Interacción entre Aplicaciones con xml: Intercambio de documentos e invocación remota

Una vez conocido el soporte que ofrecen analizadores de xml como SAX o DOM con independencia del lenguaje de programación, o JDOM en el caso de Java, se van a explorar las opciones para el diseño de una aplicación que utiliza xml para intercambiar datos e invocar acciones en procesos remotos, de la misma o distinta organización o empresa. Se pueden encontrar dos casos:

1) En ocasiones el objetivo de la aplicación es generar un documento comercial en un formato estipulado por alguna comunidad de negocios (por ejemplo las redes interbancarias), organismo internacional (como EDIFACT de las Naciones Unidas) o proveedor grande que impone un formato a sus proveedores (también llamado "integración de empresas", como en la industria del automóvil, aviación, comunicaciones, etc.). En este caso, se ha de complicar el programa que construye y manipula el documento xml para que la entrada o salida cumplan con el formato xml especificado.
2) En otras ocasiones, se puede utilizar xml como formato de codificación y serialización de datos para invocar funciones o métodos remotos entre sistemas diversos. Son aplicables en situaciones en que también se podría usar Corba. En este caso, la programación puede ser muy simple (y notar poco que los objetos están separados), a cambio de delegar el formato y la estructura de los datos a serializadores y deserializadores automáticos.

Caso 1

Para la tienda de muebles se ha creado una clase Stock que sirve para tratar de forma sencilla el stock de un producto:

Stock.java

```
public class Stock {
  String nombre;
  int num;
  int precio;

  public Stock(String no, int nu, int pr) {  // constructor con precio y num
    nombre = no;  num = nu;  precio = pr;
  }
  public Stock(String no) {
    this (no, 0, 0);
  }
  void show(String op) {                  // constructor sólo con nombre
    System.out.println(" stock op="+op+" nom="+nombre+" num="+num+"
precio="+precio);
  }
  void supply(int nu, int pr) {
    num += nu;  precio = pr;
  }
  int order(int nu) {
    if (nu == 0) return num; // número de unidades en stock
    if (nu <= num) {
       num -= nu;  return precio*nu; // importe total
    } else return -1;
  }
}
```

El código siguiente lee y genera árboles xml utilizando JDOM para implementar las operaciones de la tienda de muebles.

TiendaWebApp.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import org.jdom.input.*;
import org.jdom.output.*;
import org.jdom.*;
```

```
public class TiendaWebApp {
  private static int op_id = 0;  // contador identificadores operación
  private static int granTotal = 0; // total ventas acumulado
  private static boolean debug = false;
  private static String _u = "juan", _pass = "nauj";
  private static Stock[] stock = { new Stock("mesas"), new Stock("sillas"),
new Stock("lámparas") };

  private static Element listado;

  private String Usuari = new String("pablo");
  private String Clau = new String("casa");

  public static void main(String[] args) {

   PrintStream out = System.out;
   FileOutputStream outf;
   PrintStream p;

   String comanda, nomcamp, valor;
   boolean idOk = false, opOk = false;

   listado = new Element("list");

   for(int f=0; f<args.length; f++) {

    if (debug) out.println("Hay " + stock[0].order(0) + " mesas, " +
       stock[1].order(0) + " sillas, " + stock[2].order(0) + " lámparas.");

   try {   // cargar el árbol !!!
    SAXBuilder builder = new SAXBuilder(false); // sin validación
    Document doc = builder.build(new File(args[f]));
    Element root = doc.getRootElement(); // Obtener elemento root

    String u = root.getAttributeValue("u");
    String pass = root.getAttributeValue("pass");
    if (u != null && u.equals(_u) && pass.equals(_pass)) idOk = true;

    String op = root.getAttributeValue("name");

    // respuesta:
    Element r = new Element("res");
    r.addAttribute("name", op);
    Document rr = new Document( r );

    if (op.equals("Supply")) { // in: t_objeto, precio, unidades; out:
ok|error
      List l = root.getChildren("item");
      if (debug) out.println("item: "+l.size()+" elementos");
      for(Iterator i = l.iterator(); i.hasNext(); ) {
        Element prod = (Element) i.next();
        int precio = Integer.parseInt(prod.getAttributeValue("precio"));
        int uds = Integer.parseInt(prod.getAttributeValue("unidades"));
        int id = Integer.parseInt(prod.getAttributeValue("id"));
        stock[id].supply(uds,precio);
      }
      opOk = true;
    }
    else if (!idOk) {
        r.addContent("Identificaci¢n err¢nea");
        opOk = true;
    } else if (op.equals("Order")) {
      int precioTotal = 0;
      List l = root.getChildren("item");
      if (debug) out.println("item: "+l.size()+" elementos");

      if (l.size() > 0) {
```

```
        op_id++;          // asignar op_id
        r.addContent(
           new Element("Order").addAttribute("id",Integer.toString(op_id)));

        for(Iterator i = l.iterator(); i.hasNext(); ) {
          Element prod = (Element) i.next();
          int uds = Integer.parseInt(prod.getAttributeValue("unidades"));
          int id = Integer.parseInt(prod.getAttributeValue("id"));
          int precio = stock[id].order(uds);
          precioTotal += precio;
          Element e = (Element) prod.clone();
          e.addAttribute("precioTotal",Integer.toString(precio));
          r.getChild("Order").addContent(e);
        }
      }

      r.getChild("Order").addContent(
        new Element("total").setText(Integer.toString(precioTotal))
      );

      Element e = (Element) r.getChild("Order").clone();
      listado.addContent(e); // guardar en listado hist¢rico

      granTotal += precioTotal;

      opOk = true;
    } else if (op.equals("List")) {
      r.addContent(listado);
      r.addContent(
          new Element ("total").setText(Integer.toString(granTotal))
      );
      opOk = true;
    }
    if (! opOk) r.addContent("Operación errónea");

    XMLOutputter outxml = new XMLOutputter();
    outxml.output(r, out);

  } catch (Exception e) {
    e.printStackTrace();
  }
 }
}
}
```

El código anterior puede convertirse en un servlet con unos pequeños cambios:

TiendaWebServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
...

public class TiendaWebServlet extends HttpServlet {
  private /*static*/ int op_id = 0;  // contador identificadores operación
...

  public void doPost(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {   // antes main

   res.setContentType("text/html");
   PrintWriter out = res.getWriter();
   BufferedReader in = req.getReader();
```

El servlet anterior colocado en Tomcat (Linux o Windows) funciona tal como se muestra en el siguiente ejemplo donde se miran los pedidos (List), se suministran muebles (Supply), se hace un pedido (Order), y finalmente se pide un listado de pedidos (List).

Petición:

```
telnet tiendaMuebles.upc.es 8080
POST /test/servlet/TiendaWeb HTTP/1.0
Content-Length: 61

<?xml version="1.0" ?>
<op name="List" u="juan" pass="nauj" />
```

Respuesta:

```
HTTP/1.0 200 OK
Date: Tue, 10 Dec 2000 23:25:36 GMT
Status: 200
Servlet-Engine: Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.3.0;
Windows 98 4.10 x86; java.vendor=Sun Microsystems Inc.)
Content-Type: text/html
Content-Language: en

<res name="List"><list /><total>0</total></res>
```

Petición:

```
telnet tiendaMuebles.upc.es 8080
POST /test/servlet/TiendaWeb HTTP/1.0
Content-Length: 180

<?xml version="1.0" ?>
<op name="Supply">
<item precio="123456" unidades="60" id="0">mesa comedor haya</item>
<item precio="13457" unidades="200" id="1">silla cocina nogal</item>
</op>
```

Respuesta:

```
HTTP/1.0 200 OK
Date: Tue, 10 Dec 2000 23:27:00 GMT
Status: 200
Servlet-Engine: Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.3.0;
Windows 98 4.10 x86; java.vendor=Sun Microsystems Inc.)
Content-Type: text/html
Content-Language: en

<res name="Supply" />
```

Petición:

```
telnet tiendaMuebles.upc.es 8080
POST /test/servlet/TiendaWeb HTTP/1.0
Content-Length: 166

<?xml version="1.0" ?>
<op name="Order" u="juan" pass="nauj">
<item unidades="1" id="0">mesa comedor haya</item>
<item unidades="4" id="1">silla cocina nogal</item>
</op>
```

Respuesta:

```
HTTP/1.0 200 OK
Date: Tue, 10 Dec 2000 23:27:40 GMT
Status: 200
Servlet-Engine: Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.3.0;
Windows 98 4.10 x86; java.vendor=Sun Microsystems Inc.)
Content-Type: text/html
Content-Language: en

<res name="Order"><Order id="1"><item unidades="1" id="0"
precioTotal="123456">mesa comedor haya</item><item unidades="4" id="1"
precioTotal="53828">silla cocina
nogal</item><total>177284</total></Order></res>
```

Petición:

```
telnet tiendaMuebles.upc.es 8080
POST /test/servlet/TiendaWeb HTTP/1.0
Content-Length: 61
```

```
<?xml version="1.0" ?>
<op name="List" u="juan" pass="nauj" />
```

Respuesta:

```
HTTP/1.0 200 OK
Date: Tue, 12 Dec 2000 23:28:56 GMT
Status: 200
Servlet-Engine: Tomcat Web Server/3.1 (JSP 1.1; Servlet 2.2; Java 1.3.0;
Windows 98 4.10 x86; java.vendor=Sun Microsystems Inc.)
Content-Type: text/html
Content-Language: en

<res name="List"><list /><total>177284</total></res>
```

Caso 2

Se puede utilizar el servidor Tomcat, instalando el servicio SOAP, que puede obtenerse en `http://xml.apache.org`

Tomando un ejemplo mínimo pueden verse los pasos necesarios para programar un servidor, para describir el servicio, para programar un cliente, y para desplegar (instalar y poner en marcha) el servicio.

TiendaWeb.java (servidor)

```
import java.io.*;
import org.apache.soap.util.xml.*;

public class TiendaWeb {
  float precio = 10.0F;

  public float verPrecio (String producto) {
    return precio;
  }
}
```

El servicio ha de describirse en un fichero xml aparte:

tienda.xml

```
<isd:service xmlns:isd="http://xml.apache.org/xml-soap/deployment"
             id="urn:precio-producto">
  <isd:provider type="java"
                scope="Application"
                methods="verPrecio">
    <isd:java class="TiendaWeb"/>
  </isd:provider>
</isd:service>
```

verPrecio.java (cliente)

```
import java.io.*;
import java.net.*;
import java.util.*;
import org.apache.soap.util.xml.*;
import org.apache.soap.*;
import org.apache.soap.rpc.*;

public class verPrecio {
  static XMLParserLiaison xpl = new XercesParserLiaison ();

  public static void main (String[] args) throws Exception {
    if (args.length != 2 && (args.length != 3 || !args[0].startsWith("-"))){
      System.err.println ("Usage: java " + verPrecio.class.getName () +
                          " [-encodingStyleURI] SOAP-router-URL producto");
      System.exit (1);
    }

    int offset = 3 - args.length;       // procesar argumentos
    String encodingStyleURI = args.length == 3
                              ? args[0].substring(1)
                              : Constants.NS_URI_SOAP_ENC;
```

```
    String producto = args[2 - offset];

    Call call = new Call ();   // Construir llamada
    call.setTargetObjectURI ("urn:precio-producto");
    call.setMethodName ("verPrecio");
    call.setEncodingStyleURI(encodingStyleURI);
    Vector params = new Vector ();
    params.addElement (new Parameter("producto", String.class, producto,
null));
    call.setParams (params);

    // Invocación: action URI vacío. El rpc router de XML-SOAP no lo mira
    Response resp = call.invoke (/* router URL */ url, /* actionURI */ "" );

    if (resp.generatedFault ()) {       // Detectar fallos en la respuesta
      Fault fault = resp.getFault ();
      System.out.println ("Ouch, the call failed: ");
      System.out.println ("  Fault Code   = " + fault.getFaultCode ());
      System.out.println ("  Fault String = " + fault.getFaultString ());
    } else {
      Parameter result = resp.getReturnValue ();
      System.out.println (result.getValue ());
    }
  }
}
```

Para instalarlo hay que utilizar el servicio para gestionar aplicaciones instalables en el servidor de SOAP (en este caso Tomcat+SOAP):

```
> java org.apache.soap.server.ServiceManagerClient
Usage: java org.apache.soap.server.ServiceManagerClient url operation
arguments
where
        url is the Apache-SOAP router's URL whose services are managed
        operation and arguments are:
                deploy deployment-descriptor-file.xml
                list
                query service-name
                undeploy service-name
```

Para desplegar el servicio anterior de tienda sencilla (en una línea):

```
> java org.apache.soap.server.ServiceManagerClient
http://localhost:8080/soap/servlet/rpcrouter deploy tienda.xml
```

Esta implementación de SOAP permite 3 estilos de codificación (serialización): codificación SOAP v1.1, XML Literal, y XMI.

XMI (a partir de Java 1.2.2) serializa y deserializa automáticamente cualquier objeto (tal como ocurre en RMI).

La codificación SOAP tiene soporte para multitud de tipos: tipos primitivos, String, JavaBeans, vectores, enumeraciones, arrays de 1 dimensión de estos tipos. Para otros tipos hay que escribir nuevos serializadores/deserializadores.

La codificación literal permite enviar elementos XML como parámetros.

La implementación de SOAP en Apache funciona sobre transporte HTTP o SMTP (por correo)