Formación

Suministro Información XML

Jesús Pardal Gómez

Tipo de documento Público

Índice de contenido

Suministro de Información XML

Introducción

1. Conceptos Básicos

XML (Extended Markup Language) es un metalenguaje extensible de etiquetas desarrollado por W3C que permite definir la gramática de lenguajes específicos (de la misma manera que HTML). Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

1.1. Estructura Básica

Las etiquetas son el componente de XML que permite definir los elementos que conformarán un documento de la siguiente forma:

<etiqueta>Valor</etiqueta>

Como se puede observar los elementos se formarán usando una etiqueta de inicio, otra de fin, delimitadas mediante los caracteres "<" y ">", y que comparten el identificador textual pero añadiendo el carácter "/" al principio. En medio de las etiquetas de inicio y fin del elemento se representará el contenido que se desee almacenar en ese elemento. Este contenido puede a su vez englobar otros elementos. Por otro lado, los elementos pueden no contener ningún valor, pero en ese caso se deberá usar solamente la etiqueta de finalización.

Se considera que en el elemento XML engloba todo lo que se encuentra entre las correspondientes etiquetas de inicio y de fin y pueden contener tanto otros elementos como simplemente texto o una combinación de ambos.

Los elementos pueden asimismo contener atributos, los cuales se especificarán en la etiqueta de inicio del elemento. El objetivo de los atributos es poder proporcionar una información adicional sobre un elemento concreto. La sintaxis para representar los atributos consiste en especificar el nombre del atributo dentro de la etiqueta de inicio, a continuación un símbolo "=" y finalmente el valor del atributo delimitado por comillas dobles o por comillas simples:

<etiqueta atributo="valor">Valor</etiqueta>

Es importante tener en cuenta, para la construcción del contenido de las etiquetas de un XML, una serie de símbolos reservados para el propio lenguaje:

Código HTML	Carácter
"	“
&	&
'	`
<	<
>	>
** **	Espacio en blanco

1.2. Espacios de Nombres

De forma general, los documentos XML se suelen combinar con otros documentos XML existentes. Esta modularidad es una característica esencial de XML y permite al desarrollador poder reutilizar código existente o lo más interesante de todo, aplicar XSD.

Un espacio de nombres se define como una referencia URI (Uniform Resource Identifier), que servirá para identificar los elementos que pertenecen a dicho espacio de nombres. Otra forma de verlo es que los elementos tendrán un nombre compuesto por dos partes: una primera con su nombre y una segunda con el nombre de espacio de nombres:

<etiqueta xmlns:ns1=”http://....” >

<ns1:elemento>valor</ns1:elemento>

</etiqueta>

Al centrarnos únicamente en el aspecto de “suministro de información”, la parte de DTD, XSD y demás anotaciones las dejaremos a un lado. Se verán en otras asignaturas impartidas por el resto de compañeros.

2. Aplicaciones y casos de uso

El lenguaje XML se creó para que cumpliera varios objetivos:

- Que fuera idéntico a la hora de servir, recibir y procesar la información que el HTML, para aprovechar toda la tecnología implantada para este último.
- Que fuera formal y conciso desde el punto de vista de los datos y manera de guardarlos.
- Que fuera extensible, fácil de leer y editar.
- Que fuera fácil de implantar, programar y aplicar a los distintos sistemas.

Por este conjunto de características, este lenguaje de marcas se puede usar para infinidad de trabajos y aporta numerosas ventajas en amplios escenarios:

- Comunicación de datos. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella, independientemente de la tecnología con la que estén desarrolladas.
- Gestión de recursos. A través de XML se puede, por ejemplo, detallar el conjunto de librerías que usa un proyecto JAVA y organizarlas en un único servidor. De esta forma se elimina la dependencia "física" de las librerías y nos garantizamos que todo estará en el mismo versionado. Una herramienta muy efectiva y extendida en este caso es MAVEN.
- Servicios Web. Volviendo al primer punto pero atendiendo a un entorno cliente-servidor, los servicios web son una de las aplicaciones más extendidas hoy en día y se basan en XML para funcionar. Algunos de los frameworks más usados son Axis2 (JAVA), CodeIgniter/WSO2 (PHP) o Windows Communication Foundation (PHP).
- Protocolo RPC (Remote Procedure Call) o XML-RPC. Este protocolo permite a un equipo (cliente) ejecutar código en otro equipo (servidor) sin necesidad de preocuparse por el tipo de comunicación. Aunque fue creado en 1998, al ser tan simple, este protocolo basado en XML ha ido evolucionando y recibiendo nuevas funcionalidades hasta convertirse en lo que hoy se conoce como SOAP (Simple Object Access Protocol).
- XHTML. Uno de los lenguajes más relevantes que se crearon a partir de XML, fue XHTML (Extensible HyperText Markup Language) cuya implementación tuvo como objetivo mejorar las deficiencias que presentaba HTML en su momento, como la mejora del procesamiento (no sólo en navegadores) o el uso de otros recursos.

3. Caso práctico

Para centrarnos en un ejemplo básico de suministro de información a través de XML y profundizar algo más en las tecnologías actuales, realizaremos un pequeño caso práctico usando JAVA y AXIS2 para diseñar y consumir un pequeño servicio web a bajo nivel. Como entorno de programación usaremos Eclipse y como contenedor de aplicaciones usaremos Tomcat. A continuación se detalla paso a paso como integrar el framework en la herramienta.

Para ello, necesitaremos realizar los siguientes pasos previos:

- Instalar Eclipse (cualquier versión).
- Instalar JAVA 8 (cualquier versión sería válida, pero vamos a realizar el caso con esta).
- Instalar AXIS2 (carpeta axis2-1.5.2 de ruta compartida)
- Instalar TOMCAT 8.
- Desplegar carpeta axis2 (carpeta axis2 de ruta compartida) en la carpeta webapps de nuestro TOMCAT 8.
- Crear las variables de entorno JAVA_HOME y AXIS2_HOME.
- Importar las librerías del directorio axis2/lib (carpeta axis2 de ruta compartida) en Eclipse.

3.1. Definición de una interface Java a partir de la cual generaremos el servicio web

Creamos el paquete “es.aytos.main”. Dentro de este paquete creamos una interface llamada “Calculadora” con el siguiente código:

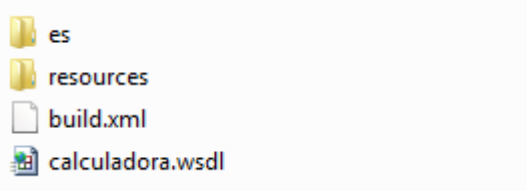
```
package es.aytos.main;

public interface Calculadora {
    public int sumar(int op1, int op2);

}
```

3.2. Generar la definición del servicio web a partir de la interface

Abrimos una consola de comandos y nos dirigimos al directorio SRC de nuestro proyecto JAVA. Una vez en el directorio, ejecutamos la siguiente instrucción para generar el fichero XML que describirá nuestro servicio web (WSDL):

%AXIS2_HOME%\bin\java2wsdl -of calculadora.wsdl -cp "./" -cn es.aytos.main.Calculadora

Una vez ejecutado el comando, tendremos nuestro fichero WSDL generado en SRC:



3.3. Generar el esqueleto del servicio web a partir del WSDL

De nuevo en la consola de comandos y en el directorio SRC ejecutamos el siguiente comando:

%AXIS2_HOME%\bin\wsdl2java -p es.aytos.main.server -S . -or -ss -sd -u -uri calculadora.wsdl

- El parámetro p indica el nombre del paquete donde deseamos que se generen las clases.
- El parámetro S indica el nombre de la carpeta en donde deseamos que se generen las clases (por defecto crea una carpeta src)
- El parámetro or indica que se sobrescriban los archivos.
- El parámetro ss indica que se generen las clases del servicio web (Por defecto NO se generan)
- El parámetro sd indica que se genere el descriptor de despliegue del servicio web.
- El parámetro u indica que no se generen innerclases sino que cada clase está en un archivo propio.
- El parámetro uri indica el WSDL de partida.

Este comando generara a partir de nuestro fichero WSDL toda la lógica de servidor para implementar el servicio web. A su vez, también genera un fichero llamado services.xml (carpeta resources) que describirá el despliegue a realizar en el servidor.

3.4. Implementar la lógica de negocio del servicio web

Hasta este momento se han generado todos los componentes necesarios para integrar el método de la interfaz que hemos creado en el primer punto como parte de un servicio que será consumido a través de XML a modo de servicio web. Antes de continuar, debemos dar funcionamiento a nuestro servicio web. Para ello nos dirigimos a la clase "CalculadoraSkeleton" del paquete es.aytos.main.server y programamos el método ya existente "es.aytos.main.SumarResponse sumar(es.aytos.main.Sumar sumar)" incluyendo el siguiente código:

```
es.aytos.main.SumarResponse response = new es.aytos.main.SumarResponse();
response.set_return(sumar.getArgs0() + sumar.getArgs1());

return response;
```

Acto seguido, compilamos el proyecto completo a través de la opción "Build All" de Eclipse y verificamos que no haya errores.

3.5. Generar el archivo del servicio web AAR (Axis Archive)

Los archivos AAR, son como los archivos WAR, EAR. Es decir son archivos JAR o ZIP que contienen archivos y directorios bajo una estructura definida.

Empezaremos por crear una nueva carpeta en el directorio raíz de nuestro proyecto llamada "Calculadora" y seguimos los siguientes pasos:

- Copiamos el directorio <RUTA_RAIZ>/src/resources y renombramos la carpeta a "META-INF".
- Copiamos el directorio <RUTA_RAIZ>/bin/src/es.

En la consola de comandos nos dirigimos a la nueva carpeta que hemos creado y ejecutamos el siguiente comando:

"%JAVA_HOME%\bin\jar" -cvf Calculadora.aar *

3.6. Desplegar el servicio web

Existen varios métodos de despliegue de un Servicio Web en Axis2, el más común es copiar el fichero "AAR" que acabamos de generar al directorio :

<AXIS_WEBAPP_HOME>\repository\services (Siendo AXIS_WEBAPP_HOME el directorio donde está desplegado Axis2 en TOMCAT)

3.7. Generar un cliente

Hasta este punto hemos generado toda la lógica y código necesarios para dar funcionalidad a nuestro servicio web (parte servidora). Ahora debemos generar el código cliente para atacar al mismo. Esta parte se podría realizar en cualquier tecnología, a través del fichero WSDL que tenemos desplegado en TOMCAT. Este fichero lo podremos comprobar con el siguiente enlace:

http://127.0.0.1:8080/axis2/services/Calculadora?wsdl

Para generar el cliente abrimos una consola de comandos y nos dirigimos al directorio SRC de nuestro proyecto. Acto seguido ejecutamos el siguiente comando:

%AXIS2_HOME%\bin\wsdl2java -p es.aytos.main.client -d adb -S . -or -uri http://127.0.0.1:8080/axis2/services/Calculadora?wsdl

Esto generará en el paquete "es.aytos.main.client" todo lo necesario para implementar del lado del cliente un acceso a nuestro servicio web.

3.8. Invocar el servicio web

Para invocar el servicio web, creamos un nuevo paquete en "es.aytos.example" y creamos una clase Main con el siguiente código:

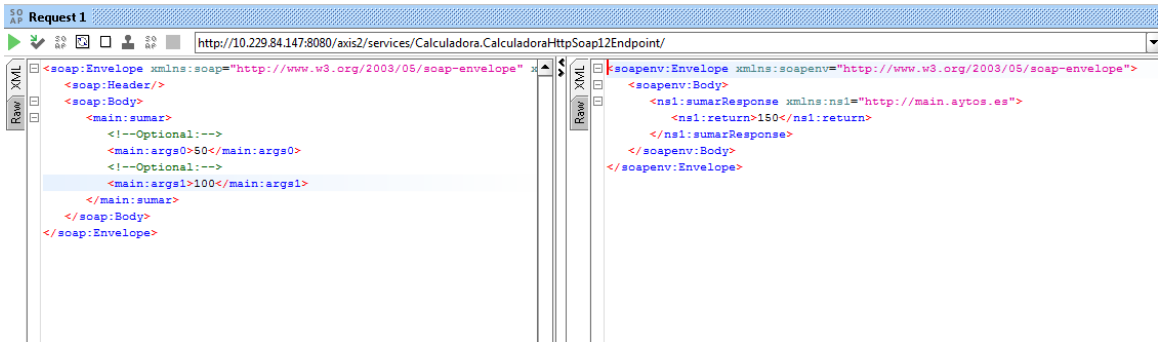
```
package es.aytos.example;

import es.aytos.main.client.CalculadoraStub;
import es.aytos.main.client.CalculadoraStub.Sumar;
import es.aytos.main.client.CalculadoraStub.SumarResponse;

public class Main {

    public static void main(String[] args) throws Exception {
        CalculadoraStub stub = new CalculadoraStub();
        Sumar operacion = new Sumar();
        SumarResponse response = null;

        // Establecemos los parámetros de la operación
        operacion.setArgs0(100);
        operacion.setArgs1(200);

        // Invocamos el WS
        response = stub.sumar(operacion);

        // Mostramos el resultado
        System.out.println(response.get_return());
    }
}
```

Ejecutamos la clase y para este ejemplo en concreto obtendremos la siguiente salida:

```
<terminated> Main (2) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2 may. 2018 17:44:33)
log4j:WARN No appenders could be found for logger (org.apache.axis2.description.AxisService).
log4j:WARN Please initialize the log4j system properly.
300
```

Hasta este punto puede parecer que la importancia del lenguaje XML no está presente, pero nada más lejos de la realidad. Dependiendo del cliente que usemos, se encapsulará la generación de XML de una forma u otra. Sin embargo, consumiendo nuestro servicio web a través de una

aplicación externa, por ejemplo SoapUI, podremos comprobar que tanto envío como respuesta tienen presentes este tipo de lenguaje:



4. Práctica Avanzada

Implementar funciones de resta, división y multiplicación en el servicio web. Ejemplos de uso y resultados reales.