Annexe A

Les Services Web en pratique

La version HTML de ce document, ainsi que les ressources qu'il utilise sont accessibles via : http ://www-clips.imag.fr/mrim/User/marie-christine.fauvet/icar06/

Avant de commencer ce TP copier l'archive `webserv_prat_1.tar`, puis extraire les fichiers qu'elle contient. Cela aura pour effet de créer l'arborescence et les fichiers nécessaires au TP (voir section A.8.1).

Objectifs :

Nous proposons de mettre en pratique le cours dédié aux services Web par le biais d'une activité qui consiste à programmer et implanter des services web simples.

A.1 Préambule

Attention :

Les utilisateurs d'autres systèmes d'exploitation que unix/linux doivent adapter en conséquence les notations utilisées ci-après.

Outils à installer :

Pour ce TP, les outils qui doivent avoir été installés sur la machine sont listés ci-après. Les versions utilisées pour ces outils sont valides au 31 juillet 2006. Elles sont bien sûr amenées à évoluer avec le temps.

1. JDK 1.5
2. Tomcat 5.5 (à rendre accessible via le port 8080)
3. Beehive
4. Ant 1.6.5
5. Axis 1.4

Les variables d'environnement listées ci-après doivent avoir été correctement positionnées (consulter les documentations d'installation, et positionner les variables en fonction de l'installation locale) :

– AXIS_HOME
– AXIS_LIB

– AXISCLASSPATH
– ANT_HOME
– BEEHIVE_HOME
– CATALINA_HOME
– CATALINA_LIB
– CATALINACLASS
– JAVA_HOME
– CLASSPATH

NB : Les services Web seront déployés dans un serveur Apache Tomcat sur la machine locale accessible via le port 8080.

A.2 Beehive et Axis

Beehive : Le projet Beehive est lancé par Apache Group pour le développement rapide d'applications Web entre autres. La notion de programmation avec annotations est introduite par le projet JSR (Java Specification Request). L'idée estt d'introduire des annotations dans du code java afin de décrire plus facilement le service Web, ses méthodes et ses paramètres. Beehive fait appel à l'outil de compilation Ant qui permet de compiler et de déployer le service Web développé en Java et JSR. Beehive s'appuie sur les bibliothèques fournies par Axis. Voir la documentation annotations JWS : `http ://e-docs.bea.com/wls/docs92/webserv/annotations.html`

Axis : Le projet Axis est aussi un projet de Apache Group pour le développement et le déploiement de services Web. Axis est une bibliothèque de classes sur laquelle on s'appuie pour le développement du service et celui du client, les deux utilisant SOAP (Simple Object Application Protocol).

A.3 Déploiement d'un service

Le service `TimeNow` donne l'heure système du serveur sur lequel il s'exécute, il possède une seule opération non paramétrée : `getTimeNow`. Pour concevoir ce service Web nous allons dans un premier temps écrire le code Java de la classe `TimeNow`.

1. Dans le répertoire `webserv_prat_1/ws_time/WEB-INF/src-ws/web` créer un fichier `TimeNow.java` (voir son contenu, section A.8.2). Respecter le nom.
2. Il faut noter que ce qui est en gras après le symbole `@` sont des annotations qui permettent de signaler que la classe `TimeNow` représente un service Web (`@WebService`). L'autre annotation utilisée est `@WebMethod` qui permet d'indiquer parmi les méthodes celles qui sont exposées par le service Web.
3. Dans le répertoire `webserv_prat_1/ws_time/WEB-INF/src` se trouve le fichier `build.properties`. Editer ce fichier pour modifier les lignes suivantes et les positionner comme suit (remplacer le chemin d'accès à `beehive.home` par la valeur correspondant à l'installation locale) :

```
----------------build.properties------------
beehive.home=/home/fauvet/Beehive/apache-beehive-incubating-1.0m1/
service.name=TimeNow
```

```
---------------------------------------------
```

4. Aller dans le répertoire `webserv_prat_1/ws_time` et dans le fichier `index.html` vérifier les lignes signalées ci-après :

```
-----------------index.html---------------
<a href="web/TimeNow.jws?wsdl">WSDL</a>
<a href="web/TimeNow.jws?method=getTimeNow">appel opération</a>
------------------------------------------
```

5. Compilation et déploiement avec `ant` :
   Aller dans le répertoire `webserv_prat_1/ws_time/WEB-INF/src` et exécuter la commande suivante :
   `ant clean build war`
6. Avec l'outil `Tomcat Web Application Manager` (*`WAR file to deploy`*) déployer l'application contenue dans le fichier `TimeNowWS.war` (généré par `ant` dans le répertoire `webserv_prat_1/`).
7. Observer les informations rendues disponibles au travers du lien `http ://127.0.0.1 :8080/TimeNowWS/` ..

A.4 Déploiement d'un client

Il s'agit ici de développer en java un client qui va exécuter la méthode du service `TimeNow` déjà étudiée. Cette fois l'appel au service va être effectué à partir d'un programme java et non plus via un navigateur.

1. Dans le répertoire `webserv_prat_1/ws_client` créer un fichier java nommé `ClientTimeNow.java` (voir son contenu, sectionA.8.3). Respecter le nom.
2. Compiler le fichier : `$ javac ClientTimeNow.java`
3. Avant d'exécuter la classe compilée il faut avoir bien défini dans la variable d'environnement `CLASSPATH` le chemin d'accès au répertoire où le fichier `ClientTimeNow.class` a été placé.
4. Dans le répertoire de `ClientTimeNow.class`, exécuter l'instruction suivante :
   `$ java ClientTimeNow`. A la date près, le résultat doit ressembler à ce qui suit :
   `$ java ClientTimeNow`
   Bonjour d'Autrans ! ! Ici il est : Sun Mar 05 17 :38 :49 CET 2006

   Le message d'erreur :
   `- Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled. est sans conséquence.`

A.5 Un service avec une opération paramétrée, et un client

Ce service Web affiche un message concaténé au nom passé en paramètre de l'opération `sayHelloWorldInParam(String name)` et de la date système. Cette dernière est obtenue par l'appel au service `TimeNow` réalisé précédemment. Pour concevoir ce service Web nous allons dans un premier temps écrire le code Java de la classe Hello.

1. Dans le répertoire `webserv_prat_1/_hello/WEB-INF/src-ws/web/` créer un fichier `Hello.java` (voir son contenu, section A.8.4). Respecter le nom.
2. Noter que ce qui est en gras après le `@` sont des annotations qui permettent de signaler que la classe `Hello` représente un service Web (`@WebService`). L'autre annotation utilisée est `@WebMethod` qui permet d'indiquer les méthodes du service Web. Dans cet exemple il n'y a qu'une méthode paramétrée (`sayHelloWorldInParam(String name)`) qui renvoie un bonjour au nom passé en paramètre (`@WebParam`).
3. Dans le répertoire `webserv_prat_1/ws_hello/WEB-INF/src/` se trouve le fichier `build.properties`. Editer ce fichier pour modifier les lignes suivantes et la positionner comme suit (remplacer le chemin d'accès à `beehive.home` par la valeur correspondant à l'installation locale) :

```
----------------build.properties------------
beehive.home=/home/fauvet/Beehive/apache-beehive-incubating-1.0m1/
service.name=Hello
--------------------------------------------
```

4. Aller dans le répertoire `webserv_prat_1/ws_hello` et dans le fichier `index.html` vérifier les lignes signalées ci-après :

```
-----------------index.html---------------
<a href="web/Hello.jws?wsdl">WSDL</a>
<a href="web/Hello.jws?method=sayHelloWorldInParam&name=you">
                              appel opération</a>
------------------------------------------
```

5. Compilation et déploiement avec `ant` :
   Aller dans le répertoire `webserv_prat_1/ws_hello/WEB-INF/src/` et exécuter la commande suivante :
   `ant clean build war`
6. Avec l'outil `Tomcat Web Application Manager` (*`WAR file to deploy`*) déployer l'application contenue dans le fichier `HelloWS.war` (généré par ant dans le répertoire `webserv_prat_1`).
7. Observer les informations rendues disponibles au travers du lien `http ://127.0.0.1 :8080/HelloWS/`.

Implantation d'un client :

1. Dans le répertoire `webserv_prat_1/ws_client` créer le fichier `ClientHelloInParam.java` (voir son contenu, section A.8.5).
2. Compiler et exécuter `ClientHelloInParam` avec les paramètres adéquats.

A.6 Perfect Glass Inc. et l'entrepôt

A.6.1 Version 1 : paramètres simples

Les fichiers `qDemandeeLocal.java` (voir son contenu, section A.8.6) et `EntrepotLocal.java` (voir son contenu, section A.8.7) contiennent respectivement la description des opérations `qDemandee` implantée par Perfect Glass Inc. et `Disponibilité` implantée par l'Entrepôt (étude de cas vue dans le cours).

Les codes fournis utilisent des actions de saisie définies dans `Saisie.java` à compiler et à rendre accessible (voir son contenu, section A.8.8).

Dans ces codes, on a considéré que le client (Perfect Glass Inc.) et le fournisseur (Entrepôt) étaient accesssibles en local, sur la même machine.

Modifier les codes fournis afin d'implanter le fournisseur via un service web : sur le modèle de l'arborescence `ws_time`, compléter l'arborescence de racine `Entrepot` (dans le répertoire `webserv_prat_1`).

1. Créer `Entrepot.java` dans le répertoire `webserv_prat_1/Entrepot/WEB-INF/src-ws/web/`
2. Adapter les fichiers `build.properties` et `index.html`.
3. Compiler et déployer le service.
4. Tester le déploiement via le serveur web (Tomcat).

Implanter le client :

1. Créer `qDemandee.java` dans un répertoire, par exemple `webserv_prat_1/PerfectGlass`. Veiller à ce que ce dernier répertoire soit dans la variable d'environnement `CLASSPATH`.
2. Compiler et tester le client.

Voir le corrigé côté Perfect Glass Inc. et côté Entrepôt. Penser à modifier si nécessaire les paramètres concernés dans le fichier `build.properties`.

A.6.2 Version 2 : paramètres complexes

L'objectif est d'étendre le service `Hello` et son client `ClientHello`. Cette extension consiste à paramétrer l'opération `sayHello` exposée par le service `Hello` de manière à ce qu'elle reçoive un objet de la classe `Personne` et renvoie une chaîne de caractères.

Il s'agit ici d'exploiter la possibilité offerte par Axis de sérialiser/désérialiser des classes Java implantées selon le patron des *Java Beans* qui fixe le format d'écriture des sélecteurs et des constructeurs.

1. Copier et décompresser l'archive `ws_helloObjet.tar` (cliquer ici) sous la racine `webserv_prat_1/`.
2. Observer l'arborescence ainsi créée : le service s'appuie sur classe `Personne` dont le code est fourni dans le paquetage `hello` (voir `ws_helloObjet/WEB-INF/src/hello/Personne.java`).
3. Créer le client `ClientHelloObjet.java` (voir son contenu, section A.8.9) et la classe `Personne.java` (voir son contenu, section A.8.10) dans le répertoire `webserv_prat_1/ws_client`. Compiler `Personne.java` et `ClientHelloObjet.java`

4. Personnaliser les paramètres dans `build.properties` (placé dans le répertoire `webserv_prat_1/ws_helloObjet/WEB-INF/`).
5. Compiler et déployer le service. Tester.

Sur le modèle fourni, modifier le service `Entrepot` et son client `PerfectGlass`, de manière à ce que l'opération `Disponibilite` admette un paramètre de type `Produit` (à définir).

Voir le corrigé client côté Perfect Glass Inc., (classe `Produit`) et (cliquer ici) pour obtenir l'archive qui contient le service `EntrepotObjet` .

Penser à modifier en conséquence les paramètres concernés dans le fichier `build.properties`.

A.7 Composition de services

A.7.1 Composition à base d'opérations bi-directionnelles

1. Modifier le service `Hello` afin qu'il réalise l'appel au service `TimeNow` et retourne au client le message issu de la concaténation du salut et la date système. Le diagramme de séquence modélisant les interactions est décrit par la figue A.1.
2. Implanter le client (sur a base du client `ClientHelloInParam.java` déjà étudié).

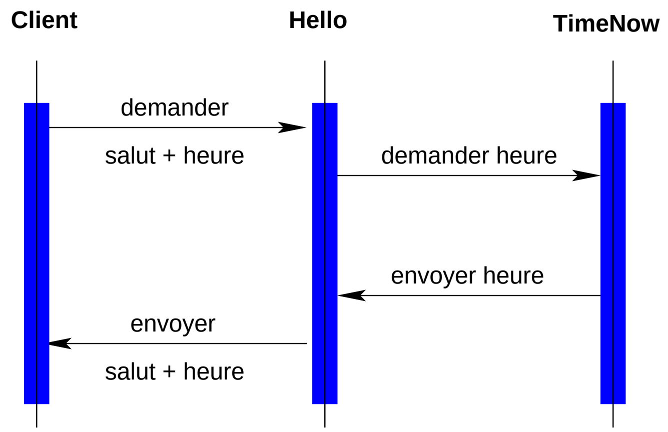

Figure A.1 – Interactions entre les services sur la base d'opérations bidirectionnelles.

Voir le corrigé pour le service et pour le client. Penser à modifier en conséquence les paramètres concernés dans le fichier `build.properties`.

A.7.2 Composition à base d'opérations uni-directionnelles

L'objectif est d'étudier l'implantation de services dont les interactions s'appuient sur des opérations unidirectionnelles. Le diagramme de séquence modélisant ces interactions, dans le cadre des échanges entre le département Ventes de Perfect Glass Inc. et l'entrepôt est décrit par la figue A.2.

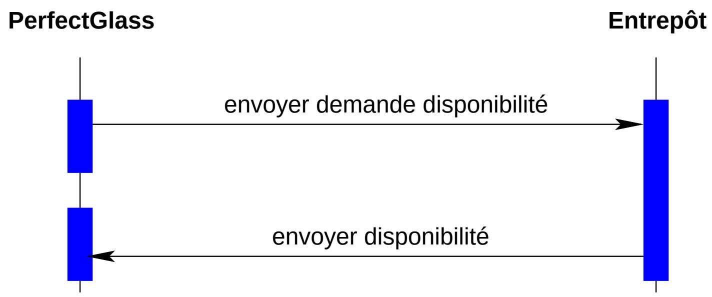

Figure A.2 – Interactions entre les services sur la base d'opérations unidirectionnelles.

1. Copier et décompresser l'archive `EntrepotUnidir.tar` (cliquer ici) sous la racine `webserv_prat_1/`
2. Copier et décompresser l'archive `PerfectUnidir.tar` (cliquer ici) sous la racine `webserv_prat_1/`. Cette archive contient le client `PerfectGlassClient.java` qui initialise les interactions, et l'arborescence du service `PerfectGlassUnidirService` qui traite la réception de la réponse envoyée par l'entrepôt.
3. Compiler le client `PerfectGlassClient.java` ainsi que la classe `Produit.java`.
4. Personnaliser les paramètres dans `build.properties`, pour chacun des deux services `EntrepotUnidir` et `PerfectGlassUnidir`.
5. Compiler et déployer les services. Tester (le message de retour est enregistré dans le fichier `PerfectGlass` placé dans `/tmp`).

Sur le modèle fourni implanter les interactions décrites dans la figure A.3.

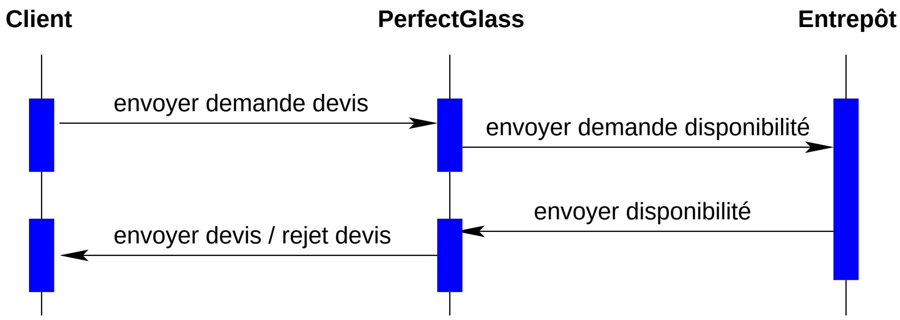

Figure A.3 – Interactions entre les services Client, PerfectGlass et Entrepôt.

A.8 Ressources

A.8.1 Arborescence de `webserv_prat_1/` (telle que fournie dans l'énoncé)

```
webserv_prat_1/
webserv_prat_1/EntrepotLocal/
```

```
webserv_prat_1/ws_hello/
webserv_prat_1/ws_hello/WEB-INF/
webserv_prat_1/ws_hello/WEB-INF/server-config.wsdd
webserv_prat_1/ws_hello/WEB-INF/src/
webserv_prat_1/ws_hello/WEB-INF/src/build.xml
webserv_prat_1/ws_hello/WEB-INF/src/build.properties
webserv_prat_1/ws_hello/WEB-INF/web.xml
webserv_prat_1/ws_hello/WEB-INF/src-ws/
webserv_prat_1/ws_hello/WEB-INF/src-ws/web/
webserv_prat_1/ws_hello/index.html
webserv_prat_1/ws_client/
webserv_prat_1/PerfectGlassLocal/
webserv_prat_1/Entrepot/
webserv_prat_1/Entrepot/WEB-INF/
webserv_prat_1/Entrepot/WEB-INF/server-config.wsdd
webserv_prat_1/Entrepot/WEB-INF/src/
webserv_prat_1/Entrepot/WEB-INF/src/build.xml
webserv_prat_1/Entrepot/WEB-INF/src/build.properties
webserv_prat_1/Entrepot/WEB-INF/web.xml
webserv_prat_1/Entrepot/WEB-INF/src-ws/
webserv_prat_1/Entrepot/WEB-INF/src-ws/web/
webserv_prat_1/Entrepot/index.html
webserv_prat_1/ws_time/
webserv_prat_1/ws_time/happyaxis.jsp
webserv_prat_1/ws_time/WEB-INF/
webserv_prat_1/ws_time/WEB-INF/server-config.wsdd
webserv_prat_1/ws_time/WEB-INF/src/
webserv_prat_1/ws_time/WEB-INF/src/build.xml
webserv_prat_1/ws_time/WEB-INF/src/build.properties
webserv_prat_1/ws_time/WEB-INF/web.xml
webserv_prat_1/ws_time/WEB-INF/src-ws/
webserv_prat_1/ws_time/WEB-INF/src-ws/web/
webserv_prat_1/ws_time/index.html
webserv_prat_1/PerfectGlass/
```

A.8.2 Code du service `TimeNow`

```
package web;
import javax.jws.WebMethod;
import javax.jws.WebService;
import java.util.Date;

@WebService
public class TimeNow {
   @WebMethod
   public String getTimeNow() {
      Date d=new Date();
      String t=d.toString();
      return t;
   }
}
```

A.8.3 Code du client ClientTimeNow

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;

import javax.xml.namespace.QName;

public class ClientTimeNow {
    public static void main(String [] args) {
        try {

            // l'URI a contacter
            String endpointURL = "http://localhost:8080/TimeNowWS/web/TimeNow.jws";

            // Le service à executer
            Service  service = new Service();
            Call  call    = (Call) service.createCall();
            call.setTargetEndpointAddress( new java.net.URL(endpointURL) );

            // l'operation du service
            call.setOperationName( new QName("TimeNow", "getTimeNow") );

            // L'appel
            String ret = (String) call.invoke( new Object[] { } );

            System.out.println("Bonjour d'Autrans !! Ici, il est " + ret);
          } catch (Exception e) {
            System.err.println("erreur "+ e.toString());
}   }     }
```

A.8.4 Code du service Hello

```
package web;

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

@WebService
public class Hello {

   @WebMethod
   public String sayHelloWorldInParam(@WebParam String name ) {
       try {

           return "Hello, " + name + ". ";

       } catch (Exception e) {
           return "Erreur " + e.toString();
}  }   }
```

A.8.5 Code du client ClientHelloInParam

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;

import javax.xml.namespace.QName;
import javax.xml.rpc.ParameterMode;

public class ClientHelloInParam {
    public static void main(String [] args) {
        try {
            // on attend un parametre
            Options options = new Options(args);
            String textToSend;

            args = options.getRemainingArgs();
            if ((args == null) || (args.length < 1)) {
                textToSend = "l'inconnu";
            } else {
                textToSend = args[0];
            }

            // Le necessaire pour realiser l'appel :
            Service  service = new Service();
            Call     call    = (Call) service.createCall();

            // L'URI du service a appeler
            String url = "http://127.0.0.1:8080/HelloWS/web/Hello.jws";
            call.setTargetEndpointAddress( new java.net.URL(url) );

            // L'operation a executer avec ses parametres d'entree et de sortie
            call.setOperationName( new QName("Hello", "sayHelloWorldInParam") );
            call.addParameter( "s", XMLType.XSD_STRING, ParameterMode.IN);
            call.setReturnType( org.apache.axis.encoding.XMLType.XSD_STRING );

            // L'appel
            String ret = (String) call.invoke( new Object[] { textToSend } );

            System.out.println(ret);

        } catch (Exception e) {
            System.err.println(e.toString()+" ici");
        }
    }
}
```

A.8.6 Code du programme qDemandeeLocal

```
import java.io.* ;

public class qDemandee{
    public static void main(String args[]) {
        try {
            String produit; // la référence du produit
            int quantite ; // la quantité demandée

            produit = Saisie.lire_String ("Saisie de la référence produit : ");
            quantite = Saisie.lire_int ("Saisie de la quantité demandée : ");

            System.out.println ("Vérification de disponibilité.......");

            String ret = Entrepot.Disponibilite (produit, quantite);

            System.out.println(ret);

        } catch (Exception e) {
            System.err.println(e.toString()+" ici");
        }
    }
}
```

A.8.7 Code du programme EntrepotLocal

```
class Produit {
    private String reference ;
    private int quantite ;

    public Produit() {
    }

    public void setRef(String ref) {
        reference = ref ;
    }
    public void setQuantite (int qte) {
        quantite = qte ;
    }

    public String getRef() {
        return reference;
    }

    public int getQuantite() {
        return quantite ;
    }
}

public class Entrepot {
       public static String Disponibilite (String produit, int quantite) {
```

```
        try {
            String res ;
            Produit v = new Produit () ;
            Produit a = new Produit ();

            v.setRef("verres") ;
            v.setQuantite(200);

            a.setRef("assiettes") ;
            a.setQuantite(4500);

            res = "Le produit " + produit + ", en quantité " + quantite ;
            if ((v.getRef().equals (produit) && v.getQuantite() >= quantite) ||
                (a.getRef().equals (produit) && a.getQuantite() >= quantite)) {
                res = res + ", est disponible.";
            } else {
                res = res + ", est indisponible" ;
            }
            return res ;
        } catch (Exception e) {
            return "Erreur " + e.toString();
        }
    }
}
```

A.8.8 Code des procédures de Saisie

```
import java.io.*;
        class Saisie {
        public static String lire_String ()
        {
                String ligne_lue=null ;
                try {
                InputStreamReader isr = new InputStreamReader(System.in);
                BufferedReader br = new BufferedReader(isr);
                ligne_lue = br.readLine ();
                }
                catch (IOException e) {System.err.println(e);}
                return ligne_lue;
        }
        public static String lire_String (String question)
        {
                System.out.print (question);
                return (lire_String());
        }
        public static int lire_int ()
        {
                return Integer.parseInt (lire_String ());
        }
        public static int lire_int (String question)
        {
```

```
                System.out.print (question);
                return (lire_int());
        }

        public static double lire_double()
        {
                return Double.parseDouble (lire_String ());
        }
        public static double lire_double (String question)
        {
                System.out.print (question);
                return (lire_double());
        }

        public static float lire_float()
        {
                return Float.parseFloat (lire_String ());
        }
        public static float lire_float (String question)
        {
                System.out.print (question);
                return (lire_float());
        }

        public static char lire_char()
        {
                String reponse = lire_String ();
                return reponse.charAt(0);
        }
        public static char lire_char (String question)
        {
                System.out.print (question);
                return (lire_char());
        }
}
```

A.8.9 Code du client `ClientHelloObjet`

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;

import javax.xml.namespace.QName;
import javax.xml.rpc.ParameterMode;

public class ClientHelloObjet {
    public static void main(String [] args) {
        try {
            String nom ;
            int age ;
```

```
            String adresse ;
            nom = Saisie.lire_String ("Donner un nom : ");
            age = Saisie.lire_int ("Donner un age : ") ;
            adresse = Saisie.lire_String ("Donner une adresse : ") ;

            Personne personne = new Personne ();
            personne.setAge(age);
            personne.setAdresse(adresse);
            personne.setNom(nom);

            System.out.println ("Connexion ...........");

            Service  service = new Service();
            Call     call    = (Call) service.createCall();

            String url = "http://127.0.0.1:8080/HelloObjetWS/web/HelloObjet.jws";
            call.setTargetEndpointAddress( new java.net.URL(url) );

            // Le necessaire pour le parametre en entree
            // qn est la correspondance XML de la classe Personne
            QName personneXML = new QName( "http://HelloObjetWS/web", "Personne" );

            call.registerTypeMapping(Personne.class, personneXML,
                      new org.apache.axis.encoding.ser.BeanSerializerFactory(
                          Personne.class, personneXML),
                      new org.apache.axis.encoding.ser.BeanDeserializerFactory(
                          Personne.class, personneXML));

            // L'operation a executer avec ses parametres d'entree et de sortie
            call.setOperationName( new QName("HelloObjet", "sayHello") );
            call.addParameter( "p", personneXML, ParameterMode.IN);
            call.setReturnType( org.apache.axis.encoding.XMLType.XSD_STRING );

            String ret = (String) call.invoke( new Object[] { personne } );

            System.out.println(ret);

        } catch (Exception e) {
            System.err.println(e.toString()+" ici");
        }
    }
}
```

A.8.10 Code de la classe Personne

```
public class Personne {
    private String nom ;
    private int age ;
    private String adresse ;

    public Personne () {
```

```
    }

    public void setNom(String n) {
        nom = n ;
    }
    public void setAdresse (String a) {
        adresse = a ;
    }
    public void setAge (int a) {
        age = a ;
    }

    public String getAdresse() {
        return adresse;
    }

    public String getNom() {
        return nom;
    }

    public int getAge() {
        return age;
    }

}
```