Sharing the Database Connection

Often in a web application, all the servlets or Java server pages access the same database. Setting up and removing connections takes more time than most other operations that the servlets perform. So it makes sense to have the connection established when the application is started and then shared among the servlets that use it.

Web applications provide a ServletContext object for storing information used by all the parts of the application. Since a Java server page is translated into a Java servlet before it is executed, JSP's can also use the ServletContext. It is included in the Java program created for each JSP. Here it is declared by

```
ServletContext application = null;
```

Therefore the Java server page can use the variable, application, without further identification, the same way it uses request, response, out, and session.

Among other things, the ServletContext object can store attributes consisting of keys and values. This is similar to the attributes in the HttpSession class. The ServletContext provides getAttribute and setAttribute methods to handle these keys and values. A connection to a database can be created in a servlet and then stored in the ServletContext object using a setAttribute method. Other servlets in the application can then access it using a getAttribute method.

A Java bean can also be used for creating a connection, but it usually makes more sense to use a Java servlet. This servlet can be listed in web.xml (unlike a bean) and have a low load-on-startup designation.

```
<servlet>
    <servlet-name>ConnectionServlet</servlet-name>
    <servlet-class>view.ConnectionServlet</servlet-class>
    <load-on-startup>10</load-on-startup>
</servlet>
```

Since this servlet will be loaded when the application is loaded, the connection will be created right away and so be available to all the servlets and Java server pages.

The Connection Servlet

The code for the connection servlet is quite simple. It only has to get a connection and place it into the servlet context. It does not even have a doGet or doPost method. Instead it just has an init method. The init method is executed when the servlet is loaded. So the connection is immediately available. (A servlet must over-ride one of the methods in HttpServlet. Two of these are init and destroy.)

```
package view;

import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

// The Connection Servlet gets a connection to the database and stores it in the ServletContext.
public class ConnectionServlet extends HttpServlet
{
    public static String ConnectionKey = "database.key";
    public static String JDBCConnectionURL = "jdbc:odbc:deli";

    public void init ()
    {
```

```
            Connection con = null;
            try
            {
                // Get a jdbc-odbc bridge and connect to the database.
                Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
                con = DriverManager.getConnection (JDBCConectionURL);
            } catch (ClassNotFoundException e){System.out.println ("Class Not Found exception.");}
              catch (SQLException e){System.out.println ("SQL Exception");}

            // Get the ServletContext and store the connection in it.
            ServletContext context = getServletContext ();
            context.setAttribute (ConnectionKey, con);
        } // init
} // ConnectionServlet
```

We could make this more general by using a configuration file to read in the name of the database driver. Such a file can be stored in the same folder as the servlet. It can have other information as well that can be used to configure the application.

Using the Connection Servlet

Other servlets in the application no longer have to open a connection to the database, an inherently slow operation. Instead they can use the connection key to get a connection from the servlet context. An example from a find servlet follows.

```
package view;

import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/*    FindServlet  finds a specific product and displays the data on the output page. */
public class FindServlet extends HttpServlet
{
    public static String ConnectionKey = "database.key";

    public  void doGet (HttpServletRequest request, HttpServletResponse response)
    {
        String table = null;
        try
        {
            PrintWriter out = response.getWriter ();

            // Get the database connection from the servlet context.
            ServletContext context = getServletContext ();
            Connection con = (Connection) context.getAttribute (ConnectionKey);

            /* Get the table name from the request.  In this example it is contained in a drop down list
            box, so we use getParameterValues to retrieve it. */
            String keyName = request.getParameter ("keyName");
```

```
            String [] tables = request.getParameterValues ("table");
            for (int count = 0; count < tables.length; count++)
                if (tables [count] != null) table = tables [count];

            Page.createHeader (out, table + " List");
            boolean found = findProduct (con, out, keyName, table);
            if (!found) out.println ("Product not in database.");
            Page.createFooter (out);
        } catch (IOException ex) {System.out.println ("<h3>IO Exception.</h3>");}

    } // doGet

    private boolean findProduct (Connection con, PrintWriter out, String keyName, String table)
    {
        boolean found = false;
        try
        {
            Statement stmt = con.createStatement ();
            String query = "Select * From " + table + " Where name = '" + keyName + "'";
            ResultSet rs = stmt.executeQuery (query);
            if (rs.next ())
            {
                found = true;
                ResultSetMetaData metaData = rs.getMetaData ();
                int columns = metaData.getColumnCount ();
                out.println("<table border='1' cellspacing='5'>");

                // Display heading.
                out.println("<caption>Produce List</caption>");
                out.println("<thead><tr>");
                for (int count = 1; count <= columns; count ++)
                    out.println ("<td>"+metaData.getColumnName (count)+"</td>");
                out.println ("</tr></thead>");

                // Displa y data.
                out.println ("<tr>");
                for (int count = 1; count <= columns; count ++)
                    out.println ("<td>"+rs.getString (count)+"</td>");
                out.println ("</tr></table>");
            }
        } catch (SQLException es) {out.println ("SQL Find Exception");}
        return found;
    } // findProduct

} // class FindServlet
```

Sharing the connection using JSP

You can also use connection sharing with JSP files. We can still use a connection servlet, since a JSP file is compiled to a Java servlet, and this servlet has access to the servlet context. In fact, the servlet context has the name, application, in the compiled servlet, so it should not be separately declared. (This is similar

to out, request, response, and session.) The following scriptlet code can be used to get the connection from the servlet context and store it in the bean. Note that this uses <%@ for imports, <%! for declarations, and <% for straight code.

```
<%@ page import="java.sql.*" %>
<%! Connection con;
    String ConnectionKey; %>
<%
    application = getServletContext ();
    ConnectionKey = "database.key";
    Object object = application.getAttribute (ConnectionKey);
    con = (Connection) object;
    findBean.setConnection (con);
%>
```

The Connection object is defined in the package, java.sql, which has to be imported into the JSP. The same key, ConnectionKey, used to store the connection must be used to retrieve it from the ServletContext object, application. Since the method, getAttribute, returns an Object, it must be cast to a Connection object. This is done in the following line. Finally the connection is sent to the bean using the bean's setConnection method.

The code for find.jsp follows:

```
<html>
<head><title> Find Produce JSP. </title></head>

<body bgcolor="white">

<jsp:useBean id="findBean" scope="session" class="view.FindBean" />
<jsp:setProperty name="findBean" property="*" />

<!-- The Connection object is in java.sql, which must be imported.-->
<%@ page import="java.sql.*" %>

<!-- The next lines are declarations for the connection and its key. -->
<%! Connection con;
    String ConnectionKey; %>
<%
    application = getServletContext ();
    ConnectionKey = "database.key";
    Object object = application.getAttribute (ConnectionKey);
    con = (Connection) object;
    findBean.setConnection (con);
%>

<!-- After processing the request, we can either output the results or display an error message. -->
<% findBean.processRequest(request); %>
<% if (findBean.getFound ()) { %>
<h4>Produce Table</h4>
<table border = "1" cellspacing = "5">
    <tr>
```

```
        <td><% out.println (findBean.getId()); %></td>
        <td><% out.println (findBean.getName()); %></td>
        <td><% out.println (findBean.getVariety()); %></td>
        <td><% out.println (findBean.getQuantity()); %></td>
        <td><% out.println (findBean.getPrice()); %></td>
        </tr>
</table>
<% } else { %>
        <h4>The product is not in the database.</h4>
<% } %>
<hr>
<p><a href="../deli">Return</a></p>
</font>
</body>
</html>
```

The bean that does the actually database contact is listed below. It gets the connection from find.jsp, which has already retrieved it from the servlet context.

```
package view;

import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

/*    FindBean is a bean that works with find.jsp to locate a name in the database and return all the data
stored with it. It returns a boolean with the value of true if the name is in the database, and with the value
false otherwise. */
public class FindBean
{
        private String id, name, variety, table;
        private int quantity;
        private double price;
        private boolean found;
        private Connection con;

        public void FindBean () {} // constructor

        // Accessor methods
        public String getId() {return id;}
        public String getName() {return name;}
        public String getVariety() {return variety;}
        public int getQuantity () {return quantity;}
        public double getPrice() {return price;}
        public boolean getFound () {return found;}

        // Mutator methods
        public void setConnection (Connection con) {this.con = con;}
        public void setName (String n) {name = n;}

        /*    processRequest gets the name and the table and finds the product. */
```

```
    public void processRequest (HttpServletRequest request)
    {
        try
        {
            /* Get the table name from the request.  In this example it is contained
                in a drop down list box.  getParameterValues is used to retrieve it. */
            String [] tables = request.getParameterValues ("table");
            for (int count = 0; count < tables.length; count++)
                if (tables [count] != null) table = tables [count];

            // Set up the query and get a result set.
            Statement stmt = con.createStatement ();
            String query = "Select * From " + table + " Where name = '" + name + "'";
            ResultSet rs = stmt.executeQuery (query);

            if (rs.next ()) // The name is in the database.
            {
                id = rs.getString ("id");
                name = rs.getString ("name");
                variety = rs.getString ("variety");
                quantity = rs.getInt ("quantity");
                price = rs.getDouble ("price");
                found = true;
            }
            else found = false;  // The name was not in the database.
        } catch (SQLException e){System.out.println ("SQL Exception");}
    } // processRequest
} // FindBean
```

The html form that is used to send data to the Java server page follows:

```
    <form method = "get" action="../deli/find.jsp">
        <p><h4>Select a Table</h4>
        <select name = "table" size = "1" >
            <option /> fruit
            <option /> vegetables
        </select></p>
        <input name="name" type="text" value = "" size = "10" /> Product Name
        <p><input type="submit" value="Find Product" /></p>
    </form>
```