Connecting an ArcIMS Site to an External Database

By Mohammed A. Hoque, West Springfield, Massachusetts

Although ArcIMS wizards can help users create and deploy interactive maps on the Web quickly and easily, they do not possess the functionality to access an external database. In some cases, this functionality may be required. For example, a county may store parcel polygons as GIS layers in one department and maintain other attributes in an external database in another department. To provide access to an external databases, ArcIMS Web sites can be customized to read external databases.

While JavaScript with HTML tags can be used to make Web pages interactive on the client side, creating Web pages by accessing a serverside database is not allowed for security reasons. However, a serverside cross-platform technology called JavaServer Pages (JSP) with Java Database Connectivity (JDBC) can be used to dynamically generate Web pages with data from a relational database. *[JDBC is an API that provides cross-DBMS connectivity to a wide range of SQL databases.]*

JSP can be thought of as an HTML extension with embedded Java code for dynamic contents. The Java code gets converted to a Java servlet, executed on the server, and passed back as pure HTML to the client. This approach keeps the code invisible to the user. Jakarta-Tomcat, open source software, is the official implementation of Sun's JSP technology and was used as the JSP engine.

This article focuses on how to retrieve data from Oracle using JSP with JDBC as the database connector on the Linux operating system. Software specified for this strategy includes ArcIMS 4.1, Red Hat Linux 7.1 as the Server OS, Apache 2.0.42 as the Web server, Tomcat 4.0.5 with J2SDK 1.4.0 as the Servlet/JSP Engine, and JDBC for Oracle as the database connector. Samba should be installed on Linux for file sharing and printing to Windows. *[Samba is an Open Source software suite that allows for interoperability between Linux/UNIX servers and Windows-based clients.]* LinNeighborhood should be able to access any shared folder on a Windows server. *[LinNeighborhood, another open source software program, can be used to mount a shared folder from a Windows server on a Linux server so it can read and pass files to a client.]* Virtual Network Computing (VNC) software can be used to connect to a Linux server from a Windows workstation.

Installation and Configuration

A technical document available from the ESRI Support Center, *Install Apache 2.0.42 with Tomcat 4.0.5 Using J2SDK 1.4.0 for ArcIMS on Linux,* provides step-by-step instructions on downloading, installing, and configuring ArcIMS with the Apache Web server and the Jakarta-Tomcat servlet engine.

1. Store all application-related JSP pages in a separate directory under webapps directory. Each application directory must have a subdirectory called WEB-INF.

2. Use mkdir command to create myJSPs and WEB-INF directories in $CATALINA_HOME/webapps.

3. Restart Tomcat to register the newly created directory as a Web application. $CATALINA_HOME is a system variable that points to Tomcat's installation directory. This variable is set when ArcIMS is installed and configured as described in the ESRI technical document previously cited. See Listing 1.

```
cd $CATALINA_HOME/webapps
mkdir myJSPs
cd myJSPs
mkdir WEB-INF
```

Listing 1: Configuring ArcIMS with the Apache Web server and the Jakarta-Tomcat servlet engine.

Create a JSP Page

JSP pages are composed of regular HTML tags with embedded Java codes. When a JSP page is executed, it passes only Java code output as strings along with other HTML code to the client. To distinguish the Java code in a JSP page from HTML code, JSP syntax is used. Java code, enclosed by scriptlet tags (<%! %>, <% %>, <%= %>, or <%@ %> , is called a JSP statement. In the example shown in Listing 2, the text is enclosed by <%= %>). The text within the scriptlet tags, called a JSP expresssion, is converted to a string and passed to the client.

1. Open any text editor (e.g., *vi* or KEdit) and type the HTML/JSP code in Listing 2 and save it as myJSP.jsp in myJSPs directory.

```
<HTML>
<TITLE>Parcel Information</TITLE>
<BODY>

<%
 String id = request.getParameter("id");
// note the statement is ended with ";" like C or C++
%>

<H1> ID passed to JSP is <%= id %> </H1>
</BODY>
</HTML>
```

Listing 2: Syntax for JSP code.

2. Open a Web browser and type the URL in Listing 3 to access the JSP page. This may require replacing the server's IP address because Tomcat executes on TCP port 8080. That's why the server name or IP address is followed by a colon and 8080. The Web application directory name is followed by the JSP file name. The id is the argument name that the JSP page expects followed by an equal sign (=) and a value. The argument pair and Web address are separated by a question mark (?). The output Web page should say "ID passed to JSP is 066-001-003."

```
Http://127.0.0.1:8080/myJSPs/myJSP.jsp?id=066-001-003
```

Listing 3: URL to access the Web page.

Installing the JDBC Driver

Oracle automatically installs the JDBC driver on the database server so the only other thing required for installation is to copy two files to the server where JSP files are located. The JDBC driver exists as a JAR file in $ORACLE_HOME\ora92\jdbc\lib. (*Note: The folder name may be a little different depending on the version of Oracle used.*) Copy classes12.jar and nls_charset12.jar files from $ORACLE_HOME\ora92\jdbc\lib to the $CATALINA_HOME/lib directory on the Linux server. Restart Tomcat.

Connect to an Oracle Database Using JDBC

Finding the correct driver and connection string is very important. The driver name is oracle.jdbc.driver.OracleDriver and the connection string for Oracle is jdbc:oracle:thin:@<server-name/address>:<port>:<oracle-service-name>. In the connection string, replace <server-name/address> with the server name or IP address. For the default port number, 1521 can be used. Replace <oracle-service-name> with the database name. Port

number and service name can also be found in tnsnames.ora file located in the $ORACLE_HOME\ora92\network\admin folder. See Listing 4 for sample code.

```
Connection conn=null;    // defining variable
Class.forName("oracle.jdbc.driver.OracleDriver");  //loading driver
conn = DriverManager.getConnection("jdbc:oracle:thin:@serverName:1521:myDB", "userName", "Password");
                                                   // connection to the database
```

Listing 4: Sample code for connecting to an Oracle database.

Querying the Database

Two objects, Statement and ResultSet, need to be created. (ResultSet is similar to Visual Basic's Recordset object.) Statement is created by the createStatement() function of the Connection object. ResultSet is populated with queried records when the executeQuery() function of the Statement object is executed. The executeQuery() function has a SQL statement as an argument. The next() function with a while() statement is used to loop through all records. To extract data, use functions such as getString(), getInt(), or getFloat() (as appropriate for the field type) with either the field name or field index. Note that the field index starts from 1, not 0. Examples of these functions are rset.getString("ID") or rset.getFloat(3). See the sample code in Listing 5.

```
Statement stmt = conn.createStatement();      // creating Statement
ResultSet rset = null;  // declaring ResultSet for queried records
String sqlStmnt = "SELECT ID, GIS_ID, ADDRESS FROM REALMAST WHERE "
+ "GIS_ID = '" + id + "'"; // SQL string
/* ResultSet object contains an array of records returned by SQL query */
rset = stmt.executeQuery(sqlStmnt);

while (rset.next()) // loop
{
  // extract and print value from field named "ID"
  out.println(rset.getString("ID");
}
```

Listing 5: Querying the database.

Handling Unexpected Errors

It is not unusual to get unexpected errors while accessing a database. It is preferable to create another JSP page that is called when an error occurs. In this example, add a JSP directive on top of myJSP.jsp page (i.e., <%@ page isErrorPage="false" errorPage="errPage.jsp" %>) and create another file called errPage.jsp. Add the code in Listing 6 and save it in the myJSPs Web application directory.

```
<%@ page import="java.sql.*" %>
<%@ page isErrorPage="true" %>
<HTML>
<HEAD><TITLE>error...</TITLE></HEAD>
<BODY>
error page...<p>

<%
  if (exception instance of SQLException)
%>
  An SQL Exception
<%
  else if (exception instance of ClassNotFoundException)
%>
  A ClassNotFoundException
<% else
{
%>
  An exception occurred while interacting with database
<% } %>
<p>
 message := <%= exception.getMessage() %>
</BODY>
</HTML>
```

Listing 6: Handling errors.

Linking JSP and ArcIMS Pages

Linking ArcIMS with JSP is relatively easy. When the user clicks on a map feature, the feature ID is retrieved using ArcIMS functions and passed as part of a URL to the JSP page. If a user is searching for features by any attributes other than the feature ID (for example, finding parcel by address), those search strings are passed to the JSP page, which causes the database to be searched and may populate a JavaScript for a <BODY> tag's onLoad() event with the IDs of any features found. When the page is loaded, the script will execute and submit XML code to the ArcIMS server, which will highlight those features on the map. See Listing 7 for a sample page.

Conclusion

This method for connecting ArcIMS to an external database is a quick and simple solution. JavaScript code can be added to a JSP page to check the validity of input data before it is passed.

For more information, contact
Mohammed A. Hoque, GIS Coordinator
Town of West Springfield, Massachusetts
Tel.: 413-263-3070
E-mail: thoque@west-springfield.ma.us
Web: gis.west-springfield.ma.us

Continued on page 58

JSP and JDBC Resources

Title	URL
Introduction to JSP Technology	www-106.ibm.com/developerworks/edu/j-dw-jsp-i.html
Dynamic Web-Based Data Access Using JSP and JDBC Technologies	www-106.ibm.com/developerworks/java/library/j-webdata/
JavaServer Pages Fundamentals, Short Course Contents	java.sun.com/developer/onlineTraining/JSPIntro/contents.html
JDBC Database Access	java.sun.com/docs/books/tutorial/jdbc/index.html
JDBC Basics	home.mindspring.com/~thornton.rose/articles/JDBC/JDBC_Basics.html
Tutorial: Java Servlets, JSP, Jakarta-Tomcat, a Database (PosgreSQL or MySQL), Apache, and Linux	www.yolinux.com/TUTORIALS/LinuxTutorialTomcat.html
Introduction to Oracle and Tomcat Installation	www.itee.uq.edu.au/~zxf/_resources/software.html

Connecting an ArcIMS Site to an External Database

Continued from page 57

```
<%-- JSP Comment:: using java library --%>
<%@ page import="java.sql.*" %>
<%@ page isErrorPage="false" errorPage="errPage.jsp" %>

<%-- JSP Comment:: start of JSP code --%>
<%
 // JAVA Comment:: defining variable
 Connection conn=null;
 // JAVA Comment:: loading driver
 Class.forName("oracle.jdbc.driver.OracleDriver");
 // JAVA Comment:: connection to the database
 conn = DriverManager.getConnection
   ("jdbc:oracle:thin:@serverName:1521:myDB", "userName", "password");
 // JAVA Comment:: creating Statement
 Statement stmt = conn.createStatement();
 // JAVA Comment:: declaring ResultSet for recordset
 ResultSet rset = null;
%>

<HTML>
<TITLE>Parcel Information</TITLE>
<BODY>

<!-- HTML Comment:: beginning of JSP statements -->
<%
 // JAVA Comment:: getting the value passed by the client
 String id = request.getParameter("id");
 if (id != null)
 {
   /* JAVA Comment:: standard SQL statement as string that may also contain other functions supported by the database */
   String sqlStmnt = "select ID, GIS_ID, ADDRESS FROM REALMAST WHERE " + "GIS_ID = '" + id + "'";
   rset = stmt.executeQuery(sqlStmnt); // JAVA Comment: SQL query

 /* JAVA Comment:: looping all records... it stops when next() reaches the end of records */
   while (rset.next())
   {
%>

<!-- HTML Comment:: a break from JSP codes-->
<!-- HTML Comment:: even though this section is out of JSP statement, it is still part of while() loop -->
     <B>ID = </B> <%= rset.getString(1) %> <BR>
     <!-- HTML Comment:: example of how JSP is embedded
                        with HTML tags -->
     <B>GIS ID = </B> <%= rset.getString(2) %> <BR>
     <B>Address = </B> <%= rset.getString(3) %> <BR>

<!-- HTML Comment:: continuing the JSP statement from the previous while() statement -->
<%
   }  // JAVA Comment:: end of while() statement
 }
 else // JAVA Comment:: if nothing passed by the client
 {
   /* JAVA Comment:: another way to printout something while inside JSP statements */
   out.println("<B><I>No value/pair passed!</I></B>");
 } // JAVA Comment:: end of if..else.. statement
%>

</BODY>
</HTML>
```

Listing 7: Source code for myJSP.jsp. Comments are shown in green, JSP/Java codes are in blue, and HTML tags and text are in black.