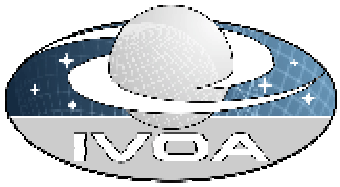

Part 1: Consuming XML Web Services in Java with or without Axis. Part2: The server Side.
Version 0.1

IVOA Working Draft
2003-10-12

This version:
http://putinURLhere

Latest version:
http://putinURLhere

Previous versions:
http://putinURLhere
http://putinURLhere

Editors:
André Schaaff

Authors:
André Schaaff

Abstract

The abstract should describe the intent of the document and summarize the main points. The abstract should typically be no more than a paragraph in length.

Status of this document

This is a Working Draft. The first release of this document was 12 October 2003.

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress." A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/docs/.

Acknowledgments

Please give credit to all major contributors to the document.

Contents

Abstract 1
Status of this document 1
Acknowledgments 2
Contents 2
1 Introduction 2
2 Accessing XML Web Services in Java 2
3 Accessing XML Web Services in Java without Axis 4
4 Inside the server 6
4.1 Axis 6
4.2 Simple deployment file example 6
5 References 7

1 Introduction

In a first step we will show how to create a Java client for a Web Service like Spectra.
In a second time we will show how to consume a Web Service in a Java Applet, without Axis or .Net support…
In the last step, after several examples (concerning the client side) given by all the contributors, we will give some information about the server side.

2 Accessing XML Web Services in Java

In this part we will show how to implement a client in Java for the Spectra Web Service.

We suppose that Java (1.4.* for example) is available on the computer and that it is accessible through the command line.

First step : Axis libraries

We must download the last (or recent) Axis package and add the contained libraries in the CLASSPATH.

Second step : WSDL

We use the WSDL description of the Web Service to generate all the Java classes (Proxy class and data structure classes):

C:\> java org.apache.axis.wsdl.WSDL2Java http://skyservice.pha.jhu.edu/devel/wave/wave.asmx?WSDL

It generates a new directory <edu>.

Third step : we can now write the Java client file

The new Java classes edu* must be added to the CLASSPATH

```
import edu.jhu.pha.skyservice.wave.*;
import edu.jhu.pha.skyservice.*;

public class Spectra {
  public Spectra() {

    try {
// Proxy, URL of the service
      System.out.println("\n-->Spectra Web Service URL");
      WaveLocator wl = new WaveLocator();
      System.out.println(wl.getWaveSoapAddress());
      WaveSoap p = wl.getWaveSoap();

// Retrieving a spectrum by its ID
      System.out.println("\n-->Retrieving a spectrum by its ID (example 200000)");
      Spectrum s = p.getSpectrum(200000, true);
      System.out.println(s.getName());
      for (int i = 0; i < 10; i++) {
        System.out.println
          ("\t" + s.getPoints().getPoint()[i].getWavelength() + " " +
           s.getPoints().getPoint()[i].getValue());
      }
// Retrieving a spectrum by its ID (in VOTable format)
      System.out.println("\n-->Retrieving a spectrum by its ID (example 200000) in
VOTable format");
      VOTABLE v = p.getSpectrumVoTable(200000, false);
      System.out.print(v.getDESCRIPTION() + ": ");
      System.out.println(v.getRESOURCE()[0].getTABLE()[0].getDATA().
                         getTABLEDATA().getTR()[0].getTD()[1].getValue());

// Returning spectra in a virtual observational field
      System.out.println("\n-->Returning spectra in a virtual observational field");
      ArrayOfSpectrum aos = p.findSpectraCone(180, 0, 5, false);
      Spectrum[] sa = aos.getSpectrum();
      System.out.println("# of objects found: " + sa.length);
      Spectrum c = null;
      for (int si = 0; si < sa.length; si++) {
```

```
        c = sa[si];
        System.out.println("\t" + c.getRa() + "\t" + c.getDec());
      }
    }
    catch (Exception f) {};
  }

  public static void main(String[] args) throws Exception {
    new Spectra();
  }
}
```

Last step : compilation and execution

3 Accessing XML Web Services in Java without Axis

The Applet Use Case

How to use an XML Web Service if you need to load jars like axis.jar which are perhaps 10 or 50 times bigger than the Applet itself ?

For this demo we propose to use a very simple Applet which has just a few fields and buttons.

Applet code source

```
.
/**
 * <p>Title: How to consume a Web Service in an Applet </p>
 * <p>Description: VizieRApplet </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: CDS</p>
 * @author André Schaaff
 * @version 1.0
 */

import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;

/*------------- Added ----------------*/
import org.ksoap.*;
import org.ksoap.transport.*;
/*------------------------------------*/

public class VizieRApplet
    extends Applet
    implements ActionListener, ItemListener {
```

```
/*------------- Added ----------------*/
static final String serviceNamespace =
    "http://cdsws.u-strasbg.fr/axis/services/VizieR";
/*------------------------------------*/

// 1 text area
TextArea ta1 = new TextArea(20, 2000);

/*------------- Added ----------------*/
public void VizieRClient() {

  try {
    System.out.println("Creating request object");
    HttpTransportSE transport = new HttpTransportSE(serviceNamespace,
        "VizieR");
    transport.debug = true;

    SoapObject request = new SoapObject(serviceNamespace, "metaAll");

    ta1.setText("Invoking....");

    String result = transport.call(request).toString();

    ta1.setText(result);
  }
  catch (Exception e) {
    System.out.println("Error: " + e.toString());
  }
}
/*------------------------------------*/

/** init */
public void init() {

  setLayout(new BorderLayout());
  Panel pc = new Panel();
  add(pc, "Center");
  Label label1 = new Label("Name to resolve");
  Label label2 = new Label("Output format (x, H, p)");
  Font bigFont = new Font("Serif", Font.PLAIN, 96);
  Font mediumFont = new Font("Serif", Font.PLAIN, 18);

  Panel basePanel = new Panel();
  add(basePanel, "West");
  basePanel.setLayout(new GridLayout(1, 1));

  basePanel.add(ta1);

  Button bRaz = new Button("Invoke");
  add(bRaz, "South");
  bRaz.addActionListener(this);
```

```
  }

  /** uniquement pour gérer le bouton "raz" */
  public void actionPerformed(ActionEvent e) {
    String s = e.getActionCommand();

    /*------------ Added ----------------*/
    VizieRClient();
    /*-----------------------------------*/
  }

  /** pour gérer les changements de base */
  public void itemStateChanged(ItemEvent e) {
  }
}
```

4 Inside the server

4.1 Axis

How to install the server side with Axis/Tomcat ?

Download the last Axis and Tomcat packages, copy the axis (axis/webapps/axis) directory into the Tomcat/webapps directory and it is done…

It is now possible to implement a Web Service.

How to deploy services ?

jws deployment : instant deployment of a Java file (renamed to .jws)

wsdd : deploy.wsdd and undeploy.wsdd files are generated with a command like :

java org.apache.axis.wsdl.WSDL2java –o . –dSession –s -v http://....

4.2 Simple deployment file example

```
<!-- Use this file to deploy some handlers/chains and services      -->
<!-- Two ways to do this:                                           -->
<!--   java org.apache.axis.client.AdminClient deploy.wsdd          -->
<!--      after the axis server is running                          -->
<!-- or                                                             -->
<!--   java org.apache.axis.utils.Admin client|server deploy.wsdd   -->
<!--      from the same directory that the Axis engine runs         -->

<deployment name="UCDList" xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
```

```
  <service name="UCDList" provider="java:RPC">

     <parameter name="wsdlTargetNamespace" value="urn:UCDList"/>
     <parameter name="wsdlServiceElement" value="SesameService"/>
     <parameter name="wsdlServicePort" value="UCDList"/>
     <parameter name="className" value="UCDList"/>
     <parameter name="wsdlPortType" value="UCDList"/>


     <operation name="UCDList" xmlns:operNS="urn:UCDList"
returnQName="return" returnType="RTypeNS:string"
xmlns:rtns="urn:UCDList">
     </operation>

      <parameter name="allowedMethods" value="*"/>
      <parameter name="scope" value="Session"/>
  </service>
</deployment>
```

And to undeploy :

```
<undeployment name="UCDList" xmlns="http://xml.apache.org/axis/wsdd/">
  <service name="UCDList"/>
</undeployment>
```

5 References

Axis - http://ws.apache.org/axis/
Java - http://java.sun.com
kSOAP- http://ksoap.enhydra.org/
SOAP - http://www.w3.org/2000/xp/Group/
Tomcat- http://jakarta.apache.org/tomcat/
WSDL - http://www.w3.org/2002/ws/