Web Services-Sending Large Data

Kristoff Cossement

Soap with mime attachments was indeed the way to go.

Remark: this only works with java 1.4.2_09 (I did not get it working with java 1.5.x)

For those who are interested I included some sample code on how to exchange large binary data between a java client and and webobjects server without taking all data in memory.
You also need the java mail and java activation framework from sun.

Client.java

```
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.*;
import java.io.*;
import org.apache.axis.attachments.AttachmentPart;
import org.apache.axis.message.*;
import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.xml.soap.SOAPElement;
import java.util.*;
import javax.mail.*;

public class Client {

  public static File resize(String sPath)
  {
    File file = new File(sPath);


    String endPoint = "http://localhost:55555/cgi-bin/WebObjects/project.woa/ws/FileUpload";
    try{
      SOAPConnection connection = SOAPConnectionFactory.newInstance().createConnection();
      SOAPMessage message = (javax.xml.soap.SOAPMessage)MessageFactory.newInstance().createMessage();
      SOAPPart part = message.getSOAPPart();
      SOAPEnvelope envelope = (org.apache.axis.message.SOAPEnvelope)part.getEnvelope();
      SOAPBody body = (org.apache.axis.message.SOAPBody)envelope.getBody();
      SOAPBodyElement operation = (org.apache.axis.message.SOAPBodyElement)body.addBodyElement(
          envelope.createName("upload",
            "ns",
            "http://localhost:55555/cgi-bin/WebObjects/project.woa/ws/FileUpload"));
      operation.setEncodingStyle("http://schemas.xmlsoaporg/soap/encoding/");

      DataHandler dh = new DataHandler(new FileDataSource(file));
      AttachmentPart attachment = (org.apache.axis.attachments.AttachmentPart)message.createAttachmentPart(dh);
      SOAPElement filename = operation.addChildElement("filename","");
      SOAPElement source = operation.addChildElement("source","");
      message.addAttachmentPart(attachment);
      filename.addTextNode(file.getName());
      source.addTextNode(attachment.getContentId());

      SOAPMessage result = connection.call(message,endPoint);
      System.out.println(result);
      part = result.getSOAPPart();
      envelope = (org.apache.axis.message.SOAPEnvelope)part.getEnvelope();
      body = (org.apache.axis.message.SOAPBody)envelope.getBody();

      if(!body.hasFault())
      {
        System.out.println("answer : "+body);

      }
    }
    catch(Exception e)
    {
```

```
      e.printStackTrace();
    }
    return null;
  }

  public static void main(String[] args) {

    try {

      resize(args[0]);

    }
    catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }
}
```

Webobjects Application .java

override dispatchRequest

```
 public WOResponse dispatchRequest(WORequest request)
{
  WOResponse result = null;
  String sURI = request.uri();
  NSLog.debug.appendln("Accessing " + sURI);

  Pattern p = Pattern.compile("/ws/FileUpload");
  Matcher m = p.matcher(sURI);
  if(m.find())
  {
    String sContType = request.headerForKey("content-type");
    p = Pattern.compile("multipart/related");
    m = p.matcher(sContType);
    if(m.find())
    {
      result = Dispatcher.handleFileUpload(request);
    }
    else
    {
      result = super.dispatchRequest(request);
    }
  }
  else
  {
    result = super.dispatchRequest(request);
  }

  return result;

}
```

Dispatcher.java

```
 //
// Dispatcher.java
// project
//
// Created by admin on 5/5/06.
// Copyright 2006 __MyCompanyName__. All rights reserved.
//
import com.webobjects.foundation.*;
import com.webobjects.appserver.*;
import com.webobjects.eocontrol.*;
```

```
import java.io.*;
import java.util.regex.*;
import org.w3c.dom.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.dom.*;
import javax.xml.parsers.*;
import org.apache.axis.attachments.*;


public class Dispatcher {

  public static WOResponse handleFileUpload(WORequest request)
  {
    String sContType = request.headerForKey("content-type");
    String sSoapXml = "";
    InputStream requestStream = request.contentInputStream();

    String sTempDir = System.getProperty("java.io.tmpdir");
    String timestamp =(new Long(System.currentTimeMillis())).toString();
    File fTempFile = new File(sTempDir+"/"+timestamp);
    File fSavedFile = new File(sTempDir+"/"+timestamp+".out");

    try
    {
      BufferedOutputStream fOut = new BufferedOutputStream(new FileOutputStream(fTempFile));
      byte[] buffer = new byte[32 * 1024];
      int bytesRead = 0;
      while ((bytesRead = requestStream.read(buffer)) != -1)
      {
        fOut.write(buffer, 0, bytesRead);
      }
      fOut.close();
    }
    catch (Exception e)
    {
      NSLog.debug.appendln(e.getMessage());
    }

    try
    {
      InputStream iStream = new FileInputStream(fTempFile);

      MultiPartRelatedInputStream mis = new MultiPartRelatedInputStream(sContType,iStream);
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      Document doc = factory.newDocumentBuilder().parse(mis);
      sSoapXml = toString(doc);
      NSLog.debug.appendln("SOAP Envelope: " + sSoapXml);
      mis.close();
      iStream.close();

      Node nEnvelope = getNamedChildNode(doc,"Envelope");
      if(null != nEnvelope)
      {
        Node nBody = getNamedChildNode(nEnvelope,"Body");
        if(null != nBody)
        {
          NSArray nSubNodes = getElementChildNodes(nBody);
          if((null != nSubNodes) && (nSubNodes.count() > 0))
          {
            Node nFileUpload = (Node)nSubNodes.get(0);
            nSubNodes = getElementChildNodes(nFileUpload);
            if((null != nSubNodes) && (nSubNodes.count() > 1))
            {
              Node nFileName = (Node)nSubNodes.get(0);
              Node nData = (Node)nSubNodes.get(1);

              String sFileName = getTextFromNode(nFileName);
              String sFileMimeID = getTextFromNode(nData);

              fSavedFile = new File(sTempDir+"/"+sFileName);
```

```
              Pattern pattern = Pattern.compile("(boundary=\".+?\")");
              Matcher match = pattern.matcher(sContType);
              if(match.find())
              {
                String boundary= match.group(1);
                String sSubContentType = "multipart/related; type=\"text/xml\"; start=\"<"+sFileMimeID+">\";
"+boundary;
                iStream = new FileInputStream(fTempFile);
                mis = new MultiPartRelatedInputStream(sSubContentType,iStream);
                try
                {
                  BufferedOutputStream fOut = new BufferedOutputStream(new FileOutputStream(fSavedFile));
                  byte[] buffer = new byte[32 * 1024];
                  int bytesRead = 0;
                  while ((bytesRead = mis.read(buffer)) != -1)
                  {
                    fOut.write(buffer, 0, bytesRead);
                  }
                  mis.close();
                  fOut.close();
                }
                catch (Exception e)
                {
                  System.out.println(e.getMessage());
                  //throw new IOException(me + " failed, got: " + e.toString());
                }
              }
            }
          }
        }
      }
    }
    catch(Exception e)
    {
      e.printStackTrace();
    }


    WOResponse result = new WOResponse();
    result.setContent( "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"+
        "<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoaporg/soap/envelope/\" xmlns:xsd=\"http://www.
w3.org/2001/XMLSchema\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">\n"+
        "<soapenv:Body>\n"+
        "<ns:uploadResponse soapenv:encodingStyle=\"http://schemas.xmlsoaporg/soap/encoding/\" xmlns:ns=\"
http://localhost:55555/cgi-bin/WebObjects/project.woa/ws/FileUpload\">\n"+
        "<ns:uploadReturn xsi:type=\"xsd:boolean\">true</ns:uploadReturn>\n"+
        "</ns:uploadResponse>\n"+
        "</soapenv:Body>\n"+
        "</soapenv:Envelope>");


    result.setHeader("text/xml; charset=utf-8","content-type");


    return result;

  }


  static public String toString(Document document) {
    String result = null;

    if (document != null) {
      StringWriter strWtr = new StringWriter();
      StreamResult strResult = new StreamResult(strWtr);
      TransformerFactory tfac = TransformerFactory.newInstance();
      try {
        Transformer t = tfac.newTransformer();
        t.setOutputProperty(OutputKeys.ENCODING, "utf-8");
        t.setOutputProperty(OutputKeys.INDENT, "yes");
```

```
        t.setOutputProperty(OutputKeys.METHOD, "xml"); //xml, html, text
        t.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
        t.transform(new DOMSource(document.getDocumentElement()), strResult);
      } catch (Exception e) {
        System.err.println("XML.toString(Document): " + e);
      }
      result = strResult.getWriter().toString();
    }

    return result;
  }

  static public Node getNamedChildNode(Node n, String sChildNodeName)
  {

    NodeList nl = n.getChildNodes();

    for(int i=0;i<nl.getLength();i++)
    {
      Node s = nl.item(i);
      String sChild = s.getNodeName();
      if(sChildNodeName.equalsIgnoreCase(sChild) || sChild.endsWith(":"+sChildNodeName))
      {
        NSLog.debug.appendln("childnode = " + s.getNodeName());
        return s;
      }
    }

    return null;

  }
  static public NSArray getElementChildNodes(Node n)
  {
    NSMutableArray elements = new NSMutableArray();
    NodeList nl = n.getChildNodes();

    for(int i=0;i<nl.getLength();i++)
    {
      Node s = nl.item(i);
      if(s.getNodeType() == Node.ELEMENT_NODE)
      {
        NSLog.debug.appendln("childnode = " + s.getNodeName());
        elements.addObject(s);
      }
    }

    return elements;
  }
  static public String getTextFromNode(Node n)
  {
    String text = "";

    NodeList nl = n.getChildNodes();

    for(int i=0;i<nl.getLength();i++)
    {
      Node s = nl.item(i);
      if(s.getNodeType() == Node.TEXT_NODE)
      {
        return s.getNodeValue();
      }
    }
    return text;
  }
}
```