code") int code)
21     {
22         return new Compte(code, solde: Math.random()*9000, new Date());
23     }
24
25     @WebMethod
26     public List<Compte> listCompte()
27     {
28         return List.of(
29             new Compte( code: 1, solde: Math.random()*9000, new Date()),
30             new Compte( code: 2, solde: Math.random()*9000, new Date()),
31             new Compte( code: 3, solde: Math.random()*9000, new Date())
32         );
33     }
34 }
```

## 2. Déployer le Web service « BanqueWS » :

Créer mon propre serveur http JaxWS qui permet d'accéder au web service « BanqueService »

```
1 import jakarta.xml.ws.Endpoint;
2 import ws.BanqueService;
3
4 public class ServerJWS
5 {
6     public static void main(String[] args)
7     {
8         //permet de démarrer un petit serveur http qui va utiliser le port 9191
9         // Ce serveur http est utilisé pour consulter uniquement le ws "BanqueService"
10        Endpoint.publish( address: "http://0.0.0.0:9191/", new BanqueService());
11        System.out.printf("Web service déployé sur http://0.0.0.0:9191/ ");
12    }
13
14 }
```

## 3. Visualiser le WSDL avec un Browser HTTP :

WSDL c'est un fichier xml qui permet de faire la description de web service dont on trouve les méthodes et leurs entrées/sorties ...

localhost:9191/?wsdl

localhost:9191/?wsdl

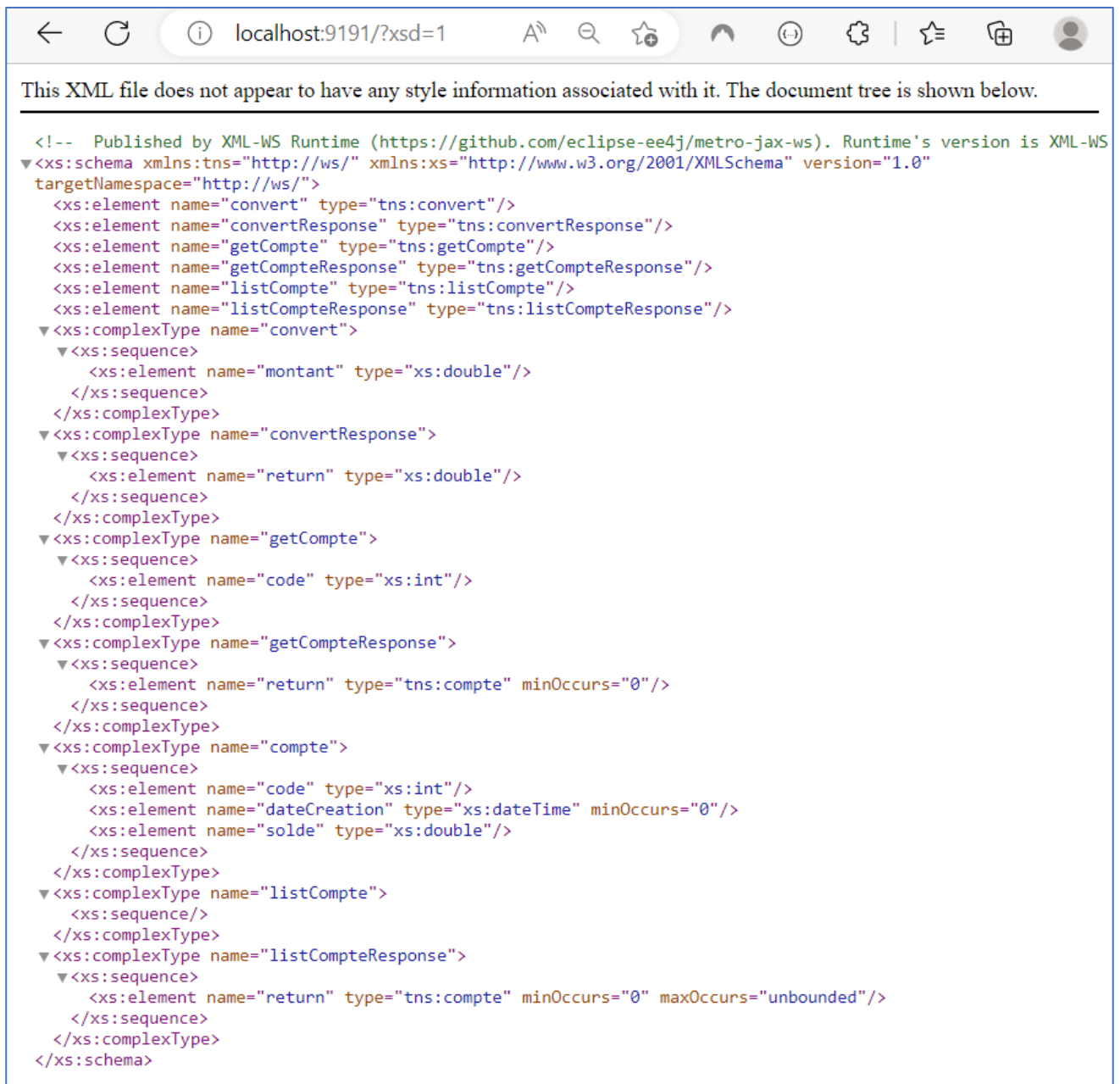
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<!-- Published by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is XML
<!-- Generated by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is XML
▼ <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://ws/" name="BanqueWS">
  ▼ <types>
    ▼ <xsd:schema>
      <xsd:import namespace="http://ws/" schemaLocation="http://localhost:9191/?xsd=1"/>
    </xsd:schema>
  </types>
  ▼ <message name="convert">
    <part name="parameters" element="tns:convert"/>
  </message>
  ▼ <message name="convertResponse">
    <part name="parameters" element="tns:convertResponse"/>
  </message>
  ▼ <message name="getCompte">
    <part name="parameters" element="tns:getCompte"/>
  </message>
  ▼ <message name="getCompteResponse">
    <part name="parameters" element="tns:getCompteResponse"/>
  </message>
  ▼ <message name="listCompte">
    <part name="parameters" element="tns:listCompte"/>
  </message>
  ▼ <message name="listCompteResponse">
    <part name="parameters" element="tns:listCompteResponse"/>
  </message>
  ▼ <portType name="BanqueService">
    ▼ <operation name="convert">
      <input wsam:Action="http://ws/BanqueService/convertRequest" message="tns:convert"/>
      <output wsam:Action="http://ws/BanqueService/convertResponse" message="tns:convertResponse"/>
    </operation>
    ▼ <operation name="getCompte">
      <input wsam:Action="http://ws/BanqueService/getCompteRequest" message="tns:getCompte"/>
      <output wsam:Action="http://ws/BanqueService/getCompteResponse" message="tns:getCompteResponse"/>
    </operation>
    ▼ <operation name="listCompte">
      <input wsam:Action="http://ws/BanqueService/listCompteRequest" message="tns:listCompte"/>
      <output wsam:Action="http://ws/BanqueService/listCompteResponse" message="tns:listCompteResponse"/>
    </operation>
  </portType>

```

Le schéma XML se compose essentiellement de déclarations d'éléments et d'attributs et de définitions de types de chaque fonctions et attributs du service web « BanqueWS »



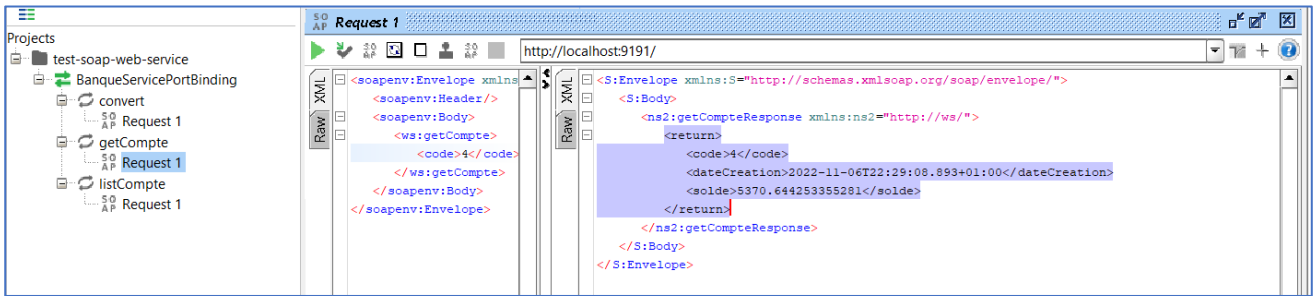
```
<!-- Published by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is XML-WS
<xs:schema xmlns:tns="http://ws/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
targetNamespace="http://ws/"
  <xs:element name="convert" type="tns:convert"/>
  <xs:element name="convertResponse" type="tns:convertResponse"/>
  <xs:element name="getCompte" type="tns:getCompte"/>
  <xs:element name="getCompteResponse" type="tns:getCompteResponse"/>
  <xs:element name="listCompte" type="tns:listCompte"/>
  <xs:element name="listCompteResponse" type="tns:listCompteResponse"/>
  <xs:complexType name="convert">
    <xs:sequence>
      <xs:element name="montant" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="convertResponse">
    <xs:sequence>
      <xs:element name="return" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getCompte">
    <xs:sequence>
      <xs:element name="code" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getCompteResponse">
    <xs:sequence>
      <xs:element name="return" type="tns:compte" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="compte">
    <xs:sequence>
      <xs:element name="code" type="xs:int"/>
      <xs:element name="dateCreation" type="xs:dateTime" minOccurs="0"/>
      <xs:element name="solde" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="listCompte">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="listCompteResponse">
    <xs:sequence>
      <xs:element name="return" type="tns:compte" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

#### 4. Tester le Web Service de type SOAP en utilisant SoapUI :

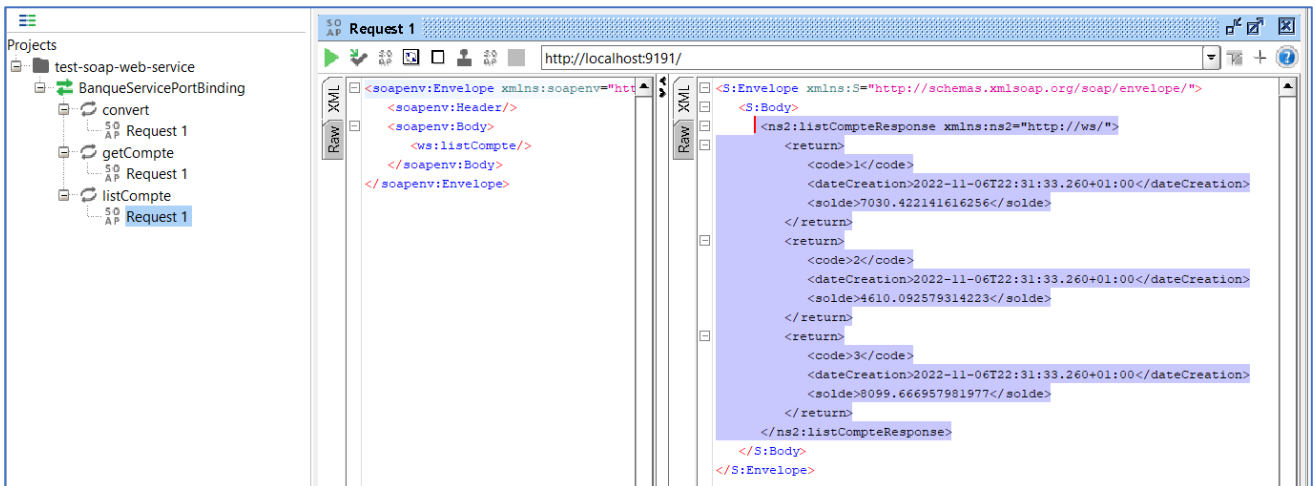
- Tester la méthode « **convert** » de web service SOAP



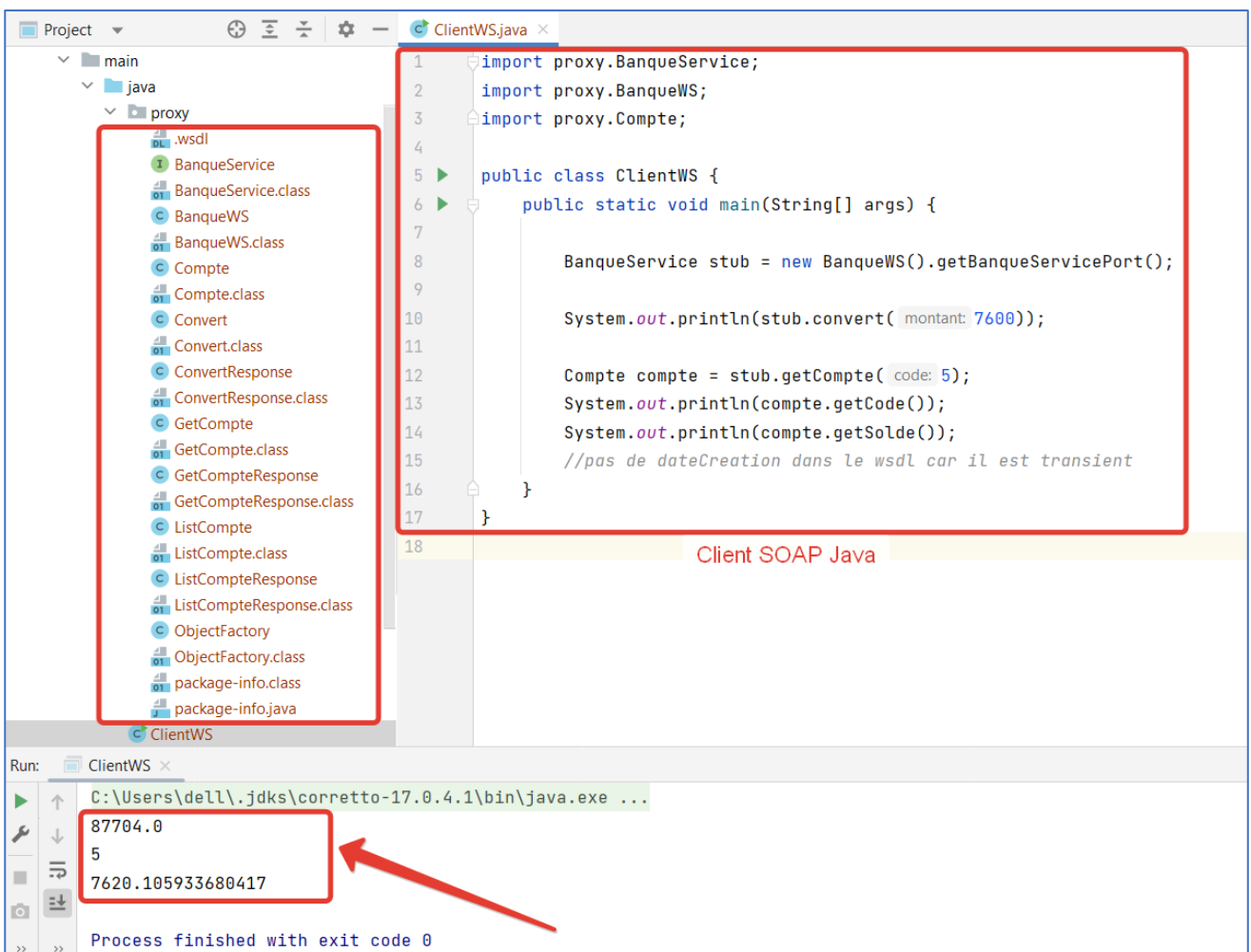
➤ Tester la méthode « **getCompte** » de web service SOAP



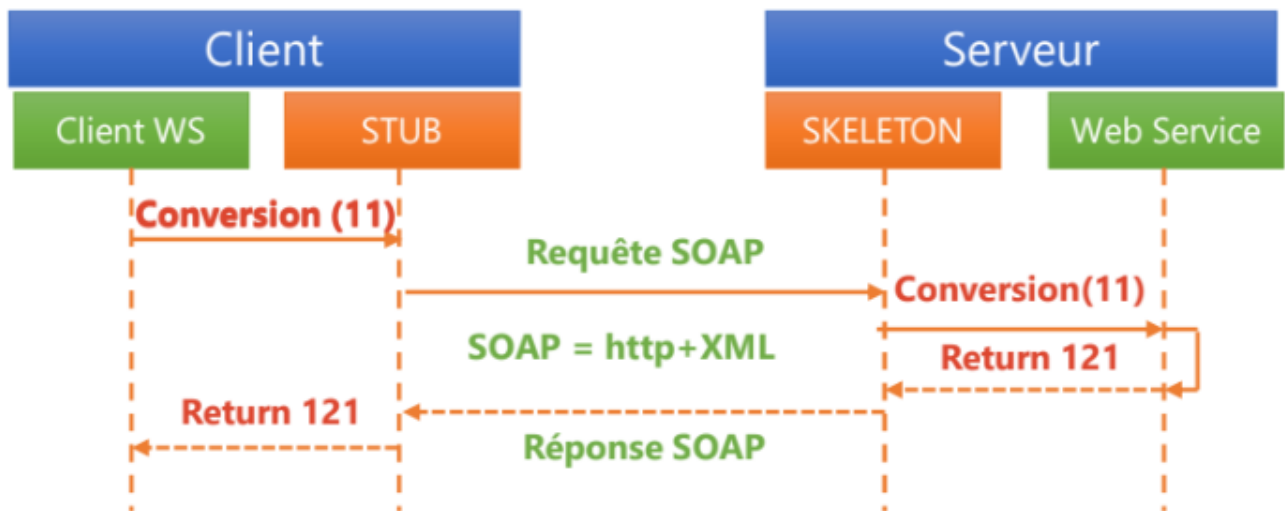
➤ Tester la méthode « **listCompte** » de web service SOAP



## 5. Création d'un Client SOAP Java :



## Résumé



- 1) Le client demande au stub de faire appel à la méthode conversion(12)
- 2) Le Stub se connecte au Skeleton et lui envoie une requête SOAP
- 3) Le Skeleton fait appel à la méthode du web service
- 4) Le web service retourne le résultat au Skeleton
- 5) Le Skeleton envoie le résultat dans une la réponse SOAP au Stub
- 6) Le Stub fournie le résultat au client