http://www.jugsardegna.org

Java Web Services con Axis

I Parte

di

Nicola Mura

nikhol@tiscali.it

Introduzione

Si assiste negli ultimi anni al diffondersi crescente delle tecnologie e approcci Web Services. E' inoltre sempre più facile impiegarli grazie anche ai numerosi tool che in vari linguaggi di programmazione permettono di implementare e "deployare" servizi senza nemmeno rendersi conto della presenza di SOAP (e HTTP) nel livello sottostante. Per quanto riguarda Java sono disponibili vari strumenti di sviluppo di WS, tra cui i tool open source Java Web Services Development Pack (JWSDP) e Apache Axis.
Verrà preso in considerazione Axis appartenente alla famiglia dei progetti sviluppati dall'Apache Software Foundation. In questo e nei prossimi articoli verranno illustrati degli esempi che gradualmente aiuteranno il lettore ad addentrarsi nel mondo dei Web services.

Web Services

Si tratta di un caso particolare dell'Architettura Service Oriented (SOA) dedicata al mondo delle tecnologie e standard Web. Un Web Service è una risorsa computazionale erogata da un particolare service provider, il quale espone anche una sua rappresentazione univoca e standardizzata tramite il linguaggio WSDL (Web Service Description Language). In base a questa interfaccia il servizio può essere richiesto anche dinamicamente da un client ad hoc senza vincoli sulle tecnologie impiegate. La comunicazione può avvenire secondo i modelli RPC e messaging attraverso il protocollo SOAP, ovvero uno standard basato su XML che riveste i canali reali di comunicazione in rete (HTTP, HTTPS, SMTP, FTP).

Axis

Brevemente, si tratta di un'API di programmazione e deployment di Web services che permette di lavorare ad un livello di astrazione elevato, evitando così di dover maneggiare direttamente l'envelope SOAP. Con Axis è possibile, dunque, implementare Web services e anche sviluppare client di servizi di terzi. Per semplicità, negli esempi che tratteremo viene esposto solo l'aspetto di funzionalità RPC e non quello relativo ai vari approcci di messaging supportati da Axis. Ciò è comunque in linea con la maggior parte della casistica Web services che si basa su meccanismi request/response a dispetto del modello asincrono.

Installazione di Axis

Dal sito ufficiale (1) è possibile scaricare il file di installazione. All'interno di questo, oltre alla documentazione, esempi, etc., è presente la cartella *axis* (sotto la cartella *webapps*) che contiene tutte le risorse necessarie al funzionamento di Axis. Essa va inserita nel Servlet Engine di cui si dispone; in questo articolo utilizzeremo il classico e diffusissimo Apache Tomcat (2).
In questo caso, dunque, la cartella va brutalmente copiata sotto la *webapps* di Tomcat. In questo modo avremo dotato il nostro server di un Web service engine (di fatto una struttura Servlet) in grado di esaudire richieste provenienti da qualsiasi client WS.

L'esempio (un semplice Web service)

Il primo esempio che verrà illustrato riguarda l'implementazione di un semplice Web services che restituisce un messaggio di saluto.
Dal **listato 1** si può notare la semplicità di questo servizio rappresentato dal solo metodo *saluto()*. Il parametro richiesto, impiegato per comporre il messaggio, e quello restituito sono entrambi di tipo *String*.

Listato 1 – Classe SalutoWS

```
package miopackage;

public class SalutoWS {
  public String saluto(String nome) {
    return "Ciao "+nome+"!";
  }
}
```

Una volta compilata la classe dovrà essere disposta secondo la configurazione:

<TOMCAT_HOME>\webapps\axis\WEB-INF\classes\miopackage\SalutoWS.class

A questo punto, Axis non è ancora "cosciente" del fatto che deve impiegare tale classe per l'esposizione di un servizio. Perciò è necessario effettuare il deployment del servizio. In questo modo si istruisce l'engine di Axis affinché in corrispondenza di una determinata richiesta (sottoforma di messaggio SOAP), istanzi la classe e ne richiami il relativo metodo.
In generale "deployare" un servizio richiede la stesura di un file xml di deployment (con estensione *wsdd*: Web Service Deployment Description); per la nostra classe tale file è definito nel **listato 2.**

Listato 2 – deploy1.wsdd

```
<deployment
  xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="urn:SalutoWS" provider="java:RPC">
    <parameter name="className" value="miopackage.SalutoWS"/>
    <parameter name="allowedMethods" value="saluto"/>
    <parameter name="scope" value="Request"/>
  </service>
</deployment>
```

Soffermiamoci un attimo su ciò che è stato inserito all'interno dei tag *<service></services>*. l'attributo *name* rappresenta il nome che noi vogliamo assegnare al servizio e che servirà ad identificarlo univocamente. L'attributo *provider* è stato valorizzato con *"java:RPC"* che indica ad Axis di pubblicare il servizio secondo un meccanismo RPC. Di seguito troviamo una sezione composta da tre tag *<parameter>*. Il primo indica il nome (comprensivo di package) della classe da noi implementata. Il secondo descrive i metodi della suddetta classe che si vuole esporre (nel nostro caso l'unico metodo implementato). Il terzo definisce il ciclo di vita della classe che in questo caso viene istanziata, utilizzata e distrutta ad ogni richiesta (altri valori possibili sono *Application* e *Session*).

Per Lanciare il deployment occorre, innanzi tutto, inserire nel classpath le librerie:

- *axis.jar*
- *jaxrpc.jar*
- *saaj.jar*
- *commons-logging.jar*
- *commons-discovery.jar*
- *log4j-1.2.8.jar*

che sono tutte reperibili dentro la distribuzione di Axis.
A questo punto occorre eseguire Tomcat e il comando descritto nel **listato 3**:

Listato 3 – Istruzione di deployment del servizio

```
java org.apache.axis.client.AdminClient deploy1.wsdd
```

La classe *AdminClient* eseguirà il deployment leggendo i metadati dal file *deploy1.wsdd* indicato e comunicandoli automaticamente all'Axis engine. Verifica del buon esito di questa operazione, è controllare che il servizio sia presente nell'elenco di quelli "deployati" aprendo un browser all'URL:

http://localhost:8080/axis/servlet/AxisServlet

Verrà visualizzata la seguente pagina:

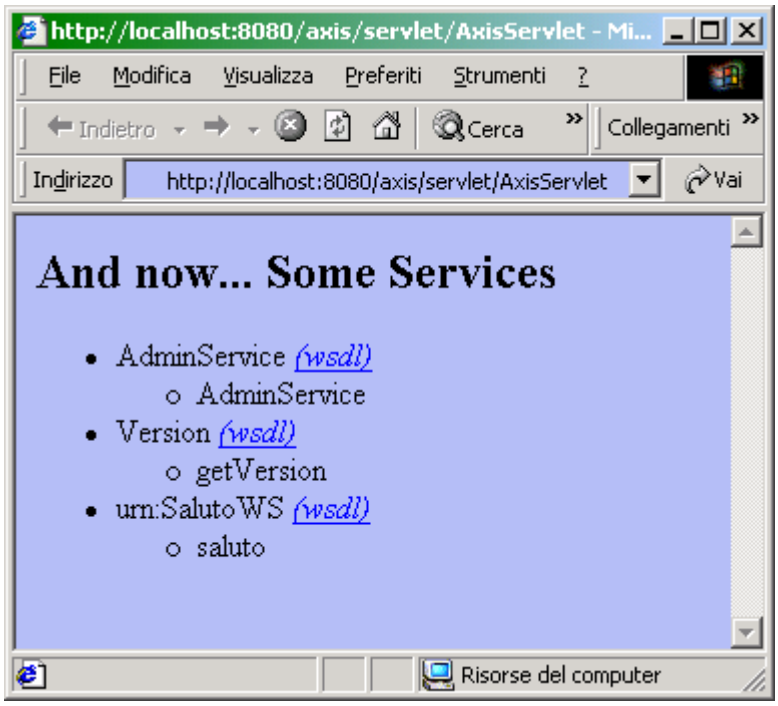

Figura 1 – Elenco dei WS" deployati"

Si può notare come sia realmente presente il servizio sotto il nome da noi assegnato nel file *deploy1.wsdd*. A lato e tra parentesi è presente il link *wsdl*; cliccando su di esso verrà visualizzato il listato WSDL (Web Service Descritpion Language). Infatti, durante la fase di deployment Axis genera automaticamente la rappresentazione WSDL del nostro servizio. In base a questo un ipotetico utilizzatore è in grado di realizzare un client ad hoc.

Per testare il corretto funzionamento del servizio occorre implementare un client. In pratica si potrà scegliere il linguaggio che si vuole per richiamare il Web service appena pubblicato. Axis mette a disposizione alcuni strumenti per effettuare chiamate RPC su SOAP in base alle specifiche definite dall'API JAX-RPC. Noi sceglieremo l'approccio DII (Dynamic Invocation Interface) tramite il quale risulterà assai semplice realizzare il client per il WS appena pubblicato. Nel **Listato 4** è illustrato il codice del client Axis (il classpath è quello visto sopra).

Listato 4 – Classe ClientSalutoWS

```
import java.net.*;
import java.rmi.*;

import javax.xml.namespace.*;
import javax.xml.rpc.*;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;

public class ClientSalutoWS {
  public static void main(String[] args) {
    String messaggio = "";
    try {
      Call call = (Call) new Service().createCall();
      call.setTargetEndpointAddress(new
           URL("http://localhost:8080/axis/services/"));
      call.setOperationName(new QName("urn:SalutoWS", "saluto"));
      Object rispostaWS = call.invoke(new Object[]{"Nicola"});
      messaggio = "il Web service ha risposto: "+(String) rispostaWS;
    }
    catch (MalformedURLException ex) {
      messaggio = "errore: l'url non è esatta";
    }catch (ServiceException ex) {
      messaggio = "errore: la creazione della chiamata è fallita";
    }catch (RemoteException ex) {
      messaggio = "errore: l'invocazione del WS è fallita";
    }finally{
      System.out.println(messaggio);
    }
  }
}
```

Si può notare come venga impiegata la classe *Call* di Axis per impostare la chiamata; dotandola delle informazioni necessarie al raggiungimento del nostro specifico WS. Viene, dapprima settato l'endpoint http, che in base alla nostra configurazione di Axis su Tomcat deve essere:

http://localhost:8080/axis/services,

infine, il nome del WS e del metodo da richiamare che sono rispettivamente *urn:SalutoWS* e *saluto.*
La RPC prende vita per effetto dell'istruzione *call.invoke()* cui passiamo il parametro *Nicola* che sarà a sua volta impiegato dal WS. Se nessuna delle tre eccezioni previste viene lanciata viene stampato il messaggio di risposta del WS:

il Web service ha risposto: Ciao Nicola!

In pratica il livello SOAP risulta totalmente trasparente al programmatore.
Di seguito è mostrato lo scambio di messaggi registrato dal TCPMonitor di Axis (descritto ampiamente nella documentazione del sito ufficiale (1)).

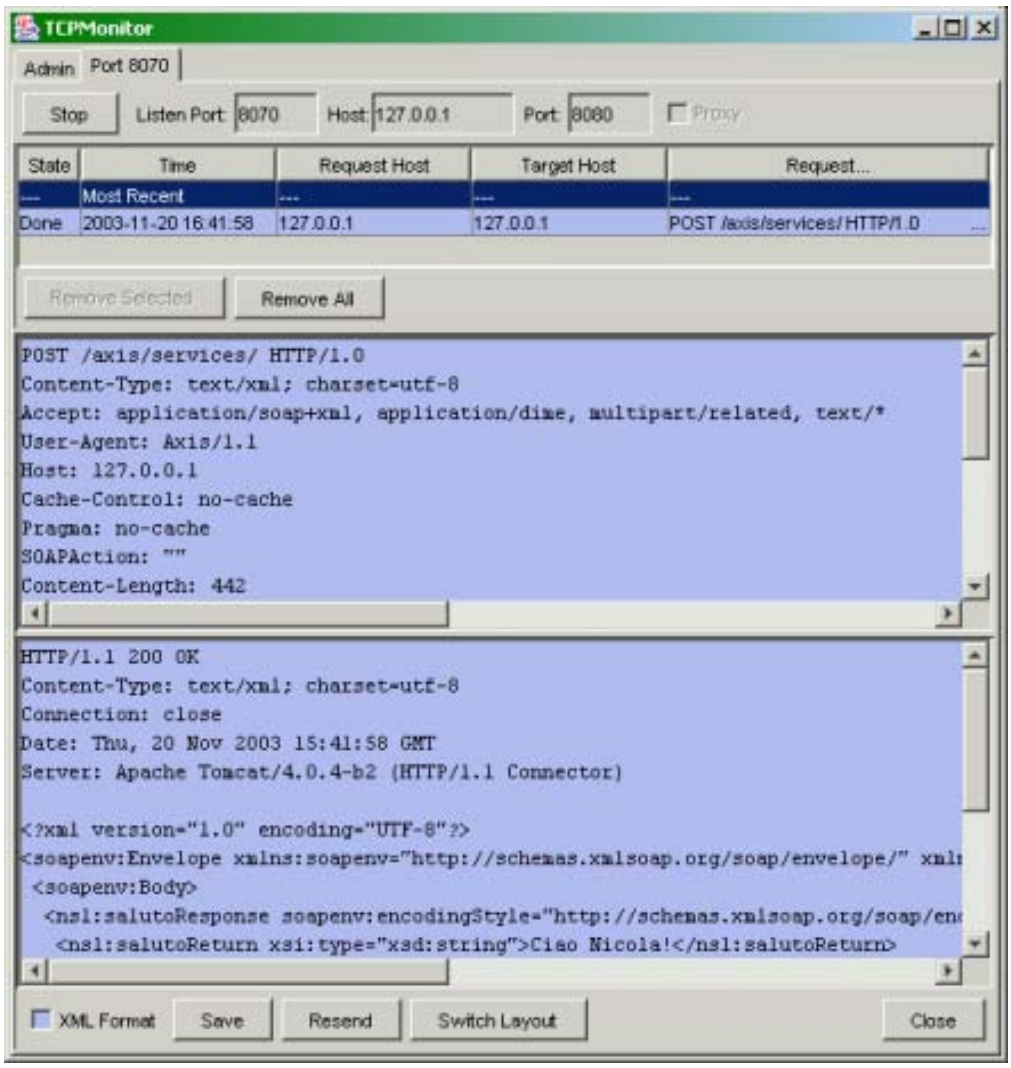

Figura 2 – TCPMonitor di Axis

Il messaggio di chiamata remota del WS così come viene visualizzato dal TCPMonitor ha il seguente aspetto:

Listato 5 – Messaggio Soap di chiamata al WS

```
POST /axis/services/ HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: 127.0.0.1
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 442

<?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>
```

```
    <ns1:saluto soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
         xmlns:ns1="urn:SalutoWS">
      <ns1:arg0 xsi:type="xsd:string">Nicola</ns1:arg0>
    </ns1:saluto>
  </soapenv:Body>
</soapenv:Envelope>
```

Mentre il messaggio di risposta al client risulta essere il seguente:

Listato 6 – Messaggio Soap di risposta del WS

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Connection: close
Date: Fri, 21 Nov 2003 16:10:50 GMT
Server: Apache Tomcat/4.0.4-b2 (HTTP/1.1 Connector)

<?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>
    <ns1:salutoResponse
         soapenv:encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
         xmlns:ns1="urn:SalutoWS">
      <ns1:salutoReturn xsi:type="xsd:string">Ciao Nicola!</ns1:salutoReturn>
    </ns1:salutoResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

In neretto sono visibili il parametro inviato al WS nel **Listato 5** e la stringa di risposta nel **listato 6** all'interno del tag *<ns1:salutoReturn>*.

Conclusioni

In questo articolo è stato affrontato il tema Web services che rappresenterà, in futuro, una realtà sempre più presente e di carattere strategico nell'universo dei sistemi Web based.
In questo semplice esempio, sono state fornite le basi per l'impiego di Axis nello sviluppo di applicazioni basate sui Web Services al fine di rendere più facile la comprensione del materiale, ufficiale e non, reperibile sulla rete.

Bibliografia

(1) David A. Chappell & Tyler Jewell, "Java Web Services", O'RELLY HOPS, 2002

Riferimenti

(1) Apache Axis: http://ws.apache.org/axis/
(2) Apache Tomcat: http://jakarta.apache.org/tomcat/