Chapter 1. How to use Axis with Enhydra applications

Table of Contents

Enhydra is now Axis enabled, so it is possible to build Axis based web service applications in the Enhydra environment. Actually, you can expose any method in your Ehydra application, so that it provides web services. The concept is illustrated by PhoneBook and PhoneBookClient applications.

PhoneBook is quite an ordinary Enhydra application. It uses a database to store and retrieve some data, and it presents these data via Enhydra presentation objects. There is no special code which makes it a web service server application, except the server-config.wsdd file, which describes its web service capabilities – methods exposed to remote (Axis) calls.

On the other hand PhonBookClient application consists of a standard Enhydra presentation layer and a business layer, which incorporates some (generated) Axis related classes, located in the phoneList.business.axis package. It uses these classes to communicate with the PhoneBook Enhydra application (which can be located on the same host, or on the remote one), thus obtaining the necessary data, which it presents to the user via its own presentation objects.

The purpose of this document is to show you how to build the service description server-config.wsdd and how to use it to build the client side Axis applications. As an example, we shall use the PhoneBook application, and assume that we want to expose (some of) the methods from the phoneList.business.PhoneList class. This is the class which communicates with the database through applications data layer. We shell expose the methods for listing, adding, deleting and modifying the data, so that they can be used from a remote client via Axis.

Setting the classpath

First you must set the classpath, so that the necessary axis jars are accessible, as well as the class which will the clients access. Provided that you have set the ENHYDRA_HOME environment variable (e.g. set ENHYDRA_HOME=C:\enhydra-enterprise\multiserver), do something like:

```
set ENHYDRA_HOME=C:\enhydra-enterprise\multiserver
```

```
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\tools\axis-ant.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\commons\jonas\axis\axis.jar
    set class-
path=%classpath%;%ENHYDRA_HOME%\lib\commons\jonas\jakarta-commons\commons-discovery.jar
    set class-
path=%classpath%;%ENHYDRA_HOME%\lib\commons\jonas\jakarta-commons\commons-logging-api.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\commons\j2ee\jaxrpc.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\commons\j2ee\saaj.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\commons\j2ee\wsdl4j.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\dods\dbmanager.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\dods\dbmanager-api.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\dods\dsconnection.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\dods\stdcaches.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\dods\stdconnection.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\dods\stdtransaction.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\enhydra\eaf_api.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\common\j2ee\mail.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\common\j2ee\activation.jar
    set class-
path=%classpath%;%ENHYDRA_HOME%\webapps\autoload\phoneBook\WEB-INF\lib\phoneBook.jar
    set classpath=%classpath%;%ENHYDRA_HOME%\lib\ext\commons-logging.jar
```

Building the Web Service Description Language file pertaining to the selected class

Start the following command (one line):

```
    java org.apache.axis.wsdl.Java2WSDL -o phoneBook.wsdl  -m
getLists,modifyPerson,getPersonData,deletePerson,addPerson -l
 "http://localhost:9000/phoneBook/PhoneBook" -n "urn:phoneBook"  phoneList.spec.PhoneList
```

This command builds phoneBook.wsdl file that describes the phoneList.business.PhoneList class, which provides a web service located at the url

http://localhost:9000/phoneBook/PhoneBook. [http://localhost:9000/phoneBook/PhoneBook]

To make things simple and to avoid writing special serialisers/deserialisers we have exposed all the methods except getById method, (you can use methods returning simple data types, as well as arrays and jbeans without writing the serialiser/deserialiser methods):

Build the client classes and deploy/undeploy descriptors

Use the following (one line) command to generate the client classes, as well as deployment descriptors, using the phoneBook.wsdl file as input:

```
    java org.apache.axis.wsdl.WSDL2Java -o . -d Application -S false -p
phoneList.business.axis phoneBook.wsdl
```

Deploy the service

If Enhydra is not running, start it and then issue the following command (assuming Enhydra uses the port 9000):

```
    java org.apache.axis.client.AdminClient  -l http://localhost:9000/phoneBook/PhoneBook
phoneList\business\axis\deploy.wsdd
```

This builds the necessary server-config.wsdd file in the Enhydra working directory (e.g. %ENHYDRA_HOME%\webapps\phoneBook\WEB-INF). You should put it into the corresponding war file, so that it is not lost across restarts of the Enhydra server.

Test the service

Point your browser to the web service url, with “?wsdl” appended – if everything is working, this should yield the wsdl description of your newly deployed service:

http://localhost:9000/phoneBook/PhoneBook?wsdl

Build the client side application

You have already generated an axis part of the clinet application in step 4 – package **phoneList.business.axis**. To build a simple web service client application, use this package and add a simple main class to it, like this:

```
package phoneBook;
import java.util.Vector;

public class ATest {

  public static void main(String[] args) {

    try {
      PhoneListService service = new PhoneListServiceLocator();
      PhoneList phoneBook;
      phoneBook=service.getPhoneBook();
      Object [][] v ;

      v = phoneBook.getLists();
      for (int i = 0; i < v.length; i++) {
        for (int j = 0; j < v[i].length; j++)
          System.out.println("Vector values: " + i + " " + j + " " + v[i][j]);
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

What should your clients know about your web service?

The clients should know only the url of the web service. Since they can obtain its wsdl description by pointing to the aforementioned url, it is enough to get everything they need to build a client application. Furthermore, if they use an IDE capable of generating web service applications, they can supply this url to the IDE, and thus generate the client classes, instead of doing it manualy.

TableApp and TableAppClient applications

The TableApp and TableAppClient pair of applications are also includede into EnhydraDemos. These applications transfer Document objects from server to client. The client then extracts a table node from the Document object and inserts it in its own presentation object.