SAAJ - Web Service SOAP Client

http://techdive.in/web-service/saaj-web-service-soap-client

Submitted by arunraj on Sat, 09/25/2010 - 17:40

In this article, lets discuss about accessing a Web Service using SAAJ.

What is SAAJ?

SOAP with Attachments API for Java (SAAJ) is mainly used for dealing directly with SOAP Request/Response messages which happens behind the scenes in any Web Service API. It allows the developers to directly send and receive soap messages instead of using JAX-WS.

Let's see an example of how to access the Web Service created in the Axis2 - Simple Web Service Example.

Now, we are going to develop Web Service Client using SAAJ API. Have a look at the following Web Service Client code.

SaaJSoapClient.java

```
/**
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS''
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS
 * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */
import javax.xml.soap.MessageFactory;
import javax.xml.soap.SOAPBody;
import javax.xml.soap.SOAPConnection;
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPElement;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPMessage;
import javax.xml.soap.SOAPPart;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
```

```
/**
 * SOAP Client Implementation using SAAJ Api.
 */
public class SaaJSoapClient
{
	/**
	 * Method used to create the SOAP Request
	 */
	private static SOAPMessage createSOAPRequest() throws Exception
	{
		MessageFactory messageFactory = MessageFactory.newInstance();
		SOAPMessage soapMessage = messageFactory.createMessage();
		SOAPPart soapPart = soapMessage.getSOAPPart();

		/*
		Construct SOAP Request Message:
		<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
			xmlns:sam="http://samples.axis2.techdive.in">
		  <soap:Header/>
		  <soap:Body>
		    <sam:getStudentName>
		      <!--Optional:-->
		      <sam:rollNumber>3</sam:rollNumber>
		    </sam:getStudentName>
		  </soap:Body>
		</soap:Envelope>
		 */

		// SOAP Envelope
		SOAPEnvelope envelope = soapPart.getEnvelope();
		envelope.addNamespaceDeclaration("sam", "http://samples.axis2.techdive.in");

		// SOAP Body
		SOAPBody soapBody = envelope.getBody();
		SOAPElement soapBodyElem = soapBody.addChildElement("getStudentName", "sam");
		SOAPElement soapBodyElem1 = soapBodyElem.addChildElement("rollNumber", "sam");
		soapBodyElem1.addTextNode("3");

		soapMessage.saveChanges();

		// Check the input
		System.out.println("Request SOAP Message = ");
```

```
        soapMessage.writeTo(System.out);
        System.out.println();
        return soapMessage;
    }

    /**
     * Method used to print the SOAP Response
     */
    private static void printSOAPResponse(SOAPMessage soapResponse) throws Exception
    {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        Source sourceContent = soapResponse.getSOAPPart().getContent();
        System.out.println("\nResponse SOAP Message = ");
        StreamResult result = new StreamResult(System.out);
        transformer.transform(sourceContent, result);
    }

    /**
     * Starting point for the SAAJ - SOAP Client Testing
     */
    public static void main(String args[])
    {
        try
        {
            // Create SOAP Connection
            SOAPConnectionFactory soapConnectionFactory = SOAPConnectionFactory.newInstance();
            SOAPConnection soapConnection = soapConnectionFactory.createConnection();

            //Send SOAP Message to SOAP Server
            String url = "http://localhost:8080/axis2/services/Student?wsdl";
            SOAPMessage soapResponse = soapConnection.call(createSOAPRequest(), url);

            // Process the SOAP Response
            printSOAPResponse(soapResponse);

            soapConnection.close();
        }
        catch (Exception e)
        {
            System.err.println("Error occurred while sending SOAP Request to Server");
            e.printStackTrace();
```

```
            }
        }
}
```

Here we create a SOAP request for accessing the Student details. Look in to the createSOAPRequest() method. It creates a SOAP request message envelope, which includes name space, web service method name and necessary parameters for the same.

This method returns a Soap Message, which is sent as a request to Web Service. The SOAP response received is printed using the printSOAPResponse(..) method.

Output
Here is the sample output:

```
Request SOAP Message =
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sam="http://samples.axis2.techdive.in"><SOAP-ENV:Header/><SOAP-ENV:Body><sam:getStudentName><sam:rollNumber>3</sam:rollNumber></sam:getStudentName></SOAP-ENV:Body></SOAP-ENV:Envelope>

Response SOAP Message =
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Body><ns1:getStudentNameResponse xmlns:ns1="http://samples.axis2.techdive.in"><ns1:return>Jack</ns1:return></ns1:getStudentNameResponse></soapenv:Body></soapenv:Envelope>
```

Web Service