JAX-WS attachment with MTOM

By mkyong | November 22, 2010 | Updated : August 29, 2012 | Viewed : 171,071 | +210 pv/w

A complete JAX-WS SOAP-based example to show how to use **Message Transmission Optimization Mechanism (MTOM)** and **XML-Binary Optimized Packaging (XOP)** technique to send a binary attachment (image) from server to client and vice verse.

> **Note**
> There are ton of articles about what's MTOM and the benefits of using it (see reference sites below), but it's lack of complete example to use **MTOM in JAX-WS**, hope this example can fill in some missing pieces in the whole picture.

Enabling MTOM on server

Enable server to send attachment via MTOM is very easy, just annotate the web service implementation class with `javax.xml.ws.soap.MTOM`.

1. WebService Endpoint

Here's a RPC-style web service, published two methods, `downloadImage(String name)` and `uploadImage(Image data)`, to let user upload or download an image file.

File : ImageServer.java

```
package com.mkyong.ws;

import java.awt.Image;

import javax.jws.WebMethod;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.jws.soap.SOAPBinding.Style;

//Service Endpoint Interface
@WebService
@SOAPBinding(style = Style.RPC)
public interface ImageServer{

    //download a image from server
    @WebMethod Image downloadImage(String name);

    //update image to server
    @WebMethod String uploadImage(Image data);

}
```

File : ImageServerImpl.java

```java
package com.mkyong.ws;

import java.awt.Image;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.jws.WebService;
import javax.xml.ws.WebServiceException;
import javax.xml.ws.soap.MTOM;

//Service Implementation Bean
@MTOM
@WebService(endpointInterface = "com.mkyong.ws.ImageServer")
public class ImageServerImpl implements ImageServer{

    @Override
    public Image downloadImage(String name) {

        try {

            File image = new File("c:\\images\\" + name);
            return ImageIO.read(image);

        } catch (IOException e) {

            e.printStackTrace();
            return null;

        }
    }

    @Override
    public String uploadImage(Image data) {

        if(data!=null){
            //store somewhere
            return "Upload Successful";
        }

        throw new WebServiceException("Upload Failed!");

    }

}
```

File : ImagePublisher.java

```java
package com.mkyong.endpoint;

import javax.xml.ws.Endpoint;
import com.mkyong.ws.ImageServerImpl;

//Endpoint publisher
public class ImagePublisher{

    public static void main(String[] args) {

    Endpoint.publish("http://localhost:9999/ws/image", new ImageServerImpl());

    System.out.println("Server is published!");

    }

}
```

2. WebService Client

Here's a web service client, to access the published web service at URL "*http://localhost:9999/ws/image*".

File : ImageClient.java

```
package com.mkyong.client;

import java.awt.Image;
import java.io.File;
import java.net.URL;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
import javax.xml.ws.Service;
import javax.xml.ws.soap.MTOMFeature;
import javax.xml.ws.soap.SOAPBinding;

import com.mkyong.ws.ImageServer;

public class ImageClient{

    public static void main(String[] args) throws Exception {

    URL url = new URL("http://localhost:9999/ws/image?wsdl");
        QName qname = new QName("http://ws.mkyong.com/", "ImageServerImplService");

        Service service = Service.create(url, qname);
        ImageServer imageServer = service.getPort(ImageServer.class);

        /************  test download  ***************/
        Image image = imageServer.downloadImage("rss.png");

        //display it in frame
        JFrame frame = new JFrame();
        frame.setSize(300, 300);
        JLabel label = new JLabel(new ImageIcon(image));
        frame.add(label);
        frame.setVisible(true);

        System.out.println("imageServer.downloadImage() : Download Successful!");

    }

}
```

3. HTTP and SOAP traffic

Here's the HTTP and SOAP traffic generated by the client, captured by traffic monitoring tool.

P.S The first wsdl request is omitted to save space.
Client send request :

```
POST /ws/image HTTP/1.1
SOAPAction: ""
Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-Type: text/xml; charset=utf-8
User-Agent: Java/1.6.0_13
Host: localhost:9999
Connection: keep-alive
Content-Length: 209

<?xml version="1.0" ?>
    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
        <S:Body>
            <ns2:downloadImage xmlns:ns2="http://ws.mkyong.com/">
                <arg0>rss.png</arg0>
            </ns2:downloadImage>
        </S:Body>
    </S:Envelope>
```

Server send response :

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: multipart/related;
start="<rootpart*c73c9ce8-6e02-40ce-9f68-064e18843428@example.jaxws.sun.com>";
type="application/xop+xml";
boundary="uuid:c73c9ce8-6e02-40ce-9f68-064e18843428";
start-info="text/xml"

--uuid:c73c9ce8-6e02-40ce-9f68-064e18843428
Content-Id: <rootpart*c73c9ce8-6e02-40ce-9f68-064e18843428@example.jaxws.sun.com>
Content-Type: application/xop+xml;charset=utf-8;type="text/xml"
Content-Transfer-Encoding: binary

<?xml version="1.0" ?>
  <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
     <S:Body>
    <ns2:downloadImageResponse xmlns:ns2="http://ws.mkyong.com/">
      <return>
        <xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
        href="cid:012eb00e-9460-407c-b622-1be987fdb2cf@example.jaxws.sun.com">
        </xop:Include>
      </return>
    </ns2:downloadImageResponse>
     </S:Body>
   </S:Envelope>
--uuid:c73c9ce8-6e02-40ce-9f68-064e18843428
Content-Id: <012eb00e-9460-407c-b622-1be987fdb2cf@example.jaxws.sun.com>
Content-Type: image/png
Content-Transfer-Encoding: binary

Binary data here.............
```

Enabling MTOM on client

Enable client to send attachment via MTOM to server is required some extra efforts, see following example :

```
//codes enable MTOM in client
BindingProvider bp = (BindingProvider) imageServer;
SOAPBinding binding = (SOAPBinding) bp.getBinding();
binding.setMTOMEnabled(true);
```

1. WebService Client

Here's a webservice client to send an image file to published endpoint above (http://localhost:8888/ws/image) via MTOM.

File : ImageClient.java

```
package com.mkyong.client;

import java.awt.Image;
import java.io.File;
import java.net.URL;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.xml.namespace.QName;
import javax.xml.ws.BindingProvider;
import javax.xml.ws.Service;
import javax.xml.ws.soap.MTOMFeature;
import javax.xml.ws.soap.SOAPBinding;

import com.mkyong.ws.ImageServer;

public class ImageClient{

    public static void main(String[] args) throws Exception {

    URL url = new URL("http://localhost:8888/ws/image?wsdl");
        QName qname = new QName("http://ws.mkyong.com/", "ImageServerImplService");

        Service service = Service.create(url, qname);
        ImageServer imageServer = service.getPort(ImageServer.class);

        /************  test upload ****************/
        Image imgUpload = ImageIO.read(new File("c:\\images\\rss.png"));

        //enable MTOM in client
        BindingProvider bp = (BindingProvider) imageServer;
        SOAPBinding binding = (SOAPBinding) bp.getBinding();
        binding.setMTOMEnabled(true);

        String status = imageServer.uploadImage(imgUpload);
        System.out.println("imageServer.uploadImage() : " + status);

    }

}
```

2. HTTP and SOAP traffic

Here's the HTTP and SOAP traffic generated by the client, captured by traffic monitoring tool.

P.S The first wsdl request is omitted to save space.

Client send request :

```
POST /ws/image HTTP/1.1
Content-type: multipart/related;
start="<rootpart*751f2e5d-47f8-47d8-baf0-f793c29bd931@example.jaxws.sun.com>";
type="application/xop+xml";
boundary="uuid:751f2e5d-47f8-47d8-baf0-f793c29bd931";

start-info="text/xml"
Soapaction: ""
Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
User-Agent: JAX-WS RI 2.1.6 in JDK 6
Host: localhost:9999
Connection: keep-alive
Content-Length: 6016

--uuid:751f2e5d-47f8-47d8-baf0-f793c29bd931
Content-Id: <rootpart*751f2e5d-47f8-47d8-baf0-f793c29bd931@example.jaxws.sun.com>
Content-Type: application/xop+xml;charset=utf-8;type="text/xml"
Content-Transfer-Encoding: binary

<?xml version="1.0" ?>
  <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
     <S:Body>
    <ns2:uploadImage xmlns:ns2="http://ws.mkyong.com/">
      <arg0>
        <xop:Include xmlns:xop="http://www.w3.org/2004/08/xop/include"
        href="cid:2806f201-e15e-4ee0-8347-b7b4dffad5cb@example.jaxws.sun.com">
        </xop:Include>
      </arg0>
    </ns2:uploadImage>
    </S:Body>
  </S:Envelope>

--uuid:751f2e5d-47f8-47d8-baf0-f793c29bd931
Content-Id: <2806f201-e15e-4ee0-8347-b7b4dffad5cb@example.jaxws.sun.com>
Content-Type: image/png
Content-Transfer-Encoding: binary

Binary data here.............
```

Server send response :

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: multipart/related;
start="<rootpart*188a5835-198b-4c28-9b36-bf030578f2bd@example.jaxws.sun.com>";
type="application/xop+xml";
boundary="uuid:188a5835-198b-4c28-9b36-bf030578f2bd";
start-info="text/xml"

--uuid:188a5835-198b-4c28-9b36-bf030578f2bd
Content-Id: <rootpart*188a5835-198b-4c28-9b36-bf030578f2bd@example.jaxws.sun.com>
Content-Type: application/xop+xml;charset=utf-8;type="text/xml"
Content-Transfer-Encoding: binary

<?xml version="1.0" ?>
  <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
     <S:Body>
    <ns2:uploadImageResponse xmlns:ns2="http://ws.mkyong.com/">
       <return>Upload Successful</return>
    </ns2:uploadImageResponse>
     </S:Body>
  </S:Envelope>

--uuid:188a5835-198b-4c28-9b36-bf030578f2bd--
```

Complete WSDL document

For those who are interested to study the WSDL file, you can get the wsdl file via URL : *http://localhost:9999/ws/image?wsdl*

Sample of the ImageServer WSDL file

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://ws.mkyong.com/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://ws.mkyong.com/"
    name="ImageServerImplService">

<types></types>

<message name="downloadImage">
    <part name="arg0" type="xsd:string"></part>
</message>
<message name="downloadImageResponse">
    <part name="return" type="xsd:base64Binary"></part>
</message>

<message name="uploadImage">
    <part name="arg0" type="xsd:base64Binary"></part>
</message>
<message name="uploadImageResponse">
    <part name="return" type="xsd:string"></part>
</message>

<portType name="ImageServer">

    <operation name="downloadImage">
        <input message="tns:downloadImage"></input>
        <output message="tns:downloadImageResponse"></output>
    </operation>
    <operation name="uploadImage">
        <input message="tns:uploadImage"></input>
        <output message="tns:uploadImageResponse"></output>
    </operation>

</portType>

<binding name="ImageServerImplPortBinding" type="tns:ImageServer">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc">
        </soap:binding>

    <operation name="downloadImage">
        <soap:operation soapAction=""></soap:operation>
        <input>
            <soap:body use="literal" namespace="http://ws.mkyong.com/"></soap:body>
        </input>
        <output>
            <soap:body use="literal" namespace="http://ws.mkyong.com/"></soap:body>
        </output>
    </operation>

    <operation name="uploadImage">
        <soap:operation soapAction=""></soap:operation>
        <input>
            <soap:body use="literal" namespace="http://ws.mkyong.com/">
                        </soap:body>
        </input>
        <output>
            <soap:body use="literal" namespace="http://ws.mkyong.com/">
                        </soap:body>
        </output>
    </operation>

</binding>

<service name="ImageServerImplService">
<port name="ImageServerImplPort" binding="tns:ImageServerImplPortBinding">
<soap:address location="http://localhost:9999/ws/image"></soap:address>
</port>
```

```
</service>
</definitions>
```

Download Source Code

Download It – JAX-WS-Attachment-MTOM-Example.zip (20KB)

Reference

1. http://www.devx.com/xml/Article/34797/1763/page/1
2. http://download.oracle.com/docs/cd/E17802_01/webservices/webservices/docs/2.0/jaxws/mtom-swaref.html
3. http://www.crosschecknet.com/intro_to_mtom.php
4. http://download.oracle.com/docs/cd/E12840_01/wls/docs103/webserv_adv/mtom.html
5. http://www.theserverside.com/news/1363957/Sending-Attachments-with-SOAP
6. http://metro.java.net/guide/Binary_Attachments__MTOM_.html

attachment | jax-ws | mtom | web services

About the Author

mkyong

Founder of Mkyong.com, love Java and open source stuff. Follow him on Twitter, or befriend him on Facebook or Google Plus. If you like my tutorials, consider make a donation to these charities.

Comments

Guest

subuhi

above syntax must be generated in WSDL that is absent in the given WSDL .

+ 0 − Reply 1 year ago