CUAHSI

universities allied for water research

INGESTING NWIS DATA USING JAVA

July 11, 2008

by:

Apurv Bhartia
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 NSF Grant No. EAR-0413265. 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

1 Introduction

In this chapter you will build a Java class that accesses the NWIS Daily Values web service to obtain daily streamflow values for Big Rock Creek near Valyermo, California, for the year 2001. The class will output the site code and site name for this location, as read from the web service, as well as the time series of streamflow values.

Computer and Skill Requirements

To complete this exercise, your computer must meet the following requirements:

- Working Internet connection
- Java SE 5: download from http://java.sun.com/
- NetBeans IDE 5.5: download from http://www.netbeans.org

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

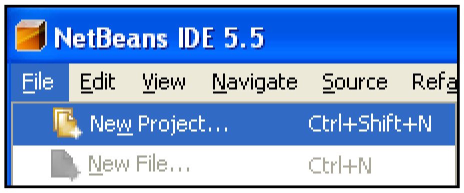
NOTE: The source code for the nwis.java class created in this exercise is located in Appendix A.

2 Procedure

2.1 Creating a New Project

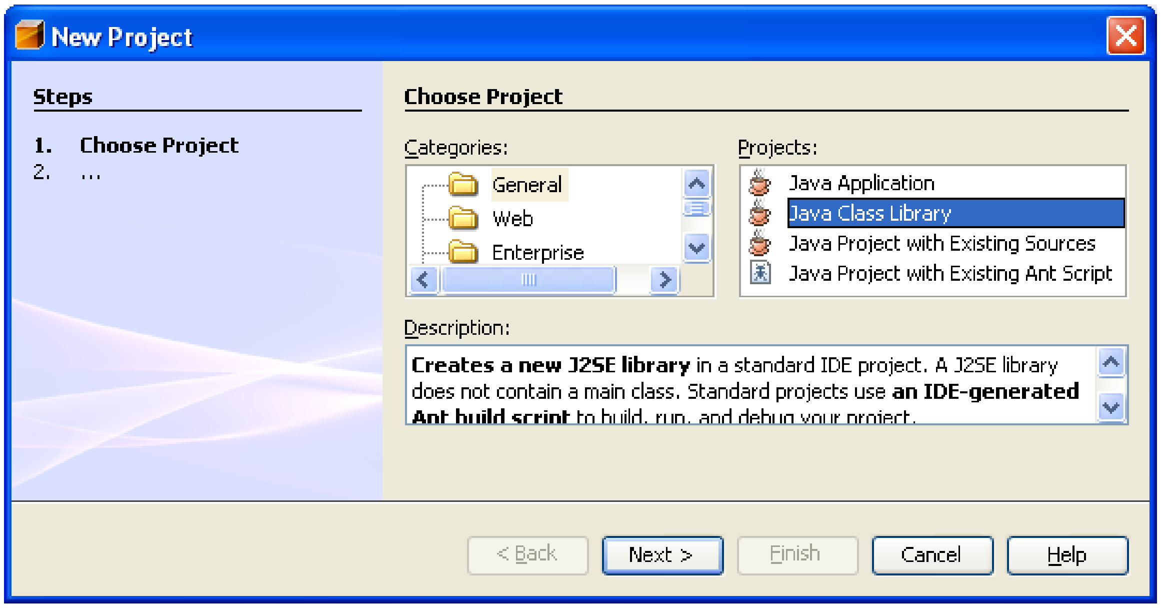
First, you will create a new Java project.

1. Start NetBeans IDE.
2. Click the File | New Project…



Creating a New Project

3. In the New Project dialog, select General in the Categories pane. In the Projects pane, select Java Class Library.


New Project

Steps

1. Choose Project
2. ...

Choose Project

Categories: General, Web, Enterprise

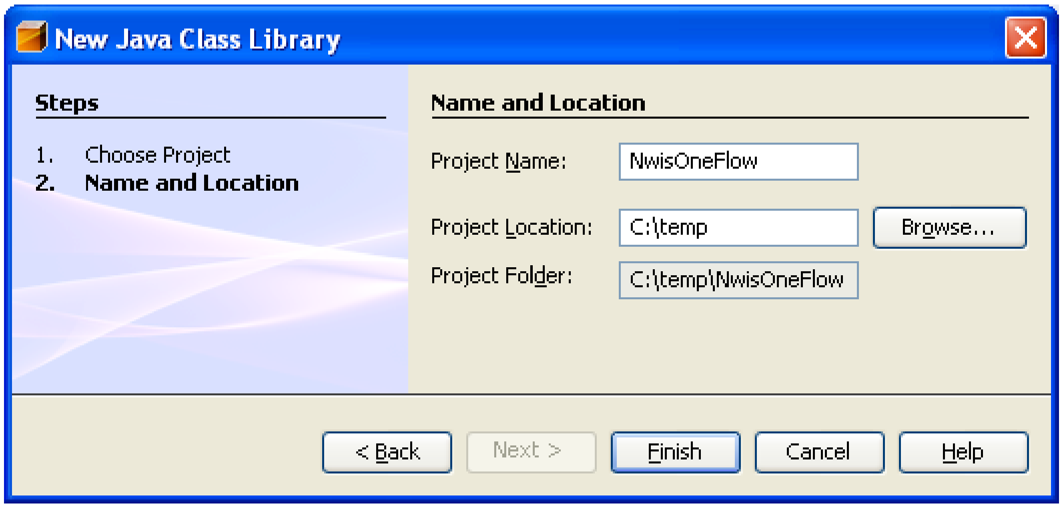
Projects: Java Application, Java Class Library, Java Project with Existing Sources, Java Project with Existing Ant Script

Description:

Creates a new J2SE library in a standard IDE project. A J2SE library does not contain a main class. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Creating a Java Class Library

4. Click Next.
5. In the New Java Class Library dialog, enter “NwisOneFlow” 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 “NwisOneFlow”, which will contain all files used in the project.
7.


New Java Class Library

Steps

1. Choose Project
2. Name and Location

Name and Location

Project Name: NwisOneFlow

Project Location: C:\temp

Project Folder: C:\temp\NwisOneFlow

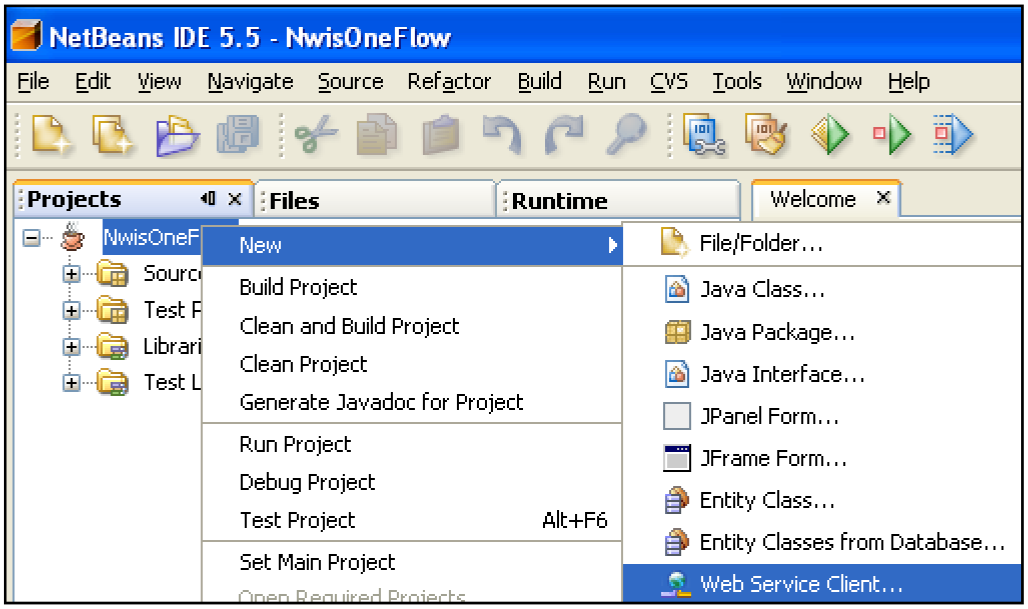
Java Class Library Wizard

8. Click Finish.

2.2 Creating a Web Service Client

With the project set up, you will now create a client for the NWIS web service.

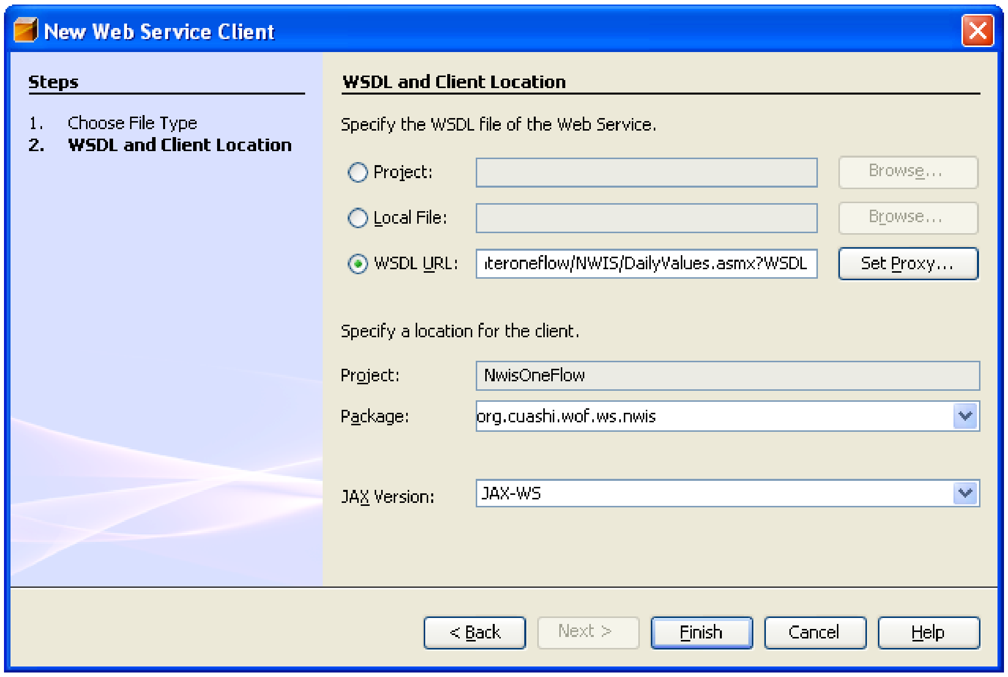
1. In the Projects window, right click on NwisOneFlow | New | Web Service Client…



Adding a Web Service Client

2. In the New Web Service Client dialog, enter
   http://river.sdsc.edu/waterOneFlow/NWIS/DailyValues.asmx

3. For the Package, enter “org.cuashi.wof.ws.nwis”.
4. For the JAX Version, select JAX-WS.



Specify the WSDL file of the Web Service.

Project:

Local File:

[✓ WSDL URL:] iteroneflow/NWIS/DailyValues.asmx?WSDL

Specify a location for the client.

Project: NwisOneFlow

Package: org.cuashi.wof.ws.nwis

JAX Version: JAX-WS

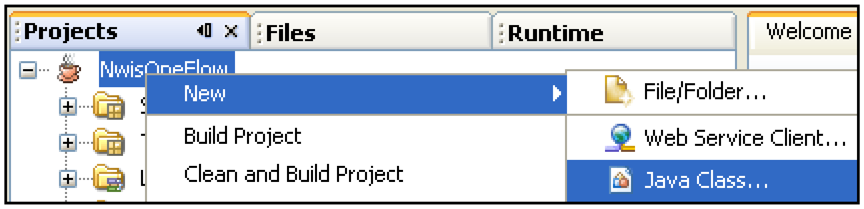
New Web Service Client Wizard

5. Click Finish to compile the web service client class.

2.3 Creating a Class to Consume a Web Service

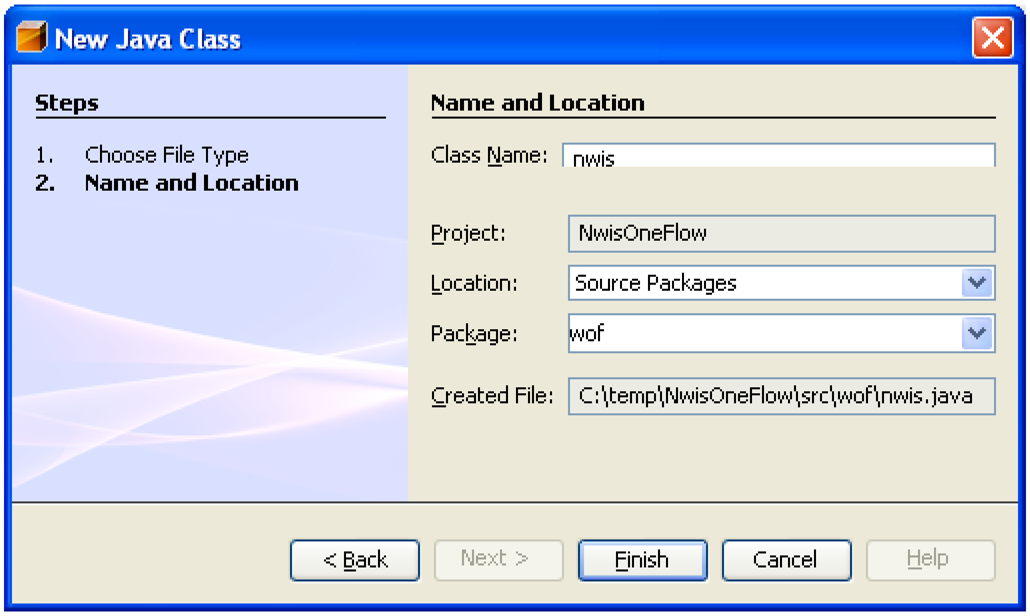
In this section you will create a new java class, wof.nwis, that accesses the web service client you just created. The class will download a time series of daily streamflow values at Big Rock Creek near Valyermo, California, for the year 2001. The site code for this location is 10263500, and the variable code for streamflow is 00060. You will hard code both of these values into the class. 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, its site code, and the time series of values.

1. In the Projects window, right click on the NwisOneFlow project |New | Java Class…



Creating a New Java Class

2. In the New Java Class dialog, enter "*nwis*" as the Class Name, and "*wof*" as the *Package*.



Class Name: nwis

Project: NwisOneFlow

Location: Source Packages

Package: wof

Created File: C:\temp\NwisOneFlow\src\wof\nwis.java

New Java Class Wizard

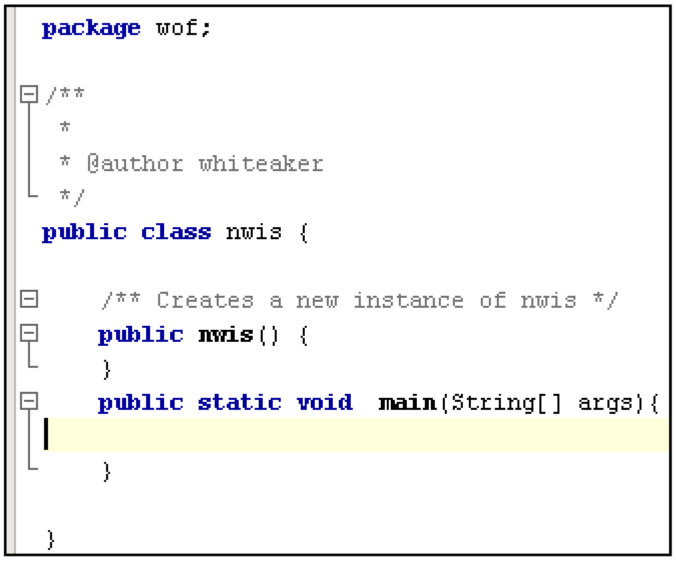
3. Click Finish.

Now you will add a "*main*" procedure, which is the default procedure that will run when the class is invoked. It is within this procedure that you will eventually add the code to retrieve the time series values.

4. At the end of the source code for the *nwis* class, enter the following code.

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

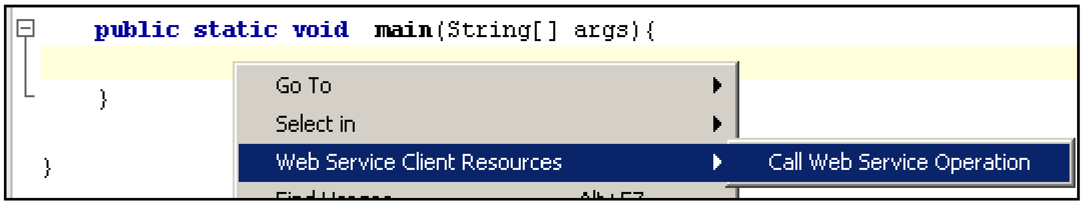
The screenshot below shows the result.



```
package wof;

/**
 *
 * @author whiteaker
 */
public class nwis {

    /** Creates a new instance of nwis */
    public nwis() {
    }
    public static void  main(String[] args){

    }

}
```

Now you will create the code for calling the web service. Fortunately, the IDE can create most of the code for you.

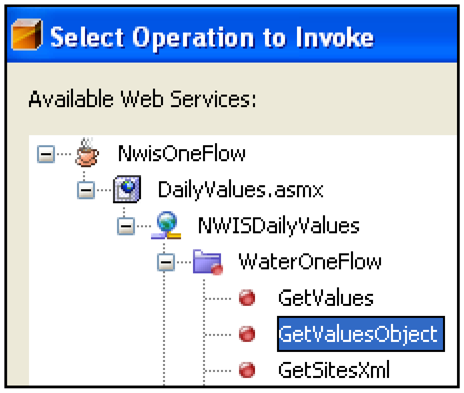
5. Right click within the code for the main method, point to Web Service Client Resources, and then click Call Web Service Operation.



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

    }

}
```

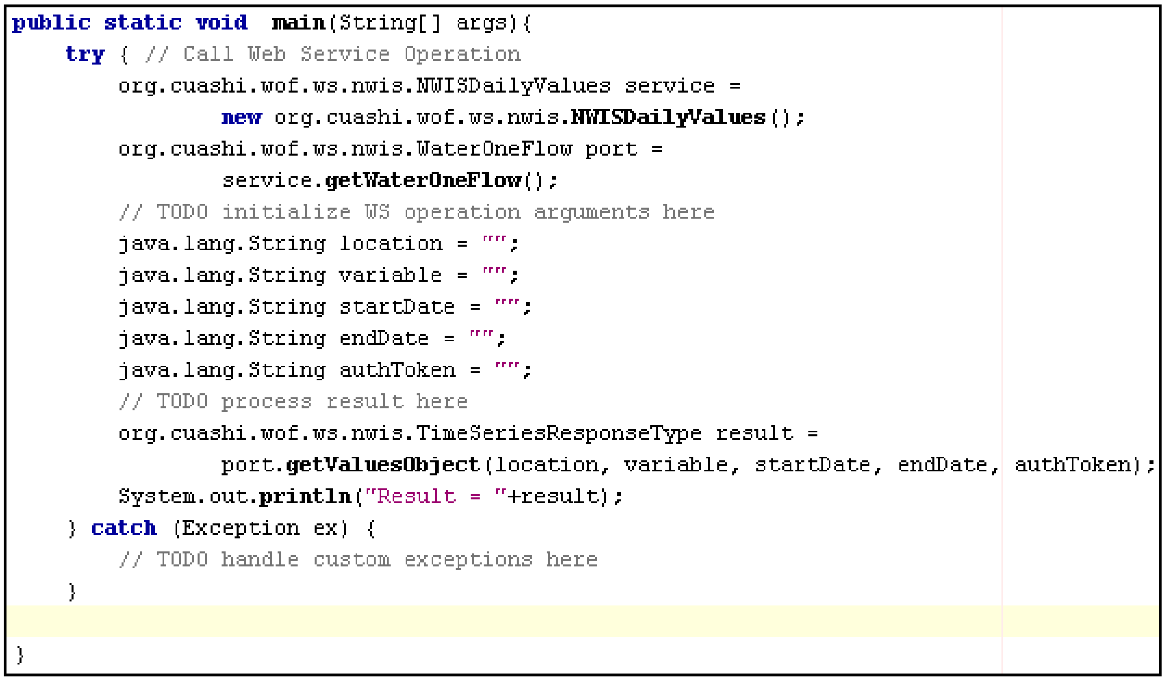
Calling Web Service Operation

6. In the Select Operations to Invoke dialog | GetValuesObject | OK.



Invoking GetValuesObject from NwisOneFlow

After you click OK, the IDE generates code for calling the *GetValuesObject* method from the NWIS web service. The IDE creates variables (e.g., location) for storing the parameters that will be sent to the web service, but leaves them empty for you to fill in later. A screenshot from the code editor is shown below.



```
public static void  main(String[] args){
    try { // Call Web Service Operation
        org.cuashi.wof.ws.nwis.NWISDailyValues service =
                new org.cuashi.wof.ws.nwis.NWISDailyValues();
        org.cuashi.wof.ws.nwis.WaterOneFlow port =
                service.getWaterOneFlow();
        // TODO initialize WS operation arguments here
        java.lang.String location = "";
        java.lang.String variable = "";
        java.lang.String startDate = "";
        java.lang.String endDate = "";
        java.lang.String authToken = "";
        // TODO process result here
        org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
                port.getValuesObject(location, variable, startDate, endDate, authToken);
        System.out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }

}
```

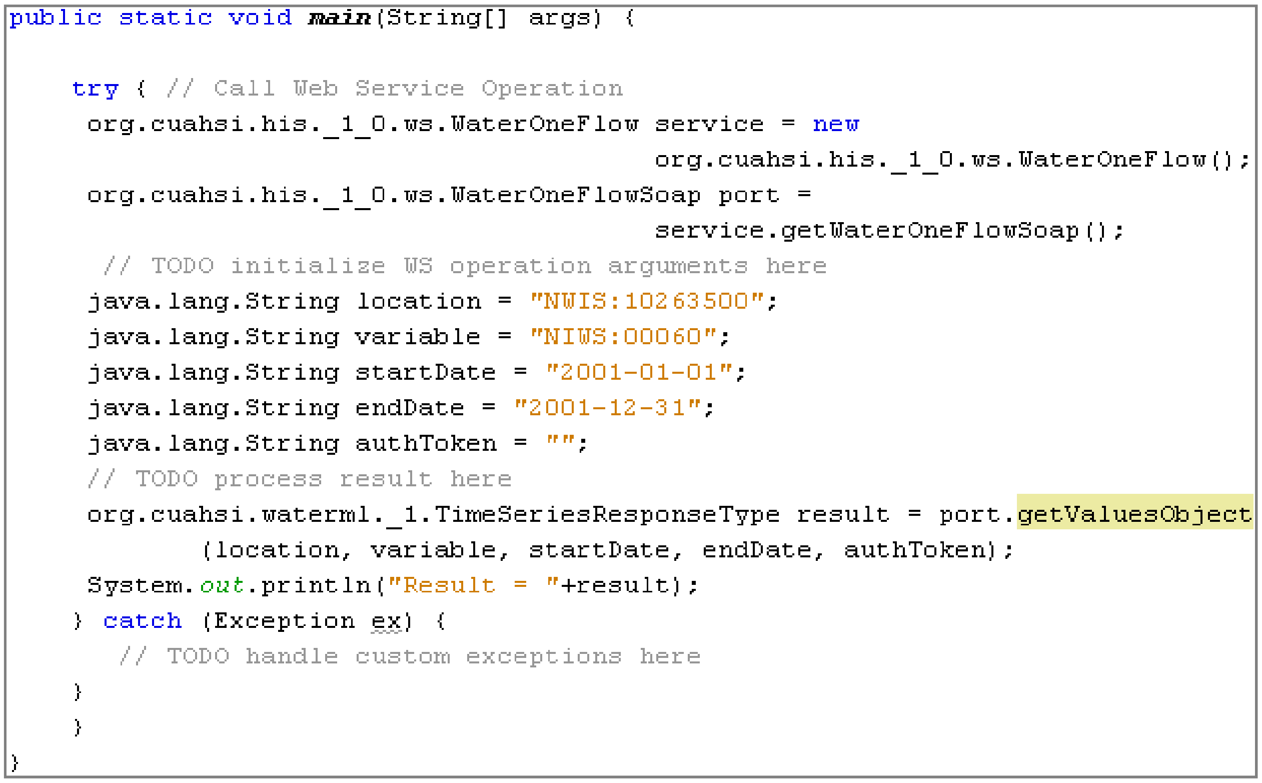
Screenshot

When executed, the above code creates a web service, gets an instance, and then calls *GetValuesObject*. Now you will hard code the parameters for our site of interest. 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.

7. Fill in the parameters for calling the web method.

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

A screenshot from the code editor is shown below.



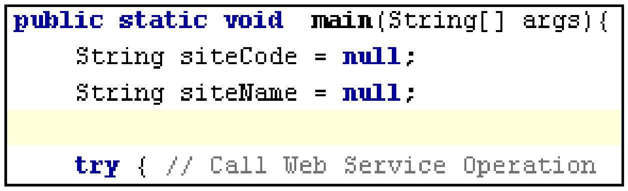
```
public static void main(String[] args) {

    try { // Call Web Service Operation
     org.cuahsi.his._1_0.ws.WaterOneFlow service = new
                                    org.cuahsi.his._1_0.ws.WaterOneFlow();
     org.cuahsi.his._1_0.ws.WaterOneFlowSoap port =
                                    service.getWaterOneFlowSoap();
      // TODO initialize WS operation arguments here
     java.lang.String location = "NWIS:10263500";
     java.lang.String variable = "NIWS:00060";
     java.lang.String startDate = "2001-01-01";
     java.lang.String endDate = "2001-12-31";
     java.lang.String authToken = "";
     // TODO process result here
     org.cuahsi.waterml._1.TimeSeriesResponseType result = port.getValuesObject
            (location, variable, startDate, endDate, authToken);
     System.out.println("Result = "+result);
    } catch (Exception ex) {
       // TODO handle custom exceptions here
    }
    }
}
```

Now you will create variables to store the site code and name as read from the web service.

8. In the main procedure, above the try statement, add the following lines of code.

```
          String siteCode = null;
          String siteName = null;
```

A screenshot of the code editor is shown below.



```
public static void  main(String[] args){
    String siteCode = null;
    String siteName = null;

    try { // Call Web Service Operation
```

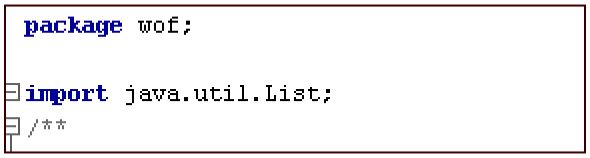
Screenshot

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

9. Below the package declaration, add an import statement for the *List* library.

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

A screenshot from the code editor is shown below.



```
package wof;

import java.util.List;
/**
```

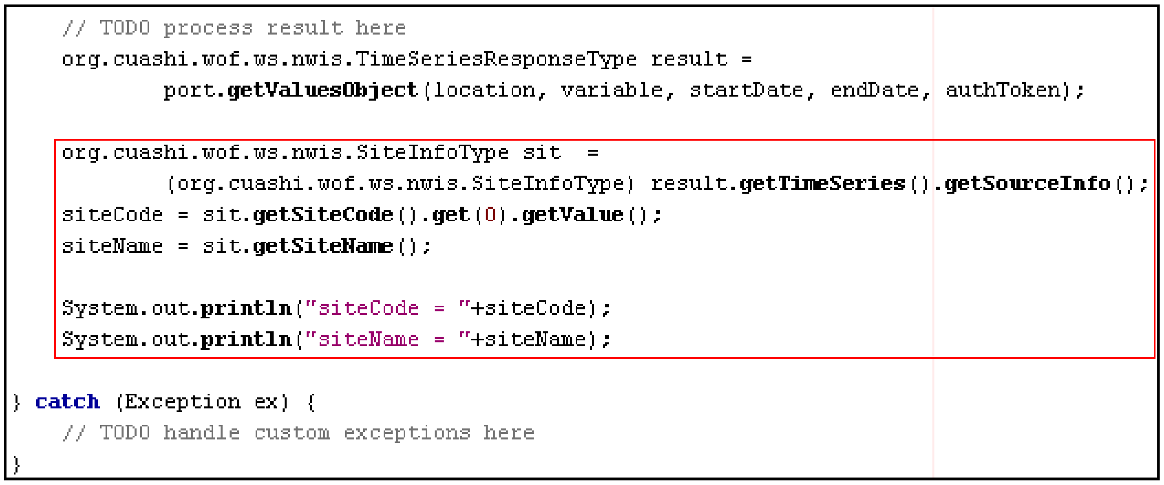
Screenshot

You will now tell the IDE to output the site code and site name to the Output window of the IDE.

10. At the end of the try statement, replace the line that begins with "*System.out.println*" with the following lines of code, in order to output the site information:

```
org.cuashi.wof.ws.nwis.SiteInfoType sit  =
                         (org.cuashi.wof.ws.nwis.SiteInfoType)
                               result.getTimeSeries().getSourceInfo();
siteCode = sit.getSiteCode().get(0).getValue();
siteName = sit.getSiteName();

System.out.println("siteCode = "+siteCode);
System.out.println("siteName = "+siteName);
```

A screenshot from the code editor is shown below.



```
    // TODO process result here
    org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
            port.getValuesObject(location, variable, startDate, endDate, authToken);

    org.cuashi.wof.ws.nwis.SiteInfoType sit  =
            (org.cuashi.wof.ws.nwis.SiteInfoType) result.getTimeSeries().getSourceInfo();
    siteCode = sit.getSiteCode().get(0).getValue();
    siteName = sit.getSiteName();

    System.out.println("siteCode = "+siteCode);
    System.out.println("siteName = "+siteName);

} catch (Exception ex) {
    // TODO handle custom exceptions here
}
```

Screenshot

Some notes on the above code logic: You are working with objects, so you need to do some type casting in order to get the correct object. The line below takes a *sourceInfo* type and casts it to a *siteInfo* type.

```
org.cuashi.wof.ws.nwis.SiteInfoType sit  =
                      (org.cuashi.wof.ws.nwis.SiteInfoType)
                               result.getTimeSeries().getSourceInfo();
```

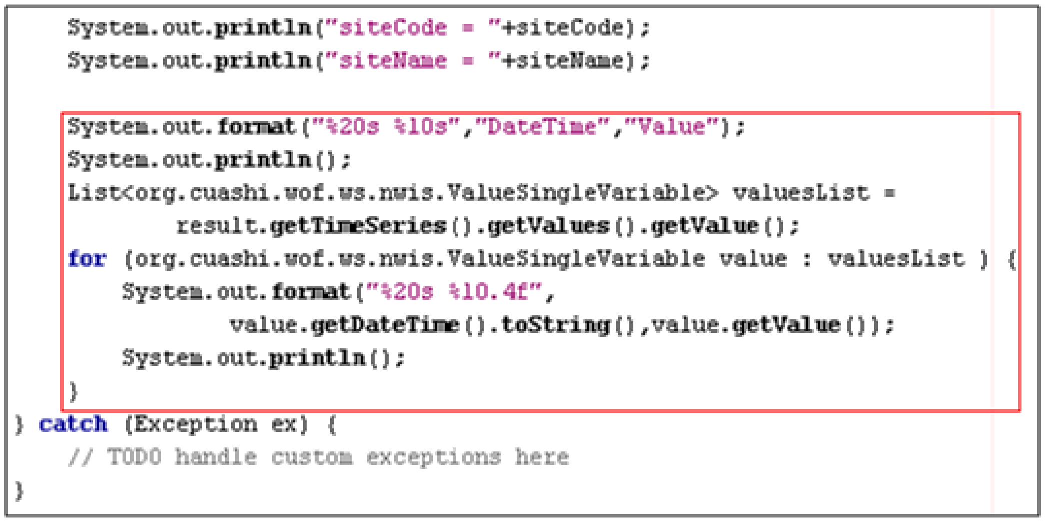
At present, there are two possible *sourceInfo* types: *siteInfoType*, and *dataSetInfoType*. If we were writing a more complete generic parser, we would use *getClass*().*getName*(), and cast based on the object type.

Finally, you will add code to output the time series values.

11. Add the flowing code after the last "*System.out.println*" line that you just added.

```
System.out.format("%20s %10s","DateTime","Value");
System.out.println();
List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
                         result.getTimeSeries().getValues().getValue();
for (org.cuashi.wof.ws.nwis.ValueSingleVariable value : valuesList ) {
    System.out.format("%20s %10.4f",
                     value.getDateTime().toString(),value.getValue());
    System.out.println();
}
```

A screenshot from the code editor is shown below.



```
    System.out.println("siteCode = "+siteCode);
    System.out.println("siteName = "+siteName);

    System.out.format("%20s %10s","DateTime","Value");
    System.out.println();
    List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
            result.getTimeSeries().getValues().getValue();
    for (org.cuashi.wof.ws.nwis.ValueSingleVariable value : valuesList ) {
        System.out.format("%20s %10.4f",
                value.getDateTime().toString(),value.getValue());
        System.out.println();
    }
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
```

Screenshot

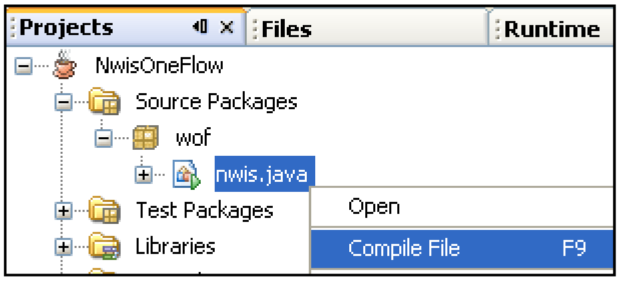
In the above code, we use a *List* and a *for* loop, which are features of java 1.5 and above. We loop through the set of values, and output formatted strings.

When finished, the code for the *main* method should look as follows. Note that text for long lines is wrapped.

```
public static void main(String[] args){
        String siteCode = null;
        String siteName = null;

        try { // Call Web Service Operation
            org.cuashi.wof.ws.nwis.NWISDailyValues service =
                    new org.cuashi.wof.ws.nwis.NWISDailyValues();
            org.cuashi.wof.ws.nwis.WaterOneFlow port =
                    service.getWaterOneFlow();
            // TODO initialize WS operation arguments here
            java.lang.String location = "NWIS:10263500";
            java.lang.String variable = "NIWS:00060";
            java.lang.String startDate = "2001-01-01";
            java.lang.String endDate = "2001-12-31";
            java.lang.String authToken = "";
            // TODO process result here
            org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
                    port.getValuesObject(location, variable, startDate,
                    endDate, authToken);

            org.cuashi.wof.ws.nwis.SiteInfoType sit  =
                    (org.cuashi.wof.ws.nwis.SiteInfoType)
                                 result.getTimeSeries().getSourceInfo();
            siteCode = sit.getSiteCode().get(0).getValue();
            siteName = sit.getSiteName();

            System.out.println("siteCode = "+siteCode);
            System.out.println("siteName = "+siteName);
            System.out.format("%20s %10s","DateTime","Value");
            System.out.println();
            List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
                       result.getTimeSeries().getValues().getValue();
            for (org.cuashi.wof.ws.nwis.ValueSingleVariable value :
                                                         valuesList ) {
                System.out.format("%20s %10.4f",
                        value.getDateTime().toString(),value.getValue());
                System.out.println();
            }
        } catch (Exception ex) {
            // TODO handle custom exceptions here
        }

    }
```

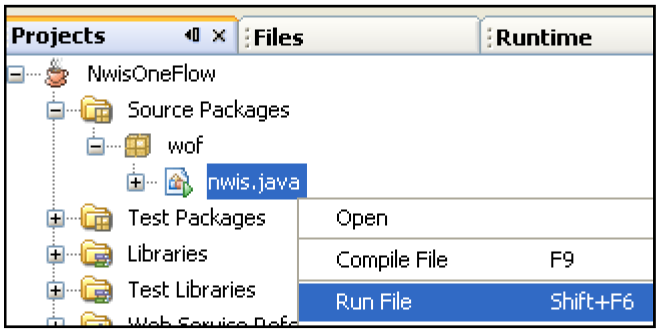
With the code finished, all that is left is to compile and run the file.

12. In the *Projects* window, right click on *nwis.java* | Compile File.



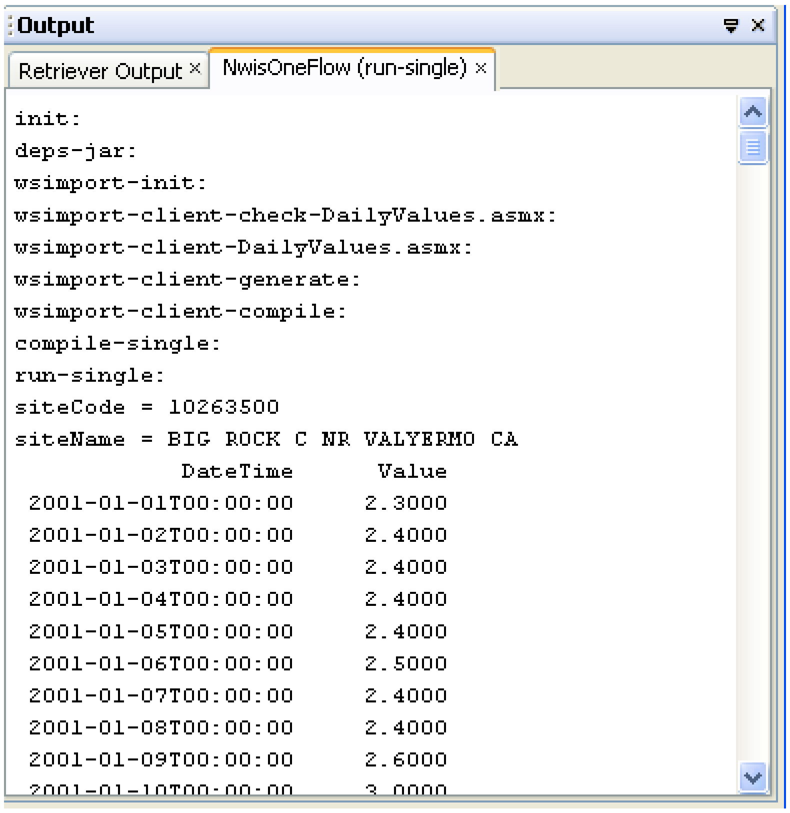
Compile File Option

13. In the *Projects* window, right click on *nwis.java* and then | Run File.



Run File Option

In a moment, you will see the results of the *GetValuesObject* call as text in the Output window.



```
init:
deps-jar:
wsimport-init:
wsimport-client-check-DailyValues.asmx:
wsimport-client-DailyValues.asmx:
wsimport-client-generate:
wsimport-client-compile:
compile-single:
run-single:
siteCode = 10263500
siteName = 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
 2001-01-08T00:00:00      2.4000
 2001-01-09T00:00:00      2.6000
 2001-01-10T00:00:00      3.0000
```

Output Window

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

3 APPENDIX: SOURCE CODE FOR NWIS.JAVA CLASS

```
/*
 * nwis.java
 *
 * Created on November 10, 2006, 1:15 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package wof;

import java.util.List;

public class nwis {

    /** Creates a new instance of nwis */
    public nwis() {
    }
    public static void  main(String[] args){
        String siteCode = null;
        String siteName = null;

        try { // Call Web Service Operation
            org.cuashi.wof.ws.nwis.NWISDailyValues service =
                    new org.cuashi.wof.ws.nwis.NWISDailyValues();
            org.cuashi.wof.ws.nwis.WaterOneFlow port =
                    service.getWaterOneFlow();
            // TODO initialize WS operation arguments here
            java.lang.String location = "NWIS:10263500";
            java.lang.String variable = "NIWS:00060";
            java.lang.String startDate = "2001-01-01";
            java.lang.String endDate = "2001-12-31";
            java.lang.String authToken = "";
            // TODO process result here
            org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
                    port.getValuesObject(location, variable, startDate,
endDate, authToken);

            org.cuashi.wof.ws.nwis.SiteInfoType sit  =
                    (org.cuashi.wof.ws.nwis.SiteInfoType)
result.getTimeSeries().getSourceInfo();
            siteCode = sit.getSiteCode().get(0).getValue();
            siteName = sit.getSiteName();

            System.out.println("siteCode = "+siteCode);
            System.out.println("siteName = "+siteName);
```

```
            System.out.format("%20s %10s","DateTime","Value");
            System.out.println();
            List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
                    result.getTimeSeries().getValues().getValue();
            for (org.cuashi.wof.ws.nwis.ValueSingleVariable value :
valuesList ) {
                System.out.format("%20s %10.4f",
                        value.getDateTime().toString(),value.getValue());
                System.out.println();
            }
        } catch (Exception ex) {
            // TODO handle custom exceptions here
        }

    }

}
```