

1- Services Web SOAP avec CxF

a. Questions :

Avant de commencer ce tuto, essayez de répondre à ces questions :

- C'est quoi une architecture orientée service ?
- C'est quoi un service Web ?
- C'est quoi WSDL et SOAP ?

b. Compétences visées :

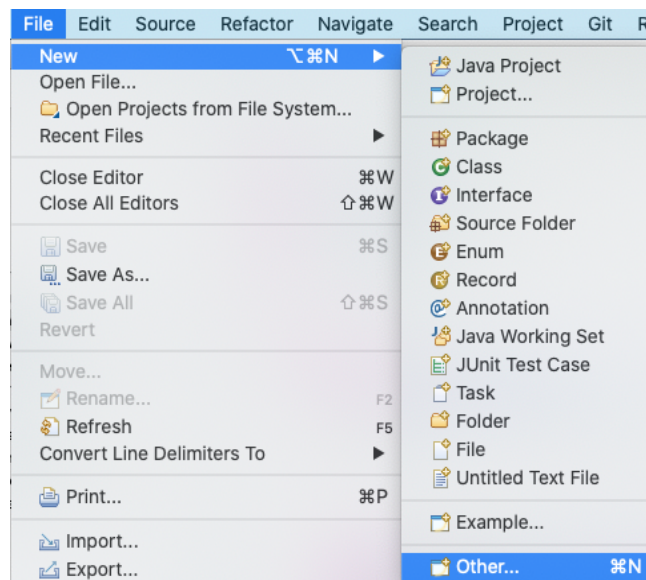
En suivant les étapes suivantes, vous devriez être capable de : (1) créer un service Web SOAP à partir d'une classe Java, (2) de comprendre les détails de son WSDL, et (3) de le tester.

c. Création d'un service Web suivant une approche bottom/up

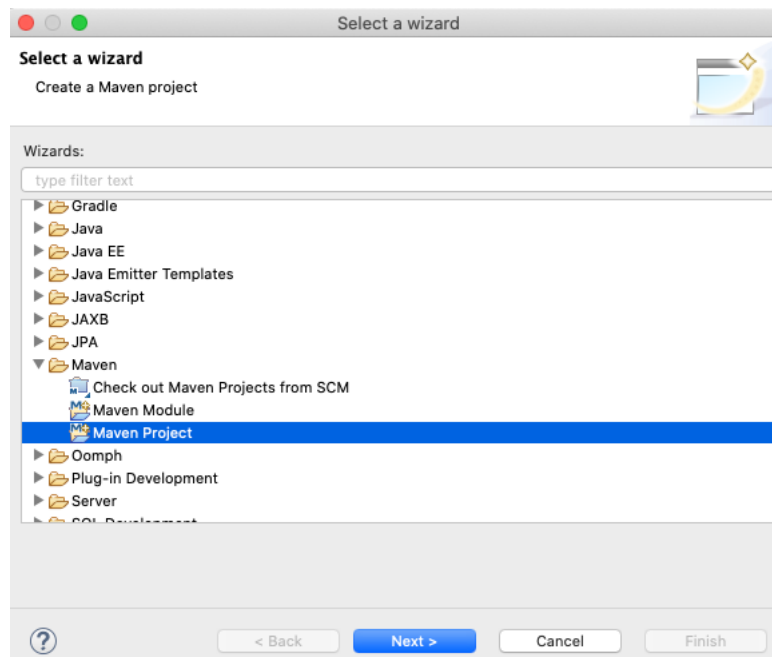
Une première méthode pour la création d'un service Web est celle appelée « bottom/up ». Elle consiste à créer une classe Java puis de l'exposer sous forme d'un service Web.

Le service que vous allez créer contient une opération. Cette opération a comme paramètre d'entrée une chaîne de caractère et retourne sa longueur (int). Pour ce faire, vous allez créer une classe java classique. Suivez les étapes suivantes :

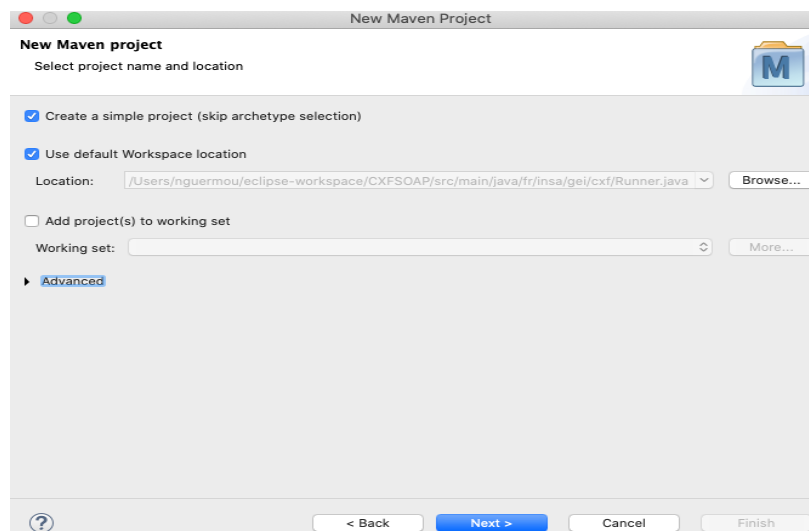
Créer un projet Java Maven : File/New/Other.



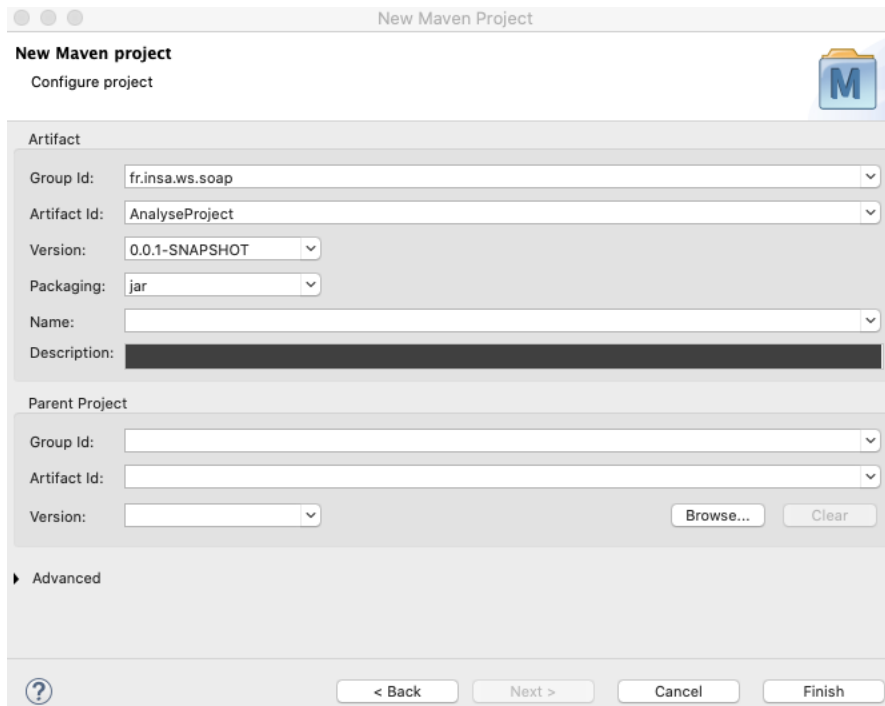
Puis allez dans Maven/Maven Project



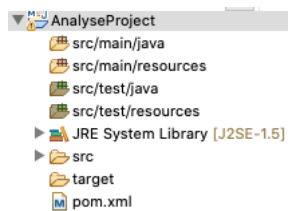
Vous allez créer un projet simple. Pour cela, cochez « Create a simple project (skip archetype selection) ». Vous allez par la suite ajouter les dépendances dont vous aurez besoin dans le fichier pom.xml de votre projet. Cliquez sur Next.



Puis renseignez les informations de votre projet. Vous êtes libres de le nommer comme vous le souhaitez. Ici on l'appelle AnalyseProject.



Cliquez sur finish. Un projet Maven est créé.



Ouvrez le fichier pom.xml de votre projet :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.insa.ws.soap</groupId>
  <artifactId>AnalyseProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

Vous allez y ajouter des dépendances. D'abord, vous allez ajouter le JDK (ici la version 11) et la dépendance *jaxws-rt* qui vous permettra de développer des services Web.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.insa.ws.soap</groupId>
  <artifactId>AnalyseProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/com.sun.xml.ws/jaxws-rt -->
    <dependency>
      <groupId>com.sun.xml.ws</groupId>
      <artifactId>jaxws-rt</artifactId>
      <version>2.3.3</version>
    </dependency>
  </dependencies>
</project>
```

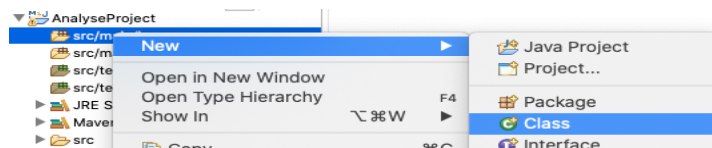
Le code du fichier pom.xml est comme suit :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>fr.insa.ws.soap</groupId>
  <artifactId>AnalyseProject</artifactId>
  <version>0.0.1-SNAPSHOT</version>

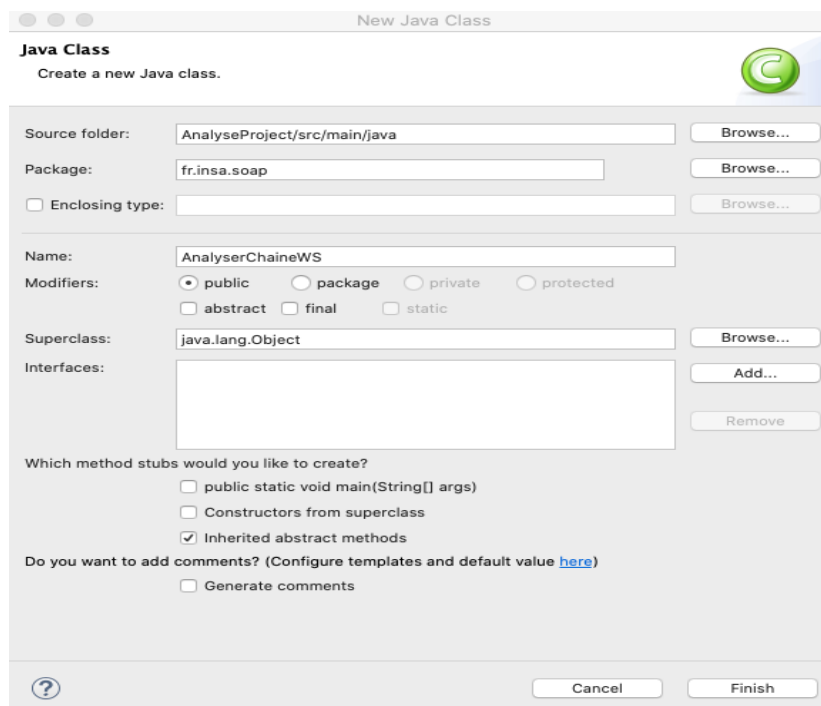
  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/com.sun.xml.ws/jaxws-rt -->
    <dependency>
      <groupId>com.sun.xml.ws</groupId>
      <artifactId>jaxws-rt</artifactId>
      <version>2.3.3</version>
    </dependency>
  </dependencies>
</project>
```

Créer une classe, qui va implémenter votre service Web.



Nommez-la comme vous le souhaitez. Ici on a créé la classe *AnalyserChaineWS* qui est dans le package fr.insa.soap



Cette classe contient la méthode *analyser* qui a comme paramètre d'entrée une chaîne de caractère et retourne sa longueur, i.e., le nombre de caractère qu'elle contient.

```
package fr.insa.soap;

public class AnalyserChaineWS {
    public int analyser(String chaine) {
        return chaine.length();
    }
}
```

Vous allez maintenant exposer votre classe sous forme d'un service Web, nommé *analyser*. Ajoutez l'annotation `@WebService(serviceName="analyser")` du package `javax.jws.WebService`.

Vous allez aussi exposer la méthode de votre classe (i.e., la méthode *analyser*) comme une opération appelée *compare* du services Web *analyser*. Pour cela, vous allez utiliser l'annotation `@WebMethod(operationName="compare")` du package `javax.jws.WebMethod`. Enfin, le paramètre de la méthode *analyser* (i.e., chaîne) sera exposé comme un paramètre chaîne et ce via l'annotation `@WebParam(name="chain")`.

Votre classe doit être comme suit :

```
package fr.insa.soap;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

@WebService(serviceName="analyser")
public class AnalyserChaineWS {
    @WebMethod(operationName="compare")
    public int analyser(@WebParam(name="chain") String chaine) {
        return chaine.length();
    }
}
```

Vous allez maintenant écrire une classe vous permettant de lancer votre service Web. Créez une classe (nommée ici *AnalyserChaineApplication*). Dans cette classe, on déclare le serveur (le host ici est *localhost*) et on veut utiliser le port 8089 (utiliser le port que vous souhaitez).

La méthode *demarrerService* permet tout simplement de publier le service (une instance) sur le serveur spécifié.

Définissez la méthode *main* et y instancier la classe *AnalyserChaineApplication* et lancez la méthode *demarrerService*. Le code de classe *AnalyserChaineApplication* est comme suit :

```
package fr.insa.soap;

import java.net.MalformedURLException;

import javax.xml.ws.Endpoint;

public class AnalyserChaineApplication {

    public static String host="localhost";
    public static short port =8089;

    public void demarrerService() {
        String url="http://"+host+": "+port+"/";
        Endpoint.publish(url, new AnalyserChaineWS());
    }

    public static void main(String [] args) throws MalformedURLException {

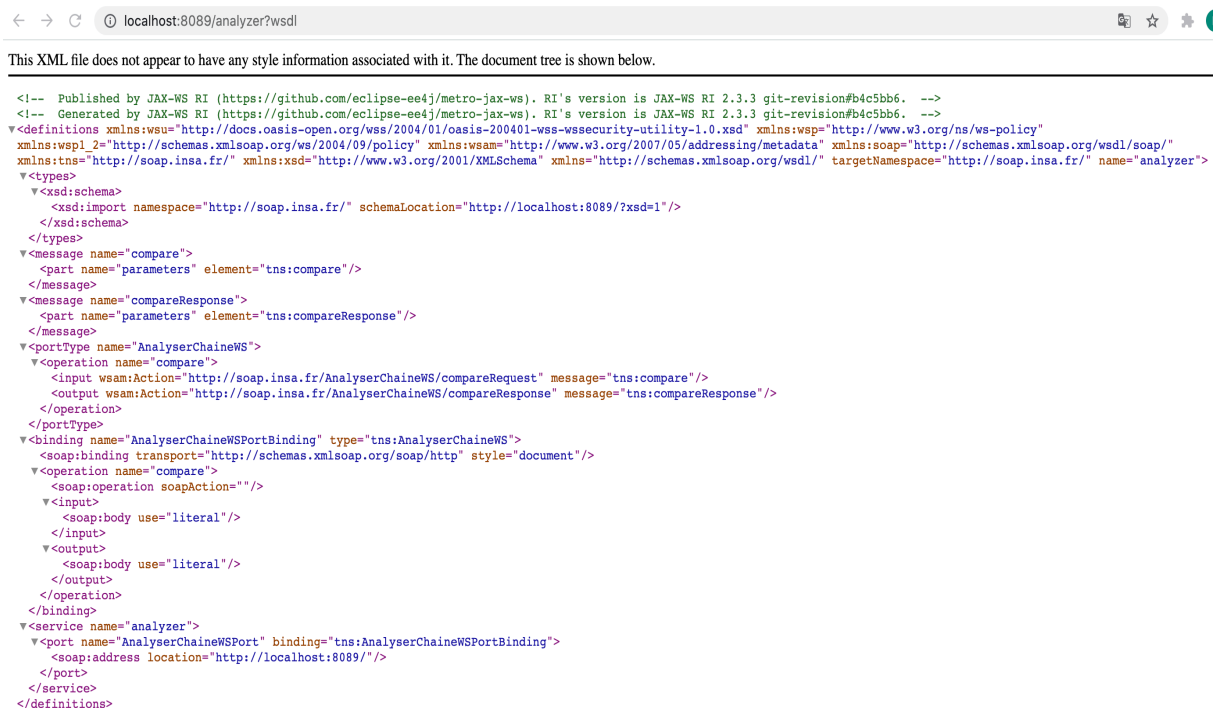
        new AnalyserChaineApplication().demarrerService();
        System.out.println("Service a démarré");
    }
}
```

Exécutez votre classe (La classe contient la méthode main donc Run As/Java Application). Le serveur est lancé et le service Web est déployé sur le port 8089.

La description WSDL de votre service peut être consulté à l'URL :
http://host:port/nom_service?wsdl.

Via votre navigateur, ouvrez l'URL : <http://localhost:8089/analyser?wsdl>

Vous pouvez examiner le WSDL de votre service Web :



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

```
<!-- Published by JAX-WS RI (https://github.com/eclipse-ee4j/metro-jax-ws). RI's version is JAX-WS RI 2.3.3 git-revision#b4c5bb6. -->
<!-- Generated by JAX-WS RI (https://github.com/eclipse-ee4j/metro-jax-ws). RI's version is JAX-WS RI 2.3.3 git-revision#b4c5bb6. -->
<?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:wspl_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://soap.insa.fr/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://soap.insa.fr/" name="analyzer">
  <types>
    <xsd:import namespace="http://soap.insa.fr/" schemaLocation="http://localhost:8089/?xsd=1"/>
  </types>
  <message name="compare">
    <part name="parameters" element="tns:compare"/>
  </message>
  <message name="compareResponse">
    <part name="parameters" element="tns:compareResponse"/>
  </message>
  <portType name="AnalyserChaineWS">
    <operation name="compare">
      <input wsam:Action="http://soap.insa.fr/AnalyserChaineWS/compareRequest" message="tns:compare"/>
      <output wsam:Action="http://soap.insa.fr/AnalyserChaineWS/compareResponse" message="tns:compareResponse"/>
    </operation>
  </portType>
  <binding name="AnalyserChaineWSPortBinding" type="tns:AnalyserChaineWS">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="compare">
      <soap:operation soapAction="">
        <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
  </service name="analyzer">
    <port name="AnalyserChaineWSPort" binding="tns:AnalyserChaineWSPortBinding">
      <soap:address location="http://localhost:8089/">
    </port>
  </service>
</definitions>
```

Les différents types de données, l'opération du service, le binding, ...etc.

Par exemple, l'élément portType décrit l'opération *compare* qu'offre le service. Pour invoquer cette opération, le service doit recevoir le message *compare* et répond en envoyant le message *compareResponse*.

```
<portType name="AnalyserChaineWS">
  <operation name="compare">
    <input wsam:Action="http://soap.insa.fr/AnalyserChaineWS/compareRequest" message="tns:compare"/>
    <output wsam:Action="http://soap.insa.fr/AnalyserChaineWS/compareResponse" message="tns:compareResponse"/>
  </operation>
</portType>
```

Regardez plus haut dans le document WSDL pour voir la définition de ces messages.

```
<message name="compare">
  <part name="parameters" element="tns:compare"/>
</message>
<message name="compareResponse">
  <part name="parameters" element="tns:compareResponse"/>
</message>
```

Le message `compare` permet de transporter une donnée *compare* et le message `compareResponse` une donnée *compareResponse*. Ces deux éléments sont décrits dans la partie *types*.

```
<types>
  <xsd:schema>
    <xsd:import namespace="http://soap.insa.fr/" schemaLocation="http://localhost:8089/?xsd=1"/>
  </xsd:schema>
</types>
<message name="compare">
  <part name="parameters" element="tns:compare"/>
</message>
<message name="compareResponse">
  <part name="parameters" element="tns:compareResponse"/>
</message>
```

Le document xml schéma qui décrit les éléments `compare` et `compareResponse` est importé depuis l'URL indiquée dans `schemaLocation`. Ouvrez cette URL :

```
<!-- Published by JAX-WS RI (https://github.com/eclipse-ee4j/metro-jax-ws). RI's version is JAX-WS RI 2.3.3 git-revision#b4c5bb6. -->
<xs:schema xmlns:tns="http://soap.insa.fr/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" targetNamespace="http://soap.insa.fr/">
  <xs:element name="compare" type="tns:compare"/>
  <xs:element name="compareResponse" type="tns:compareResponse"/>
  <xs:complexType name="compare">
    <xs:sequence>
      <xs:element name="chain" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="compareResponse">
    <xs:sequence>
      <xs:element name="return" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

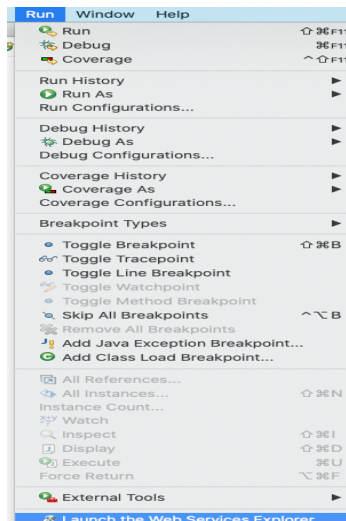
Vous remarquerez les deux éléments `compare` et `compareResponse` de type `compare` et `compareResponse` respectivement.

Ces deux types complexes sans définis dans les éléments `complexType`. Par exemple, le type *compare* contient un élément *chain* de type *String*.

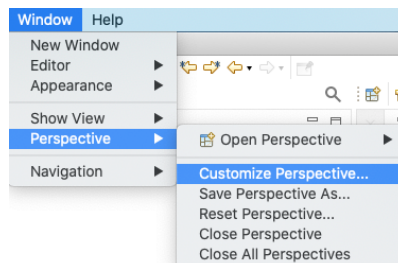
Faites une synthèse de ce que vous avez fait jusqu'à maintenant. Assurez vous de comprendre ce que vous avez fait.

d. Tester le service développé :

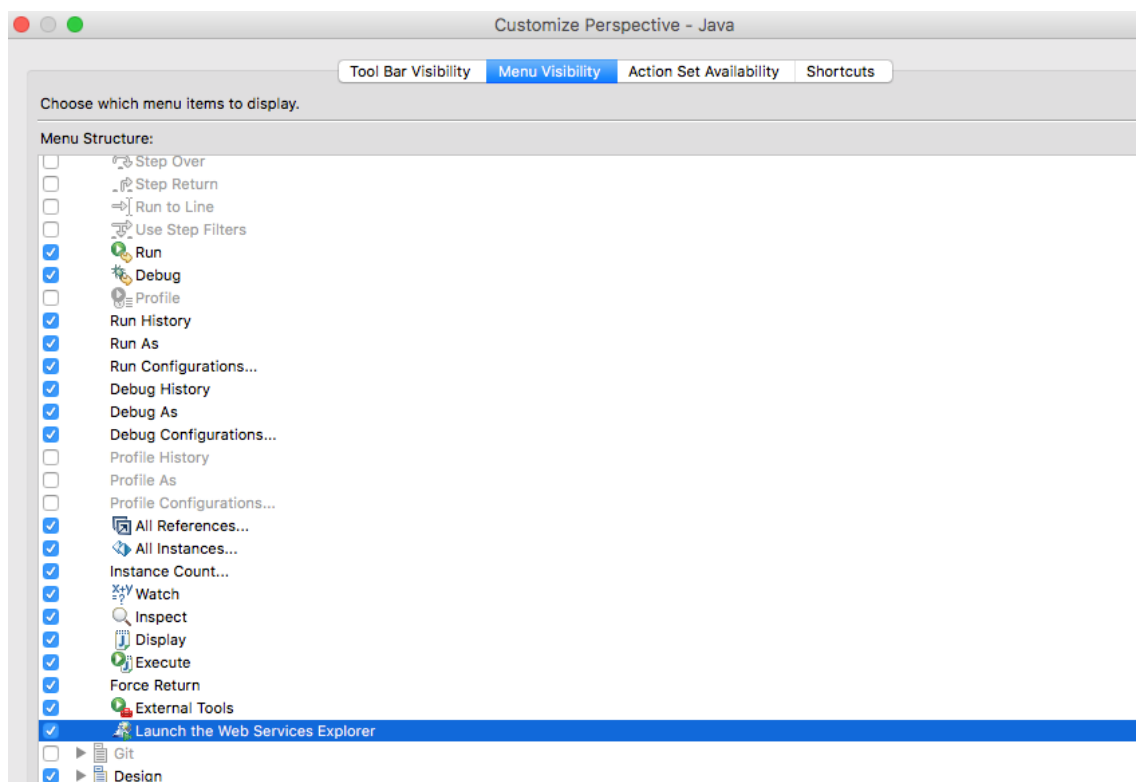
Maintenant, vous allez tester votre service. Pour ce faire, vous allez lancer « Web Services Explorer » comme le montre la figure suivante (si vous ne le trouvez pas, allez à l'étape suivante pour l'ajouter).



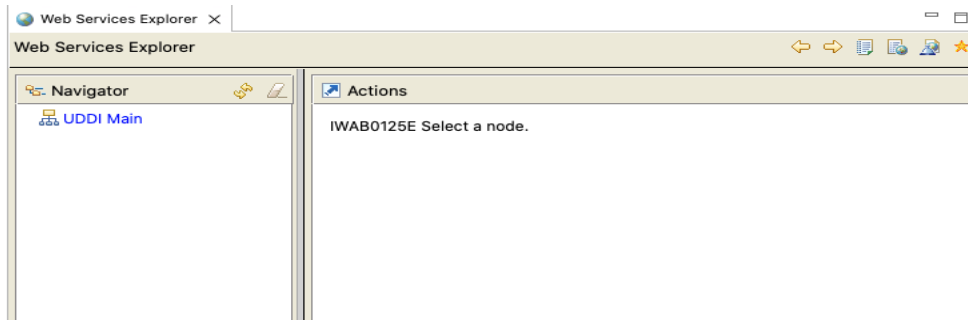
Si vous ne le trouvez pas, allez dans window/Perspective/Customize Perspective.



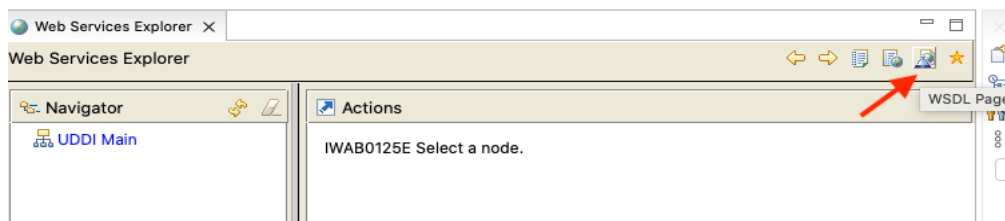
Ensuite, vous allez dans l'onglet "Menu Visibility", puis Run. Cochez « Lanch Web service explorer » afin de l'ajouter au menu « Run » de votre IDE.



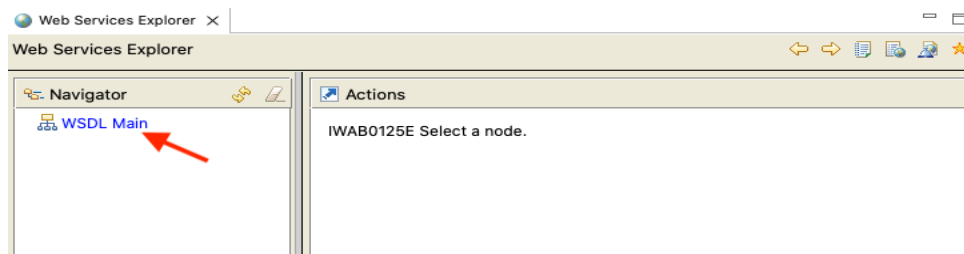
Lancez l'outil de test depuis Run. L'explorateur de service s'ouvre comme suit :



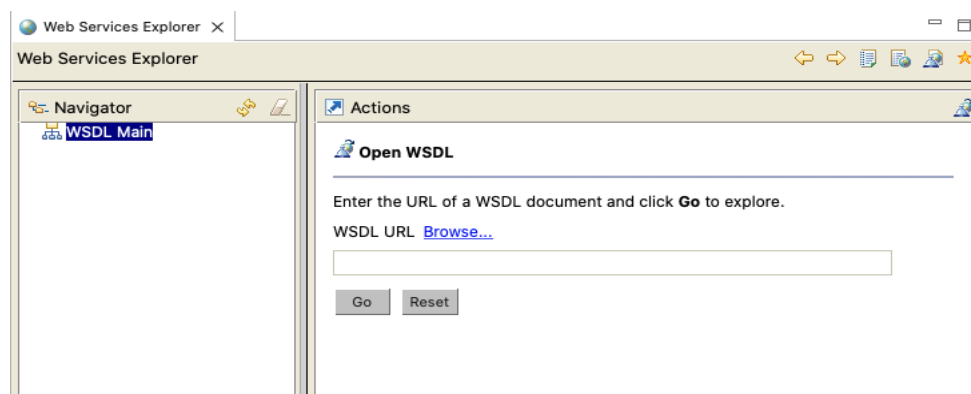
Pour tester le Web service que vous avez développé, vous avez besoin de son WSDL. Rappelez vous que pour invoquer un service Web, l'utilisation de son WSDL est primordiale. Il contient toutes les informations nécessaires. Pour ce faire, cliquer sur l'icône dédiée à l'exploration de fichiers WSDL comme le montre la figure suivante (flèche rouge).



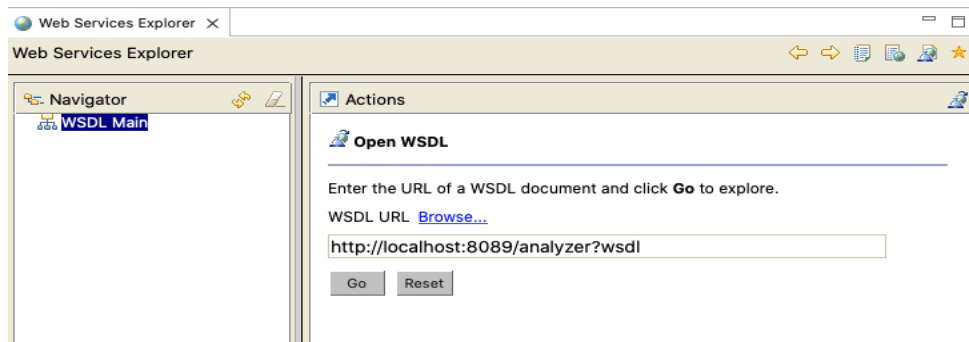
Vous obtenez la fenêtre suivante.



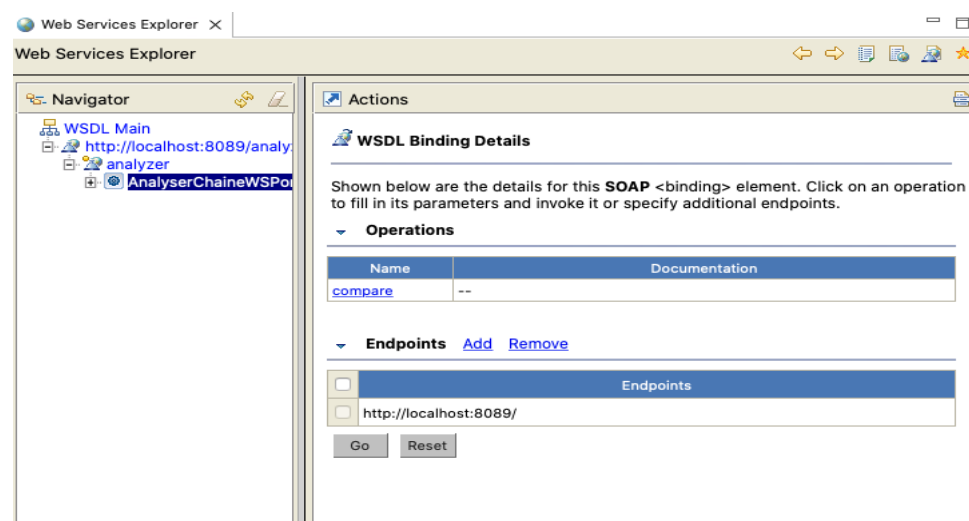
Cliquez sur *WSDL Main* depuis le menu gauche. Saisissez l'URL



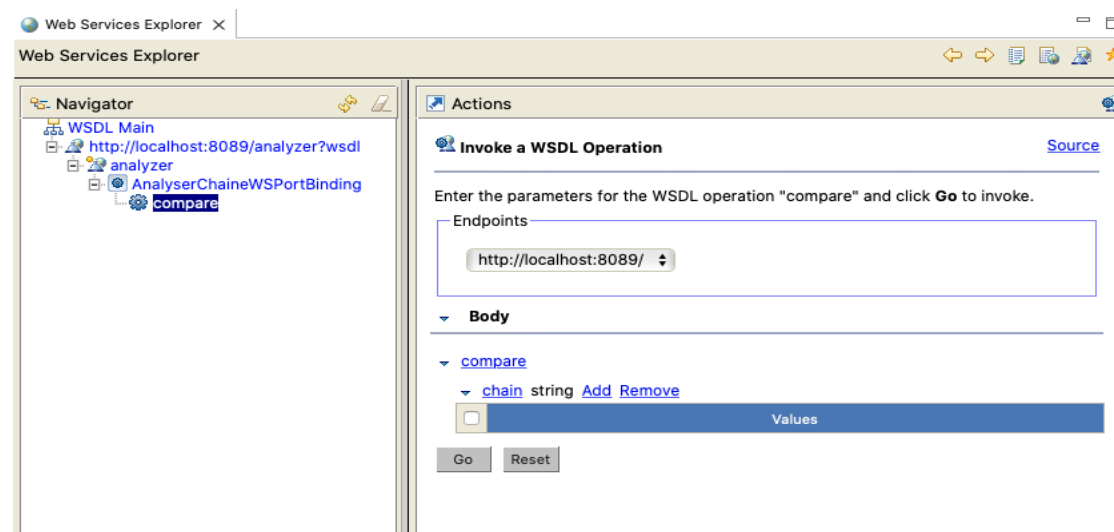
Récupérez l'adresse du WSDL de votre fichier et saisissez la dans le WSDL URL. Ici l'URL est : <http://localhost:8089/analyser?wsdl>



Essayez de décrire ce que vous obtenez.



Comme vous pouvez le voir, le service analyzer a un binding (AnalyserChaineWSPortBinding). Via ce binding, on peut invoquer l'opération compare. Cliquez sur l'opération :



Cette opération attend comme paramètre une chaîne de caractère (*chain*). Cliquez sur *add* pour renseigner le paramètre d'entrée. Entrer une chaîne de caractère puis cliquez sur *go*. Vous pouvez voir le résultat de l'invocation du service dans la section « Statuts »

The screenshot shows the 'WSDL Main' interface. On the left, the 'Navigator' pane lists the hierarchy: WSDL Main, http://localhost:8089/analyzer?wsdl, analyzer, AnalyserChaineWSPortBinding, and compare. The 'Actions' pane on the right is titled 'Invoke a WSDL Operation' and includes a 'Source' link. It instructs the user to enter parameters for the 'compare' operation and click 'Go'. The 'Endpoints' section shows 'http://localhost:8089/'. The 'Body' section is expanded to show the 'compare' operation with a 'chain' parameter of type 'string'. There is an 'Add' button and a 'Remove' button. Below this, a table with the header 'Values' contains one row with the value 'aaaaaa'. At the bottom of the 'Body' section are 'Go' and 'Reset' buttons. Below the 'Actions' pane is the 'Status' pane, which is also expanded to show the 'compareResponse' section with the result 'return (int): 6'. A 'Source' link is also present in the 'Status' pane.

Vous pouvez voir les requêtes SOAP reçues et envoyées. Pour cela, examiner la **source** de la requête

The screenshot shows the 'Status' pane with a 'Form' link. It displays the SOAP Request Envelope and the SOAP Response Envelope. The SOAP Request Envelope is shown in a text area with the following XML content:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://soap.insa.fr/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:compare>
      <chain>aaaaaa</chain>
    </q0:compare>
  </soapenv:Body>
</soapenv:Envelope>
```

The SOAP Response Envelope is also shown in a text area with the following XML content:

```
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:compareResponse xmlns:ns2="http://soap.insa.fr/">
      <return>6</return>
    </ns2:compareResponse>
  </S:Body>
</S:Envelope>
```