Applicazioni Java – WS con Axis

Laurea Specialistica in Ingegneria Informatica per la Gestione d'Azienda
Corso di Sistemi Informativi per le Aziende

sistema di tracciabilità agroalimentare

Ing. Mario G.C.A. Cimino

Dicembre 2004

Installazione di Axis su Jakarta Tomcat

- ✓ Axis (http://ws.apache.org/axis/) è un SOAP engine, un framework per costruire SOAP processor (come client, server, gateway...)
- ✓ È sviluppato come java servlet, per cui si installa su servlet engine come Tomcat. Il package (http://apache.fis.uniroma2.it/ws/axis/1_2RC1/axis-1_2RC1-bin.zip) non contiene un file formato web appication (axis.war) ma si installa automaticamente copiando la cartella *webapps\axis* nella corrispondente cartella *webapps* di Tomcat.
- ✓ Non e` necessario riavviare Tomcat: in pochi secondi viene rilevata automaticamente l'applicazione e stampati i relativi messaggi sulla shell, tra cui la mancanza di alcuni package da aggiungere manualmente come di seguito.
- ✓ Copiare i 4 files *.jar* dell'XML parser della *Xerces* nella cartella *axis/WEB-INF/lib*
- ✓ Accedere mediante il browser alla url http://localhost:8080/axis/ quindi eseguire il test di validazione *happyaxis* (http://localhost:8080/axis/happyaxis.jsp) cliccando sul link *Validate*.
- ✓ Normalmente, viene rilevata la mancanza degli ulteriori packages descritti di seguito, da copiare sempre nella cartella *axis/WEB-INF/lib*.

PACKAGE	CONTENUTO NEL FILE	SCARICABILE DA
activation.jar	jaf-1_0_2-upd.zip	http://java.sun.com/products/javabeans/glasgow/jaf.html
mail.jar	javamail-1_3_2.zip	http://java.sun.com/products/javamail/downloads/index.html
xmlsec.jar	xml-security-bin-1_1_0.zip	http://xml.apache.org/security/dist/java-library/xml-security-bin-1_1_0.zip

- ✓ Successivamente si testa un SOAP endpoint. Sebbene lo standard SOAP usi richieste *HTTP POST* per inviare una richiesta XML ad un endpoint, Axis supporta anche un elementare meccanismo *HTTP GET*, utile a scopo di testing. Accedere da browser alla url http://localhost:8080/axis/services/Version?method=getVersion. Dovrebbe apparire qualcosa del genere:

```
<?xml version="1.0" encoding="UTF-8" ?>
<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>
 <getVersionResponse
  soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <getVersionReturn
   xsi:type="xsd:string">
   Apache Axis version: 1.1 Built on Apr 04, 2003 (01:30:37 PST)
  </getVersionReturn>
 </getVersionResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

- ✓ Testiamo ora un Java Web Service (JWS) Endpoint. JWS è un meccanismo che permette di generare un WS Server automaticamente da codice Java. Accedere da

browser alla url http://localhost:8080/axis/EchoHeaders.jws?method=list. Dovrebbe apparire qualcosa di simile:

```
<?xml version="1.0" encoding="UTF-8" ?>
<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>
  <listResponse
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <listReturn xsi:type="soapenc:Array"
    soapenc:arrayType="xsd:string[6]"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <item>accept:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*</item>
    <item>accept-language:en-us</item>
    <item>accept-encoding:gzip, deflate</item>
    <item>user-agent:Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)</item>
    <item>host:localhost:8080</item>
    <item>connection:Keep-Alive</item>
   </listReturn>
  </listResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Generazione di un WS tramite JWS

✓ I file *.jws* sono file *.java* con diversa estensione, che vengono riconosciuti da *Axis* e compilati al primo accesso. Scrivere la seguente classe Java e copiarla nella cartella *webapps/axis*.

```
// calcolatore.jws

public class Calcolatore {
  public int somma(int i1, int i2) {
    return i1 + i2;
  }

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

✓ Accedere da browser alla url http://localhost:8080/axis/Calcolatore.jws.Axis compila automaticamente la classe e converte le chiamate SOAP in invocazioni di metodo della classe. Il file *Calcolatore.class* verrà posto in *axis/WEB-INF/jwsClasses*.

✓ E` possibile esaminare il wsdl su http://localhost:8080/axis/Calcolatore.jws?wsdl

```
<?xml version="1.0" encoding="UTF-8" ?>
  <wsdl:definitions targetNamespace="http://localhost:8080/axis/Calcolatore.jws"
      xmlns:apachesoap="http://xml.apache.org/xml-soap"
      xmlns:impl="http://localhost:8080/axis/Calcolatore.jws"
      xmlns:intf="http://localhost:8080/axis/Calcolatore.jws"
```

```
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!-- WSDL created by Apache Axis version: 1.2RC2 Built on Nov 16, 2004 (12:19:44 EST)  -->
<wsdl:message name="sommaRequest">
  <wsdl:part name="i1" type="xsd:int" />
  <wsdl:part name="i2" type="xsd:int" />
</wsdl:message>
<wsdl:message name="sottrazioneResponse">
  <wsdl:part name="sottrazioneReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="sommaResponse">
  <wsdl:part name="sommaReturn" type="xsd:int" />
</wsdl:message>
<wsdl:message name="sottrazioneRequest">
  <wsdl:part name="i1" type="xsd:int" />
  <wsdl:part name="i2" type="xsd:int" />
</wsdl:message>
<wsdl:portType name="Calcolatore">
  <wsdl:operation name="somma" parameterOrder="i1 i2">
    <wsdl:input message="impl:sommaRequest" name="sommaRequest" />
    <wsdl:output message="impl:sommaResponse" name="sommaResponse" />
  </wsdl:operation>
  <wsdl:operation name="sottrazione" parameterOrder="i1 i2">
    <wsdl:input message="impl:sottrazioneRequest" name="sottrazioneRequest" />
    <wsdl:output message="impl:sottrazioneResponse" name="sottrazioneResponse" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CalcolatoreSoapBinding" type="impl:Calcolatore">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="somma">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="sommaRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                     namespace="http://DefaultNamespace" use="encoded" />
    </wsdl:input>
    <wsdl:output name="sommaResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                     namespace="http://localhost:8080/axis/Calcolatore0u106 ?ws" use="encoded" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="sottrazione">
    <wsdlsoap:operation soapAction="" />
```

```
        <wsdl:input name="sottrazioneRequest">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="http://DefaultNamespace" use="encoded" />
        </wsdl:input>
        <wsdl:output name="sottrazioneResponse">
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                         namespace="http://localhost:8080/axis/Calcolatore.jws" use="encoded" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CalcolatoreService">
      <wsdl:port binding="impl:CalcolatoreSoapBinding" name="Calcolatore">
        <wsdlsoap:address location="http://localhost:8080/axis/Calcolatore.jws" />
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

- ✓ Per testare il funzionamento del nostro WS Server, sviluppiamo un WS Client come di seguito:

```
// CalcolatoreClient.java

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.rpc.ParameterMode;

public class CalcolatoreClient
{
   public static void main(String [] args) throws Exception {
        Options options = new Options(args);

        String endpoint = "http://localhost:" + options.getPort() +
                          "/axis/Calcolatore.jws";

        args = options.getRemainingArgs();

        if (args == null || args.length != 3) {
            System.err.println("Uso: CalcolatoreClient <somma|sottrazione> arg1 arg2");
```

```
            return;
        }

        String method = args[0];
        if (!(method.equals("somma") || method.equals("sottrazione"))) {
            System.err.println("Uso: CalcolatoreClient <somma|sottrazione> arg1 arg2");
            return;
        }

        Integer i1 = new Integer(args[1]);
        Integer i2 = new Integer(args[2]);

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

        call.setTargetEndpointAddress( new java.net.URL(endpoint) );
        call.setOperationName( method );
        call.addParameter( "op1", XMLType.XSD_INT, ParameterMode.IN );
        call.addParameter( "op2", XMLType.XSD_INT, ParameterMode.IN );
        call.setReturnType( XMLType.XSD_INT );

        Integer ret = (Integer) call.invoke( new Object [] { i1, i2 });

        System.out.println("Risultato : " + ret);
    }
}
```

✓ Il seguente script provvere a compilare ed eseguire il client.

```
rem make.bat
@echo off
set CLASSPATH=
setlocal
set lib=C:\tomcat5\webapps\axis\WEB-INF\lib\
set libs=%lib%axis.jar;%lib%jaxrpc.jar;%lib%saaj.jar;%lib%commons-logging.jar;%lib%wsdl4j.jar;
         %lib%commons-discovery.jar; %lib%activation.jar;%lib%mail.jar;
javac -classpath "%libs%" *.java
java -classpath "%libs%" CalcolatoreClient -p8080 somma 3 7
java -classpath "%libs%" CalcolatoreClient -p8080 sottrazione 9 5
endlocal
```

```
@echo on
```

✓ Producendo il seguente risultato

```
C:myws> make
Risultato : 10
Risultato : 4
```

Generazione di un WS tramite Java2WSDL

✓ Java2WSDL permette la creazione di WS con maggior flessibilità in fase di sviluppo, mediante il formato *Web Service Deployment Descriptor* (WSDD).

✓ Supponiamo di avere i tre file di testo seguenti:

```
C:myws\make.bat
C:myws\CalcolatoreClient2.java
C:myws\samples\calcolatore\Calcolatore.java
```

```
// Calcolatore.java
package samples.calcolatore;

public interface Calcolatore {
    public int somma(int i1, int i2);
    public int sottrazione(int i1, int i2);
}
```

✓ Mostreremo man mano il file *make.bat*. Prima di tutto, compiliamo il codice java:

```
rem make.bat
@echo off
set CLASSPATH=
```

```
setlocal
set lib=C:\tomcat5\webapps\axis\WEB-INF\lib\
set libs=%lib%axis.jar;%lib%jaxrpc.jar;%lib%saaj.jar;%lib%commons-logging.jar;%lib%wsdl4j.jar;
         %lib%commons-discovery.jar;%lib%activation.jar;%lib%mail.jar;
javac -classpath "%libs%" samples/calcolatore/*.java
```

- ✓ Quindi si crea il WSDL usando java2SWDL, con le seguenti opzioni: -o *<nome del file wsdl generato>*, -l *<collocazione del servizio>*, -n *<namespace degli elementi del wsdl>* *-p<namespace e package associato>*

```
set service_location=http://localhost:8080/axis/services/Calcolatore
set target_namespace=urn:calcolatore
set package=samples.calcolatore

java -classpath "%libs%" org.apache.axis.wsdl.Java2WSDL -o Calcolatore.wsdl
     -l"%service_location%" -n"%target_namespace%" -p"%package%" "%target_namespace%"
     samples.calcolatore.Calcolatore
```

- ✓ Viene generato un file `C:myws\Calcolatore.wsdl` dal seguente contenuto:

```
<?xml version="1.0" encoding="UTF-8"?>
  <wsdl:definitions targetNamespace="urn:calcolatore"
                 xmlns:impl="urn:calcolatore"
                 xmlns:intf="urn:calcolatore"
                 xmlns:apachesoap="http://xml.apache.org/xml-soap"
                 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <!--WSDL created by Apache Axis version: 1.2RC2 Built on Nov 16, 2004 (12:19:44 EST)-->
   <wsdl:message name="sommaResponse">
      <wsdl:part name="sommaReturn" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="sottrazioneRequest">
      <wsdl:part name="in0" type="xsd:int"/>
      <wsdl:part name="in1" type="xsd:int"/>
```

```
</wsdl:message>
<wsdl:message name="sottrazioneResponse">
   <wsdl:part name="sottrazioneReturn" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="sommaRequest">
   <wsdl:part name="in0" type="xsd:int"/>
   <wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:portType name="Calcolatore">
   <wsdl:operation name="somma" parameterOrder="in0 in1">
      <wsdl:input name="sommaRequest" message="impl:sommaRequest"/>
      <wsdl:output name="sommaResponse" message="impl:sommaResponse"/>
   </wsdl:operation>
   <wsdl:operation name="sottrazione" parameterOrder="in0 in1">
      <wsdl:input name="sottrazioneRequest" message="impl:sottrazioneRequest"/>
      <wsdl:output name="sottrazioneResponse" message="impl:sottrazioneResponse"/>
   </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CalcolatoreSoapBinding" type="impl:Calcolatore">
   <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="somma">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="sommaRequest">
         <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:calcolatore"/>
      </wsdl:input>
      <wsdl:output name="sommaResponse">
         <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:calcolatore"/>
      </wsdl:output>
   </wsdl:operation>
   <wsdl:operation name="sottrazione">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="sottrazioneRequest">
         <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:calcolatore"/>
      </wsdl:input>
      <wsdl:output name="sottrazioneResponse">
         <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                        namespace="urn:calcolatore"/>
      </wsdl:output>
   </wsdl:operation>
</wsdl:binding>
<wsdl:service name="CalcolatoreService">
```

```
      <wsdl:port name="Calcolatore" binding="impl:CalcolatoreSoapBinding">
         <wsdlsoap:address location="http://localhost:8080/axis/services/Calcolatore"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

✓ Si usa il WSDL generato per creare tutti i componenti necessari, con WSDL2Java:

```
java -classpath "%libs%" org.apache.axis.wsdl.WSDL2Java -o . -d Session -s -S true
     -N"%target_namespace%" samples.calcolatore Calcolatore.wsdl
```

✓ Vengono generati i file seguenti:

C:myws\samples\calcolatore\Calcolatore.class	
C:myws\samples\calcolatore\Calcolatore.java	interfaccia di partenza resa remota con RMI
C:myws\samples\calcolatore\CalcolatoreService.java	interfaccia di servizio lato client
C:myws\samples\calcolatore\CalcolatoreServiceLocator.java	implementazione del servizio lato client
C:myws\samples\calcolatore\CalcolatoreSoapBindingImpl.java	implementazione di default del server (completare)
C:myws\samples\calcolatore\CalcolatoreSoapBindingSkeleton.java	skeleton lato server
C:myws\samples\calcolatore\CalcolatoreSoapBindingStub.java	stub lato client
C:myws\samples\calcolatore\deploy.wsdd	descrittore di deployment
C:myws\samples\calcolatore\undeploy.wsdd	descrittore di undeployment

a noi interessa solo `CalcolatoreSoapBindingImpl.java`: occorre semplicemente implementare i corpi dei metodi, in cui viene ritornato un valore di default -3.

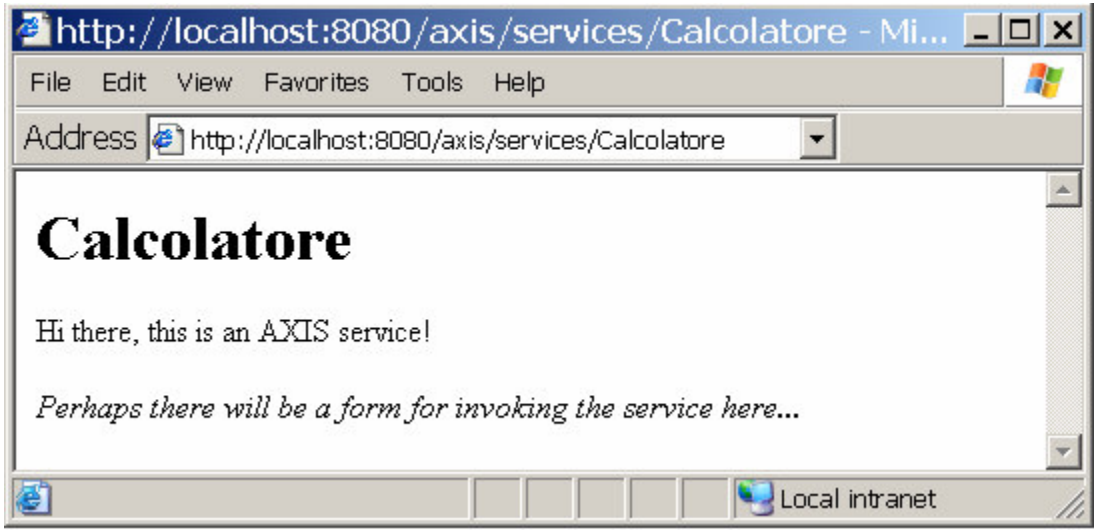
```
// CalcolatoreSoapBindingImpl.java

package samples.calcolatore;

public class CalcolatoreSoapBindingImpl implements samples.calcolatore.Calcolatore {
    public int somma(int in0, int in1) throws java.rmi.RemoteException {
        return -3;
    }

    public int sottrazione(int in0, int in1) throws java.rmi.RemoteException {
```

```
            return -3;
        }
    }
```

- ✓ Fatto ciò, si può fare il deployment del servizio mediante l'applicazione *AdminClient* e compilare le classi generate da *WSDL2Java* collocando i file *.class* nella cartella *WEB-INF\classes* di Axis.

```
java -classpath "%libs%" org.apache.axis.client.AdminClient samples\calcolatore\deploy.wsdd
set deploydir=C:\tomcat5\webapps\axis\WEB-INF\classes
javac -classpath "%libs%" samples/calcolatore/*.java -d %deploydir%
```

- ✓ Accedendo da browser alla url http://localhost:8080/axis/services/Calcolatore compare un messaggio del genere:



- ✓ Questo indica che il server è attivo. Analogamente a quanto fatto per l'endpoint jws, aggiungendo alla fine della URL il suffisso *?wsdl* Axis genera la descrizione wsdl del servizio che può essere salvata da chi deve sviluppare il client. Come client possiamo adoperare quello visto per il meccanismo jws, rinominando la classe come *CalcolatoreClient2* e modificando solo l'endpoint nel seguente modo:

```
String endpoint = "http://localhost:" + options.getPort() +
                  "/axis/services/Calcolatore";
```

- ✓ Fatto ciò si può ricompilare e rieseguire il client, ottenendo ovviamente i medesimi risultati del server jws. Ecco l'ultima parte del file *make.bat*:

```
javac -classpath "%libs%" *.java
java -classpath "%libs%" CalcolatoreClient2 -p8080 somma 3 7
java -classpath "%libs%" CalcolatoreClient2 -p8080 sottrazione 9 5
endlocal
@echo on
```