**TP Web services avec Tomcat et Axis2 : Mise en place de l'environnement**

* Installez Axis2 :
  + Positionnez la variable d'environnement **$TPWS** qui pointera vers un répertoire de votre choix et qui sera le répertoire de Axis2
  + Copier Axis2 dans ce répertoire après l’avoir dézippé
  + Positionnez les variables d'environnement en exécutant : **set\_env.bat**  
    Ajouter cette ligne suivante dans le fichier .bashrc (pour l’environement Linux)
* Vérifiez que votre variable d'environnement **PATH** est bien positionnée pour accéder à l'environnement de programmation java jdk.   
  Dans une fenêtre "Terminal" exécutez la commande **java -version**
* Pour lancer le serveur Axis2 exécutez la commande : **axis2server.sh**

**Développement d'un premier exemple d'un service Web avec Axis2**

Le but de ce TP est de développer un premier exemple d'un service Web en utilisant Axis2. Il s'agit d'afficher tout simplement le célèbre : HELLO WORLD.

* **Etape 1 :**  
  Définir la structure répertoire/fichier suivante:
* - helloService
* - META-INF
* - services.xml
* - hello
* - HelloWorld.java
* **Etape 2**  
  Définir la classe du service HelloWorld.java
* **Etape 3**  
  Définir le fichier de déploiment comme suit:
* <service name="HelloWorldService" scope="application">
* <description>
* Hello World Service
* </description>
* <messageReceivers>
* <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only" class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
* <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out" class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
* </messageReceivers>
* <parameter name="ServiceClass">
* hello.HelloWorld
* </parameter>
* </service>
* **Etape 4 : Déploiement du service sur Axis2**  
  Cette étape consiste à déployer le service au sein d'un fournisseur de services web. L'environnement d'exécution et de déploiement des services web que nous utilisons est l'outil Axis2.   
  Pour réaliser le déploiement, il suffit de copier le répertoire **helloService** dans le répertoire **$AXIS2\_HOME/repository/services/**de Axis2. Vous devez donc effectuer :  
  **cp -r helloService $AXIS2\_HOME/repository/services/**
* **Etape 5 : Accès au service sur Axis2**  
  Vous êtes désormais en mesure de voir le service HelloWorldService dans la liste des services de Axis2 accessibles sur la page dont l'URL est la suivante :

http://localhost:8080/axis2/services/

Si vous cliquez sur le lien du service **HelloWorldService**, vous verrez la définition WSDL (générée automatiquement par Axis2) de votre service web.

* **Etape 3 : Exécution du service**  
  La dernière étape consiste à mettre en oeuvre votre service qui est désormais accessible à travers tout le net !   
  Pour exécuter une méthode de votre service et obtenir la réponse SOAP correspondante, vous tapez l'expression suivante dans votre navigateur:

http://localhost:8080/axis2/services/HelloWorldService/hello

Par rapport à l'expression précédente, vous précisez ici le nom de la méthode à exécuter.

**Développement d'un client d'un service Web avec Axis2 (en utilisant Axis Data Bindings)**

L'objectif de ce TP est d'utiliser/tester le service web développé dans le [TP précédent](http://www-inf.it-sudparis.eu/cours/CSC5522/WebServices/TP_WS/Axis2/TP_Hello/service.html). Pour ce faire, nous allons développer un client Axis en utilisant les outils de génération automatique d'interface Java/XML fournis par cet outil.  Nous utiliserons l'utilitaire WSDL2Java, qui permet de générer à partir de la description WSDL d'un service les différentes classes et interfaces clientes nécessaires à l'appel de ce service côté client.

* **Etape 1 :**  
  La description WSDL du l'exemple HelloWorld est accessibles à l'adresse : <http://localhost:8080/axis2/services/HelloWorldService?wsdl>   
  Générer les interfaces Java clientes du service HelloWorld en utilisant la commande suivante :  
  **wsdl2java.bat -uri http://localhost:8080/axis2/services/HelloWorldService?wsdl -d adb**   
  L'utilitaire WSDL2Java génère un certain nombre de classes et interfaces Java dont la classe **HelloWorldServiceStub.java**. Cette classe contient les différentes méthodes exposées par ce service. Au sens de la spécification WSDL, il s'agit d'un type de port (<<portType>>) qui implémente plusieurs <<opérations>> (les méthodes de notre service). La classe **HelloWorldServiceStub.java** contient une classe pour chaque type de données complexe éventuellement défini par le service (dans le fichier .wsdl).
* **Etape 2 :**  
  Le code du client dépend des classes générées. Dans le code client, il suffit de
  1. instancier le stub
  2. préparer les paramettres d'entr ée
  3. appeler une méthode du stub

Voilà un exemple de code:

import hello.HelloWorldServiceStub;

import hello.HelloWorldServiceStub.HelloResponse;

public class Client {

public static void main (String [] args){

try{

//creation d'un stub pour le service Web Hello World

HelloWorldServiceStub stub = new HelloWorldServiceStub();

//invocation de la methode hello

HelloResponse resp = stub.hello();

// affichage de resultat

System.out.println(resp.get\_return());

}catch(Exception e){

System.out.println(e);

}

}

}

* **Etape 3 :**  
  Tester le client.

**Développement d'un service Web (Client + Serveur) : calcul du double et du carré d'un entier**

En procédant comme précédemment (exercice de Hello World), on vous demande de définir un service web qui calcule le carré et le double d'un nombre.   
Le service retourne cette fois-ci un nombre La méthode à invoquer possède un parameter.   
Pour traiter les méthodes avec paramètres, supposons que votre service possède la méthode Carre ayant la signature suivante :

public class Carre {   
 public int calcule(int nb){   
 return nb\*nb;   
 }   
 }

L'invocation sera alors réalisée de la façon suivante : 

http://localhost:8080/axis2/services/CarreWorldService/carre?args0=10

Vous obtiendrez la réponse suivante (toujours sous un format SOAP XML) dont voici un extrait :   
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">   
   < soapenv:Body>  
     <calculeResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">  
       <calculeReturn xsi:type="xsd:int">100</calculeReturn>   
     </calculeResponse>   
   </soapenv:Body>   
 </soapenv:Envelope>

**Développement d'un client**

Développer un client java comme précédemment (exercice de Hello World) ou Servlet qui donne le carré et le double d'un entier.

**Développement d'un service Web à partir d'une description WSDL**

1. Copier cet exemple [Addition.wsdl](http://www-inf.it-sudparis.eu/cours/CSC5522/WebServices/TP_WS/Axis2/TP_InOut/Addition.wsdl)
2. Définir la structure répertoire/fichier suivante:
3. - Addition
4. - META-INF
5. - Addition.wsdl
7. Générer les fichiers Java côté serveur en utilisant la commande suivante :

wsdl2java.sh -uri META-INF/Addition.wsdl -p addition -d adb -s -ss -sd -ssi

1. Ajouter la logique de traitement de service à la classe d'implémentation serveur (*AdditionServiceSkeleton.java*)
2. int a = additionner0.getEntier1();
3. int b = additionner0.getEntier2();
4. calculette.AdditionnerResponse resp = new calculette.AdditionnerResponse();
5. resp.setResultat1(a+b);
6. return resp;
7. lancer un serveur axis avec le service en exécutant **ant start.server**
8. Déployer le service sur Axis en copiant **build/lib/AdditionService.aar**dans le répertoire **/repository/services/** de axis.
9. Ecrire un client pour interroger ce service (en exécutant **wsdl2java.sh -uri http://localhost:8080/axis2/services/AdditionService?wsdl -d adb**