Java SOAP Web Services

September 30, 2009 / Gavin Jackson

java jax-ws programming soap

Recently I was required to call some remote services (written in Microsoft.net) using SOAP from Java. There is a fair amount going on in the Java Web Services space (and an amazing number of acronyms).

The three primary web service implementations for the Java platform consist of:

- Apache Axis
- Sun JAX-WS and
- Codehaus CXF (previously known as Crossfire)

I decided to use JAX-WS, which is the Sun reference implementation (which now ships standard with JDK6. For this example I'm only interested in talking to an existing web service (not creating my own), the web service in question has a published WSDL file (an xml interface specification).

The first step was to generate the java stubs from the external WSDL file. This is done using the "wsimport" command. The generated files are then used by your (client) code to marshal, transmit, receive and unmarshal the SOAP calls.

I was having a lot of trouble figuring out how to add data to the SOAP headers (in this case authentication details). After a day of banging my head against a brick wall I stumbled across the "-XadditionalHeaders" option.

This really saved me - and I sincerely hope that this helps others out there!

So this was the command I used to generate the web service client java code:

```
wsimport -p au.net.lesmills.zenkai.education.generated -s
~/work/zenkai/trunk/zenkai/BizEducation/src/
https://servicestest.lesmills.com:444/programkitcodes.asmx?WSDL -httpproxy:proxy:8080 -extension -
XadditionalHeaders
```

I then wrote the following test client to call the validate method via SOAP:

```
package au.net.lesmills.zenkai.education.test;

import javax.xml.bind.JAXBException;
import javax.xml.ws.Holder;
import au.net.lesmills.zenkai.education.generated.*;
import com.sun.xml.internal.bind.api.JAXBRIContext;

public class ProgramKitCodesTest{

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

 System.setProperty("https.proxyHost", "proxy.lesmills.net.au");
 System.setProperty("https.proxyPort", "8080");

 System.out.println("Started");

 ProgramKitCodes pkc = new ProgramKitCodes();
 ProgramKitCodesSoap pkcs = pkc.getProgramKitCodesSoap();

 String username = "USERNAME";
 String password = "PASSWORD";

 ObjectFactory of = new ObjectFactory();

 AuthenticationHeader ah = of.createAuthenticationHeader();
 JAXBRIContext jbc = (JAXBRIContext)
JAXBRIContext.newInstance("au.net.lesmills.zenkai.education.generated");
 ah.setUsername(username);
 ah.setPassword(password);

 Holder h = new Holder(ah);

 String kitCode = "KA35-LN0V-KG8B-0109-RPM42";
 ValidateResult result = pkcs.validate(kitCode, h);
 System.out.println("Completed - result is :" +
 result.getProgram() + " " +
 result.getQuarter() + " " +
 result.getReleaseNo());

 }

}
```

Note how ProgramKitCodes is generated from the wsdl file - this provides an interface to all of the remote methods. The AuthenticationHeader JAXB object is part of the SOAP header specification (defined in the WSDL).

Also, note how the above code uses a proxy server to talk to the external web services server (via ssl). No real magic here, you can use this technique in any java application.

Although you won't be able to compile and test against the above SOAP implementation, I hope that this helps you get the ball rolling.