

Département d'Informatique

Filière : IAAD

A,U :2023-2024



Université Moulay Ismail faculté des Sciences Meknès

Département d'Informatique

TP N 5 : web services SOAP WDSL

Module : Systèmes Distribués

Réalisé par :

- ***Illa Meryeme***

Dans la première partie, on a parlé des concepts fondamentaux des Web Services. Principalement, les trois concepts fondamentaux SOAP, WSDL, UDDI. Il est question de mettre en œuvre un exemple de Web Service basé sur SOAP en utilisant Jax-ws. Pour cela, nous allons utiliser une spécification javaxws qui fait partie des spécifications JEE. Pour créer un web service basé sur SOAP pour les applications Java, en utilisant Javax Ws, c'est très simple. Ici, on va créer une classe banque service. Tout simplement, c'est une classe qui va être utilisée des annotations Jax-ws. La notation qui permet d'indiquer qu'il s'agit d'un web service, c'est WebService. Pour spécifier le nom du web service, on peut utiliser ServiceName. Par exemple, Banque WS. Et chaque méthode, par exemple, j'ai une méthode conversion qui reçoit un montant en input et qui permet de retourner le montant, plutôt de convertir ce montant de l'euro en dirham. Et donc chaque méthode, pour qu'il fasse partie des méthodes du web service, il va utiliser la notation web méthode. Et donc pour attribuer un nom à l'opération, on va utiliser operation name. Donc, operation name, si vous voulez ici, operationName, ça permet de spécifier le nom de l'opération.

Si vous mettez pas operationName par défaut, il va prendre le même nom que la méthode. Et après, il y a les paramètres. Ici, le paramètre s'appelle MT. Au niveau du web service, je voudrais l'appeler Montant. Donc, je vais utiliser l'annotation WEPARAM. Donc, WEPARAM_Name = Montant. Vous voyez, les annotations principales pour créer un web service sont WEPERVICE. Web méthode et Webparam. La même chose ici. Vous avez une méthode test, une méthode get_compte, une méthode qui permet de consulter la liste des comptes. On va essayer de montrer comment créer d'abord en premier lieu, voir comment créer le web service. Ça, c'est la première étape. Deuxième étape, comment déployer le web service à travers un serveur Jax-ws, simple dans un premier temps. Et par la suite, on va voir comment on travaille avec Spring, comment déployer un web service basé sur SOAP dans une application Spring. Après, en troisième lieu, on va essayer de voir avec un browser, on va consulter le WSDL pour l'analyser et voir sa structure. Et puis, on va voir comment tester les web services en utilisant un outil qui s'appelle SOAPUI, c'est un outil de test des web services. Et puis, on va créer un client Java qui permet de consommer le web service. Et après, vous aurez quelques demandes supplémentaires par la suite où vous pouvez consulter si vous voulez voir comment faire ça avec ces c# et comment le faire avec avec PHP.

On crée un projet Java, Maven sans passer par spring

Dans un package ws, je vais créer une classe que je vais appeler banque service et une classe Compte.

```
BanqueService.java x Compte.java x
1 package ws;
2
3 import java.time.LocalDate;
4 import java.util.List;
5
6 public class BanqueService {
7     public double conversion(double mt) {
8         return mt*11;
9     }
10
11     public Compte getCompte(int code) {
12         return new Compte(code, solde: Math.random()*60000, LocalDate.now());
13     }
14     public List<Compte> listComptes() {
15         return List.of(
16             new Compte( code: 1, solde: Math.random()*60000, LocalDate.now()),
17             new Compte( code: 2, solde: Math.random()*60000, LocalDate.now()),
18             new Compte( code: 3, solde: Math.random()*60000, LocalDate.now())
19         );
20     }
}
```

```
Service.java x Compte.java x
package ws;

import java.time.LocalDate;

public class Compte {
    private int code;
    private double solde;
    private LocalDate dateCreation;

    public Compte(int code, double solde, LocalDate dateCreation) {
        this.code = code;
        this.solde = solde;
        this.dateCreation = dateCreation;
    }

    public Compte() {
    }

    public int getCode() {
        return code;
    }
}
```

Alors maintenant, pour que cette classe devienne un web service, j'ai besoin d'utiliser Jax-ws. Et pour cela, j'ai besoin d'utiliser la notation WebService et pour utiliser Jax-ws, il faut l'ajouter comme dépendance au projet

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/com.sun.xml.ws/jaxws-ri -->
  <dependency>
    <groupId>com.sun.xml.ws</groupId>
    <artifactId>jaxws-ri</artifactId>
    <version>4.0.2</version>
    <type>pom</type>
  </dependency>
</dependencies>
```

On utilise l'annotation WebService et on va attribuer un nom BanqueWS

```
BanqueService.java
2
3 import jakarta.jws.WebService;
4
5 import java.time.LocalDate;
6 import java.util.List;
7 @WebService(serviceName = "BanqueWS")
8 public class BanqueService {
9     public double conversion(double mt) {
10         return mt*11;
11     }
12 }
```

Chaque méthode doit être annotée webMethod et on attribue un nom à l'opération operationName et on affecte pour chaque paramètre l'annotation @WebParam

```

BanqueService.java
4  import jakarta.jws.WebParam;
5  import jakarta.jws.WebService;
6
7  import java.time.LocalDate;
8  import java.util.List;
9  @WebService(serviceName = "BanqueWS")
10 public class BanqueService {
11     @WebMethod(operationName = "ConversionEuroToDH")
12     public double conversion(@WebParam(name = "montant") double mt) {
13         return mt*11;
14     }
15     @WebMethod
16     public Compte getCompte(@WebParam(name="code") int code) {
17         return new Compte(code, solde: Math.random()*60000, LocalDate.now());
18     }
19     @WebMethod
20     public List<Compte> listComptes() {
21         return List.of(
22             new Compte( code: 1, solde: Math.random()*60000, LocalDate.now()),
23             new Compte( code: 2, solde: Math.random()*60000, LocalDate.now()),

```

Maintenant, on a besoin de déployer le website et pour ce faire, on va utiliser à nous de créer notre propre serveur jax-ws

```

Project
└─ ws-soap-fsm C:\Users\lenovo\IdeaProjects\ws-soap-fsm
   └─ src
      └─ main
         └─ java
            └─ ws
               └─ BanqueService
                  └─ Compte
                     └─ ServerJWS
                        └─ resources
                           └─ test
                              └─ target
                                 └─ .gitignore
                                    HELP.md
                                    mvnw
                                    mvnw.cmd

ServerJWS.java
1  import jakarta.xml.ws.Endpoint;
2  import ws.BanqueService;
3
4  public class ServerJWS {
5      public static void main(String[] args) {
6
7          String url= "http://0.0.0.0:9090/";
8          Endpoint.publish(url, new BanqueService());
9          System.out.println("Web service déployé sur "+url);
10     }
11 }
12 }

Run: ServerJWS
C:\Users\lenovo\.jdk\temurin-17.0.11\bin\java.exe ...
Web service déployé sur http://0.0.0.0:9090/

```

On démarre le serveur jax-ws et on va démarrer un navigateur web et on va demander le WSDL (un document XML qui permet la description de l'interface du website)



Services Web

Adresse	Informations
Nom de service : {http://ws/}BanqueWS Nom de port : {http://ws/}BanqueServicePort	Adresse : http://localhost:9090/ WSDL: http://localhost:9090/?wsdl Classe d'implémentation : ws.BanqueService

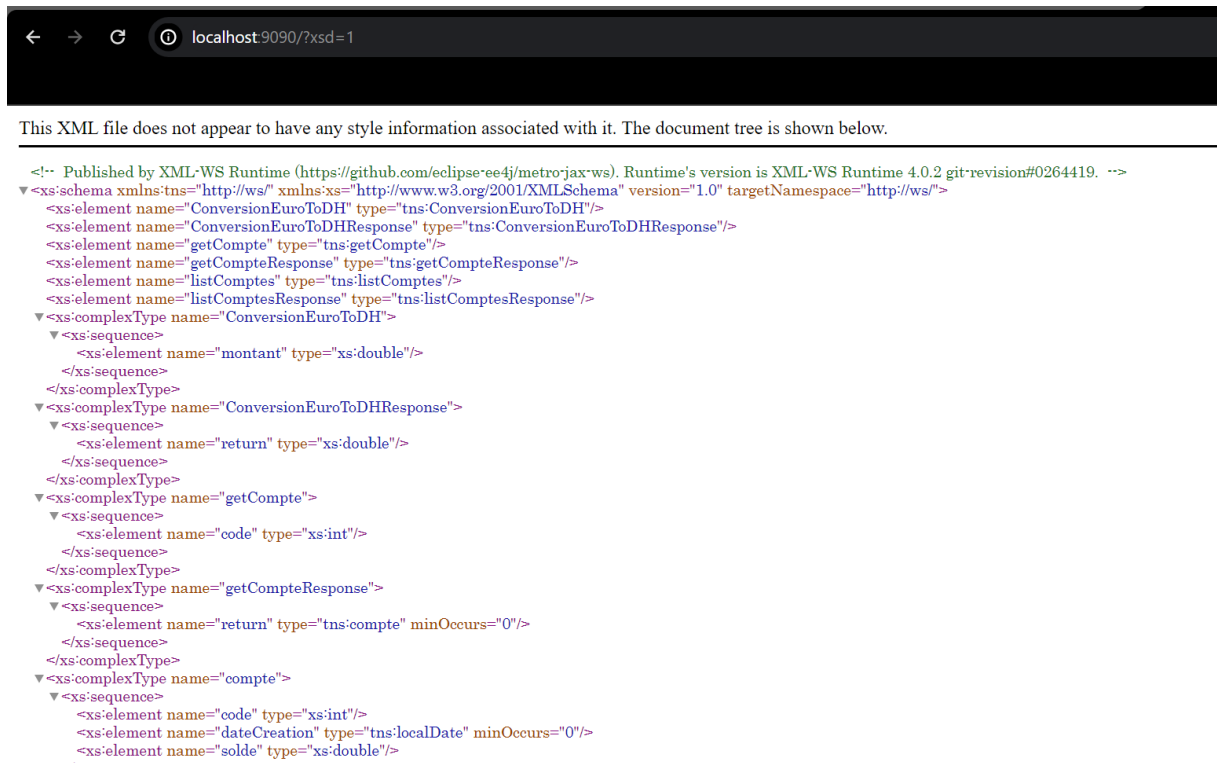


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Published by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is XML-WS Runtime 4.0.2 git-revision#0264419. -->
<!-- Generated by XML-WS Runtime (https://github.com/eclipse-ee4j/metro-jax-ws). Runtime's version is XML-WS Runtime 4.0.2 git-revision#0264419. -->
<?xml version="1.0" encoding="UTF-8" ?>
<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: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:9090/?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="ConversionEuroToDH">
    <part name="parameters" element="tns:ConversionEuroToDH"/>
  </message>
  <message name="ConversionEuroToDHRResponse">
    <part name="parameters" element="tns:ConversionEuroToDHRResponse"/>
  </message>
  <message name="getCompte">
    <part name="parameters" element="tns:getCompte"/>
  </message>
  <message name="getCompteResponse">
    <part name="parameters" element="tns:getCompteResponse"/>
  </message>
  <message name="listComptes">
    <part name="parameters" element="tns:listComptes"/>
  </message>
  <message name="listComptesResponse">
    <part name="parameters" element="tns:listComptesResponse"/>
  </message>
  <portType name="BanqueService">
    <operation name="ConversionEuroToDH">
      <input wsam:Action="http://ws/BanqueService/ConversionEuroToDHRequest" message="tns:ConversionEuroToDH"/>
      <output wsam:Action="http://ws/BanqueService/ConversionEuroToDHRResponse" message="tns:ConversionEuroToDHRResponse"/>
    </operation>
  </portType>
</definitions>
```

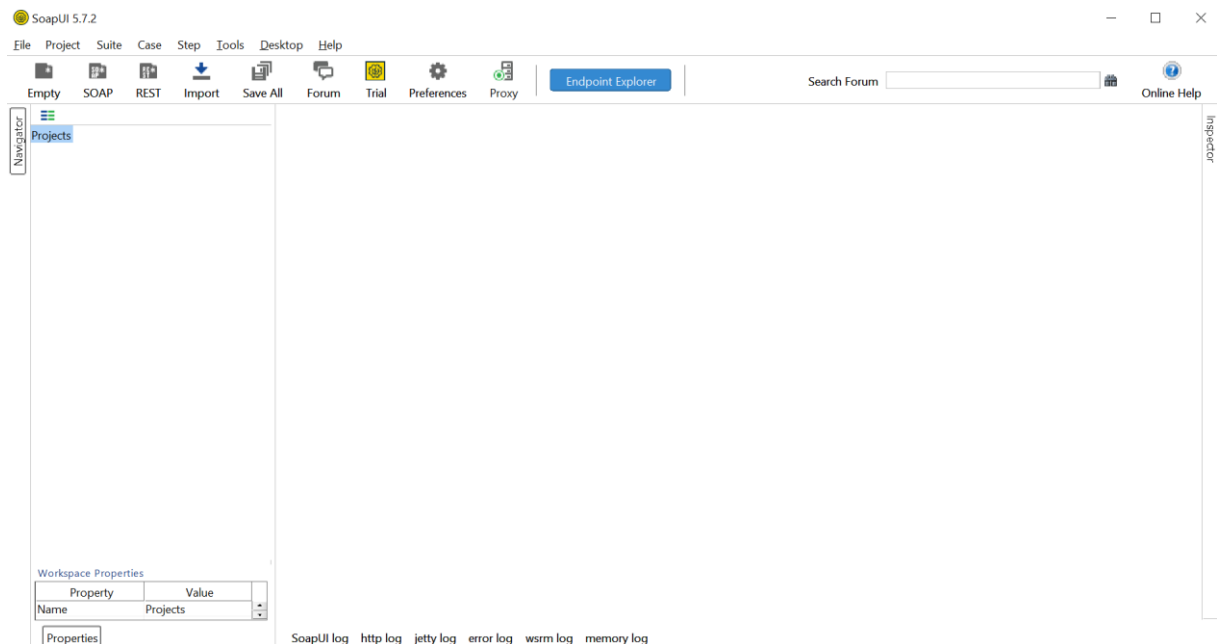
NB : Dans les anciennes versions, il faut mettre le nom du web service BanqueWS

On affiche le schéma xml :





On arrive à la quatrième étape c'est comment tester le web service et pour ce faire, on utilise un outil qui s'appelle SoapUI (un outil open source qui est utilisé pour tester les web services basés sur soap)

On lance SoapUI



On va tester un projet et pour cette étape on a besoin du wsdl (car le web service est basé sur soap)

 **New SOAP Project** X

New SOAP Project 

Creates a WSDL/SOAP based Project in this workspace


Project Name:

Initial WSDL:


Create Requests: ☒ Create sample requests for all operations?


Create TestSuite: ☐ Creates a TestSuite for the imported WSDL

Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)




On trouve qu'il y a 3 méthodes, et on teste la méthode ConversionEuroToDH et on choisit différents montants

Request 1  

Raw 

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws:ConversionEuroToDH>
      <montant>78</montant>
    </ws:ConversionEuroToDH>
  </soapenv:Body>
</soapenv:Envelope>
```

Raw 

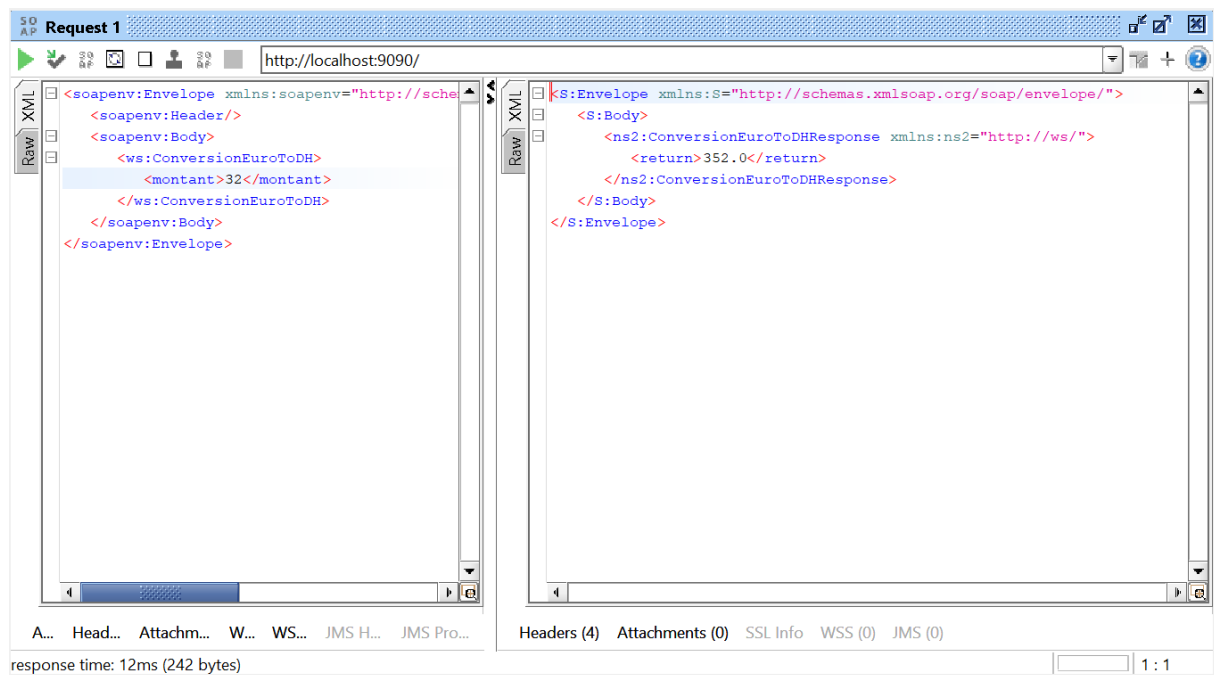
```
<?xml version='1.0' encoding='UTF-8'>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:ConversionEuroToDHResponse xmlns:ns2="http://ws/">
      <return>858.0</return>
    </ns2:ConversionEuroToDHResponse>
  </S:Body>
</S:Envelope>
```

A... Head... Attachm... W... WS... JMS H... JMS Pro...

response time: 46ms (242 bytes)

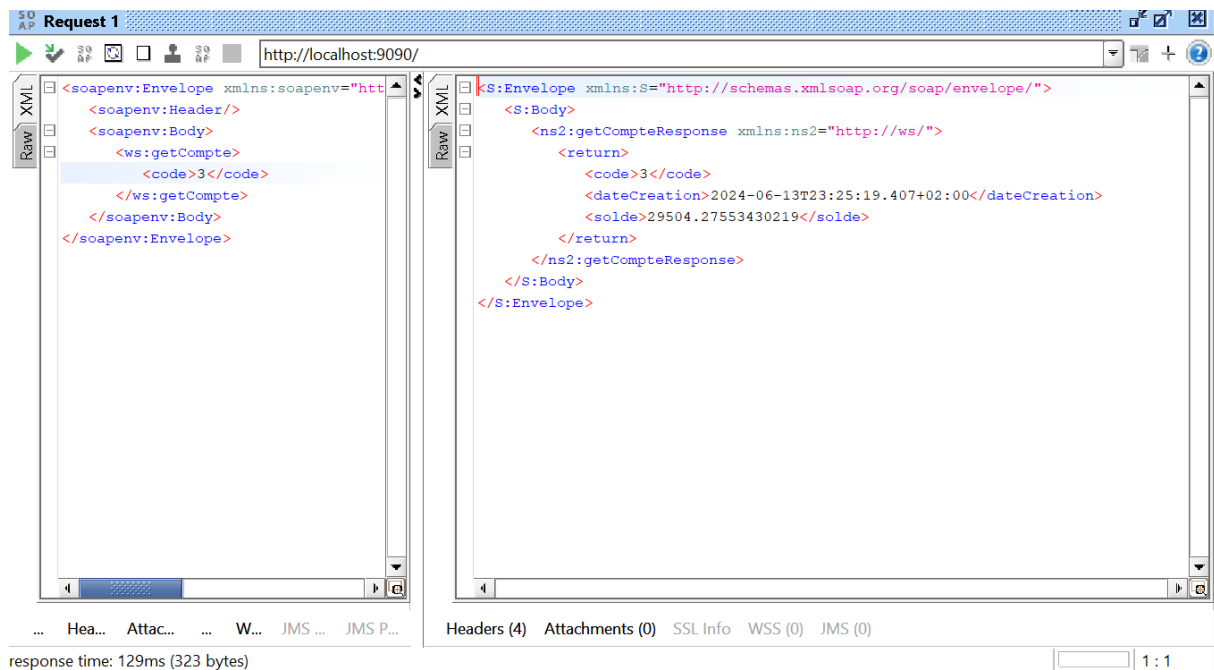
Headers (4) Attachments (0) SSL Info WSS (0) JMS (0)

1 : 1

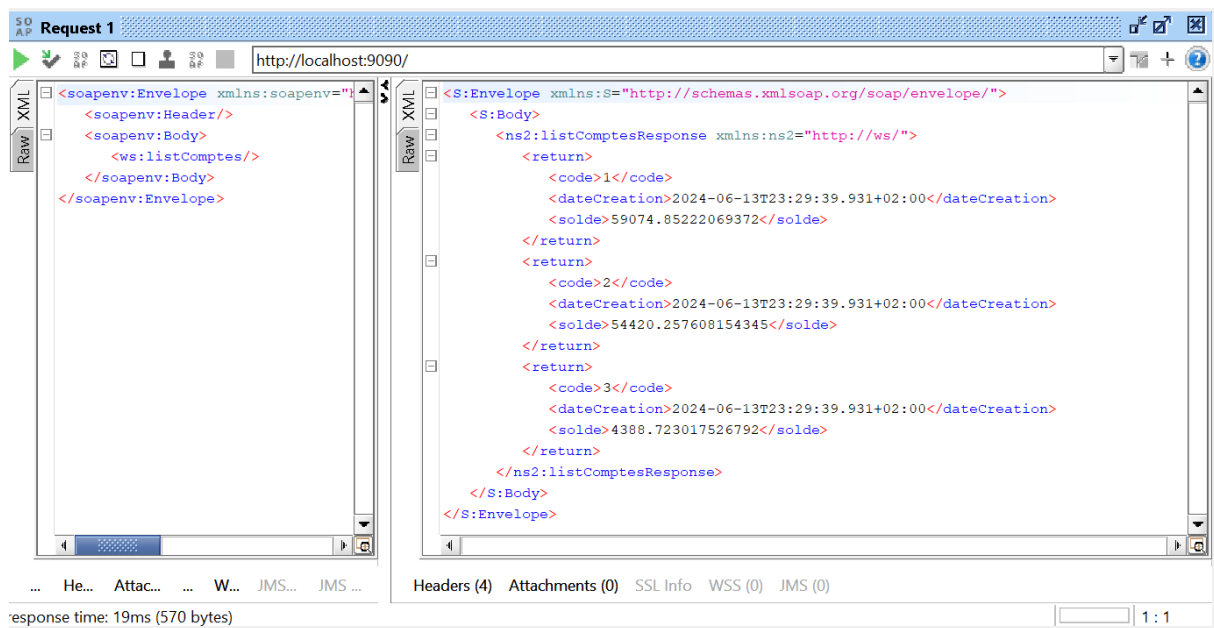


Pour la méthode `getCompte()`, il retourne un compte défini par le code, par la date de création et par le solde

On a changé l'attribut de `dateCreation` par `Date` au lieu de `LocalDate` et on crée a nouveau le projet test2 et on affiche le résultat



Pour la méthode `listComptes()`, on affiche le résultat



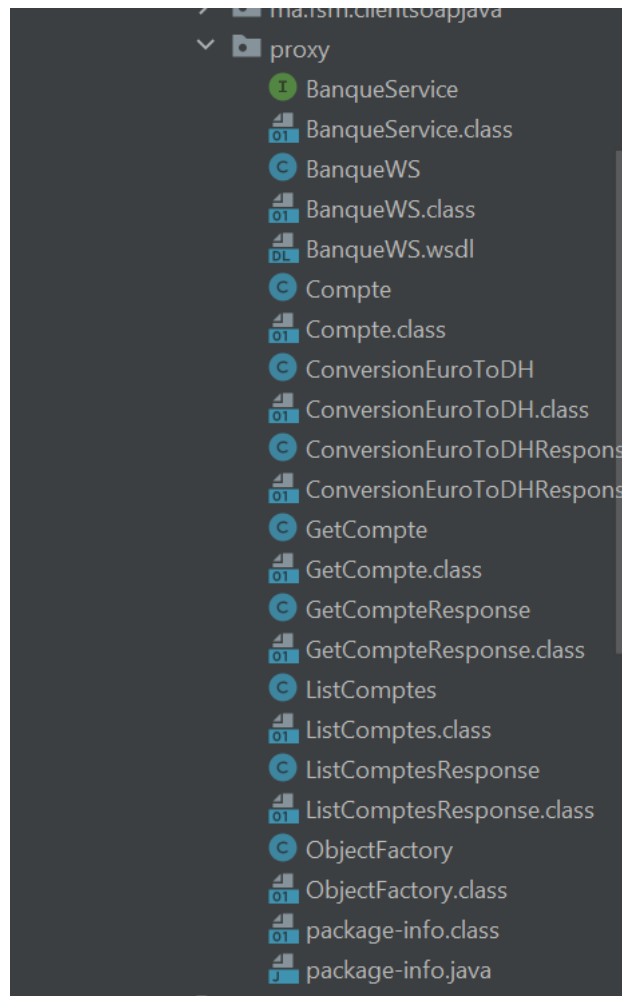
Maintenant, on va créer un client java qui permet de consommer le web service
On crée un projet fils s'appelle client-soap-java et on ajoute les dépendances de jax-ws

```

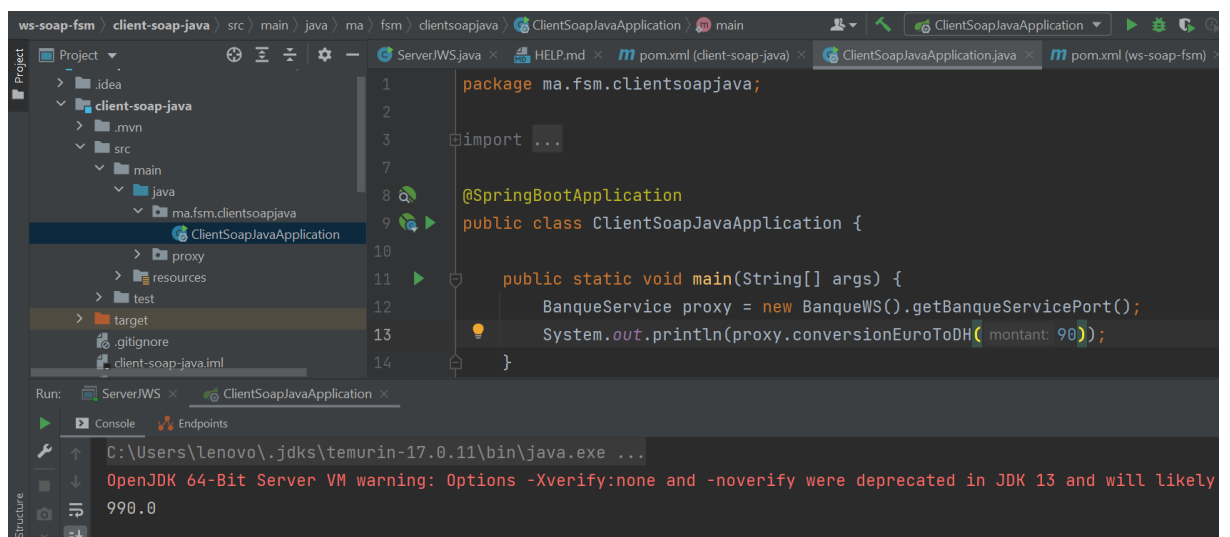
</properties>
<dependencies>
  <dependency>
    <groupId>com.sun.xml.ws</groupId>
    <artifactId>jaxws-ri</artifactId>
    <version>4.0.2</version>
    <type>pom</type>
  </dependency>
  <dependency>

```

On a besoin de wsdl pour générer se qu'on appelle un proxy (un ensemble de classes qu'on va générer à partir du wsdl et qui vont permettre à mon application java de communiquer avec le web service), on va générer des codes java à partir de wsdl



On va créer dans l'application client et on fait appelle web service par l'interface qui s'appelle BanqueService dans proxy et après on va faire appel au méthode conversionEuroToDH()



On va consulter une compte à partir de proxy et on va afficher les informations sur ce compte :

```

public class ClientSoapJavaApplication {

    public static void main(String[] args) {
        BanqueService proxy = new BanqueWS().getBanqueServicePort();
        System.out.println(proxy.conversionEuroToDH( montant: 90));
        Compte compte = proxy.getCompte( code: 4);
        System.out.println("-----");
        System.out.println(compte.getCode());
        System.out.println(compte.getSolde());
        System.out.println(compte.getDateCreation());

        proxy.listComptes().forEach(cp ->{
            System.out.println("-----");
            System.out.println(cp.getCode());
            System.out.println(cp.getSolde());
            System.out.println(cp.getDateCreation());
        });
    }
}

```

IntelliJ IDEA 2024.1.3 available

On va afficher comme résultat

```

990.0
-----
4
2086.953708569588
2024-06-14T00:23:48.121+02:00
-----
1
12603.075950432069
2024-06-14T00:23:48.149+02:00
-----
2
26797.33882831996
2024-06-14T00:23:48.149+02:00
-----
3
16710.52115582336
2024-06-14T00:23:48.149+02:00
-----

Process finished with exit code 0

```