Grouper Book - CXF web service client

CXF as a client for the Grouper Webservice

The rationale of using open standards such as webservices is that they promote interoperability. Using a client generated by the grouper-ws build process to test the server is great, but we need to go further to test using it with other implementations of webservices clients. The first one we're going to use is CXF, a popular alternative to Axis which is targetted more at modern java development idioms (preferring configuration over code, sensible defaults etc.). Here are the steps:

- Download CXF from http://cxf.apache.org/download.html (I'm using version 2.3.1, earlier versions have a library conflict which prevents wsdl2java working against the grouper wsdl file)
- Extract the download to a directory - I'll be calling this CXF_HOME
- Open a command line and go to CXF_HOME/bin (we'll be using the tools in this directory to auto-generate a client)
- Download the wsdl file from your groups-ws installation (I'm assuming you've got it installed on Tomcat on your local machine listening on port 8080, with the default name for the webapp) at http://localhost:8080/grouper-ws/services/GrouperService?wsdl You will need to autenticate to download it. Assuming you've got the tomcat-users.xml content from the quickstart in you TOMCAT_HOME/conf, use GrouperSystem/123. We are downloading the wsdl file because the CSX tools don't support authentication when retireving a WSDL file for client generation, so save the file to somewhere easy to get to.
- In CXF_HOME/bin invoke the wsdl2java command with:

```
./wsdl2java -verbose -client -d <PATH FOR GENEARATED JAVA CLIENT> <PATH TO WSDL FILE>
```

so if I downloaded the wdsl to /tmp/grouperService.xml and want the output to go into /tmp/grouperCxfClient I'd use:

Linux:

```
./wsdl2java -verbose -client -d /tmp/grouperCxfClient /tmp/grouperService.xml
```

CXF will now generate a set of client classes for accessing the webservice. For the next step I import these into a java project using may favourite IDE (Eclipse). You need to add the CXF libraries to the build path for this project - they're in CXF_HOME/lib and CXF_HOME/lib/endorsed.

Java 6 includes an earlier version of the JAXB libraries used by CXF, so you need to either:

- Use Java5
- Use Java6 and endorse the libraries in CXF_HOME/lib/endorsed

The way to endorse libraries for a java command is to use the -Djava.endorsed.dirs= switch at the command line. Setting it up in an IDE is slightly different. In Eclipse I add them to list of system libraries in the JRE definition (Help\Preferences\Java\Installed JREs\Edit the one you want to add them to, move them to the top of the list), and add a -Djava.endorsed.dirs=<Path to the endorsed libs> argument into the Default VM Arguments box for good measure (the latter does seem to be necessary, and makes it obvious to me that I'm modified this JRE)

In your IDE you can now edit the generated client files that you imported to change the location of the WSDL file to the one on the server. To do this, open up GrouperService in the edu.internet2.middleware.grouper.ws.soap package, create a class and change the wdslLocation and url variables to point to http://localhost:8080/grouper-ws/services/GrouperService?wsdl. This is not required (things will work perfectly well with the WSDL file saved locally), but it will make it easier to update. Now create a class in a test package and call it GetMembersSimple, add the code as follows:

```
import java.net.URL;
import java.util.List;

import javax.xml.namespace.QName;

import edu.internet2.middleware.grouper.ws.soap.GrouperService;
import edu.internet2.middleware.grouper.ws.soap.GrouperServicePortType;
import edu.internet2.middleware.grouper.ws.soap.xsd.WsSubject;

public class GetMembersSimple {
        private static final QName SERVICE_NAME = new QName("http://soap.ws.grouper.middleware.internet2.edu/",
"GrouperService");

        /**
         * @param args
         */
        public static void main(String[] args) {
                URL wsdlURL = GrouperService.WSDL_LOCATION;

                GrouperService ss = new GrouperService(wsdlURL, SERVICE_NAME);

        GrouperServicePortType port = ss.getGrouperServiceHttpSoap12Endpoint();

        String _getMembersLite_clientVersion = "v1_6_000";
        String _getMembersLite_groupName = "<Group name eg: stem:group>";
        String _getMembersLite_groupUuid = "";
        String _getMembersLite_memberFilter = "All";
        String _getMembersLite_actAsSubjectId = "GrouperSystem";
        String _getMembersLite_actAsSubjectSourceId = "";
        String _getMembersLite_actAsSubjectIdentifier = "";
        String _getMembersLite_fieldName = "";
        String _getMembersLite_includeGroupDetail = "T";
        String _getMembersLite_includeSubjectDetail = "F";
        String _getMembersLite_subjectAttributeNames = "";
        String _getMembersLite_paramName0 = "";
        String _getMembersLite_paramValue0 = "";
        String _getMembersLite_paramName1 = "";
        String _getMembersLite_paramValue1 = "";
        String _getMembersLite_sourceIds = "";

        edu.internet2.middleware.grouper.ws.soap.xsd.WsGetMembersLiteResult _getMembersLite__return = port.
getMembersLite(_getMembersLite_clientVersion, _getMembersLite_groupName, _getMembersLite_groupUuid,
_getMembersLite_memberFilter, _getMembersLite_actAsSubjectId, _getMembersLite_actAsSubjectSourceId,
_getMembersLite_actAsSubjectIdentifier, _getMembersLite_fieldName, _getMembersLite_includeGroupDetail,
_getMembersLite_includeSubjectDetail, _getMembersLite_subjectAttributeNames, _getMembersLite_paramName0,
_getMembersLite_paramValue0, _getMembersLite_paramName1, _getMembersLite_paramValue1,
_getMembersLite_sourceIds);
        List <WsSubject> subjects = _getMembersLite__return.getWsSubjects();
        for( WsSubject subject : subjects) {
                System.out.println(subject.getId().getValue());
        }
        }

}
```

This code is based on code found in the generated class GrouperServicePortType_GrouperServiceHttpSoap12Endpoint_Client. Remember to change the _getMembersLite_groupName to a group that exists in your Grouper repository.

Before runnng the code we need to configure CXF to authenticate to the webservice container with basic auth. I do this by creating a file cxf.xml in the root of the source code tree in my project and pasting the following into it (assumes you have default values):

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:http="http://cxf.apache.org/transports/http
/configuration"
        xmlns:sec="http://cxf.apache.org/configuration/security"
        xsi:schemaLocation="http://cxf.apache.org/configuration/security
           http://cxf.apache.org/schemas/configuration/security.xsd
           http://cxf.apache.org/transports/http/configuration
           http://cxf.apache.org/schemas/configuration/http-conf.xsd
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans.xsd">
        <http:conduit name="http://localhost:8080/grouper-ws/services/.*">
                <http:client AutoRedirect="true" Connection="Keep-Alive" />
                <http:authorization>
                        <sec:UserName>GrouperSystem</sec:UserName>
                        <sec:Password>123</sec:Password>
                </http:authorization>
        </http:conduit>
        <http:conduit
                name="{http://soap.ws.grouper.middleware.internet2.edu/}GrouperServiceHttpSoap12Endpoint.http-
conduit">
                <http:client AutoRedirect="true" Connection="Keep-Alive" />
                <http:authorization>
                        <sec:UserName>GrouperSystem</sec:UserName>
                        <sec:Password>123</sec:Password>
                </http:authorization>
        </http:conduit>
</beans>
```

The first conduit enables CXF to authenticate to retireve the WSDL file. The second to authenticate when executing methods against the web service. The documentation for conduits is at . The documentation at http://cxf.apache.org/docs/client-http-transport-including-ssl-support.html would suggest the the second conduit is not required if the URL in the first is changed to include the whole web service, but I have found that it is.

Now run the code, and you should see the list of member IDs of the group printed out. A similar pice of code that uses the non-lite method looks like:

```
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.JAXBElement;
import javax.xml.namespace.QName;

import edu.internet2.middleware.grouper.ws.soap.GrouperService;
import edu.internet2.middleware.grouper.ws.soap.GrouperServicePortType;
import edu.internet2.middleware.grouper.ws.soap.xsd.WsGetMembersResult;
import edu.internet2.middleware.grouper.ws.soap.xsd.WsGroupLookup;
import edu.internet2.middleware.grouper.ws.soap.xsd.WsParam;
import edu.internet2.middleware.grouper.ws.soap.xsd.WsSubject;
import edu.internet2.middleware.grouper.ws.soap.xsd.WsSubjectLookup;

public class GetMembers {
        private static final QName SERVICE_NAME = new QName("http://soap.ws.grouper.middleware.internet2.edu/",
"GrouperService");

        /**
         * @param args
         */
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                URL wsdlURL = GrouperService.WSDL_LOCATION;
                GrouperService ss = new GrouperService(wsdlURL, SERVICE_NAME);
        GrouperServicePortType port = ss.getGrouperServiceHttpSoap12Endpoint();
        String clientVersion = "v1_6_000";
        List<WsGroupLookup> wsGroupLookups = new ArrayList<WsGroupLookup>();
        WsGroupLookup wsGroupLookup = new WsGroupLookup();
        JAXBElement<String> value = new JAXBElement<String>(new QName("groupName"), String.class, "<Group name
e.g. stem:name>");
        wsGroupLookup.setGroupName(value);
        wsGroupLookups.add(wsGroupLookup);
        String memberFilter = "All";
        WsSubjectLookup actAsSubject = new WsSubjectLookup();
        value = new JAXBElement<String>(new QName("subjectId"), String.class, "GrouperSystem");
        actAsSubject.setSubjectId(value);
        String fieldName = "";
        String includeGroupDetail = "F";
        String includeSubjectDetail = "T";
        List<String> subjectAttributeNames = new ArrayList<String>();
        subjectAttributeNames.add("a");
        subjectAttributeNames.add("name");
        List<WsParam> params = null;
        List<String> sourceIds = null;
        edu.internet2.middleware.grouper.ws.soap.xsd.WsGetMembersResults _getMembers__return = port.getMembers
(clientVersion, wsGroupLookups, memberFilter, actAsSubject, fieldName, includeGroupDetail,
includeSubjectDetail, subjectAttributeNames, params, sourceIds);
        List<WsGetMembersResult> results = _getMembers__return.getResults();
        for(WsGetMembersResult result : results) {
                List <WsSubject> subjects = result.getWsSubjects();
                for( WsSubject subject : subjects) {
                        System.out.println(subject.getId().getValue());
                }
            }
        }

}
```

The sample code above does things that I wouldn't necessarily do in production, such as process the WSDL just before executing the method, but it is a working base from which to start.