

Série TD / TP 0

Les services web avec JAX-WS

I- Présentation de JAX-WS

JAX-WS¹ (Java API for XML Web Services) est une API permettant de développer des services web étendus en utilisant le protocole SOAP et le langage WSDL via la plateforme de développement JAVA.

Le développement de Services Web avec JAX-WS est basé sur des POJO (Plain Old Java Object). Les fonctionnalités de base pour le développement de Web Services avec JAXWS requièrent simplement l'utilisation d'annotations Java.

II- Installation et Démarrage

Pour les besoins de ce TP, les outils suivants seront utilisés (à installer dans l'ordre) :

- *java_ee_sdk-7u2*: Kit de développement de Java EE7. Contient, principalement, le serveur d'application Glassfish4, dans lequel nous allons déployer notre service web (à télécharger depuis oracle.com).
- *eclipse-jee-kepler-SR2-win32*: C'est l'environnement de développement Eclipse, pour Java EE (à télécharger depuis eclipse.org).
- *soapUI-x32-5.2.1.exe* : outil de test complet pour les services web.

1 Installation de Java EE SDK

Pour installer Java EE SDK, le JDK est requis. Il est préférable de suivre les instructions suivantes :

- Configurer le serveur Glassfish comme suit (configuration standard) :
 - ✓ Login : admin
 - ✓ Mot de passe : (vide)
- Accepter les valeurs des ports par défaut : Admin Port (4848) et Http Port (8080)

Il est souhaitable d'installer votre serveur d'application directement sous C: C:\...\glassfish4

Pour vérifier le bon déroulement de l'installation, accéder à la console d'administration de Glassfish en entrant l'url : <http://localhost:4848>

¹ JAX-WS : <https://jax-ws.java.net/>

2 Configuration d'Eclipse avec le serveur d'application Glassfish

Pour ajouter Glassfish comme serveur dans l'environnement Eclipse, veuillez suivre les instructions suivantes :

- Démarrer Eclipse. Dans le menu *Window*, cliquer sur *Show View*, puis sur *Servers*.
- Cliquer sur *Create a new server* pour ajouter un nouveau serveur.
- Cliquer sur *Download additional server adapters*.
- Choisir *Glassfish Tools* et cliquer sur *Next*. Accepter les conditions d'utilisation.
- Cliquer sur *Finish* et redémarrer Eclipse.
- Cliquer de nouveau sur *Create a new server*. Choisir *Glassfish4* puis *Next*.
- Naviguer jusqu'au répertoire d'installation de Glassfish.
- Cliquer sur *Next*. Laisser le domaine par défaut puis cliquer sur *Finish*.

3 Démarrer le serveur Glassfish

Pour démarrer Glassfish, deux manières de le faire :

- En utilisant l'utilitaire *asadmin* qui se trouve dans le dossier */glassfish4/bin* et la commande : *asadmin start-domain* .
- Dans Eclipse, en cliquant droit sur le serveur Glassfish dans l'onglet *Servers* puis sur *Start*.

III- Création d'un Client consommateur

1 Génération de stub (artefacts)

Le but de l'exemple qui suit est de créer une simple application Java qui utilise un service web existant, fait appel à ce service web et consomme le résultat produit par ce dernier.

Plusieurs services web sont disponibles en libre accès sur Internet. GeoIPService est utilisé pour ce test (accessible via : <http://webservicex.com/ws/WSDetails.aspx?CATID=12&WSID=64>). Ce service web permet, entre autres, de définir le pays d'une adresse IP.

1. Ouvrir Eclipse, et créer un nouveau projet de type Java Project. Appeler votre projet *IPLocationFinder*.
2. Ajouter une nouvelle classe Java à ce projet. Appeler cette dernière *IPLocationFinder* et entrer comme nom de package *org.soa.ws.tpl*
3. Taper le code suivant dans votre classe.

```
public class IPLocationFinder {
    public static void main(String [] args) {

        if (args.length != 1)
            System.out.println("Vous devez entre une adresse IP.");
        else {
            String ipAdress = args [0];
            //TODO - Stub Call
        }
    }
}
```

Pour générer le stub du service web à partir de sa description WSDL, l'outil **wsimport** du JDK peut être utilisé. Ces classes générées ne doivent pas être modifiées puisqu'elles sont générées automatiquement. Seule la classe d'implémentation peut l'être. Elle correspond à la classe implémentant le comportement du service web. Le stub représente le **SEI** (Service Endpoint Interface).

4. Créer un nouveau dossier. Appeler ce dernier *SEI*. Créer un sous-dossier *src*.
5. Exécuter la commande suivante :

```
> wsimport -keep -s src <wsdl_URL>
```

6. Importer les classes Java générées (du dossier *src*) vers le projet IPLocationFinder. Les mettre dans un nouveau package *or.soa.ws.tp1*

2 Consommer le service web

1. Pour savoir quelle classe est utilisée comme stub, regarder le WSDL du service web et extraire :

- ✓ wsdl :service name
- ✓ wsdl :port name
- ✓ wsdl : operation name

```
<wsdl:service name="GeoIPService">
  <wsdl:port name="GeoIPServiceSoap" binding="tns:GeoIPServiceSoap">...
</wsdl:port>
  <wsdl:port name="GeoIPServiceSoap12" binding="tns:GeoIPServiceSoap12"
>...</wsdl:port>
  <wsdl:port name="GeoIPServiceHttpGet" binding="tns:GeoIPServiceHttpGet"
>...</wsdl:port>
  <wsdl:port name="GeoIPServiceHttpPost" binding="tns:GeoIPServiceHttpPost"
>...</wsdl:port>
</wsdl:service>
```

```
<wsdl:portType name="GeoIPServiceSoap">
  <wsdl:operation name="GetGeoIP">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
      GeoIPService - GetGeoIP enables you to easily look up
      countries by IP addresses
    </wsdl:documentation>
    <wsdl:input message="tns:GetGeoIPSoapIn"/>
    <wsdl:output message="tns:GetGeoIPSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="GetGeoIPContext">
    <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
      GeoIPService - GetGeoIPContext enables you to easily look
      up countries by Context
    </wsdl:documentation>
    <wsdl:input message="tns:GetGeoIPContextSoapIn"/>
    <wsdl:output message="tns:GetGeoIPContextSoapOut"/>
  </wsdl:operation>
</wsdl:portType>
```

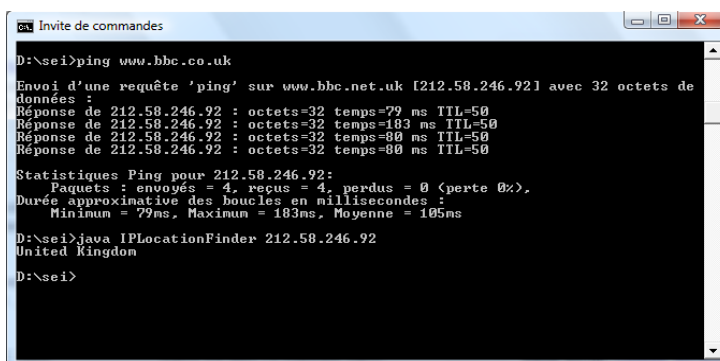
2. Compléter le code de la classe Client comme suit :

- ✓ Créer une instance de la classe Service Name.
- ✓ Utiliser cette instance pour créer une instance de la classe Port Name.
- ✓ Faire appel à cette dernière pour appeler la méthode Operation Name. Donnez l'adresse IP en argument.
- ✓ Afficher le nom du pays retourné par cette opération.

```
<wsdl:types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://www.web
servicex.net/">
    <s:element name="GetGeoIP">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="I
PAddress" type="s:string"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="GetGeoIPResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="G
etGeoIPResult" type="tns:GeoIP"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:complexType name="GeoIP">...</s:complexType>
    <s:element name="GetGeoIPContext">...</s:element>
    <s:element name="GetGeoIPContextResponse">...</s:element>
    <s:element name="GeoIP" nillable="true" type="tns:GeoIP"/>
  </s:schema>
</wsdl:types>
```

3. Exécuter votre Client avec n'importe quelle adresse IP (ex : Faire des ping pour les obtenir), et observer le résultat.

Exemple avec l'adresse IP de la BBC : **212.58.246.92**



```
D:\sei>ping www.bbc.co.uk
Envoi d'une requête 'ping' sur www.bbc.net.uk [212.58.246.92] avec 32 octets de
données :
Réponse de 212.58.246.92 : octets=32 temps=79 ms TTL=50
Réponse de 212.58.246.92 : octets=32 temps=183 ms TTL=50
Réponse de 212.58.246.92 : octets=32 temps=80 ms TTL=50
Réponse de 212.58.246.92 : octets=32 temps=80 ms TTL=50

Statistiques Ping pour 212.58.246.92:
    Paquets : envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
    Durée approximative des boucles en millisecondes :
        Minimum = 79ms, Maximum = 183ms, Moyenne = 105ms

D:\sei>java IPLocationFinder 212.58.246.92
United Kingdom
D:\sei>
```

I. Installation Logiciels

1. Java JDK 7

- Installer
- Set des variables d'environnement : Clic droit sur *Ordinateur* => Propriétés => Paramètres système avancés => Variables d'environnement => Variables système :
 - ✓ Ajouter : JAVA_HOME = C:\Program Files\Java\jdk1.7.0_79
 - ✓ Modifier : Path = ... ; C:\Program Files\Java\jdk1.7.0_79\bin

2. Java EE SDK (glassfish4) : Java_ee_sdk-7u2.zip

- Dézipper.
- Couper-Coller le dossier **java_ee_sdk-7u2** dans **C:**

3. Eclipse Java EE : eclipse-jee-kepler-SR2-win32.zip

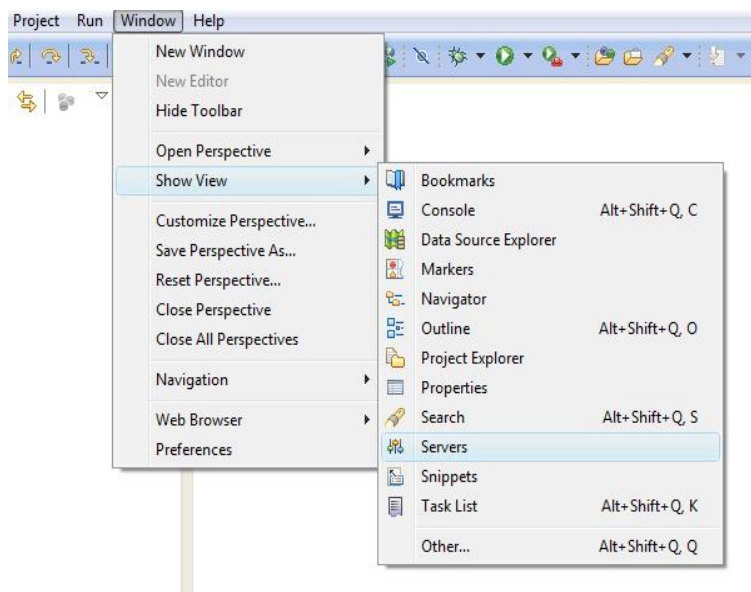
- Dézipper.
- Coller au Bureau

4. soapUI : soapUI-x32-5.2.1.exe

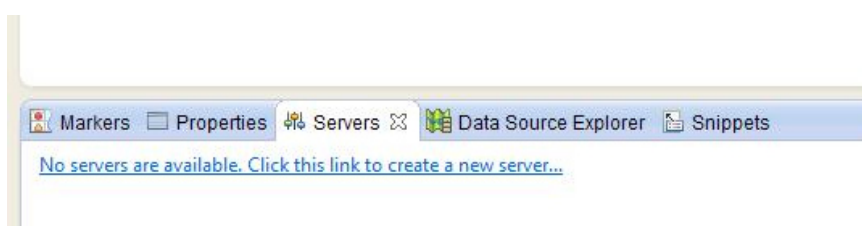
- Installer

II. Configuration de Glassfish avec Eclipse

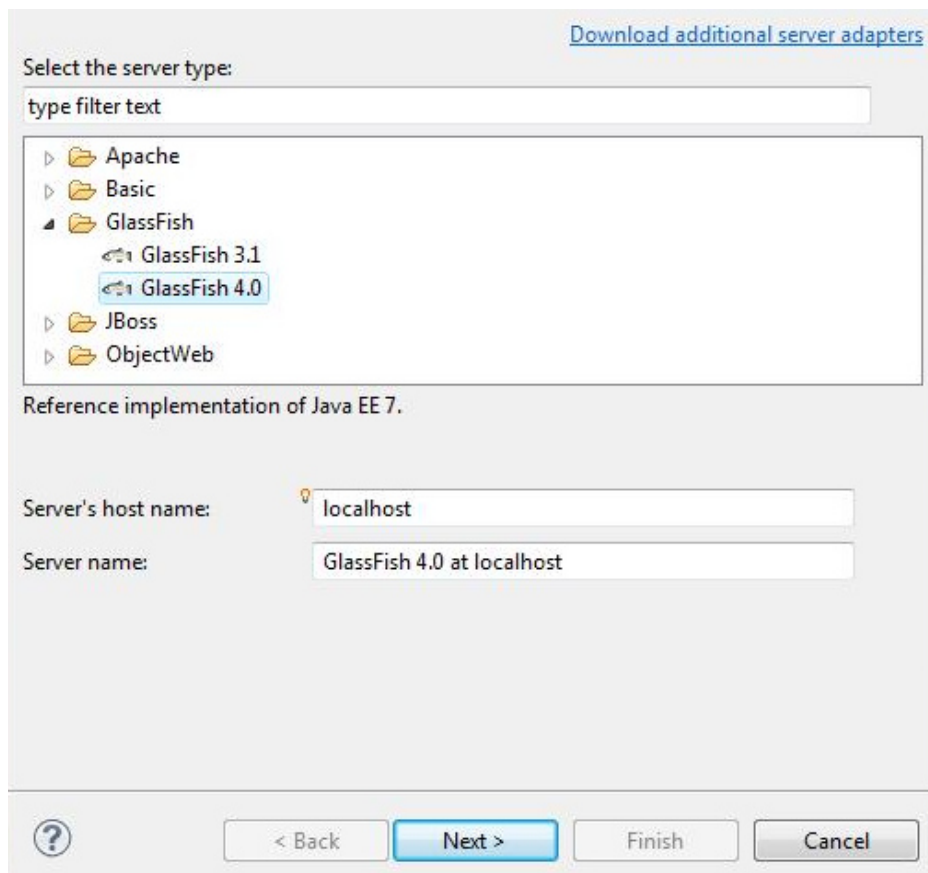
- ✓ Dans Eclipse, clic sur Servers :



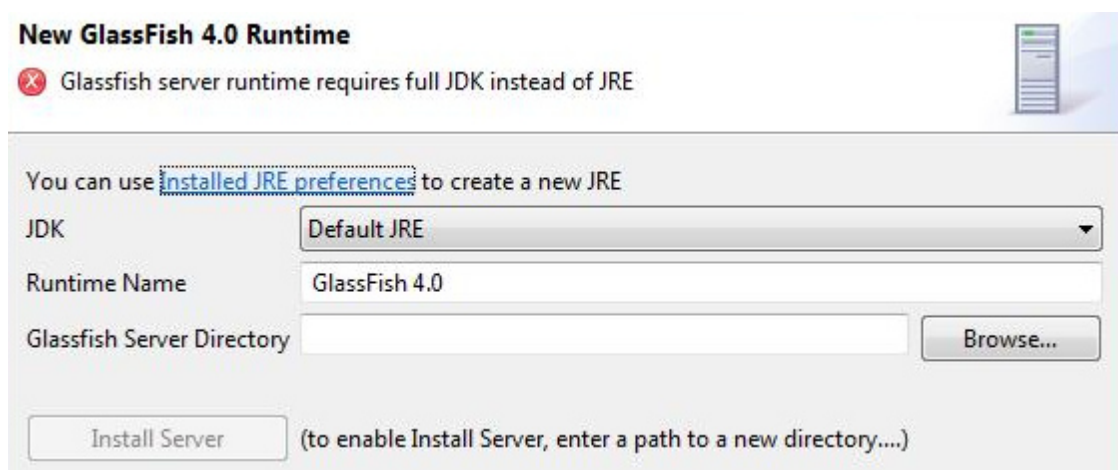
- ✓ Clic sur le lien : Create a new server ...



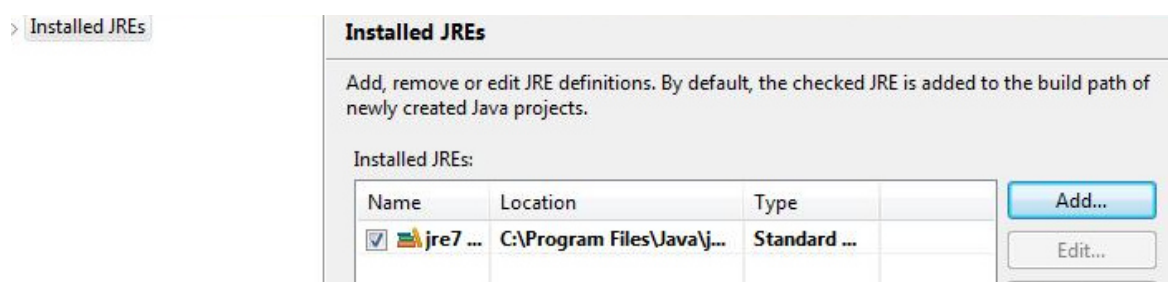
- ✓ Choisir GlassFish 4.0, puis clic sur Next :



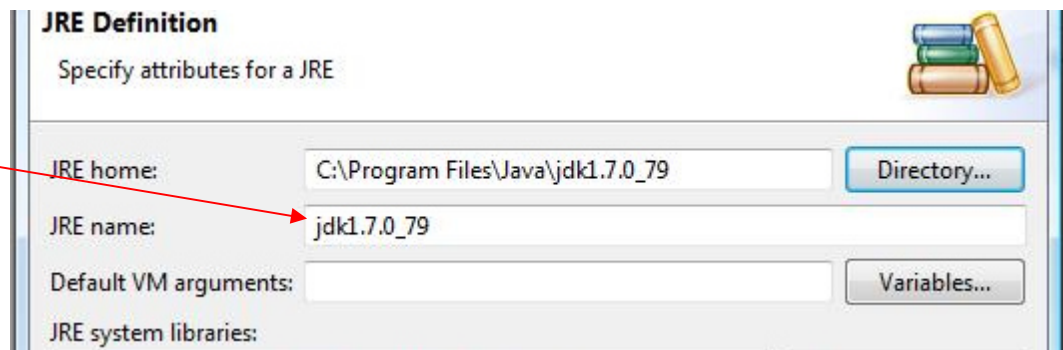
- ✓ Clic sur Installed JRE preference :



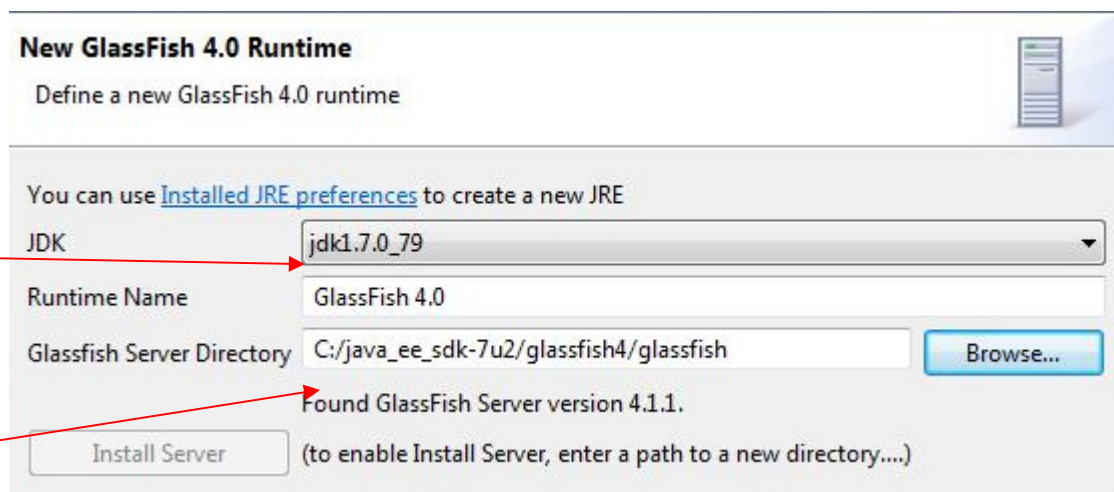
- ✓ Clic sur Add :



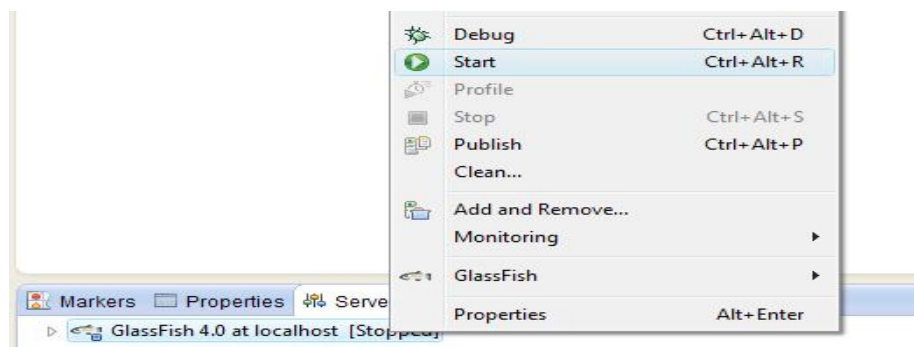
- ✓ Choisir Standard VM puis Next.
- ✓ Ajouter le dossier d'installation du jdk 7 en cliquant sur Directory. Puis Finish.



- ✓ Cocher le jdk nouvellement ajouté. Clic sur Ok.
- ✓ Choisir le nouveau jdk 1.7.
- ✓ Dans Glassfish Server Directory, naviguer (Browse...) et ajouter le dossier d'installation sdk glassfish. Clic sur Next puis sur Finish.



- ✓ Glassfish configuré. Démarrer le Serveur comme suit :



Pour vérifier le bon déroulement de l'installation, accéder à la console d'administration de Glassfish en entrant l'url : <http://localhost:4848>