SOAP

Autors: Joan Manuel Marquès i Leandro Navarro.

Introducció

En aquesta sessió de problemes crearem un servei Web que parli SOAP utilitzant Axis. Això ens ajudarà entendre el funcionament d'SOAP.

El guió d'aquesta sessió està basant en els articles:

- Axis User's Guide: http://xml.apache.org/axis/ (de fet, n'hem copiat algunes parts literalment)
- Creating Web Services with Apache Axis (especialment interessant per la part de WSDL ja que conté un exemple detallat molt fàcil de seguir): http://www.onjava.com/lpt/a//onjava/2002/06/05/axis.html

Uns altres articles interessants de consultar –sobretot per la part d'instal·lació- són:

- Axis: The next generation of Apache Soap: http://www.javaworld.com/javaworld/jw-01-2002/jw-0125-axis.html
- Axis Installation Guide: http://xml.apache.org/axis/

Us recomanem que per qualsevol dubte que tingueu recorreu a aquests documents. Al ser més extensos que aquest guió us poden ajudar a entendre els detalls que aquí no apareixen.

What is SOAP? (http://xml.apache.org/axis/)
SOAP is an XML-based communication protocol and encoding format for inter-application communication.

SOAP is widely viewed as the backbone to a new generation of cross-platform cross-language distributed computing applications, termed Web Services.

What is Axis? (http://xml.apache.org/axis/)
Axis is essentially a SOAP engine -- a framework for constructing SOAP processors such as clients, servers, gateways, etc. The current version of Axis is written in Java, but a C++ implementation of the client side of Axis is being developed.

But Axis isn't just a SOAP engine -- it also includes:

- *a simple stand-alone server,*
- *a server which plugs into servlet engines such as Tomcat,*
- *extensive support for the Web Service Description Language (WSDL),*
- *emitter tooling that generates Java classes from WSDL.*
- *some sample programs, and*
- *a tool for monitoring TCP/IP packets.*

Axis now delivers the following key features:

- ***Speed**. Axis uses SAX (event-based) parsing to acheive significantly greater speed than earlier versions of Apache SOAP.*
- ***Flexibility**. The Axis architecture gives the developer complete freedom to insert extensions into the engine for custom header processing, system management, or anything else you can imagine.*
- ***Stability**. Axis defines a set of published interfaces which change relatively slowly compared to the rest of Axis.*

- ***Component-oriented deployment***. *You can easily define reusable networks of Handlers to implement common patterns of processing for your applications, or to distribute to partners.*
- ***Transport framework***. *We have a clean and simple abstraction for designing transports (i.e. senders and listeners for SOAP over various protocols such as SMTP, FTP, message-oriented middleware, etc), and the core of the engine is completely transport-independent.*
- ***WSDL support***. *Axis supports the Web Service Description Language, version 1.1, which allows you to easily build stubs to access remote services, and also to automatically export machine-readable descriptions of your deployed services from Axis.*

Objectius

- Entendre el funcionament d'SOAP.
- Saber instal·lar un servei web que funcioni utilitzant SOAP.
- Saber construir una aplicació que utilitzi una eina SOAP.

Plataforma

Donat que tot el que farem servir funciona sobre java, podeu utilitzar indistintament Linux o MS-Windows

Tasques

1. Instal·lació Tomcat

Primer de tot, cal tenir un servidor web instal·lat. Nosaltres ho farem utilitzant Tomcat (Consulta la pàgina web de l'assignatura).

> Si utilitzeu MS-Windows, podeu baixar-vos un .exe que fa la instal·lació. Cal que feu la instal·lació a c:\tomcat4.1.

A partir d'aquest punt, ens referirem al directori on heu intal·lat Tomcat com `TOMCAT_HOME`.

2. Instal·lació pàrser XML: xerces

Baixeu la versió de xerces disponible a la pàgina web de l'assignatura i descomprimiu-la. El fitxer que ens interessa, xerces.jar, estarà en l'arrel del directori de xerxes.

3. Instal·lació Axis

Baixeu la versió d'Axis disponible a la web de l'assignatura. Descomprimiu el fitxer zip i deixeu-lo en un directori que anomenarem `AXIS_HOME`.

Copieu la carpeta d'Axis `AXIS_HOME\webapps` a `TOMCAT_HOME\webapps`. Per a que el Tomcat reconegui Axis com un servei més.

Copieu `xerces.jar a TOMCAT_HOME\webapps\axis\WEB_INF\lib`. D'aquesta manera la llibreria estarà disponible per treballar des del Tomcat.

Ja per acabar, cal afegir un context al fitxer `server.xml` que es troba al directori `TOMCAT_HOME\conf`. El context s'afegeix posant aquesta línia:

```
<Context path="/axis" docBase="axis" debug="0" reloadable="true"/>
```

4. Exemple de publicació de serveis usant Axis: calculadora

(Aquest apartat està basat en la documentació en l'apartat *Publishing Web Services* with *Axis d'Axis User's Guide*: http://xml.apache.org/axis/)

En aquest apartat caldrà executar alguns programes. Per tal de poder-los executar, cal que el vostre CLASSPATH inclogui (podeu trobar un exemple complert a la pàgina web de l'assignatura):

- `axis-1_1/lib/axis.jar`
- `axis-1_1/lib/jaxrpc.jar`
- `axis-1_1/lib/saaj.jar`
- `axis-1_1/lib/commons-logging.jar`
- `axis-1_1/lib/commons-discovery.jar`
- `axis-1_1/lib/wsdl4j.jar`
- `axis-1_1/` (for the sample code)
- A JAXP-1.1 compliant XML parser such as xerces or crimson

(p.ex.: MS-WINDOWS: `set CLASSPATH=c:\axis-1_1;c:\axis-1_1\lib\axis.jar;c:\axis-1_1\lib\jaxrpc.jar;c:\axis-1_1\lib\saaj.jar;c:\axis-1_1\lib\commons-logging.jar;c:\axis-1_1\lib\commons-discovery.jar;c:\axis-1_1\lib\wsdl4j.jar;C:\Tomcat4.1\webapps\axis\WEB-INF\lib\xerces.jar;c:\Tomcat4.1\common\lib\servlet.jar`)

Let's say we have a simple class like the following:

```
public class Calculator {
  public int add(int i1, int i2){
    return i1 + i2;
  }

  public int subtract(int i1, int i2) {
    return i1 - i2;
  }
}
```

(You'll find this very class in samples/userguide/example2/Calculator.java.)

How do we go about making this class available via SOAP? There are a couple of answers to that question, but we'll start with the easiest way Axis provides to do this, which takes almost no effort at all!

5. JWS (Java Web Service) Files - Instant Deployment

OK, here's step 1 : copy the above .java file into your webapp directory, and rename it "Calculator.jws". So you might do something like this:

```
% copy Calculator.java <your-webapp-root>/axis/Calculator.jws
```

Now for step 2... hm, wait a minute. You're done! You should now be able to access the service at the following URL (assuming your Axis web application is on port 8080): http://localhost:8080/axis/Calculator.jws

Axis automatically locates the file, compiles the class, and converts SOAP calls correctly into Java invocations of your service class. Try it out - there's a calculator client in samples/userguide/example2/CalcClient.java, which you can use like this:

```
% java samples.userguide.example2.CalcClient -p8080 add 2 5
Got result : 7
```

```
% java samples.userguide.example2.CalcClient -p8080 subtract 10 9
Got result : 1
%
```

(note that you may need to replace the "-p8080" with whatever port your J2EE server is running on)

El punt més important a tenir en compte per aquesta part és tenir el CLASSPATH correcte. Per evitar problemes la millor solució és:

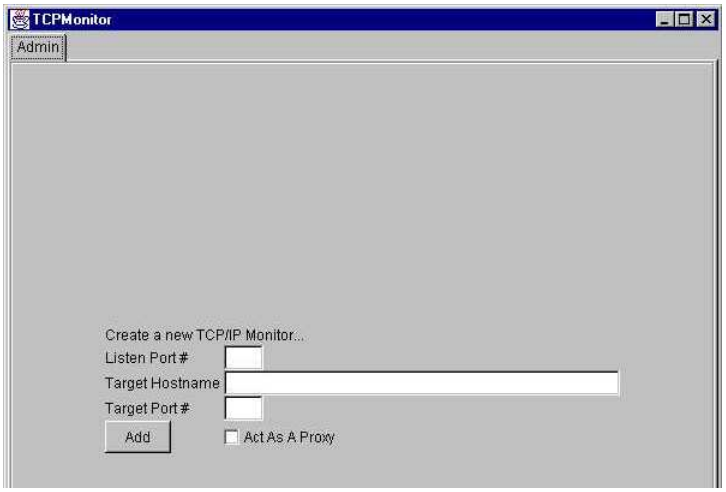
1. Configurar el CLASSPATH.
2. Al mateix terminal/consola que hagueu posat el CLASSPATH executeu el Tomcat per a que heredi les variables d'entorn.
3. Si continueu tenint problemes verifiqueu els fitxers de Log del Tomcat.

6. Monitoritzar petició i resposta: TCPMon

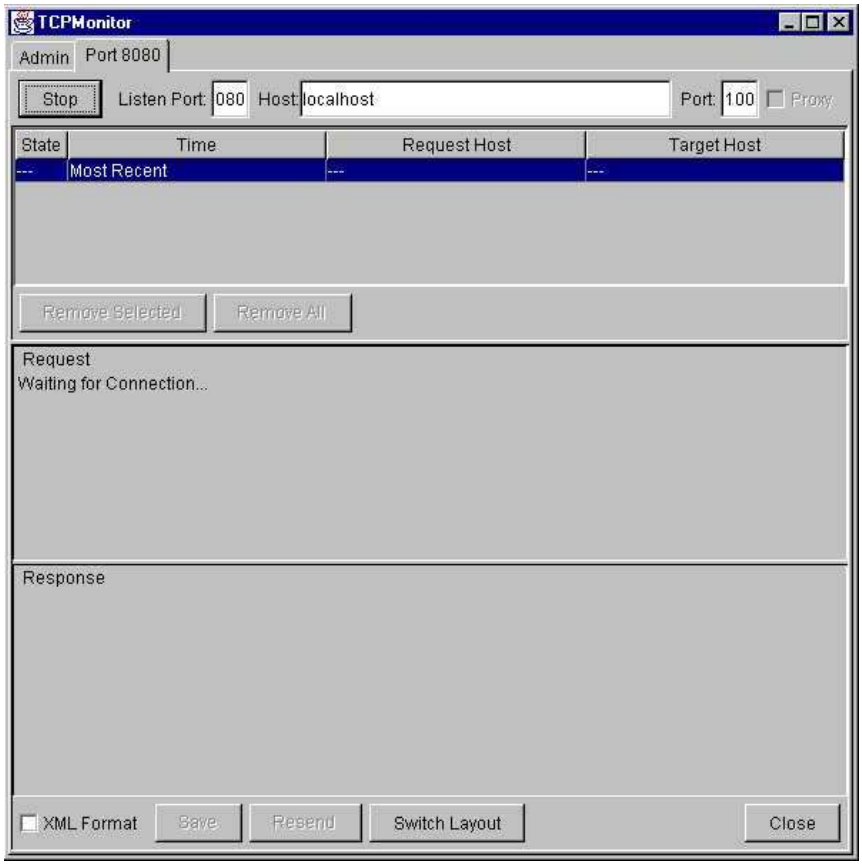
Per arrancar el monitor:

```
% java org.apache.axis.utils.tcpmon
```

(recordeu que heu de tenir el CLASSPATH tal com hem comentat en l'apartat anterior)



To use the program, you should select a local port which tcpmon will monitor for incoming connections, a target host where it will forward such connections, and the port number on the target machine which should be "tunneled" to. Then click "add". You should then notice another tab appearing in the window for your new tunneled connection. Looking at that panel, you'll see something like this:



Per a monitoritzar l'exemple anterior ho podem fer de dues maneres diferents:

Escoltant del port on hi ha el Tomcat. Això implica canviar el Tomcat d'adreça

1. Arrencar tomcat en un port diferent de 8080 (cal modificar el fitxer `server.xml` per a que arranqui d'un port diferent del 8080. P.ex. al port 11100)

```
    <!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8081 -->
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
port="11100"                minProcessors="5" maxProcessors="75"
               enableLookups="true" redirectPort="8443"
               acceptCount="10" debug="0" connectionTimeout="20000"
               useURIValidationHack="false" />
    <!-- Note : To disable connection timeouts, set connectionTimeout
value
     to -1 -->
```

2. Arrencar TCPMon de manera que:
   a. Que escolti del port 8080
   b. Que reenviï a localhost port 11100
3. Executar petició al port 8080:

```
java samples.userguide.example2.CalcClient -p8080 add 2 5
```

4. Al TCPMon pots veure la petició i la resposta en xml

Enviant peticions a un port diferent d'on hi ha el Tomcat i que el TCPMon reenviï la petició al port d'on escolta tomcat

1. Arrencar tomcat al port per defecte (8080)
2. Arrencar TCMon de manera que:
   a. Que escolti del port 9000

   b. Que reenviï a `localhost` port `8080`
3. Executar petició al port `9000`:

```
java samples.userguide.example2.CalcClient -p9000 add 2 5
```

4. Al TCPMon pots verure la petició i la resposta en xml

Construcció d'un Servei Web

Tot i que és molt senzill construir un servei web posant el codi com a .jws al directori d'Axis, necessitem altres maneres per a desenvolupar els nostres serveis web que ens permetin tenir més control sobre els serveis que oferim. Ara veurem una altra manera de fer-ho que ens ho permetrà.

A partir d'un codi que us donem, l'empaquetarem com a servei Web, i el desplegarem en un sistema Axis d'Apache. Un cop tinguem el servei al servidor, a partir només del WSDL, crearem Java stubs per a comunicar-nos amb el servidor (per si teniu interès ens saber més... l'WSDL serveix per a descriure serveis web d'una manera estructurada. Trobareu més informació a: http://www.w3.org/TR/wsdl).

Els passos que seguirem són:

1. Veure el codi que utilitzarem com exemple
2. *Java2WSDL*: generar el WSDL a partir de la interfície que proporcionem
3. *WSDL2Java*: generar l'empaquetament de la part de codi corresponent al servidor, i els stubs per a que els clients accedeixin al servei amb facilitat
4. `BotigaWebSoapBindingImpl.java`: Afegir el codi corresponent al nostre servei
5. *Deploy*: desplegar el servei a l'Axis d'Apache
6. Escriure un client que, usant l'stub generat, accedeixi amb facilitat al servei web.

Abans de començar amb l'exercici, recordeu que cal afegir al CLASSPATH el directori que utilitzeu com a base per l'exemple (el directori base del package).

1. Codi

Seguint el que hi ha a la sessió 4 de problemes, farem el codi corresponent a una botigaWeb. Tindrem dos fitxers: una interfície i la implementació d'una classe:

BotigaWeb.java

```
package botigaWeb;

public interface BotigaWeb {
    // Mètode que ens dóna el preu d'un producte
    public float veurePreu(String producte);
}
```

BotigaWebImpl.java

```
package botigaWeb;

public class BotigaWebImpl{
      float preu = 10.0F;

      public float veurePreu(String producte){
            return preu;
      }
}
```

2. Java2WSDL: generar el WSDL a partir de la interfície que proporcionem

Primer de tot cal que compileu el codi corresponent a la botiga (javac). Després utilitzarem Java2WSDL per a generar un fitxer WSDL corresponent a la interfície donada:

```
% java org.apache.axis.wsdl.Java2WSDL -o botigaWeb.wsdl -
l"http://localhost:8080/axis/services/botigaWeb" -n urn:botigaWeb -
p"botigaWeb" urn:botigaWeb botigaWeb.BotigaWeb
```

on:

- Nom del fitxer de sortida WSDL: botigaWeb.wsdl
- URL del servei web: `http://localhost:8080/axis/services/botigaWeb`
- Espai de noms destí per WSDL: `urn:botigaWeb`
- Map Java package = namespace: `botigaWeb = urn:botigaWeb`
- Classe principal: `botigaWeb.BotigaWeb`

3. WSDL2Java: generar l'empaquetat de la part de codi corresponent al servidor, i els stubs per a que els clients accedeixin al servei amb facilitat

Generarem aquest codi en un altre `package` (per mantenir-lo separat de l'original).

```
% java org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -p botigaWeb.ws
botigaWeb.wsdl
```

on:

- Directori base per la sortida: .
- Àmbit del desplegament (`Application, Request o Session`): `Session`
  - `Request` scope (default): will create a new object each time a SOAP request comes in for your service.
  - `Application` scope: will create a singleton shared object to service all requests.
  - `Session` scope: will create a new object for each session-enabled client who accesses your service.
- Activa la generació per la part del servidor : `s` (si estiguéssim accedint a un servei Web extern no ho hauríem de fer, ja que només necessitaríem l'stub del client)
- Package on posarem el codi: `botigaWeb.ws`
- Nom del fitxer WSDL: `botigaWeb.wsdl`

Després d'haver executat aquesta comanda s'ha generat bastant de codi al directori `botigaWeb/ws`:

- `BotigaWebSoapBindingImpl.java`: implementació del codi corresponent al nostre servei Web. Aquest és l'únic fitxer que haureu d'editar.
- `BotigaWeb.java`: interfície remota a la BotigaWeb (amplia Remote, i els mètodes de l'original BotigaWeb.java llencen RemoteExceptions).
- `BotigaWebService.java`: Interfície de Servei per al nostre servei Web. El ServiceLocator implementa aquesta interfície.
- `BotigaWebServiceLocator.java`: conté la classe que implementa la part del servei corresponent al client
- `BotigaWebSoapBindingSkeleton.java`: esquelet de la part corresponent al codi del servidor
- `BotigaWebSoapBindingStub.java`: codi stub del client que encapsula l'accés del client
- `deploy.wsdd`: descriptor de desplegament que passem a l'Axis per a desplegar aquests serveis.
- `undeploy.java`: descriptor de desplegament que ens permetrà desplegar el servei Web d'Axis.

4. `BotigaWebSoapBindingImpl.java`: Afegir el codi corresponent al nostre servei

Hem de modificar el fitxer `BotigaWebSoapBindingImpl.java` per a que implementi el nostre servei. (Les línies que hem afegit estan en negreta):

```
package botigaWeb.ws;

import botigaWeb.BotigaWebImpl;

public class BotigaWebSoapBindingImpl implements botigaWeb.ws.BotigaWeb{
            BotigaWebImpl botiga = new BotigaWebImpl();

  public float veurePreu(java.lang.String in0) throws
java.rmi.RemoteException
  {
     return botiga.veurePreu(in0);
  }
}
```

5. Deploy: desplegar el servei a l'Axis d'Apache

Compileu el codi del servei:

```
% javac botigaWeb/ws/*.java
```

Empaqueteu el codi:

```
% jar cvf botigaWeb.jar botigaWeb/*.class botigaWeb/ws/*.class
```

i poseul-lo a axis:

```
% mv botigaWeb.jar %TOMCAT_HOME%/webapps/axis/WEB-INF/lib
```

Desplegueu el servei Web:

```
% java org.apache.axis.client.AdminClient deploy.wsdd
```

En aquest punt ja tenim el servei corresponent a la Botiga Web executant-se al servidor.

6. Escriure un client que, usant l'stub generat, accedeixi amb facilitat al servei web.

Per a provar el servei, crearem un client que faci una consulta de preu:

```
package botigaWeb;

public class BotigaWebTester{

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

      // crear un servei
      botigaWeb.ws.BotigaWebService service = new
         botigaWeb.ws.BotigaWebServiceLocator();

      // usar el servei per a obtenir un stub del servei
      botigaWeb.ws.BotigaWeb botiga = service.getbotigaWeb();

      // cridar el servei
      System.out.println("Preu: " + botiga.veurePreu("aa"));
   }
}
```

Ara ja només queda compilar i executar la prova!!!

FEINA A FER:

Afegiu a la classe BotigaWeb una operació per a comprar un producte:

```
public float comprarProducte(int unitats, String producte);
```

Donada una quantitat de productes ens diu quan ens costarà.

Un cop afegida aquesta operació al servei web, executeu una vegada l'operació de comprar producte i intercepteu-ne la petició i resposta.

Comprimit en un zip heu de lliurar:

1. Tot el codi generat (tant el codi vostre com el que genera l'Axis) per tal que d'oferir el servei Web de la botigaWeb.
2. La petició i la resposta interceptades.

Enviament solució

Pugeu la resposta a les qüestions plantejades, l'informe i els fitxers indicats anteriorment al BSCW.

Bibliografia

- Axis: http://xml.apache.org/axis/
- Axis User's Guide: http://xml.apache.org/axis/
- Creating Web Services with Apache Axis: http://www.onjava.com/lpt/a//onjava/2002/06/05/axis.html
- Axis: The next generation of Apache Soap: http://www.javaworld.com/javaworld/jw-01-2002/jw-0125-axis.html
- Axis Installation Guide: http://xml.apache.org/axis/