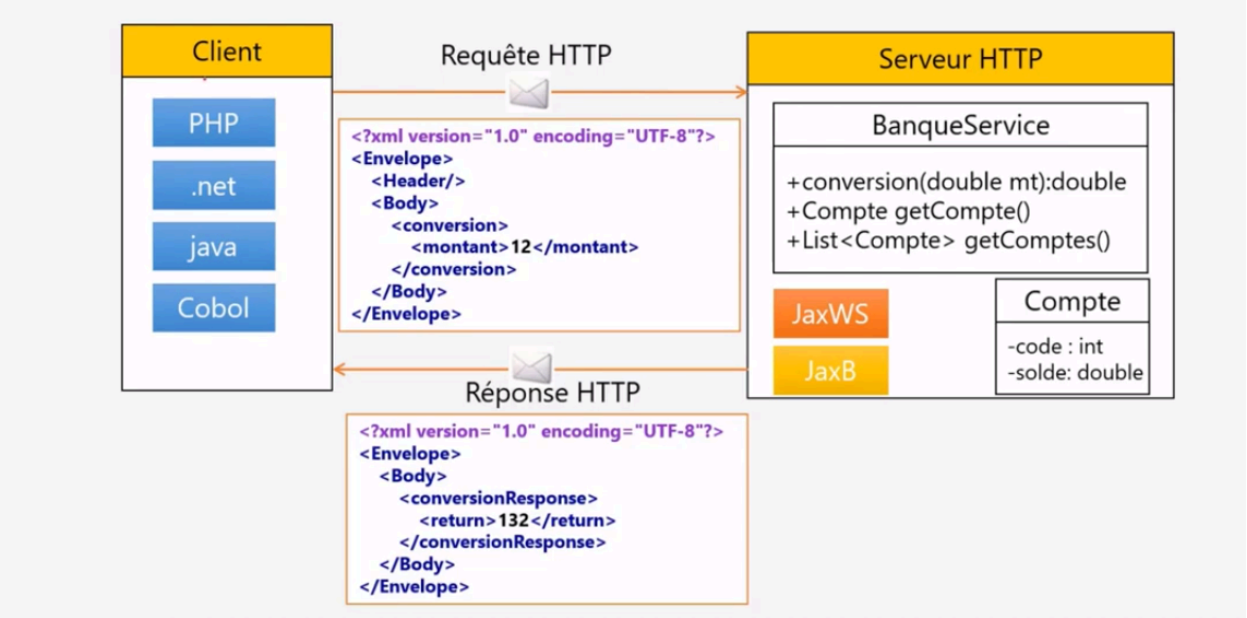
TP 1 : Création de service web SOAP

Objectifs : Dans ce TP,vous initiez à la création d'un service web SOAP.

1. Création d'unserviceweb SOAPenjava, ledéployerensuiteletester.
2. Créer un service web client qui pourra interagir avec le premier WS que vous avez créée.

Environnement de travail : IDE eclipse, Netbeans ou intellige, JDK > 8

Introduction



Qu 'est un web Service ?

1. **Un Service Web** est une application qui permet d'échanger des données avec d'autres applications web. Même si ces dernières sont construites dans des langages de programmation différents.
2. Parmi les Services Web les plus connus on peut citer SOAP, REST ou HTTP.

Quel est le format d'un message ?

1. JSON
2. XML
3. SOAP

SOAP (Simple Object Access Protocol)

1. Il est un protocole d'échange inter-applications.
2. Il échange des messages sur des réseaux entre un expéditeur (SOAP sender) et un destinataire (SOAP receiver).
3. Il est indépendant de toute plateforme.
4. Il est basé sur XML.
5. Il utilise généralement HTTP.

   SOAP = XML + HTTP

Partie 1 : Implémentation de l'architecture du serveur

Création d'un service web SOAP en java

1. **Création du projet :** Il faut créer un projet Maven et ajouter la bibliothèque **jaxws-ri** dans le fichier **pom.xml**.
2. **Structure du projet :** Créez quatre dossiers : **models**, **services**, **server** et **client**.
3. **Création des classes :** Dans le dossier models, créez deux classes : **Student** et **University**. Pour la classe **Student**, ses attributs sont :

```
    private Integer id;
    private String name;
    private String email;
    private Univerty univerty;
    private Date created;
```

Pour la classe **University**, les attributs sont :

```
    private int id;
    private String name;
```

NB: N'oubliez pas de créer les constructeurs ainsi que les getters et setters pour les deux classes.

4. **Création de l'interface :** Ensuite, il faut créer l'interface **StudentService** dans le dossier **services**, qui contient le code suivant :

```
@WebService
public interface StudentService {
    @WebMethod
    Student getStudentById(@WebParam(name = "id") Integer id);
    @WebMethod
     String getNameStudentById(@WebParam(name = "id") Integer
    id);
    @WebMethod
       String getStudentUniversityById(@WebParam(name = "id")
    Integer id);
    @WebMethod
    List<Student> getAllStudents();
    @WebMethod
       String addStudent(@WebParam(name = "student") Student
    student);
}
```

5. **Création de l'implémentation :** Après cela, créez une classe d'implémentation nommée **StudentServiceImpl**, qui contient une variable statique pour stocker tous les étudiants :

```
public static List<Student> students;
```

Ajoutez un constructeur pour initialiser la liste des étudiants :

```
public StudentServiceImpl() {
    students = new ArrayList<>();

    //Initialization Universities
    Univerty univerty1 = new Univerty(1, "ISI");
    Univerty univerty2 = new Univerty(2, "INSAT");

    //Initialization Students
        students.add(new Student(1, "Mohamed Amine Guesmi",
"amine@gmail.com", univerty1));
         students.add(new Student(1, "Foulen Ben Foulen",
"foulen@gmail.com", univerty1));
            students.add(new Student(1, "Tounsi Almami",
"almami@gmail.com", univerty2));
}
```

Ensuite, implémentez les fonctions de l'interface.

7. **Ajout de l'annotation :** Dans la classe d'implémentation, ajoutez l'annotation suivante pour déclarer le service :

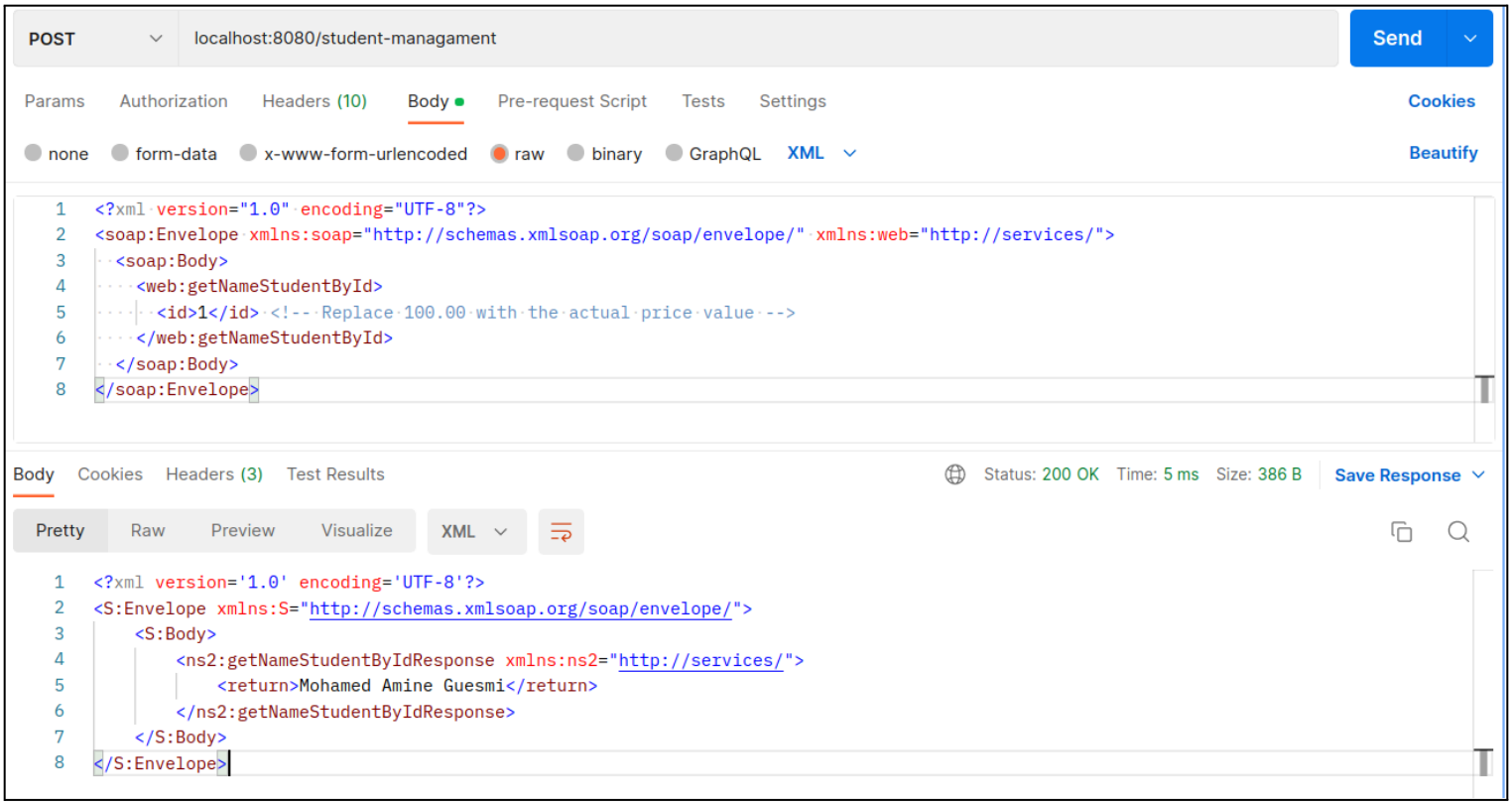
```
@WebService(serviceName = "studentsManagement",
endpointInterface = "services.StudentService")
```

9. **Publication des fonctions :** La dernière étape consiste à publier vos fonctions en utilisant le protocole SOAP. Créez une classe **Server** dans le dossier **server** et ajoutez le code suivant :

```
public class server {
    public static void main(String[] args) {
                                        String url =
    "http://localhost:8080/student-managament";
        Endpoint.publish(url, new StudentServiceImpl());
        System.out.println(url + " deployed ! ");
    }
}
```

10. **Consultation du WSDL :** Après avoir démarré le serveur, visitez l'URL suivante pour consulter le WSDL de votre serveur : http://localhost:8080/student-management

Partie 2 : Tester nos API SOAP avec Postman



L'API est accessible à l'URL suivante : http://localhost:8080/student-management . La méthode à utiliser est **POST**, et le type de corps doit être changé en XML. Voici le corps de la requête :

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://services/">
 <soap:Body>
   <web:getNameStudentById>
       <id>1</id> <!-- Replace 100.00 with the actual price
value -->
   </web:getNameStudentById>
 </soap:Body>
</soap:Envelope>
```

Partie 3 : Tester nos API SOAP avec un client JAVA

Ajouter la classe **Client** dans le dossier client et y insérer le code suivant :

```
URL wsdlURL = new URL("http://localhost:8080/student-managament?wsdl");
QName qname = new QName("http://services/", "studentsManagement");

Service service = Service.create(wsdlURL, qname);
StudentService proxy = service.getPort(StudentService.class);

System.out.println("\nGet Student name with id");
String studentName = proxy.getNameStudentById(1);
System.out.println("Student Name: " + studentName);
```