Université de Carthage **Ecole Nationale d'Ingénieurs de Carthage**	**Atelier Web service** ***SOAP, JAX-WS*** **2025-2026**	**3ING Génie Informatique**

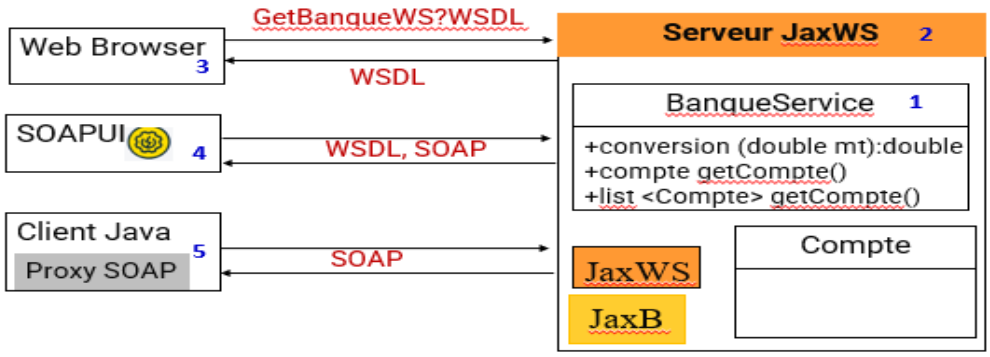
Objectifs

Les objectifs de cet atelier sont :

- Créer et documenter un service web SOAP intitulé « Service Banque » avec un fichier WSDL.
- Tester le service avec SOAP UI et développer un client Java pour le consommer.

Démarche de travail

1. Investiguer l'approche Bottom/UP et TOP Down
2. Créer et annoter un POJO
3. Déployer le Web Service avec un simple Serveur JaxWS
4. Générer la description du service
5. Tester les opérations du web service avec un outil comme SOAPUI



Partie A : L'implémentation du service

1. Créez un simple projet web maven que vous le nommez **SOAP2026** :

- Sous ce projet vous implémentez un Web service qui permet de :
  - Convertir un montant du dinar en euro

- Récupérer un compte client (un compte client est défini par un ID, solde et date création)
- Consulter la liste de comptes disponibles.

2. Ajouter les dépendances de l'API JAX-WS au fichier pom du projet

```
<!-- https://mvnrepository.com/artifact/com.sun.xml.ws/jaxws-ri -->
<dependency>
  <groupId>com.sun.xml.ws</groupId>
  <artifactId>jaxws-ri</artifactId>
  <version>4.0.3</version>
  <type>pom</type>
</dependency>
```

Génération du web service

3. Ajouter les annotations JAX-WS sur la classe POJO que vous avez créé pour générer le web service.
4. Déployer votre web service sur un serveur JWS
5. Tester le service via l'outil SOAP-UI

Utilisation de la bibithéque JAXB

6. Désérialiser quelques attributs du compte en utilisant les annotations JAXB suivantes

- @XmlTransient : indique que le champ en question n'est pas considéré dans la sérialisation.
- @XmlRootElement(name="") : Elle indique à JAXB que la classe doit être traitée comme un élément XML
- XmlAccessorType(XmlAccessType.FIELD) : Cette annotation spécifie la stratégie d'accès, c'est-à-dire elle indique que JAXB doit utiliser les champs directement (plutôt que les méthodes getter et setter) pour accéder aux données.

7. Re-testez votre web service et constatez la désérialisation réalisé suite à l'utilisation de @XmlTransient
8. Générer le fichier WSDL (la documentation de votre Web service)

Partie B : Création d'une application client SOAP pour consommer le WS

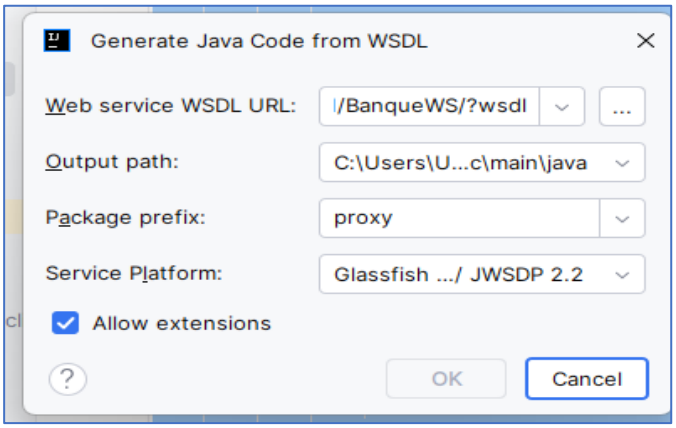
1. Créez un simple projet web maven que vous le nommez **ClientSOAP2026** en utilisant le middleware (le proxy wsimport : qui permet de générer du code Java à partir d'un fichier **WSDL**)

2. Ajouter le plugin wsimport à votre fichier pom.xml pour supporter cette fonctionnalité

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>jaxws-maven-plugin</artifactId>
    <version>2.6</version>
    <executions>
        <execution>
            <id>generate-ws-client</id>
            <goals>
                <goal>wsimport</goal>
            </goals>
            <configuration>
                <wsdlFiles>
                    <wsdlFile>src/main/re-
sources/myservice.wsdl</wsdlFile>
                </wsdlFiles>
                <packageName>com.example.proxy</packageName>
                <sourceDestDir>${project.build.directory}/generated-
sources/wsimport</sourceDestDir>
                <verbose>true</verbose>
            </configuration>
        </execution>
    </executions>
</plugin>
```

3. Générer le proxy (des classes et des interfaces que vous allez utiliser pour communiquer avec le WS) en utilisant le WSDL

   - Si vous êtes sur Intellij accédez à l'onglet help ➔Find action ➔ Generate Java Code from WSDL,
   - Saisissez l'adresse de votre fichier WSDL et choisir l'implémentation GlassFish/JAX-WS 2.2
   - Saisissez le nom de package « Proxy » là où vous allez générer les classes et les interfaces JAVA



Remarque : Il ya des versions de Intellij qui ne supportent pas la génération graphique depuis WSDL (fonction réservée à Ultimate Edition). Vous pouvez dans ce cas exécutez la fonctionnalité via le terminal de votre projet.

4. Commencez la consommation du Web service par exemple :

a. Créer une classe ClientWS ,

b. Tester, ces fonctionnalités :

```
public static void main(String[] args) {
   //créer le midelware
   BanqueService stub = new BanqueWSSOAP().getBanqueServicePort();
     System.out.println("Le montant converti est :"+ stub.convert(4500));
     Compte c=stub.getCompte(3);
     System.out.println(c.getCode());
   System.out.println(c.getSolde());
}
```

Remarques :

- **Assurez-vous que le plugin Jakarta EE : web services (JAX-WS) est bien installé**