ACCESSING WATERONEFLOW WITH JAVA

August 25, 2011

by:

Dr. Tim Whiteaker
Center for Research in Water Resources
The University of Texas at Austin

Distribution

Funding

Funding for this document was provided by the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) under National Science Foundation Grant No. EAR-0622374. In addition, much input and feedback has been received from the CUAHSI Hydrologic Information System development team. Their contribution is acknowledged here.

Table of Contents

Introduction

This walkthrough shows how to build Java classes to access streamflow data from the USGS NWIS Daily Values Web service. This service conforms to CUAHSI's WaterOneFlow standard and returns data in WaterML format. Once you learn how to access this service, accessing other WaterOneFlow services from the dozens of other organizations publishing water data using WaterOneFlow is easy since they all use the same standard.

In particular, your program will obtain daily streamflow values for Big Rock Creek near Valyermo, California, for the year 2001. The program will output the site name for this location and the time series of streamflow values. Of course this demonstration class is very simple and perhaps not that useful, but the ideas can be extended to support the particular needs of your organization.

Computer and Skill Requirements
To complete this exercise, your computer must meet the following requirements:

- **Working Internet connection**
- **Java Platform (JDK) 7** (http://java.sun.com/javase/downloads/)
- **NetBeans IDE 7** (http://www.netbeans.org)

To use Web services in NetBeans, you will need the SOAP Web Services plugin. This may have been included with your installation of NetBeans.

To check for the SOAP Web Services plugin:

1. In **NetBeans**, click **Tools | Plugins**.
2. Click the **Installed** tab. If you see **Java Web and EE** or **Soap Web Services**, make sure it is active. (Soap Web Services is included with Java Web and EE.)
3. If you don't have the plugin, click the **Available Plugins** tab.
4. In the **Search** box, enter **soap**.
5. Find the SOAP Web Services plugin in the search results and click to install it.

This exercise assumes that you have some familiarity with general programming concepts and Java.

Note
The source code for the FlowReporter class created in this exercise is located in the Appendix.

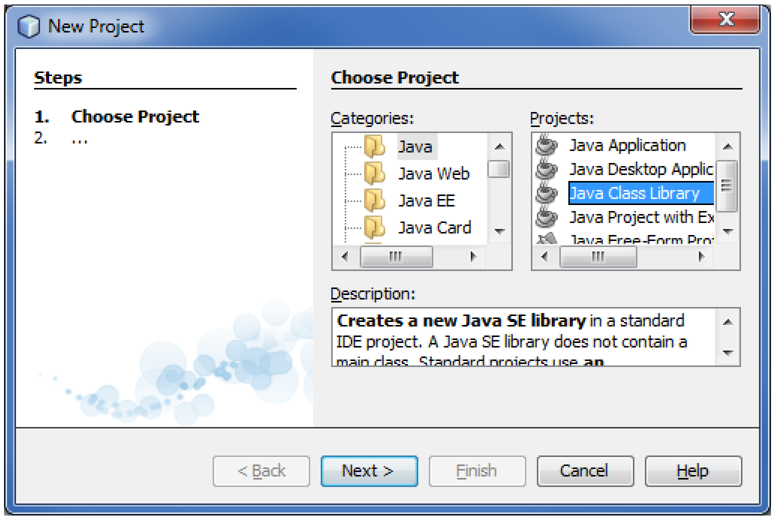
Creating a New Project

The main steps in this walkthrough are to create a new project, add a Web service client for the USGS NWIS Daily Values Web service, and create another class to use the client to access the data and display the result.

To create a new project:

1. Start **NetBeans IDE**.
2. Click the **File** menu and then click **New Project**.

3. In the New Project dialog, select **Java** in the Categories pane. In the Projects pane, select **Java Class Library** (Figure 1).
4. Click **Next**.
5. In the New Java Class Library dialog, enter **`NwisExample`** as the Project Name.
6. Enter the path in which you want the project folder to be created in Project Location. The IDE will create a subfolder at that location called NwisExample, which will contain all files used in the project.
7. Click **Finish**.



Description:

Creates a new Java SE library in a standard IDE project. A Java SE library does not contain a main class. Standard projects use **an**

Figure 1 Creating a Java Class Library

Creating a Web Service Client

With the project established, you will now create a client for the NWIS WaterOneFlow Web service. The service is described by a file called a WSDL, which tells programs how to interact with the service and what the service can do. The WSDL for the NWIS Daily Values service is located at:
http://river.sdsc.edu/waterOneFlow/NWIS/DailyValues.asmx?WSDL

The IDE will create the service client by interpreting the WSDL, resulting in a class that enables other objects in your program to communicate with the Web service.

To create the Web service client:

1. In the Projects pane, right-click **NwisExample**, point to **New**, and click **Web Service Client** (Figure 2).

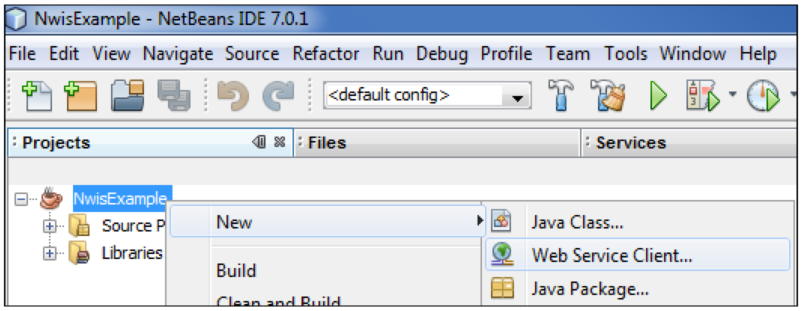
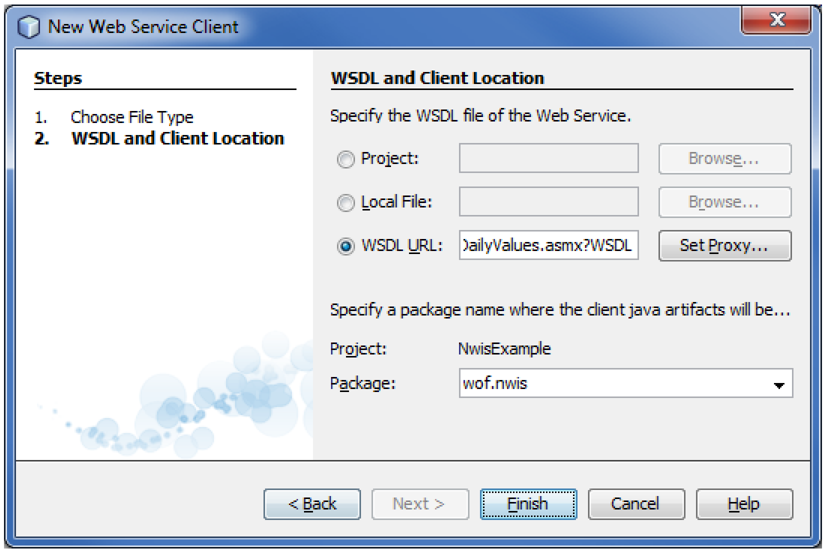


Figure 2 Adding a Web Service Client

2. In the New Web Service Client dialog, choose the **WSDL URL** option and enter the following URL: http://river.sdsc.edu/waterOneFlow/NWIS/DailyValues.asmx?WSDL
3. For the Package, enter **wof.nwis** (Figure 3).
4. Click **Finish** to create the class.


New Web Service Client

Steps
1. Choose File Type
2. **WSDL and Client Location**

WSDL and Client Location

Specify the WSDL file of the Web Service.

Project:
Local File:
WSDL URL:)ailyValues.asmx?WSDL

Specify a package name where the client java artifacts will be...

Project: NwisExample
Package: wof.nwis

Figure 3 Creating the Web Service Client

Creating a Class To Report Flow Data

In this section you will create a new class, report.FlowReporter, which accesses the Web service client you just created. The class will download a time series of daily streamflow values at Big Rock Creek near Valyermo, California, for the year 2001. The unique USGS identifier for this location is 10263500, and this identifier belongs to the NWIS network. In CUAHSI terms, that identifier is called a site code. Likewise, the variable code for streamflow is 00060, and that code belongs to the NWIS variable vocabulary. You will hard code both of these values into the class. A shorter way of saying the above is that our site is NWIS:10263500 and our variable is NWIS:00060. You will also hard code the time span (the year 2001). In a more robust application, you would let the user supply these parameters. The class will output the name of the site and the time series of values.

Tip
Refer to the Appendix for the source code if you get stuck.

To create the FlowReporter class:

1. In the Projects window, right-click **NwisExample**, point to **New**, and click **Java Class**.
2. In the New Java Class dialog, enter `FlowReporter` as the Class Name, and `report` as the Package.
3. Click **Finish**.

The class is created, and code for the class is shown in the code pane. Now you will add a method to make the Web service requests. Fortunately, the IDE can create the code for you.

To create the method that accesses the Web service client:

1. Right-click in the code for FlowReporter (e.g., after the open brace following the FlowReporter class declaration) and click **Insert Code** (Figure 4).

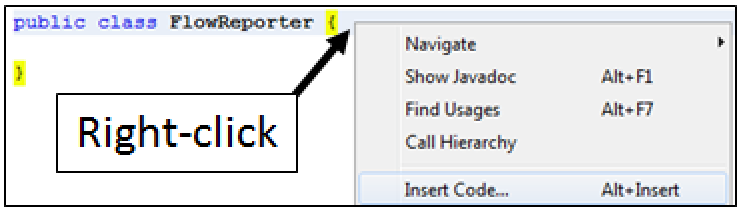


Figure 4 Inserting code

2. In the menu that opens, click **Call Web Service Operation**.
3. In the Select Operations to Invoke dialog, expand the tree for WaterOneFlow SOAP and select **GetValuesObject**.
4. Click **OK**.

After you click OK, the IDE generates code for calling the GetValuesObject method from the NWIS Web service. The method includes parameters (e.g., location) that will be sent to the service.

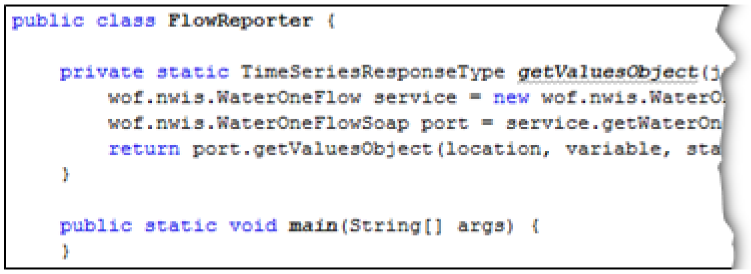
Now you will program a "main" method, which is the default method that runs when the class is invoked.

To program the main method:

1. In the source code for the FlowReporter class, add the following code.

```
public static void main(String[] args) {
}
```

Figure 5 shows the result.



```
public class FlowReporter {

    private static TimeSeriesResponseType getValuesObject(j
        wof.nwis.WaterOneFlow service = new wof.nwis.WaterO
        wof.nwis.WaterOneFlowSoap port = service.getWaterOn
        return port.getValuesObject(location, variable, sta
    }

    public static void main(String[] args) {
    }
```

Figure 5 FlowReporter class and its methods

Now you will hard code the parameters for our site of interest, namely:

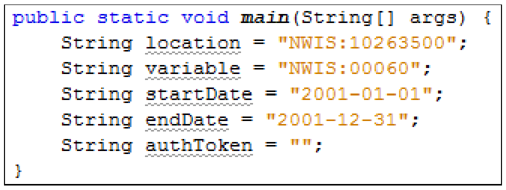
- location – the network name and site code for the site of interest, i.e., NWIS:10263500
- variable – the vocabulary and variable code for the variable of interest, i.e., NWIS:00060
- startDate – 2001-01-01
- endDate – 2001-12-31
- authToken – Token required for data access. This service is public, so the token can be blank.

Remember, you are hard coding these parameters for this simple example application, but a more robust application would read these parameters as inputs from the user.

2. In the main method, add the code below to define the parameters for time series retrieval.

```
String location = "NWIS:10263500";
String variable = "NWIS:00060";
String startDate = "2001-01-01";
String endDate = "2001-12-31";
String authToken = "";
```

A screenshot from the code editor is shown below.



```
public static void main(String[] args) {
    String location = "NWIS:10263500";
    String variable = "NWIS:00060";
    String startDate = "2001-01-01";
    String endDate = "2001-12-31";
    String authToken = "";
}
```

Figure 6 Specifying query parameters

You will now download the time series and output the site name to the Output window in the IDE.

3. To the end of the main procedure, add the following code.

```
// Download time series object
TimeSeriesResponseType result = getValuesObject(
        location, variable, startDate, endDate, authToken);
```

```
// Show site name
wof.nwis.SiteInfoType site = (wof.nwis.SiteInfoType)
        result.getTimeSeries().getSourceInfo();
System.out.println("Site name = " + site.getSiteName());
```

The code above calls the getValuesObject method to return a time series object from the Web service. This object contains the time series values and also important metadata such as descriptions of the site and variable associated with the values. Then the code assigns the source info from the time series object to a site object. The source info must be cast as a SiteInfoType because source info could be one of several types such as sites or regions in space. Finally, the code prints the site name to the IDE Output window.

To output the datetimes and values in the time series, you will use a List object.

4. Just above the existing import statement for wof.nwis.TimeSeriesResponseType, add an import statement for the List library as in the code below.

```
import java.util.List;
```

5. In the code for the main method, append the following code.

```
// Make a header for showing datetimes and values
System.out.format("%20s %10s", "DateTime", "Value");
System.out.println();

// Show datetimes and values
List<wof.nwis.ValueSingleVariable> values =
    result.getTimeSeries().getValues().getValue();

for (wof.nwis.ValueSingleVariable value : values ) {
    System.out.format("%20s %10.4f",
            value.getDateTime().toString(), value.getValue());
    System.out.println();
}
```

The code above prints a header for showing datetimes and values in the Output window. Then it accesses the value list from the time series object. Then it loops through each value and prints the result to the Output window.

6. Click the **File** menu and then click **Save All**.

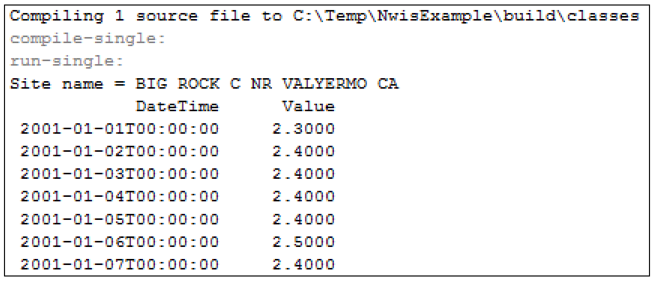
With the code finished, all that is left is to compile and run the file.

RUNNING THE CODE

To test service access, you'll run the code in the FlowReporter class. Let's make sure things are running smoothly by cleaning and building the project before running the FlowReporter code.

1. In the Projects window, right-click **NwisExample** and then click **Clean and Build**.
2. In the Projects window, right-click **FlowReporter.java** and then click **Run File**.

In a moment, you will see the results of the Web service request as text in the Output window (Figure 7). The flow values are in cubic feet per second, which you can also determine from the time series object. We left it out of this exercise to keep things simple.



```
Compiling 1 source file to C:\Temp\NwisExample\build\classes
compile-single:
run-single:
Site name = BIG ROCK C NR VALYERMO CA
              DateTime        Value
 2001-01-01T00:00:00     2.3000
 2001-01-02T00:00:00     2.4000
 2001-01-03T00:00:00     2.4000
 2001-01-04T00:00:00     2.4000
 2001-01-05T00:00:00     2.4000
 2001-01-06T00:00:00     2.5000
 2001-01-07T00:00:00     2.4000
```

Figure 7 Output from FlowReporter

Congratulations! You have created a Java class which calls the NWIS Web service to retrieve time series data. This concludes the exercise.

APPENDIX: SOURCE CODE FOR FLOWREPORTER CLASS

```
package report;

import java.util.List;
import wof.nwis.TimeSeriesResponseType;

/**
 *
 * @author CUAHSI
 */
public class FlowReporter {

    private static TimeSeriesResponseType getValuesObject(
            java.lang.String location, java.lang.String variable,
            java.lang.String startDate, java.lang.String endDate,
            java.lang.String authToken) {

        wof.nwis.WaterOneFlow service = new wof.nwis.WaterOneFlow();
        wof.nwis.WaterOneFlowSoap port = service.getWaterOneFlowSoap();
        return port.getValuesObject(
                location, variable, startDate, endDate, authToken);
    }

    public static void main(String[] args) {
        String location = "NWIS:10263500";
        String variable = "NWIS:00060";
        String startDate = "2001-01-01";
        String endDate = "2001-12-31";
        String authToken = "";

        // Download time series object
        TimeSeriesResponseType result = getValuesObject(
            location, variable, startDate, endDate, authToken);

        // Show site name
        wof.nwis.SiteInfoType site = (wof.nwis.SiteInfoType)
            result.getTimeSeries().getSourceInfo();
        System.out.println("Site name = " + site.getSiteName());

        // Make a header for showing datetimes and values
        System.out.format("%20s %10s", "DateTime", "Value");
        System.out.println();

        // Show datetimes and values
        List<wof.nwis.ValueSingleVariable> values =
            result.getTimeSeries().getValues().getValue();

        for (wof.nwis.ValueSingleVariable value : values ) {
            System.out.format("%20s %10.4f",
                    value.getDateTime().toString(), value.getValue());
            System.out.println();
        }
    }

}
```