Implementing a Web Service Client using Java

Requirements

This guide is based on implementing a Java Client using JAX-WS that comes with Java Web Services Developer Pack version 2.0 (JWSDP). This can be downloaded from the following location

http://java.sun.com/webservices/downloads/previous/index.jsp

This guide explains how to modify the sample solution in JWSDP to write a web service client that connects to PageOne SOAP Server.

The sample solution can be found on the following location

YOUR_JWSDP_INSTALLATION_ROOT\jwsdp-2.0\jaxws\samples\fromwsdl

Follow the instructions below to write your own client

Editing the build scripts

1. Before you start editing the sample we recommend creating a back up of the fromwsdl folder.

2. Copy the PageOne liquidSoap.wsdl to fromwsdl/etc folder. Also create a subfolder ‘xsd’ and copy the schema that is referred in the liquidSoap .wsdl.

3. Modify your build.xml as follows

At the beginning of the build file, edit the path to the sjsas.props file and libraries if necessary. You can remove the sections that are not required to generate clients. Please refer to the ant script at the end of the document to see the sections you will need.

Also change the project’s default parameter to ‘server’ from the default ‘help’ mode. This is the very first line of the build file.

4. Within the /etc folder you will find the other files that are used in build.xml. Edit the following files as follows

File → **build.properties**

```
server.wsdl=etc/liquidSoap.wsdl
client.wsdl=https://soap.oventus.com/LiquidWS/MessageService?WSDL
```

……

client=fromwsdl.client.LiquidSoap

File → **custom-server.xml**

wsdlLocation="liquidSoap.wsdl"
targetNamespace='http://schemas.oventus.com/'

File → **custom-client.xml**

wsdlLocation=https://soap.oventus.com/LiquidWS/MessageService?WSDL

targetNamespace='http://schemas.oventus.com/'

5. Delete the contents of src\fromwsdl\server since you do not need to generate the server

6. Also delete the client class in src\fromwsdl\client. You will be writing a fresh client class to invoke the service.

Before running the ant script, make sure that you have the required elements in your classpath.

Generating Client Stubs

7. Run your ant script. This will generate all the classes you will need to write the Web Service client. These classes will be generated within the build folder of the project root directory.

Implementing the Client

The client class uses the following classes that were generated in the previous steps.

1. MessageServicePortType

   The port type class provides and interface to access the methods offered in the web service.

2. MessageService

   This class is used to get access to the port.

3. ObjectFactory

   ObjectFactory is used to generated Object Types that are used in constructing or accessing the soap messages.

4. Following is an outline of the client class. Place this class in src\fromwsdl\client

```
package fromwsdl.client;

import com.pageone.liquidsoap.jaxb.*;

public class LiquidSoap
{
        public String invokeMyLogin()
        {
        try
        {
                MessageServicePortType port = new MessageService().getMessagePort();

                ObjectFactory objFac = new ObjectFactory();

                LoginRequest loginRequest = objFac.createLoginRequest();
                loginRequest.setUserId("username");
                loginRequest.setPwd("password");

                LoginResponse loginResponse = objFac.createLoginResponse();
                PageoneHeader pageoneHeader = objFac.createPageoneHeader();

                javax.xml.ws.Holder loginHolder = new
javax.xml.ws.Holder(loginResponse);
                javax.xml.ws.Holder headerHolder = new
javax.xml.ws.Holder(pageoneHeader);

                port.login(loginRequest,headerHolder, loginHolder );

                PageoneHeader p1Header = (PageoneHeader)headerHolder.value;
                LoginResponse logResponse = (LoginResponse) loginHolder.value;

                StatusType statusType = logResponse.getStatus();

                System.out.println("Session ID is >>> "+p1Header.getSessionId());
                System.out.println("Status  is >>> "+statusType.getValue()+"
"+statusType.getDescription());

                return p1Header.getSessionId();

        }
        catch (Exception ex)
        {
                ex.printStackTrace();
                return null;
        }

        }

        public void invokeMySendMessage(String sessionID)
        {
                try
                {
                        MessageServicePortType port = new
MessageService().getMessagePort();
```

```
                         ObjectFactory objFac = new ObjectFactory();

                         SendMessageType sendMessageType =
objFac.createSendMessageType();
                         PageoneHeader pageoneHeader = objFac.createPageoneHeader();
                         pageoneHeader.setSessionId(sessionID);

                         sendMessageType.setSourceAddress(null);
                         sendMessageType.setMessage("Hello test client sample");
                         sendMessageType.setFlashSMS(true);
                         sendMessageType.getDestinationAddress().add("VALID
ADDRESS");

                         SendMessageResponseType sendResponse =
port.sendMessage(pageoneHeader,sendMessageType);

                         System.out.println("Send Response transactionID
"+sendResponse.getTransactionID().getValue());

                 }
                 catch (Exception ex)
                 {
                         ex.printStackTrace();
                 }
         }

   public static void main (String[] args)
         {
      try
                 {
                 LiquidSoap ls = new LiquidSoap();
                 String sessionID = ls.invokeMyLogin();
                 ls.invokeMySendMessage(sessionID);

      } catch (Exception ex) {

      }
   }
}
```

NOTE

Please make sure you set a valid username and password in login request and also a valid destination address in sendMessageType. The above class creates a SOAP login request and sends to the PageOne SOAP server . Then the sessionID is extracted from the login response and a SOAP sendMessage request is sent . If you do not set these you will not be able to see the results .

Running the client

1. You can automate generating client classes as part of your ant script edit your build.xml by adding

```
<antcall target="client" />
```

in targets section at the bottom of your file.

2. Run your ant script to generate client classes.

3. Finally you can run your client class by

```
>> java  fromwsdl.client.LiquidSoap
(this class file located at   build\classes\fromwsdl\client)
```

The basic Ant script required for generating the client

```
<project basedir="." default="server" name="fromwsdl">
    <property environment="env"/>
    <property file="../../../jwsdp-shared/bin/sjsas.props"/>
      <condition property="lib.home" value="../../../jaxws/lib">
            <available file="../../../jwsdp-shared/bin/sjsas.props"/>
      </condition>
    <condition property="lib.home" value="${env.JAXWS_HOME}/lib">
      <not>
        <available file="../../../jwsdp-shared/bin/sjsas.props"/>
      </not>
    </condition>

    <property file="etc/build.properties" />
    <property name="build.home" value="${basedir}/build"/>
    <property name="build.classes.home" value="${build.home}/classes"/>


    <path id="jaxws.classpath">
        <pathelement location="${java.home}/../lib/tools.jar"/>
        <pathelement location="${lib.sample.home}/jaxwsSampleUtils.jar"/>
        <fileset dir="${lib.home}">
            <include name="*.jar"/>
            <exclude name="j2ee.jar"/>
        </fileset>
    </path>

    <taskdef name="wsimport" classname="com.sun.tools.ws.ant.WsImport">
        <classpath refid="jaxws.classpath"/>
    </taskdef>

    <target name="setup">
```

```
      <mkdir dir="${build.home}"/>
      <mkdir dir="${build.classes.home}"/>
   </target>

   <target name="clean">
      <delete dir="${build.home}" includeEmptyDirs="true" />
   </target>

   <target name="generate-client" depends="setup">
      <wsimport
         debug="${debug}"
         verbose="${verbose}"
         keep="${keep}"
         extension="${extension}"
         destdir="${build.classes.home}"
         wsdl="${client.wsdl}">
         <binding dir="${basedir}/etc" includes="${client.binding}"/>
      </wsimport>
   </target>

   <target name="client" depends="generate-client">
      <javac
         fork="true"
         srcdir="${basedir}/src"
         destdir="${build.classes.home}"
         includes="**/client/**,**/common/**">
         <classpath refid="jaxws.classpath"/>
      </javac>
   </target>


  <target name="server" depends="setup">
               <antcall target="clean" />
               <antcall target="client" />
   </target>

</project>
```