XML, la synthèse

SOAP

Simple Object Access Protocol

SOAP – Introduction

- SOAP signifie Simple Object Access Protocol
- SOAP est un protocole de communication pour l'invocation d'objets à distance basé sur XML.
- Permet aux applications et aux composants logiciels d'échanger des informations par l'invocation de service (web services) : un composant client émet une requête auprès d'un composant serveur.
- Information structurées et typées
- Principe fondamental : échange de messages
- Simple et extensible

SOAP – Introduction

- Indépendant des systèmes d'exploitation, des langages de programmation et des moyens de transport
- Protocole dès le départ prévu pour être exploité sur Internet, les versions actuelles utilisent d'ailleurs principalement le protocole HTTP et SMTP
- Environnement décentralisé et distribué

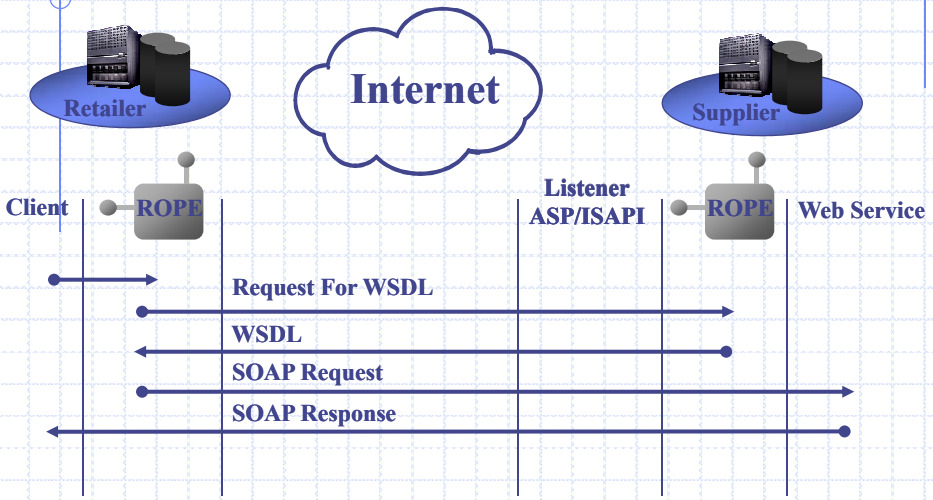
SOAP et les Web services

- SOAP et XML sont les technologies de base de l'architecture des services web ('Web Services')
- Web Services: applications modulaires et auto descriptives pouvant êtres publiées, recherchées et invoquées sur le Web

SOAP – Historique

- Proposé au W3C en mai 2000 par UserLand, Ariba, Commerce One, Compaq, Developmentor, HP, IBM, IONA, Lotus, Microsoft, and SAP
- La v1.1 considéré comme un standard de fait par certains vendeurs mais disponible uniquement sous forme de "note" auprès du W3C (aucune reconnaissance officielle)
- Premier "Working Draft" du W3C à propos de SOAP en décembre 2001.
- Version 1.2 en cours :
  - Pas encore normalisée par le consortium W3C
  - Peu supportée

SOAP – Exemple d'un échange



Composants de l'architecture

- ROPE : Remote Object Proxy Engine
  - Composant utilitaire pour la gestion des messages SOAP (création, envoi, ...) utilisé pour développer la partie cliente et serveur SOAP
- WSDL : Web Service Description Language
  - Langage permettant de connaître les possibilités et caractéristiques d'un service distant.

SOAP – Avant SOAP

- (D)COM – (Distributed) Component Object Model
  - RPC – Remote Procedure Call
- CORBA – Common Object Request Broker Architecture
  - IIOP – Internet Inter-ORB Protocol
  - RMI – Remote Method Invocation
  - Difficultés de mise en œuvre
    - Configuration, installation, administration
    - Pire si: sécurité, transaction…
    - Problèmes avec Firewall et proxy
- Certaines alternatives (RDS – Remote Data Service, XML-RPC)
  - lourde et moins performantes

SOAP – Pourquoi

- Pourquoi un nouveau protocole?
  - Volonté d'utiliser des technologies existantes
    - XML: format du message, enveloppe et mécanisme RPC
    - HTTP, SMTP…: mécanismes de transport
  - Souci de simplicité (versus CORBA, DCOM): facilité d'implémentation d'un ROPE, d'un client et d'un serveur .
  - Extensible: via l'utilisation de XML

SOAP – Avantages

- Compatible avec standards du marché
- Interopérabilité
- Découverte dynamique de services
- Supporté par différents vendeurs (MS, IBM, HP, SUN…)
- Indépendant: reconnu par W3C
  - bientôt sujet d'un RFC (Request For Comment)
- Traverse aisément les firewalls
- Structuré utilisant XML
- SOAP est très léger comme protocole
- La base texte en XML de Soap peut être envoyé et compris par la pluspart des plates-formes, OS, langage de programmation, et réseau.

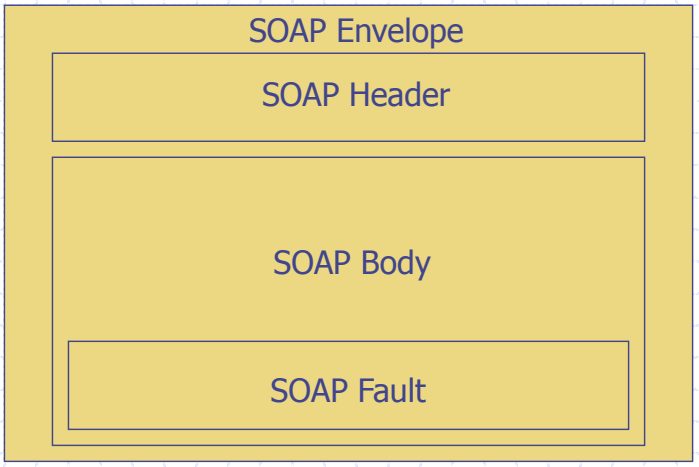
Qu'est-ce qu'un SOAP Message

- Document XML ayant son propre schéma
- Possède son propre espace de noms sur tous les éléments et attributs
  - Deux espaces de nom:
    - SOAP Envelope : http://schemas.xmlsoap.org/soap/envelope/
    - SOAP Serialisation (encodage) : http://schemas.xmlsoap.org/soap/encoding/
- Plus strict que XML
  - Ne peut pas contenir de référence à un DTD
  - Ne peut pas contenir d'instructions processeurs

Structure d'un message SOAP

- Un message SOAP :
  - **Doit** contenir un et un seul élément "Envelope"
  - **Peut** contenir un et un seul élément "Header"
  - **Doit** contenir un et un seul élément "Body"
  - Body **Peut** contenir un et un seul élément "Fault"

Structure d'un message SOAP



Exemple d'un message SOAP

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <soap:Header>
        <h:identity xlmns:h=« http://www.mysite.com/header »>me@mysite.com
        </h:identity>
    </soap:Header>
    <soap:Body>
        <m:ChercherPrix xmlns:m="http://www.mysite.com/
        ChercherPrix/">
                <m:basePrice>135</m:basePrice>
        </m:ChercherPrix>
    </soap:Body>
</soap:Envelope>
```

- Exemple de message de requête
- Dans le corps (Body) de ce message SOAP nous voyons une procédure appelée (ChercherPrix) avec passage d'un paramètre (DIS)

Exemple d'un message SOAP

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <soap:Body>
        <m:ChercherPrixResponse xmlns:m="http://www.mysite.com/
                ChercherDernierPrix/">
            <m:Prix monnaie="eur">La liste des prix</m:Prix>
        </m:ChercherPrixResponse>
    </soap:Body>
</soap:Envelope>
```

- Exemple de message de réponse
- Le nom de l'élément réponse du Body doit être le même que celui du nom de la requête avec le suffixe Response.
- Le namespace de la réponse doit être le même que celui de la requête.

SOAP Message - l'élément Envelope

- SOAP Envelope
  - Élément racine, conteneur du message
  - Définit les namespaces et l'encodage
  - Contient directement deux sous éléments:
    - <soap:Header> (optionnel)
    - <soap:Body>

SOAP Message - l'élément Header

```
<soap:Header>
   <t:Transaction xmlns:t="un-URI"
    soap:mustUnderstand="1">
        5
   </t:Transaction>
</soap:Header>
```

- L'élément "Transaction" ne fait pas partie du langage SOAP mais défini par le concepteur du message.
- L'attribut optionnel « soap:mustUnderstand" indique si le récepteur du message doit comprendre l'élément "Transaction" de l'en-tête (header) (0=faux, 1=vrai)
- Header procure une extension au message
  - Pour : authentification, info transaction, payement…

SOAP Message - l'élément Body

- Le contenu du message
  - Peut être
    - Appel ou réponse RPC
    - Message d'erreur: <soap:Fault>
    - Autre…

SOAP Message - l'élément Fault

- Utilisé pour les erreurs survenant dans une application SOAP
- Élément Fault est Optionnel, il doit être enfant direct de Body et unique

```
<soap:Fault>
    <soap:faultcode>SOAP-ENV:server </soap:faultcode>
    <soap:faultstring>Erreur du serveur</soap:faultstring>
    <soap:detail>
          <e:monDetail xmlns:e="un-URI">
                    <e:message>Ca n'a pas marché</e:message>
                    <e:errorcode>1001</e:errorcode>
          <e:monDetail>
    </soap:detail>
</soap:Fault>
```

- Doit contenir les éléments faultcode et faultstring
- Valeur standard pour faultcode: server, VersionMismatch, MustUnderstand, Client

Autre exemple d'un message SOAP et gestion des types

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <soap:Header></soap:Header>
    <soap:Body>
      <ns1:sayHelloTo xmlns:ns1="Hello" SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
       <name xsi:type="xsd:string">John</name>
      </ns1:sayHelloTo>
     </soap:Body>
</soap:Envelope>
```

- Les types de données sont supportés et exprimés grâce au langage XML Schema et XML Schema-instance

Simulation d'une requête HTTP adaptée pour SOAP

```
POST http://www.SmartHello.com/HelloApplication HTTP/1.0
Content-Type: text/xml; charset="utf-8"
Content-Length: 587
SOAPAction: "http://www.SmartHello.com/HelloApplication#sayHelloTo"
<SOAP-ENV:Envelope xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="
http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
     <SOAP-ENV:Header></SOAP-ENV:Header>
    <SOAP-ENV:Body>
         <ns1:sayHelloTo xmlns:ns1="Hello"
          SOAP-ENV:encodingStyle="
http://schemas.xmlsoap.org/soap/encoding/">
             <name xsi:type="xsd:string">Tarak</name>
         </ns1:sayHelloTo>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Simulation d'une réponse HTTP adaptée pour SOAP

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 615
<SOAP-ENV:Envelope
                      xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/"
                      xmlns:xsi="
http://www.w3.org/1999/XMLSchema-instance"
                      xmlns:xsd="http://www.w3.org/1999/XMLSchema">
     <SOAP-ENV:Body>
          <ns1:sayHelloToResponse
                     xmlns:ns1="Hello"
                     SOAP-ENV:encodingStyle="
http://schemas.xmlsoap.org/soap/encoding/">
               <return xsi:type="xsd:string">Hello Tarak, How are
you doing?</return>
          </ns1:sayHelloToResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Implémentation et API disponibles

- Java:
  - Apache SOAP (http://xml.apache.org/dist/soap/)
  - DevelopMentor's implementation (http://www.develop.com/soap/soap.jar)
  - IdooXoap from ZVON (http://www.zvon.org/index.php?nav_id=30)
- Python:
  - PythonWare (côté client uniquement) (http://www.pythonware.com/products/soap)
- C++:
  - IdooXoap from ZVON (http://www.zvon.org/?nav_id=33)

Implémentation et API disponibles

- Perl:
  - SOAP:Lite (http://www.soaplite.com/)
- ADA:
  - An ADA implementation (http://home.snafu.de/boavista/soap.html)
- Microsoft Visual Studio:
  - Microsoft SOAP toolkit. (http://msdn.microsoft.com/xml/general/toolkit_intro.asp)

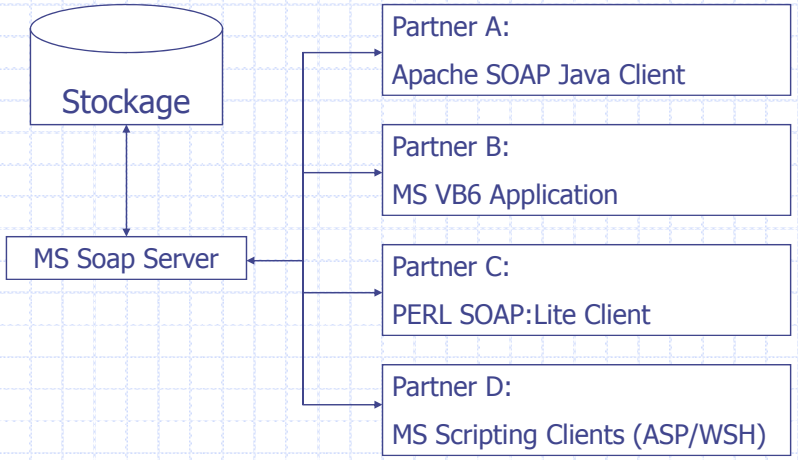
SOAP Communication

- Synchrone
  - Via HTTP ou HTTPS (encryption ssl)
  - Comme requête POST en HTML
- Asynchrone
  - Via SMTP et MIME

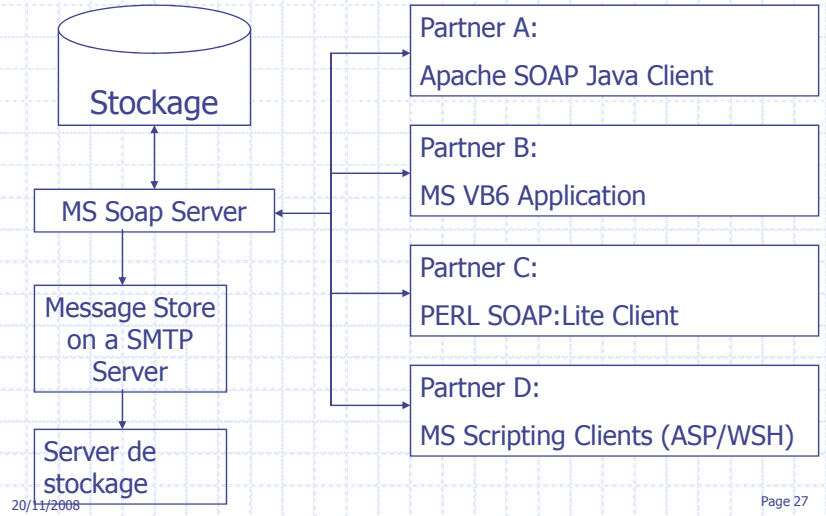
```
<port name="StockQuotePort" binding="tns:StockQuoteSoap">
   <soap:address location="mailto://subscribe@expert-it.com">
</port>
```

  - avec MIME
    - Encapsulation pour document composé (binaire…)

SOAP Exemple: Synchrone



SOAP Exemple: Asynchrone



SOAP - WSDL

- SOAP
  - définit la structure des messages échangés par les applications via Internet
- WSDL
  - standardise les schémas XML utilisés pour établir une connexion entre émetteurs et récepteurs
  - fournit un mode de description des composants applicatifs
  - permet d'invoquer leurs fonctions à distance
    - par l'échange de messages au format SOAP

UDDI

- **U**niversal **D**escription, **D**iscovery and **I**ntegration
  - Annuaire mondial d'entreprises basé sur le Web
  - Intégre toutes sortes d'entrées
    - nom, carte d'identité des sociétés, description des produits et des services, etc
  - automatise la procédure de recherche et de découverte des Web Services

Sécurité

- Importante considération de la sécurité vu les protocoles.
- Non-accepté par les entreprises si pas de sécurité.
- Privacy: être sûr que la conversation entre le client et le web service est cryptée.
- Authentification: besoin d'être identifié l'un l'autre.
- Non-repudation: garder un log file de l'invocation du client.

Sécurité (suite)

- Sécurité niveau transport: Déjà accessible aujourd'hui sous la forme de HTTPS. Suffisant pour une majorité d'applications.
- Sécurité niveau message: Besoin d'une sécurité au niveau de tous les intermédiaires présents entre le client et le Web Service.

Sécurité (suite)

- XKMS: (XML Key Management Services) nous fournit avec un standard, une distribution et un management de clés pour sécuriser les communications d'un point à un autre.
- XML Encryption: Standard qui permet d'encrypter le message de l'envoyeur et de le décrypter à l'arrivée chez le destinataire. Rend le message illisible durant le transport.
- SAML: (Security Access Markup Language) fournit un mécanisme pour déterminer les droits d'accès à l'arrivée du message.

Messaging « fiable »

- Plusieurs contraintes comme: commande de messages, delais de livraison, priorités, etc.
- HTTP est le protocole primaire utilisé comme protocole de transport, et comme nous le savons tous, n'est pas un protocole fiable.
- Des efforts sont cependant déployés pour fournir une extension fiable à l'HTTP comme la spécification HTTP Reliable (HTTP-R) de chez IBM.

Qualité de Service (QoS)

- La QoS est très importante dans le business entre deux sociétés.
- La QoS énumère certains critères à propos d'un Web Service et, si un fournisseur de Web Services dicte sa propre QoS, le consommateur a tout intérêt à s'assurer de la bonne exécution du service.

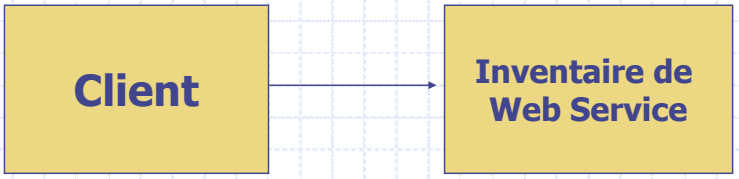
Qualité de Service (QoS) (suite)

- Critères:
  - Différentes options de support: gold service, economy service.
  - Level garanti des performances du système
  - Maximum downtime permis
  - Clauses de pénalité si un de ces critères n'est pas rencontré
  - Bande passante du réseau
  - Scénarios catastrophiques

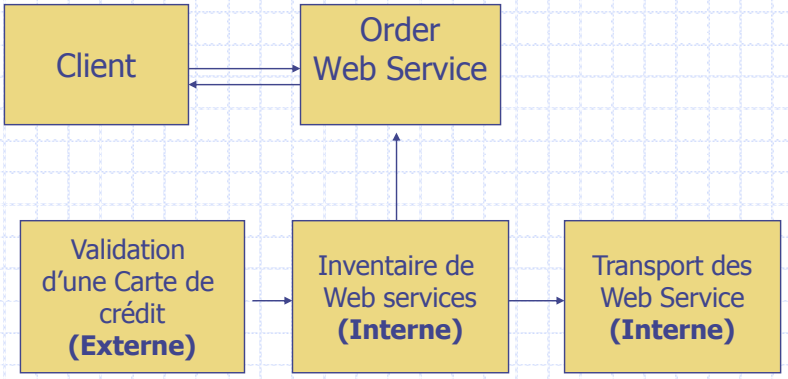
Transactions

- Les transactions sont fondamentales dans tous les business.
- Parfois des Web Services appellent d'autres Web Sevices.
- Pour l'instant pas de standard disponible pour définir les transactions via Web Services.
- Cependant un standard fait sa place: le BTP (Business Transaction Protocol).

Exemple de WorkFlow simple



Exemple de WorkFlow complexe



Performance

- Les performances des Web Services sont liés aux sites web. La raison étant que c'est le web server qui va recevoir la requête SOAP via HTTP.
- La scabilité est importante: càd capacité du Web Service à desservir une croissante demande de services.
- Solution idéale: fournir un service constant et un temps de réponse acceptable.
- Cependant non adapté dans la pratique.

Avantages de SOAP

- Traverse aisément les firewalls
- Structuré utilisant XML
- Peut-être combiné avec certains protocoles de transport comme HTTP, SMTP, JMS.
- SOAP est très léger comme protocole
- Beaucoup de supports (IBM, Microsoft, SUN)
- La base texte en XML de Soap peut être envoyé et compris par la plupart des plates-formes, OS, langages de programmation, et réseaux.

Désavantages de SOAP

- Malgré le large support, des incompatibilités subsistent entre les différentes implémentations Soap.
- Mécanismes de sécurités immature.
- Pas de garantie de livraison du message, si failure, ne sais pas renvoyer le message.
- Un client Soap ne sait pas envoyer une requête à plusieurs serveurs sans l'envoyer à tous les serveurs.

SoapAction

```
Post /rpcrouter HTTP/1.1
Host : 127.0.0.1
Content – type: text/xml; charset=«utf-8»
Content-Length: 287
SOAPAction: "http://www.mysite.com/TimeService/GetDateTime"
<soap-env:Envelope xlmns:……
```

Dans cet exemple, le SoapAction est utilisé par le firewall et le proxy pour accepter ou rejeter la requete Soap. L'URI peut aussi être une string vide : SOAPAction : ""

Installer Apache SOAP

http://jakarta.apache.org/tomcat/

- Télécharger et dézipper Tomcat

  SET TOMCAT_HOME=C:\jakarta-tomcat-X.X

- Télécharger et dézipper Apache Soap Library

  SET SOAP_HOME=C:\soap-2_2

- Configurer Tomcat pour Apache Soap

Copier soap.jar de SOAP\lib dans %TOMCAT_HOME%\lib

Copier soap.war de SOAP\webapps dans %TOMCAT_HOME%\webapps

Installer Apache SOAP (suite)

- Set CLASSPATH
  - %SOAP_HOME%\lib\soap.jar
  - %TOMCAT_HOME%\lib\xerces.jar
  - %TOMCAT_HOME%\lib\mail.jar
  - %TOMCAT_HOME%\lib\activation.jar

Tester la config côté server:

Dans IE : http://localhost:8080/soap/servlet/rpcrouter

On doit obtenir: Sorry, Idon't speakvia HTTP GET – you have to use HTTP POST to talk to me.

Tester la config côté client:

Dans cmd (d'une ligne): java org.apache.soap.server.ServiceManagerClient http://localhost:8080/soap/servlet/rpcrouter list

Retourne la liste des services déployés.

Simple Exemple

```
package hello;
public class HelloServer{
    public String sayHelloTo(String name){
            System.out.println("sayHelloTo(String name)");
            return "Hello " + name ;
    }
}
```

Créer un hello.jar contenant la HelloServer.class et le copier dans %TOMCAT_HOME%\lib

Simple Exemple (suite)

```
package hello;

import java.net.URL;
import java.util.Vector;
import org.apache.soap.SOAPException;
import org.apache.soap.Constants;
import org.apache.soap.Fault;
import org.apache.soap.rpc.Call;
import org.apache.soap.rpc.Parameter;
import org.apache.soap.rpc.Response;

public class Client
{
```

Simple Exemple (suite)

```
public static void main(String[] args) throws Exception
 {
   if(args.length == 0)
   {
     System.err.println("Usage: java hello.Client [SOAP-router-URL] ");
     System.exit (1);
   }

   try
   {
     URL url = null;
     String name = null;
     if(args.length == 2)
     {
       url = new URL(args[0]);
       name = args[1];
     }
```


Simple Exemple (suite)

```
else
       {
         url = new URL("http://localhost:8080/soap/servlet/rpcrouter");
         name = args[0];
       }

       // Build the call.
       Call call = new Call();
       call.setTargetObjectURI("urn:Hello");
       call.setMethodName("sayHelloTo");
       call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
       Vector params = new Vector();
       params.addElement(new Parameter("name", String.class, name, null));
       call.setParams(params);
```

Simple Exemple (suite)

```
// Invoke the call.
        Response resp = null;
        try
        {
           resp = call.invoke(url, "");
        }
        catch( SOAPException e )
        {
           System.err.println("Caught SOAPException (" + e.getFaultCode() + "): " + e.getMessage());
           System.exit(-1);
        }

        // Check the response.
        if( !resp.generatedFault() )
        {
           Parameter ret = resp.getReturnValue();
           Object value = ret.getValue();
           System.out.println(value);
        }
```


Simple Exemple (suite)

```
else
        {
          Fault fault = resp.getFault();
          System.err.println("Generated fault: ");
          System.out.println ("  Fault Code   = " + fault.getFaultCode());
          System.out.println ("  Fault String = " + fault.getFaultString());
        }
      }
      catch(Exception e)
      {
        e.printStackTrace();
      }
    }
}
```

Déployer le service pour l'exemple

IE : http://localhost:8080/soap/admin/index.html

Cliquer sur deploy, et ensuite configurer

ID	urn:Hello
Scope	Application
Methods	sayHelloTo
Provider Type	Java
Provider Class	hello.HelloServer
Static	No
Def Map Reg Class	org.apache.soap.DOMFaultListener