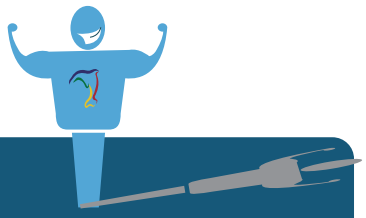
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

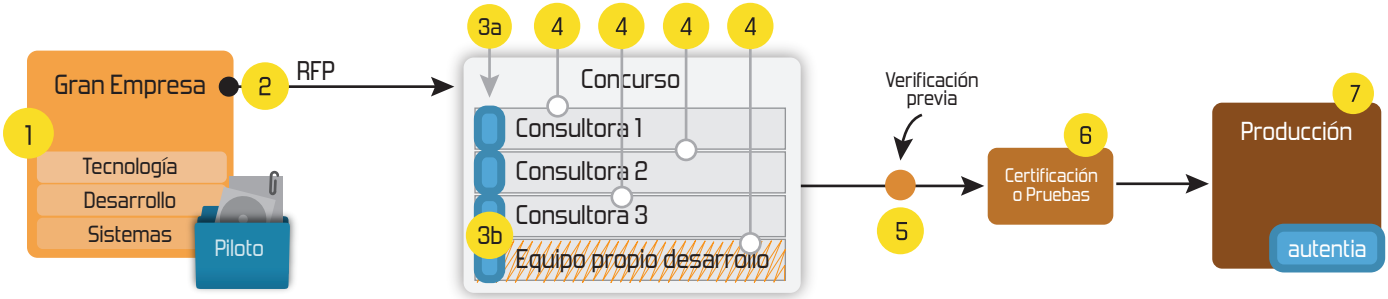
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Fecha de creación del tutorial: 2008-10-24

Web Services con Axis2. Configuración y ejemplo

Introducción

Este es un tutorial básico que introduce a los servicios web en Java, y muestra cómo configurar el equipo con las herramientas necesarias para poder crear luego un web service de ejemplo.

Utilizaremos Axis2 para publicar el servicio web, y Eclipse como herramienta para facilitarnos el proceso, no obstante el código es simple y podría hacerse todo por línea de comandos. Puedes mirar la sección Tutoriales Recomendados, al final de esta página, para profundizar y seguir aprendiendo sobre el tema (o un buen libro, ¡por supuesto!).

Requisitos

Utilizaremos el siguiente software:

- Servidor web Apache Tomcat 6.0.18, que podremos descargar desde http://tomcat.apache.org/download-60.cgi, en su versión *Binary Distributions | Core | zip*.
- Motor de servicios web Apache Axis2 1.4.1, disponible en http://ws.apache.org/axis2/download/1_4_1/download.cgi. Necesitaremos:
    - La distribución *Standard Binary Distribution | zip*.
    - La distribución *WAR (Web Archive Distribution) | zip*.
- Distribución de Java, en versión JDK 5.0 Update 16: http://java.sun.com/javase/downloads/index_jdk5.jsp.
- IDE Eclipse 3.4 (Ganymede, y edición JEE), obtenemos su última versión publicada en http://www.eclipse.org/downloads/.

Con lo anterior aseguro un correcto funcionamiento, pero si quieres utilizar otras versiones, adelante.

Configuración del software

Lo explicaré para las rutas del software que manejo en mi ordenador (obviamente sólo son orientativas, adáptalas en tu caso a tu sistema operativo y estructura de directorios).

1.- Sun Java JDK 5.0 update 16

Realizaremos la instalación y estableceremos en nuestro sistema la variable JAVA_HOME. En mi equipo en concreto es:

```
JAVA_HOME=C:\Herramientas\Java\jdk1.5.0_16
```

2.- Apache Tomcat 6.0.18

Simplemente descomprimimos el Tomcat en una carpeta, y establecemos la variable CATALINA_HOME a la ruta absoluta del directorio descomprimido. En mi caso es:

```
CATALINA_HOME=C:\TutorialWS\apache-tomcat-6.0.18
```

3.- Apache Axis2 1.4.1

Descomprimimos la versión zip de la distribución WAR, y el fichero *axis2.war* lo vamos a copiar en

```
C:\TutorialWS\apache-tomcat-6.0.18\webapps
```

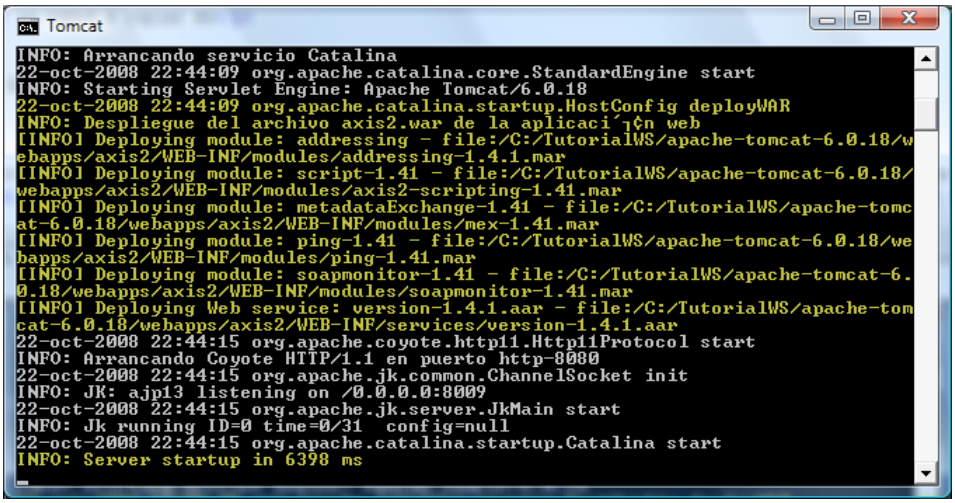
Asimismo descomprimimos la distribución binaria en una carpeta, y establecemos la variable AXIS2_HOME a la ruta absoluta del directorio descomprimido:

```
AXIS2_HOME=C:\TutorialWS\axis2-1.4.1
```

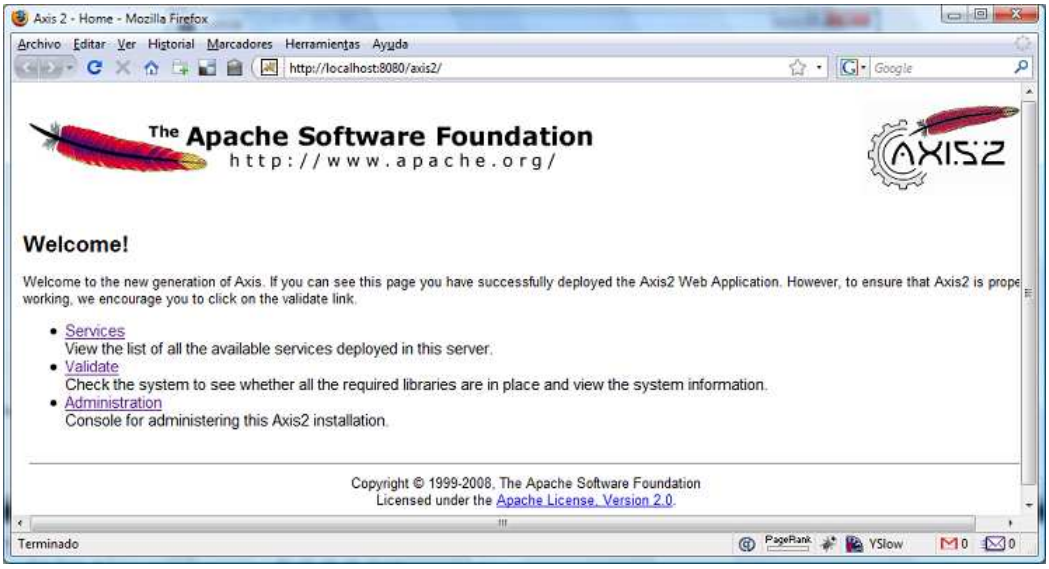
Y añadimos a la variable PATH del sistema la ruta:

```
C:\TutorialWS\axis2-1.4.1\bin
```

Para comprobar que el servidor y el motor de axis se han instalado correctamente, arrancamos el servidor ejecutando CATALINA_HOME\bin\startup.bat (ojito con los cortafuegos instalados, dar acceso a puertos y programas de este tutorial), y aparecerá una consola con las siguientes trazas:



Todo ha ido bien. Podemos comprobarlo accediendo a la URL http://localhost:8080/axis2:



Página de bienvenida de Axis2

4.- Eclipse Ganymede

Lo descomprimimos en una carpeta, por ejemplo

```
C:\Herramientas\eclipseGanymede
```

Y establecemos como workspace la ruta (si no existe, crearla antes): `C:\TutorialWS\ejercicio`

Ejemplo de un Web Service con Axis2

Vamos a hacer un servicio web servidor de *echo* (vaya, ique original!) al que le invoquemos con nuestro nombre y nos responda. Haremos también el programa cliente que haga la petición.

Recordemos brevemente que si disponemos del descriptor WSDL (Web Service Description Language), podemos crear el esqueleto del web service servidor (*proveedor)* y/o su cliente. Así, los programadores podemos desperocuparnos del manejo de los componentes, procesos y protocolos de comunicación (HTTP, FTP, SMTP, JMS, SOAP, XML...) y centrarnos en implementar la funcionalidad.

Clientes de Servicios Web con Spring

2009-03-30
Administración de sitios Moodle

2009-03-29
Empaquetamiento de aplicaciones de escritorio (standalone) con Maven

2009-03-27
Primeros pasos con Moodle

2009-03-26
Introducción a JSF Java

2009-03-25
A1 Website Analyzer

2009-03-24
Cómo ver el correo de Gmail sin conexión a Internet

2009-03-20
JasperReports Maven Plugin

2009-03-16
Creación de contenidos SCORM: eXe

2009-03-15
Spring WS: Creación de Servicios Web con Spring

2009-03-13
Instalación Alfresco (Labs)

2009-02-26
Maven JXR Plugin: publica el código fuente en el site

2009-03-15
Generación de XML Schema (XSD) y DTD a partir de documentos XML

2009-03-04
Persistencia con Spring

2009-02-26
Vistas materializadas

2009-02-03
Instalación de MySQL 5.1 en Windows

2009-03-03
Instalación de Java Virtual Machine

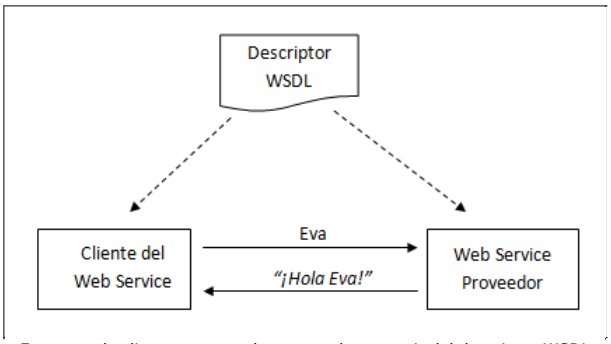
2009-03-03
Primeros Pasos con Liferay 5.2.1

2009-02-27
Edicion de video MPEG2

2009-02-26
Introducción teórica a XPath

2009-02-26
Integración Selenium / Maven 2 / Surefire / Cargo / Tomcat 6

2009-02-24
Selenium Remote Control



Esquema de cliente y proveedor generados a partir del descriptor WSDL

Pueden darse otras combinaciones:

- Que el servidor ya exista y sólo queramos crear un cliente para dialogar. Generalmente nos darán el WSDL para decirnos cómo 'hablar' con el servidor.
- Que no tengamos nada pero sepamos la funcionalidad que hará el web service. A partir de una interfaz en la que definamos las operaciones, podemos generar el WSDL, y de ahí, el procedimiento antes explicado tanto para el cliente, servidor o ambos.

La primera de ellas va a ser la aproximación para crear nuestro ejemplo. Comencemos:

1.- Crear el web service servidor

Creamos un nuevo proyecto Java en Eclipse (File | New | Java | Java Project) llamado `ServidorEchoWS`. Nos aseguramos de que estamos trabajando con Java 5 para evitar problemas si tenemos alguna otra JRE en el sistema:

- `Window | Preferences | Java | Installed JREs`, y que esté seleccionado `jre_1.5_0.16`
- `Window | Preferences | Java | Compiler`, Compiler compliance level: `1.5`

Ahora creamos un paquete `com.autentia.ws.servidor` y ahí una clase llamada `Echo`:

```
package com.autentia.ws.servidor;

/**
 * <p>
 * Echo.java <br/> Clase que implementa la logica de nuestro web service
 * </p>
 *
 *
 * @author Ivan Garcia Puebla - www.autentia.com
 * @version 1.0
 */
public class Echo {

        /**
         * Metodo que implementa la funcionalidad de saludo
         *
         * @param nombre
         *            Nombre de la persona que invoca el servicio
         * @return Cadena de saludo
         */
        public String saludar(String nombre) {

                return "Hola " + nombre;
        }

        /**
         *
         * Metodo que implementa la funcionalidad de despedida
         *
         * @param nombre
         *            Nombre de la persona que invoca el servicio
         * @return Cadena de despedida
         */
        public String despedir(String nombre) {

                return "Adios " + nombre;
        }

}
```

A nivel raíz del protecto creamos una carpeta `META-INF` y en su interior un fichero llamado `services.xml`, que será el descriptor del web service:

```
<?xml version="1.0" encoding="UTF-8"?>
<service>
        <description> Web service que emite un saludo o una despedida
        </description>
        <messageReceivers>
                <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
                        class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />
        </messageReceivers>
        <parameter name="ServiceClass">com.autentia.ws.servidor.Echo
        </parameter>
        <operation name="saludar" mep="http://www.w3.org/2004/08/wsdl/in-out" />
        <operation name="despedir" mep="http://www.w3.org/2004/08/wsdl/in-out" />
</service>
```

En este punto, la estructura del proyecto será como muestra la imagen:

2009-02-22
Integración de Groovy, JRuby y BeanShell con Spring 2

2009-02-18
Instalación de Pentaho BI Suite Community Edition 1.7.0

2009-02-18
Replicar Web PHP en máquina local

2009-02-16
Selenium Core : El motor de Selenium.

2009-02-16
Integración de JasperReports con PHP

2009-02-09
EJB 3.0 y pruebas unitarias con Maven, JUnit 4 y Embedded JBoss sobre Java 6

2009-02-09
Web Service Security

2009-02-09
Manual Avanzado de Firebug

2009-01-29
Ejemplo con Mockito

2009-01-29
Uso de Mock objects en pruebas con Mockito

2009-01-29
StrutsTestCase

2009-01-28
Eventos en Hibernate (parte III)

2009-01-28
Eventos en Hibernate (parte II)

2009-01-27
Eventos en Hibernate (parte I)

2009-01-25
Aprendiendo XMLSchema a través de ejemplos

2009-01-20
Pruebas Software con Junit 4 y Eclipse

Últimas ofertas de empleo

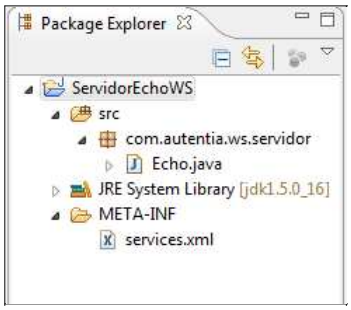
2009-03-26
Comercial - Ventas - ALMERIA.

2009-03-12
Comercial - Ventas - VALENCIA.

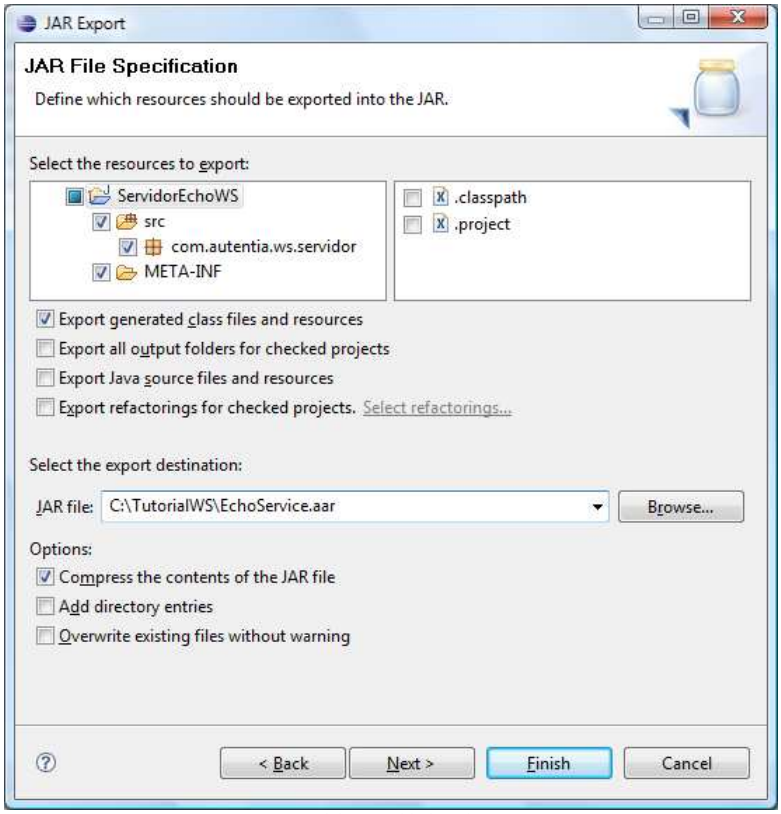
2009-03-12
Comercial - Ventas - SEVILLA.

2009-02-21
Otras - Estética/Peluquería - MADRID.

2009-02-13
T. Información - Otros no catalogados - MADRID.

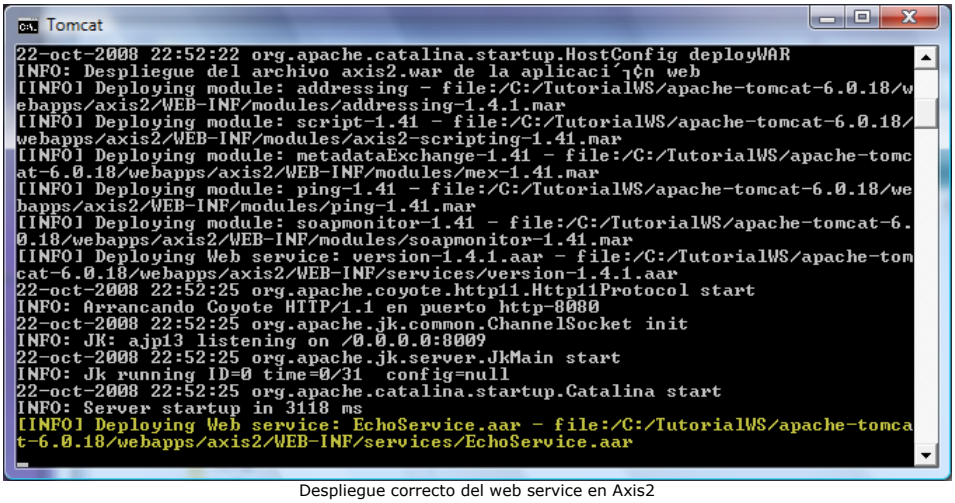


Vamos a crear el ensamblado de tipo aar (Axis ARrchive, que viene a ser el equivalente al jar de Java). Botón derecho sobre el nombre del proyecto | `Export` | `Java` | `Jar File`, pulsamos `Next` y en la siguiente pantalla deseleccionamos los ".classpath" y ".project" y como ruta en input JAR File ponemos por ejemplo `C:\TutorialWS\EchoService.aar`:



Configuracion del ensamblado EchoService.aar

Pulsamos Finish. A continuación accedemos a la carpeta C.\TutorialWS y copiamos el fichero EchoService.aar en `CATALINA_HOME\webapps\axis2\WEB-INF\services`. A los pocos segundos, Axis detectará que hay un nuevo recurso que debe ser desplegado, y en la consola de Tomcat aparecerá esta traza:



Despliegue correcto del web service en Axis2

Si accedemos a la web http://localhost:8080/axis2/ y pulsamos en *Services*, veremos que aparece el nuevo servicio en la lista:

Web service EchoService activo

Ya tenemos el web service proveedor publicado y a la espera de que un cliente lo invoque. Ahora vamos a crearlo.

4.- Implementando el cliente

Como hemos explicado antes, si queremos utilizar un servicio web, tenemos que utilizar un cliente que sepa 'dialogar' con aquel. Para saber qué ofrece y cómo comunicarse, nos valdremos de un descriptor del servicio, un fichero XML que cumple la especificación WSDL (la más reciente es la versión 2.0).

Pediremos a Axis que nos dé el WSDL de nuestro servicio EchoService invocando la siguiente dirección http://localhost:8080/axis2/services/EchoService?wsdl. Lo guardamos en un fichero con el nombre `echoservice.wsdl` (precaución al guardarlo, que no se introduzcan caracteres externos al propio xml).

Debemos abrir y examinar el contenido de echoservice.wsdl y si vemos que en las etiquetas "`<soap:address location="`" aparece un número de IP en vez de `localhost`, cambiar a `localhost`. Por ejemplo:

```
<wsdl:service name="EchoService">
  <wsdl:port name="EchoServiceHttpSoap11Endpoint" binding="ns:EchoServiceSoap11Binding">
    <soap:address
      location="http://87.105.13.213:8080/axis2/services/EchoService.EchoServiceHttpSoap11Endpoint/"/>
  </wsdl:port>
  <wsdl:port name="EchoServiceHttpSoap12Endpoint" binding="ns:EchoServiceSoap12Binding">
    <soap12:address
      location="http://87.105.13.213:8080/axis2/services/EchoService.EchoServiceHttpSoap12Endpoint/"/>
  </wsdl:port>
  <wsdl:port name="EchoServiceHttpEndpoint" binding="ns:EchoServiceHttpBinding">
    <http:address
      location="http://87.105.13.213:8080/axis2/services/EchoService.EchoServiceHttpEndpoint/"/>
  </wsdl:port>
</wsdl:service>
```

cambiar por:

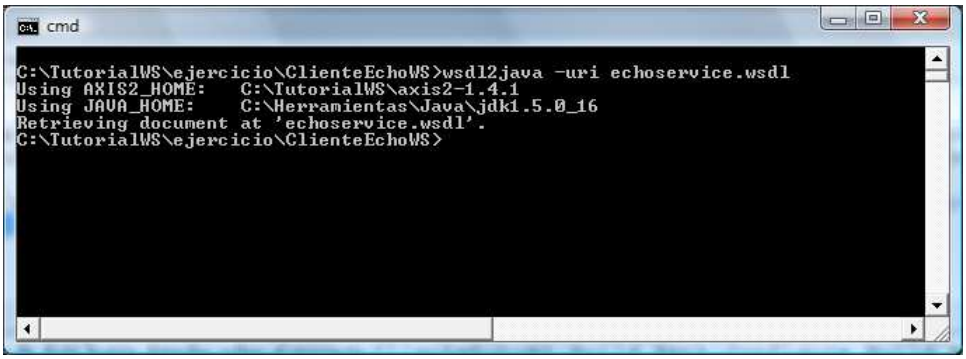
```
<wsdl:service name="EchoService">
  <wsdl:port name="EchoServiceHttpSoap11Endpoint" binding="ns:EchoServiceSoap11Binding">
    <soap:address
      location="http://localhost:8080/axis2/services/EchoService.EchoServiceHttpSoap11Endpoint/"/>
  </wsdl:port>
  <wsdl:port name="EchoServiceHttpSoap12Endpoint" binding="ns:EchoServiceSoap12Binding">
    <soap12:address
      location="http://localhost:8080/axis2/services/EchoService.EchoServiceHttpSoap12Endpoint/"/>
  </wsdl:port>
  <wsdl:port name="EchoServiceHttpEndpoint" binding="ns:EchoServiceHttpBinding">
    <http:address
      location="http://localhost:8080/axis2/services/EchoService.EchoServiceHttpEndpoint/"/>
  </wsdl:port>
</wsdl:service>
```

Volviendo a Eclipse, creamos un proyecto Java con el nombre `ClienteEchoWS` e importamos las librerías de Axis2: botón derecho sobre el proyecto `ClienteEchoWS` | `Build Path` | `Configure Build Path` | pestaña `Libraries` | `Add External JARS...` , navegamos hasta el `AXIS2_HOME\lib`, seleccionamos todos los .jar (aunque muchos no serán realmente necesarios), y OK.

Copiamos ahora el fichero echoService.wsdl en la carpeta de nuestro proyecto de eclipse de ClienteEchoWS, que estará en `C:\TutorialWS\ejercicio\ClienteEchoWS`. Abrimos una consola de línea de comandos y nos situamos en ese mismo directorio. Una vez ahí, ejecutamos el comando:

```
wsdl2java -uri echoservice.wsdl
```

Obtendremos un resultado como éste:

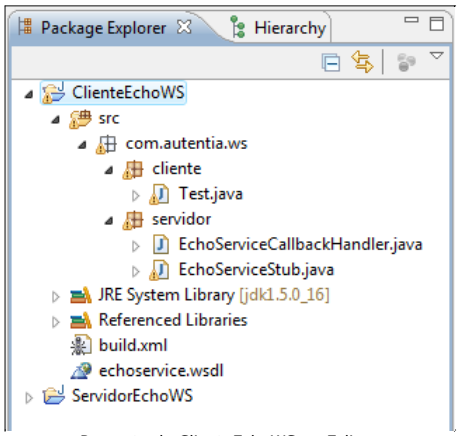


Ejecucion del comando wsdl2java

Con ello hemos creado las clases del cliente que resuelven por nosotros la comunicación con el web service. Volvemos a Eclipse, refrescamos el proyecto (boton derecho sobre `ClienteEchoWS` | Refresh) y veremos que se ha creado el paquete `com.autentia.ws.servidor` con dos clases: `EchoServiceCallbackHandler.java` y `EchoServiceStub.java`. Creamos en el proyecto un paquete `com.autentia.ws.cliente` y una clase `Test.java` con el siguiente código:

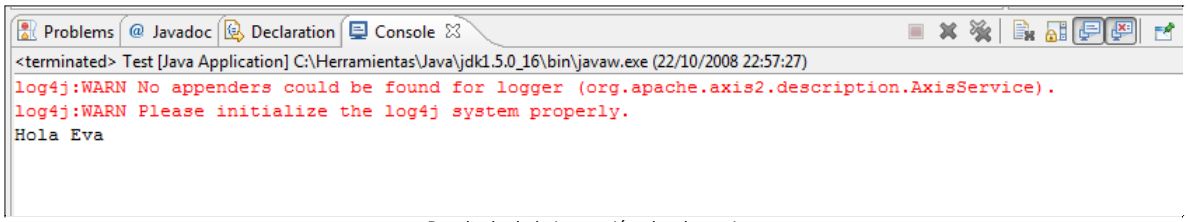
```
package com.autentia.ws.cliente;

import java.rmi.RemoteException;

import org.apache.axis2.AxisFault;

import com.autentia.ws.servidor.EchoServiceStub;

/**
 * <p>
 * Test.java <br/> Clase que prueba la invocacion a nuestro web service de echo
 * </p>
 *
 *
 * @author Ivan Garcia Puebla - www.autentia.com
 * @version 1.0
 */

public class Test {

	/**
	 * Metodo principal de la clase
	 *
	 * @param args
	 */
	public static void main(String[] args) {

		/*
		 * Utilizamos el stub generado a partir del wsdl que logran establecer
		 * la conexion con el web service proveedor.
		 */
		EchoServiceStub customer = null;
		EchoServiceStub.Saludar request = null;
		EchoServiceStub.SaludarResponse response = null;

		try {

			// creamos el soporte y la peticion
			customer = new EchoServiceStub();
			request = new EchoServiceStub.Saludar();

			// establecemos el parametro de la invocacion
			request.setNombre("Eva");

			// invocamos al web service
			response = customer.saludar(request);

			// mostramos la respuesta
			System.out.println(response.get_return());

		} catch (RemoteException excepcionDeInvocacion) {
			System.err.println(excepcionDeInvocacion.toString());
		}

	}
}
```

En este punto, nuestro proyecto en Eclipse deberá tener el aspecto de la imagen:



Proyecto de ClienteEchoWS en Eclipse

Ejecutamos el cliente pulsando con el botón derecho sobre `Test.java` | `Run As` | `Java Application`. En la vista inferior `Console` veremos el resultado de la ejecución (el texto en rojo podemos ignorarlo, no estamos usando log4j) :



Resultado de la invocación al web service

El web service nos ha respondido correctamente saludándonos :-)

¡Inténtalo ahora tú con el servicio de despedida!

Conclusión

Si este tutorial te ha servido para comenzar y romper el miedo a crear servicios web con Java, te animo a seguir aprendiendo. No olvidemos que la complejidad de las soluciones a desarrollar en proyectos reales requieren utilizar las múltiples capacidades y patrones de los web services en profundidad.

Cada día la arquitectura orientada a servicios (SOA) está más presente en las organizaciones, y dominar los web services se va convirtiendo en una necesidad, como nos muestra la alta demanda de la formación que impartimos en Autentia sobre estas tecnologías.

Anímate y coméntanos lo que pienses sobre este tutorial

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Nombre: E-Mail:

Comentario:

Enviar comentario **Texto Legal y condiciones de uso**

Autor	Mensaje
	Fecha de envío: 2009-04-15 - 07:30:05 PM
Ivan	En el WSDL aparece la IP real de tu ordenador -proveedor de internet, asignada por LAN, etc- y con ella el cliente puede que no alcance el servidor (tras router, tema puertos, etc). En nuestro ejemplo, el servidor y cliente están en la misma máquina, ponemos localhost (127.0.0.1) y un riesgo menos. JAX-WS es cómodo por el uso de anotaciones, si el proyecto lo permite. Gracias por los comentarios, un saludo.
	Fecha de envío: 2009-04-14 - 12:57:16 PM
Miguel	Fantastico el tutorial. Comentar que con el jre6 y con el compilador 1.6 no he tenido ningun problema. Un par de preguntas: ¿Porque debemos cambiar las direcciones numericas por "localhost" en el wsdl? ¿prefieres axis2 a JAXWS?
	Fecha de envío: 2009-04-06 - 05:39:08 PM
Borja	Hola Iván, Gracias por este mini tutorial para empezar con axis2. Acabo de empezar en WS para trabajar con Axis2 y me ha sido muy útil sin dejarte prácticamente ningún paso, no como pasa la mayoría de las veces que nos saltamos pasos sin querer al pensar que son obvios. Nada más, como supongo que nadie escribe estas cosas, eso, felicitarte pues. Un saludo.

Página 1 de 1 Anterior **Saltar a la página 1** Siguiente

- Puedes inscribirte en nuestro servicio de notificaciones haciendo clic aquí.
- Puedes firmar en nuestro libro de visitas haciendo clic aquí.
- Puedes asociarte al grupo AdictosAlTrabajo en XING haciendo clic aquí.
- Añadir a favoritos Technorati. ADD THIS BLOG TO MY Technorati FAVORITES

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Creación e invocación de Webservices por SSL	En este tutorial se pretende enseñar a desplegar un webservice usando SSL y a invocarlo correctamente	2006-11-29	13089	Muy bueno	2	
Servicios Web RESTful en Axis 2	En este tutorial vamos realizar una descripción de REST y vamos a ver un ejemplo práctico de un Servicio Web RESTful en AXIS 2	2008-04-03	5118	Muy bueno	8	
Creando un servicio web a partir de su interfaz WSDL	En este tutorial se resalta la importancia de definir la interfaz de un servicio web antes de implementarlo, y cómo hacer todo esto con Eclipse y Apache Axis v1.	2006-09-26	18126	Bueno	11	
Monitorización de Web Services con Glassfish Wsmonitor	En este tutorial vamos a realizar una introducción a una herramienta de monitorización de mensajes SOAP o Servicios Web en general.	2008-04-04	1621	Bueno	1	
Metro: pila de webservices de Sun.	NE este tutorial Germán nos enseñara qué es y cómo usar Metro: pila de webservices de Sun en nuestras aplicaciones	2008-04-05	3477	Bueno	5	
Generador automático de Webservices	Os mostramos como crear un servicio Web a partir de una clases, gracias a generadores automáticos de código y NetBeans	2003-10-16	40986	Bueno	6	
Web Services en tu IPAQ	Cesar Crespo nos enseña como programar accesos Web Services desde tu IPAQ en Visual C++ con PocketSOAP, Apache SOAP y Axis	2004-08-02	37254	Bueno	3	
WebServices con Axis y JBoss	En este tutorial os mostramos como realizar servicios web utilizando Axis y el contenedor de aplicaciones web JBoss	2006-04-03	21732	Regular	2	
Metro: pila de webservices de Sun. Integración con Maven 2	En este tutorial Germán nos enseñara a integrar la generación de webservices con Metro y Maven2.	2008-04-05	2118	Regular	2	
Jersey: la implemetación de RESTFull de Sun	En este tutorial Germán nos enseña cómo usar RESTFull con la tecnología de Sun.	2008-04-05	1672	-	-	

Nota: