Building Applications
Using ArcWeb Services with Open Source Tools

By Amar J. Das, Senior Programmer, NSTAR

This tutorial for ArcWeb Services V2 provides instructions on how to build an application that incorporates ArcWeb Services using open source software. The tools Axis and Ant are used, and Eclipse was chosen as the integrated development environment (IDE). This article does not attempt to explain core concepts of Web services or tools such as Apache Ant. Except for ArcWeb Services, all other software used for development is available freely on the Internet. The complete version of all listings referenced in this article is available from *ArcUser Online* at www.esri.com/arcuser.

Java software development kit (SDK) can be downloaded from Sun's Web site. Setup is self-explanatory, and installing the Eclipse SDK will also install Ant. Axis will generate error messages if it does not find activation.jar in the classpath. Because of that, Java Activation Framework is included in the setup. Except for Java SDK, all other products can be downloaded in their compressed form and can be extracted using a compress/uncompress utility such as WinZip. An ESRI Global Account is required to obtain an evaluation version of ArcWeb Services. Create an ESRI Global Account by visiting www1.arcwebservices.com/v2006/evaluate.jsp. The ArcWeb Services evaluation license expires after one month or when 5,000 credits for accessing ArcWeb Services have been used.

Setting Up the Project

Eclipse is becoming the de facto Java IDE. After acquiring the required software listed in the accompanying table, create an Eclipse project.

1. Start Eclipse, click on the Window toolbar, and choose Open Perspective > Java. In the Java Perspective, use File > New > Project to create a new project. Click Next in the New Project dialog box after selecting Java Project. This will bring up the New Java Project dialog box.

2. In the Project name box, enter arcwebtest and select Create separate source and output folders. Accept all other default values and click Next.

3. Go to the Libraries tab and select the Add External Jars button. Browse and add the activation.jar, axis.jar, axis-ant.jar, commons-discovery.jar, commons-logging.jar, jaxrpc.jar, log4j.jar, saaj.jar, and wsdl4j.jar files to the project. The version numbers will most likely be different from those shown in the illustration.

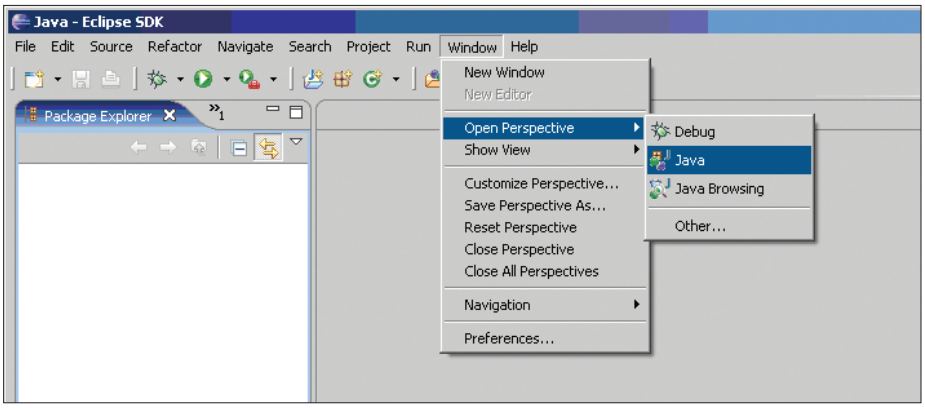
4. Click Finish and the Eclipse project is set up.

Creating the Ant Build File

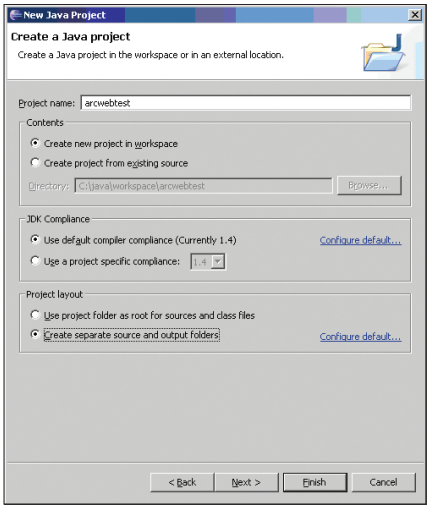
Ant allows developers to automate the application build process. Instructions for Ant are assembled in an XML file. The default name of this file is build.xml, but any name can be chosen for the build file.

Software	Version	URL
Java SDK	1.5.0_03	www.sun.com
Eclipse	3.1.0	www.eclipse.org
Ant	1.6.2	ant.apache.org
Axis	1.2.1	ws.apache.org/axis
Java Activation Framework	1.0.2	java.sun.com/products/javabeans/glasgow/jaf.html

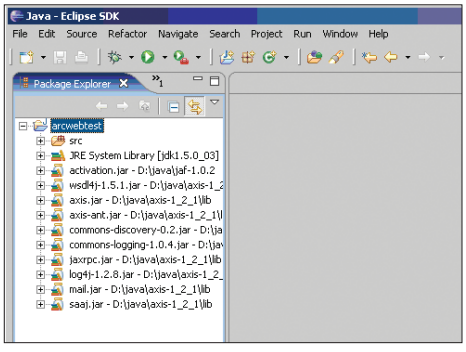
Software needed for this exercise



Start Eclipse, click on the Window toolbar, and choose Open Perspective > Java.

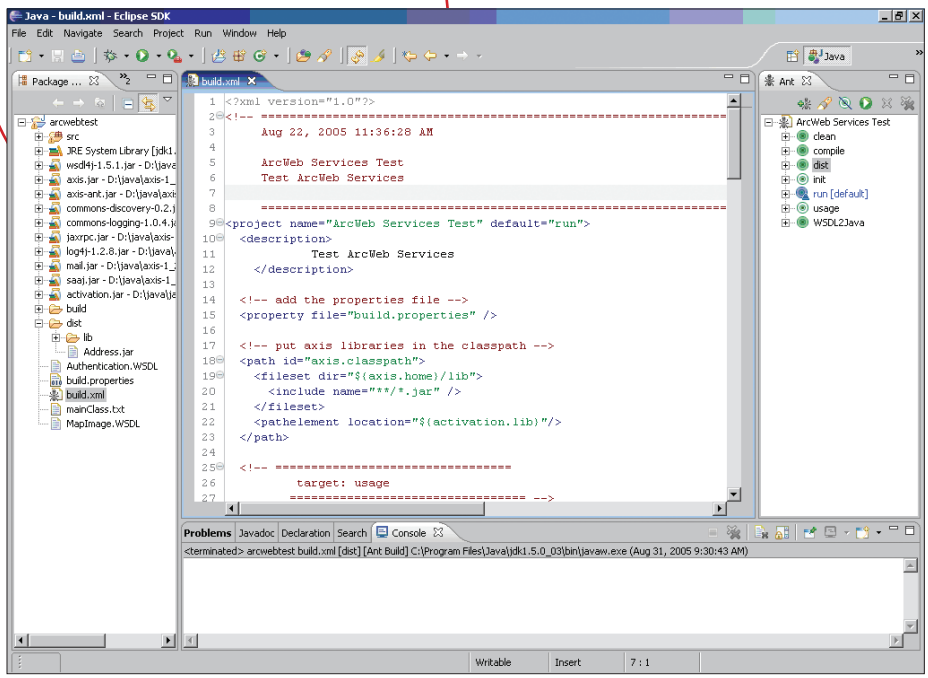


Use the wizard to create a Java project called arcwebtest.

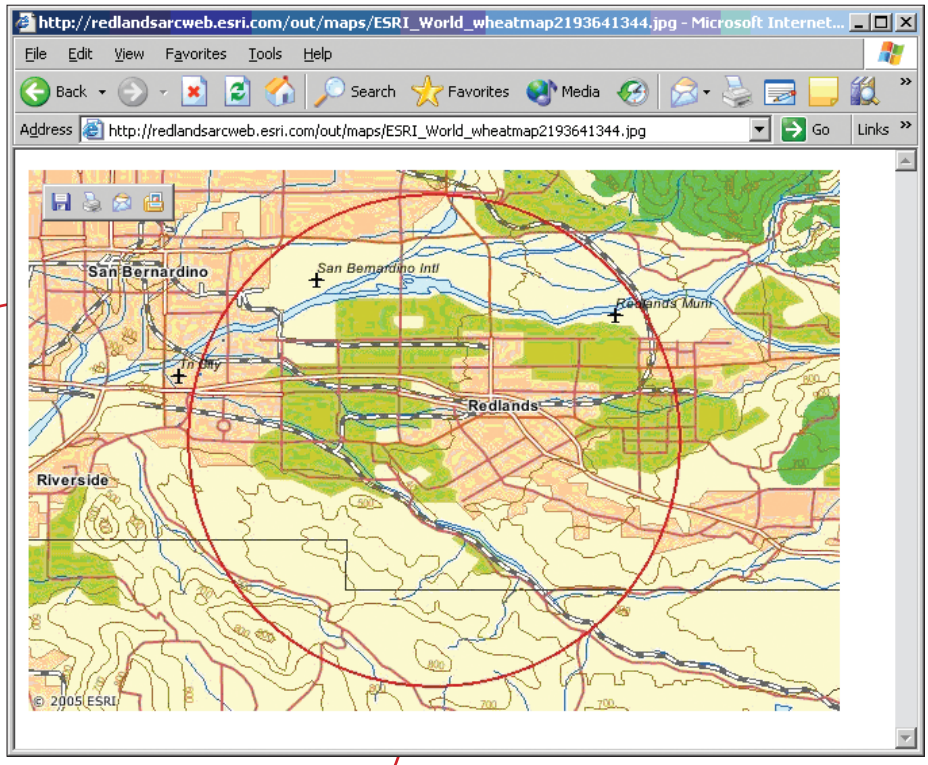


Go to the Libraries tab; select the Add External Jars button; and add the activation.jar, axis.jar, axis-ant.jar, commons-discovery.jar, commons-logging.jar, jaxrpc.jar, log4j.jar, saaj.jar, and wsdl4j.jar files to the project.

Create an ESRI Global Account by visiting www1.arcwebservices.com/v2006/evaluate.jsp



Ant allows developers to automate the application build process. Here Ant is running a task in Ant View in Eclipse.



The map output from MapImage Web service

1. To create the Ant build file for the Eclipse project, right-click on arcwebtest in the Project Explorer and choose New > File. In the File Name box, enter build.xml. Click Finish.

2. Create another file called build.properties using the same process. Close build.properties, then right-click on it in the Project Explorer and use Open With > Properties File Editor.

3. Set source, destination directories, axis home, and other properties in the build.properties file shown in Listing 1.

4. If it is not already open, open build.xml. Type "<" in the editor window. Eclipse will prompt with two options: Build File Template and Project. Choose Build File Template. Eclipse will generate a template build file. Save this file.

Every Ant build file must have <project> and <target> tags. When a build file runs, the default target is executed unless the target is explicitly mentioned. Listing 2 (see complete listing at www.esri.com/arcuser) shows the complete build file listing for this project. The build.properties file is referenced in build.xml using the <property> tag. To compile any Java source file, the classpath must be set correctly. The <path> tag sets the classpath for the project. The classpath is named as axis.classpath so that it can be referenced elsewhere.

Generating Java Stubs from WSDL

Web Services Description Language (WSDL) is an XML-based language used to describe a Web service's capabilities. ArcWeb services exposes various Web services such as the MapImage Web service, Place Finder Web service, Proximity Web services. The WSDL location for these Web services is listed at arcweb.esri.com/arcwebonline/index.htm.

To access these Web services, Java stubs must be generated from the WSDL. Axis provides a third-party Ant task to help in this conversion. It is called WSDL2Java and can be found in the axis-ant.jar. The core ant task <taskdef> is used to define the conversion task (shown in Listing 3).

In this tutorial, the targeted ArcWeb Services are Authentication and MapImage Web services. The Authentication Web service validates access to ArcWeb Services. The MapImage Web service can generate dynamic map content. A look at the WSDL links for

Continued on page 46

Building Applications

Using ArcWeb Services with Open Source Tools

Continued from page 45

Authentication (arcweb.esri.com/services/v2/Authentication.wsdl) and MapImage (arcweb.esri.com/services/v2/MapImage.wsdl) will reveal several target name spaces. The autogeneration process will create individual Java packages for all these name spaces. Using the Ant <mapping> tag shown in Listing 4, all the generated files can be placed in a single package.

To execute the WSDL2Java target, add Ant view to the Eclipse IDE.

1. Use Window menu and select Show View > Ant. On the Ant tab, click on the Add Buildfile icon.
2. Expand arcwebtest in the Buildfile Selection window and select build.xml. Click OK.
3. Select WSDL2Java in the Ant tab. Click on the icon Run the Selected Target.
4. Select arcwebtest in Package Explorer and click F5 to refresh the project.

Notice that a new Java package called com.esri.arcweb.v2 has been created under the src folder. Go through the source files in that package to get familiar with how WSDL links are converted to Java files.

Developing an ArcWeb Services Client

Now write a stand-alone client for the ArcWeb Services.

1. Right-click on src under arcwebtest in the Package Explorer and select New > Class.
2. In the Name box, type in "MapClient".
3. Click Finish.

Listing 5 (see complete listing at www.esri.com/arcuser) shows the complete source code listing for MapClient.java. Notice that the stubs generated in the last step are imported in the client code. Apache Log4j is used to log warning and error messages and debug information for this project. Log4j requires a configuration file. From your Axis installation directory, copy log4j.properties file and paste it under …/arcwebtest/src folder. Initialize the logger with the following instructions.

```
private static final Logger log
= Logger.getLogger(MapClient.
class);
```

If the Internet will be accessed using a proxy server, the authentication information for the proxy server must be provided in the Properties settings shown in Figure 6. The Properties settings can be ignored if proxy server is not used.

Listings

```
src.dir=./src
dist.dir=./dist
build.dir=./build

proxy.host=[your proxy server name or IP address]
proxy.user=[your proxy user id]
proxy.password=[your proxy password]

log4j.properties=log4j.properties

axis.home=[location where axis is installed]
activation.lib=[location of activation.jar]

authentication.wsdl=http://arcweb.esri.com/services/v2/Authentication.
WSDL
mapimage.wsdl=http://arcweb.esri.com/services/v2/MapImage.WSDL
```

Listing 1: Contents of build.properties

```
  <target name="WSDL2Java" description="Create Java file from WSDL">
            <taskdef name="axis-wsdl2java" classname="org.apache.
            axis.tools.ant.wsdl.Wsdl2javaAntTask">
                        <classpath refid="axis.classpath" />
      </taskdef>
  </target>
```

Listing 3: The core Ant task <taskdef> is used to define the conversion task.

```
  <mapping
       namespace="http://www.themindelectric.com/package/com.esri.
       is.services.common.v2/"
package="com.esri.arcweb.v2"/> etc.
```

Listing 4: Ant <mapping> tag

```
  Properties properties = System.getProperties();
  properties.put("http.proxyHost", [your proxy host name];
  properties.put("http.proxyPort", "80");
  properties.put("http.proxyUser", [your proxy account name];
  properties.put("http.proxyPassword", [your proxy account password];
  Properties newprops = new Properties(properties);
  System.setProperties(newprops);
```

Listing 6: Properties settings for the proxy server

```
  AuthenticationLocator loc = new AuthenticationLocator();
  IAuthentication auth = loc.getIAuthentication();
  String token = auth.getToken(username, password);
```

Listing 7: Obtaining a token from the Authentication Web service

Set the data source. A list of all data sources available can be found at arcweb.esri.com/arcwebonline/index.htm. Many credits are required to access some data sources, particularly for reports. This limits testing when using an evaluation license so this tutorial uses ESRI.Basemap.World as the data source.

Setting the ESRI Global Account Information

Set the ESRI Global Account information.

```
String username = [your ESRI
global account name];
String password = [your ESRI
global account password];
```

All ArcWeb Services requests expect user authentication information.

Using the ESRI Global Account, a token can be obtained from the Authentication Web service. This token can be used to access other ArcWeb Services later (see Listing 7).

The map size parameters are expressed in pixels. Both height and width of a map must be within 5 to 6,000 pixels. This exercise will generate a map of Redlands within a five-mile radius of ESRI headquarters. The point object is similar to an ArcObject point. Possible radius units are miles, kilometers, and decimal degrees (see Listing 8).

The MapImage Web Service needs information about the data source, layers, map size, and points of interest. This is done using the MapImageOption object as shown in Listing 9. The default map image format is png8 with possible values of jpg, gif, and png. Now, return the minimum map for the defined point of interest. The MapImageInfo object contains information about the map and legend URL as shown in Listing 10. This will return the minimum map for the defined point of interest.

Compiling and Creating a Distribution

The client can be run directly from Eclipse by right-clicking on MapClient.java, then selecting Run As > Java Application. The map and legend URLs are displayed on the console. Paste the URLs in a browser window and see the results. For more information, contact Amar J. Das, senior programmer, NSTAR, at amar.das@comcast.net or 781-353-6311.

```
MapImageSize miSize = new MapImageSize();
miSize.setHeight(400);
miSize.setWidth(600);

Point cirPoint = new Point();
cirPoint.setX(-117.199577);
cirPoint.setY(34.048364);

CircleDescription circle[] = new CircleDescription[1];
CircleDescription circle1 = new CircleDescription();

circle1.setCenter(cirPoint);
circle1.setRadius(5.0);
circle1.setRadiusUnits("Miles");
circle[0] = circle1;
```

Listing 8: Code for generating a map of Redlands within a five-mile radius of ESRI

```
MapImageOptions miOptions = new MapImageOptions();
miOptions.setDataSource(dataSource);
miOptions.setDisplayLayers(layers);
miOptions.setMapImageFormat("jpg");
miOptions.setMapImageSize(miSize);
miOptions.setReturnLegend(true);
miOptions.setCircles( circle);
```

Listing 9: Using the MapImageOPtions object to communicate with the MapImage Web service

```
MapImageInfo miInfo = mi.getBestMap(miOptions, auth.
getToken(username,
      password));
log.info(miInfo.getMapUrl());
log.info(miInfo.getLegendUrl());
```

Listing 10: MapImageObject