Using Blackboard Learn 9 Web Services - Part 3 Generating And Using Web Service Client Classes

Introduction

This tutorial will describe how to use Axis2--a Java framework for generating web service client classes--and the WSDL files provided by the Blackboard web services to create Java client classes that an application may use to interact with the Blackboard web services. If you're not familiar with Axis2, visit their website at http://ws.apache.org/axis2/. After generating the web service client classes, you can use the client classes to interact with the Blackboard web services.

Blackboard Learn 9.1 web services are Simple Object Access Protocol (SOAP) based. Therefore any application you have that will use the Blackboard web services must create a SOAP envelope (an XML file) with appropriate nodes to make a request of a Blackboard web service. Your application will also need to process the SOAP envelope (XML file) that the Blackboard web service returns in response.

If you are not familiar with SOAP web services, I recommend you read some of the books and/or online articles about SOAP web services. A book I found helpful is Developing Web Services with Apache CXF and Axis2 by Kent Ka lok Tong (http://www.agileskills2.org).

A SOAP web service makes available a Web Services Description Language (WSDL) file that is an XML file with nodes describing the operations (methods) the web service supports and the data types the web service uses. There are several Java frameworks that can use a web service's WSDL file to create Java web service client classes.

Your application uses the generated web service client classes to create the SOAP envelope that is sent to the web service and to process the SOAP envelope that is returned by the web service. This allows your application to focus on creating and processing objects and not the basic technical details of creating/processing the XML SOAP envelopes.

Determine Which Blackboard Web Service Classes Your Application Needs To Use

In part 2 of this tutorial (http://www.brucephillips.name/blog/index.cfm/2011/6/8/Blackboard-Learn-91-Web-Services-Tutorials-and-Documentation) I explained how you can get the JavaDoc for the Blackboard web service classes and use it to determine which web service classes your application will need to use. For this tutorial we want to create a client application that will use a proxy tool to login to our Blackboard installation and then get the Blackboard course titles a specific Blackboard user is enrolled in.

A review of the Blackboard web services and their methods indicates that our application will need to use the Context web service to initialize the session with Blackboard and to login. We can also use the Context web service to get what Blackboard courses (the course ids) a user is

enrolled in. However, since we want the actual course names (not just the course id values) the example application will also need to use the Course web service.

Authorizing The Proxy Tool For Specific Web Services And Operations

If you review part 1 (http://www.brucephillips.name/blog/index.cfm/2010/8/3/Using-Blackboard-Learn-9-Web-Services--Part-1-Logging-In-Using-A-Proxy-Tool) of this tutorial, we used a Java application (registerproxytool) to register a proxy tool that can be used to login to Blackboard web services and then make calls upon those web services. A Blackboard administrator then had to authorize the proxy tool and make it available. When a proxy tool registers with your Blackboard installation it requests access to specific web services and methods of those classes.

In part 2 of this series we explained how to specify which Blackboard web services and methods the proxy tool should be authorized to use. In our review of the Blackboard web services and operations for this application we've determine that our application needs to use these Blackboard web services and methods:

1. Context.WS:emulateUser
2. Context.WS:logout
3. Context.WS:getMemberships
4. Context.WS.initialize
5. Context.WS.loginTool
6. Context.WS.registerTool
7. Course.WS:getCourse

Since we don't already have a proxy tool that has been authorized to access all the above, our example application will need to register a new proxy tool and then a Blackboard system administrator will need to authorize that proxy tool.

Example Application

Download the example application from http://www.brucephillips.name/blackboard/blackboardcoursesforuser.zip. The download is an archived Eclipse project created using Eclipse 3.5 and Maven 2. The program demonstrates how to register a proxy tool with a Blackboard installation using the Blackboard Learn 9.1 Context web service, get the Blackboard course memberships for a Blackboard user, and then get the course name for each of those courses. The program also demonstrates how to generate the Blackboard web service client classes using Axis2 and the WSDL file.

See the ReadMe.txt file for instructions on how to build and run the application using either Eclipse (with Maven) or just with Maven.

To import the project into your Eclipse workspace you'll need to be using Eclipse 3.5 or higher with the Maven 2 plugin. You can get Eclipse at http://www.eclipse.org/. Information about

installing the Eclipse Maven 2 plugin is at http://m2eclipse.sonatype.org/installing-m2eclipse.html.

Before running the program you'll need to set the values for the properties listed in the bbws.properties file. Those values are for the Blackboard installation that you want to register the proxy tool with and Blackboard user name you want to get the list of course titles for.

Generating Blackboard Web Service Client Class

In the example application are two packages: edu.ku.it.si.bbcontextws.generated and edu.ku.it.si.bbcoursews.generated. The classes in those two packages were generated using Axis2 and the WSDL files provided by the Context and Course web services. To get a WSDL file for a Blackboard web service log in to your Blackboard installation as a System Administrator and go to System Admin - Web Services. There you'll see a list of the web services for your Blackboard installation. If you click on the hyperlink that ends in wsdl the WSDL file will be displayed in your browser.

Note that if you have made the web service discoverable then you can go to the WSDL hyperlink directly without having to login to Blackboard. The format for the WSDL hyperlink is http://*your_blackboard_domain*/webapps/ws/services/*webservicename*.WS?wsdl (for example http://ec2-50-19-165-66.compute-1.amazonaws.com/webapps/ws/services/Course.WS?wsdl).

After you have displayed the WSDL file for a Blackboard web service select view source in your web browser, select all the content (the entire XML), and copy it to your clip board. In Eclipse, right click on src/main/resources and select New - Other and under the Web Services folder select WSDL and then click the Next button. Change the file name to match the web service name, for example courseWS.wsdl; click Next and then click Finish.

Delete all the content that Eclipse put into the WSDL file and then paste into the Eclipse WSDL file all the XML you copied from the Blackboard web service WSDL file. In the example application, this work has already been done. In src/main/resources there is a contextWS.wsdl and a courseWS.wsdl.

Once we have the WSDL files for the web services in Eclipse, we can use the Axis2 framework to generate the web service client classes. If you're not familiar with Axis2 visit their website at http://ws.apache.org/axis2/.

In the example application is class AxisCodeGenerator (package edu.ku.it.si.blackboardcoursesforuser.axis). This class uses the Axis2 framework to create the web service client classes from a web service WSDL file. The arguments to the WSDL2Code.main method determine where the generated classes are written to and what WSDL file is used. See the comments in class AxisCodeGenerator for more details.

Since the example application already includes the generated web service client classes, we don't need to run the AxisCodeGenerator. But if you would like to see how it works, you can delete

the classes in package edu.ku.it.si.bbcoursews.generated and then run the AxisCodeGenerator class from within Eclipse to recreate the Course web service client classes.

Using The Generated Blackboard Web Service Client Class

If you examine the ContextWSStub class (package edu.ku.it.si.bbcontextws.generated) you'll see that it contains methods that match up with the operations the web service declared (see part 2 of this tutorial for how to find the operations of a Blackboard web service).

In addition to the methods there are numerous static inner classes. These classes represent additional data types needed to interact with the web service. For example the ContextWSStub class has a loginTool method. This method is used to login to the Blackboard web services using a proxy tool (see part 1 of this tutorial). The method requires an argument of type LoginTool and returns an object of type LoginToolResponse. Both of those data types are defined as public static inner classes in the ContextWSStub class.

To understand how to use the generated web service client class, you should review the generated code, the signature of the methods, the return type of the methods, and the additional inner classes.

Using The Context Web Service To Initialize A Session and Login To Blackboard

The first step in using any Blackboard web service is for your client application to initialize a session and login. You use the Context web service to do this work. If you examine class BlackboardCoursesForUserServiceImpl you'll see how to use the Context web service client class (ContextWSStub) to set up the security part of the SOAP envelope, initialize the session, store the returned session id in a CallBackHandler class, and then login using a previously registered proxy tool (see part 1 of this tutorial). There are extensive comments in the code to explain each step. The result of logging in will either be a boolean value (true meaning a successful login) or an exception.

Note the values for the arguments being passed into the getBlackboardCoursesForUser are originally set in the bbws.properties file.

Getting The Course Names For A Blackboard User

After logging in successfully, the code uses the ContextWSStub class's getMemberships method (see getBlackboardCoursesForUser method of class BlackboardCoursesForUserServiceImpl). See the code for how the getMemberships method is called and its return object processed. Remember that the data types used in the code are all defined in the ContextWSStub class that was generated by Axis2 using the Context web service WSDL file.

The end result of calling the Context web services getMemberships operation is the program has an array of CourseIdVO objects. Each CourseIdVO is storing a course id String, which is the external id value for a Blackboard course. Since our goal is to get the Blackboard course names

for each of these course ids, we need to use the Course web service, which has a method getCourse.

If you examine the code in BlackboardCoursesForUserServiceImpl you'll see that we setup the CourseWSStub object in the same way we setup the ContextWSStub object so that a SOAP request made to the Blackboard Course web service will have the same session id value that was previously setup by the Context web service. That session id value has been authorized to access the web services and methods of the proxy tool that was used to login. So for the CourseWSStub object we reuse the same CallBackHandler object as the ContextWSStub used.

One other step we need to take before calling the Course web service's getCourse method is to create GetCourse and CourseFilter objects. The CourseFilter object is used to specify how we want Blackboard to filter the Blackboard courses when we call the getCourse method. For example we want to filter by course ids, so we set the filter type to 3 (see courseFilter.setFilterType(3) in the example code).

How did we know what value to set the filter type to? You have to review the JavaDoc for the CourseFilter class's setFilterType method to find a list of integer values and what each value represents (see part 2 for how to find the JavaDoc for the Blackboard web service classes).

The result of calling the getCourse method of the Course web service is that we will have an array of CourseVO objects. We can loop over every CourseVO object stored in the array and get the name of the course. Again see the CourseWSStub class for the definition of the CourseVO class.

Running The Example Application

Be sure to put in your values for the properties in bbws.properties. These values are necessary so that the program knows where to find your Blackboard installation and the other information needed to register the proxy tool. See the ReadMe.txt file in the example application for further information on configuring and running the example application.

Summary

Blackboard web services are powerful, but somewhat complex to use. You have to understand SOAP-based web services, SOAP web services Security (see securing SOAP messages with Rampart), the APIs for the Blackboard web services, how to generate the web service client classes, and then how to initialize a session and login.

Creating a client application that will interact with the Blackboard web services is not a simple task. Hopefully this series of tutorials will help you develop enough of an understanding so that you can create your own applications that use Blackboard web services.