ISTANBUL KÜLTÜR UNIVERSITY, DEPARTMANT OF COMPUTER ENGINEERING
CSE5001 WEB PROGRAMMING LAB MANUAL

Lab 06: ASP.NET general concepts and ADO.NET

Objective: To learn state management options and data access operations

Summary: States and ADO.NET

Managing State

Hyper Text Transfer Protocol (HTTP) is a stateless protocol. When the client disconnects from the server, the ASP.NET engine discards the page objects. This way, each web application can scale up to serve numerous requests simultaneously without running out of server memory.
ASP.NET provides various state management options like

- View State
- Control State
- Session State
- Application State
- Cookie
- QueryStrings (URL)
- Hidden Fields

View State: The state of the page and all its controls. When a page is sent back to the client, the changes in the properties of the page and its controls are determined, and stored in the value of a hidden input field named _VIEWSTATE. When the page is again posted back, the _VIEWSTATE field is sent to the server with the HTTP request.

Control State: Control state cannot be modified, accessed directly, or disabled.
Session State: When a user connects to an ASP.NET website, a new session object is created. When session state is turned on, a new session state object is created for each new request. This session state object becomes part of the context and it is available through the page. Sessions are identified and tracked with a 120-bit SessionID, which is passed from client to server and back as cookie or a modified URL. The SessionID is globally unique and random.
Application State: The ASP.NET application is the collection of all web pages, code and other files within a single virtual directory on a web server. When information is stored in application state, it is available to all the users. To provide for the use of application state, ASP.NET creates an application state object for each application from the HTTPApplicationState class and stores this object in server memory. This object is represented by class file global.asax.
Cookie: A cookie is a small piece of text stored on user's computer. Usually, information is stored as name value pairs. Cookies are used by websites to keep track of visitors. Every time a user visits a website, cookies are retrieved from user machine and help identify the user.
Hidden fields: Hidden fields are used to store data at the page level. As its name says, these fields are not rendered by the browser. It's just like a standard control for which you can set its properties. Whenever a page is submitted to server, hidden fields values are also posted to server along with other controls on the page.

Query strings: Query strings are usually used to send information from one page to another page. They are passed along with URL in clear text.

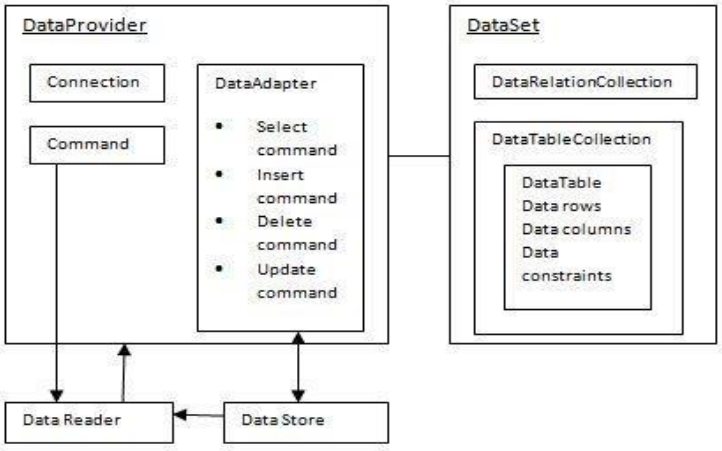
ADO (ActiveX Data Objects). NET

The ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data. The two key components of ADO.NET are **Data Providers** and **DataSet** . The .Net Framework includes mainly three Data Providers for ADO.NET.

- They are the Microsoft SQL Server Data Provider, OLEDB Data Provider and ODBC Data Provider.
- SQL Server uses the **SqlConnection** object, OLEDB uses the **OleDbConnection** Object and ODBC uses **OdbcConnection** Object respectively.

Core elements of the .NET data provider model

The four Objects from the .Net Framework provides the functionality of Data Providers in the ADO.NET.



- The **Connection** Object provides physical connection to the Data Source.
- The **Command** Object uses to perform SQL statement or stored procedure to be executed at the Data Source.
- The **DataReader** Object is a stream-based, forward-only, read-only retrieval of query results from the Data Source, which do not update the data.
- The **DataAdapter** Object, which populate a Dataset Object with results from a Data Source.

Result Sets

The **dataset** represents a subset of the database. It provides a disconnected representation of result sets from the Data Source, and it is completely independent from the Data Source.
DataSet consists of a collection of **DataTable** objects that you can relate to each other with **DataRelation** objects. The DataTable contains a collection of **DataRow** and **DataColumn** Object which contains Data. The **DataAdapter** Object provides a bridge between the DataSet and the Data Source.

Connection Strings

Connection strings are the text representation of connection properties for a data provider.

SqlServer **Standard Security** Connection Strings:
Server=myServerAddress;Database=myDataBase;User Id=myUsername;Password=myPassword;

SqlServer **Trusted Connection** Connection Strings:
Server=myServerAddress;Database=myDataBase;Trusted_Connection=True;

Procedure 01: Retain value after page refresh by **View State**

1. Open Visual Studio for Web.
2. In the **File** menu, click **New Web Site**.
   The **New Web Site** dialog box is displayed.
3. Under **Installed**, click **Visual C#** and then select **ASP.NET Empty Web Site**.
4. In the **Web location** box, select **File System**, and then enter the name of the folder where you want to keep the pages for your website.
   For example, type the folder name **C:\Lab06_StateManagementAndADONET**. Click **OK**.
5. In **Solution Explorer**, right-click the root of your Web site, and then click **Add New Item**.
6. Select **Web Form**, name the file **ViewState.aspx**, and then click **Add**.
7. To demonstrates the concept of storing **view state**. Let us keep a counter, by adding a label and button to the page.
   - *The markup file code is as follows:*

```
<h3>View State </h3>
Page Counter:
<asp:Label ID="lblCounter" runat="server" />
<asp:Button ID="btnIncrement" runat="server" Text="Add Count"/> </div>
```

-Switch to Codebehind. Create a **counter** property which is incremented each time the page is posted back by clicking a button on the page. A label control shows the value in the counter.

```
public int counter
{
   get
     {
         if (ViewState["pcounter"] != null)
                return ((int)ViewState["pcounter"]);
            else return 0;
      }
        set
        { ViewState["pcounter"] = value; }
}
protected void Page_Load(object sender, EventArgs e)
{
       lblCounter.Text = counter.ToString();
       counter++;
}
```

Procedure 02: Passing value between pages using **Session**.

*(open two new blank web page with the name **Session01.aspx** and **Session02.aspx**)*

7. To demonstrates the concept of storing **session** state. Let's create two .aspx page and pass the value one page to another.
   - There is a button on the page, a text box to enter string and a label to display the text stored from last session.
   The markup file code is as follows (Session01.aspx):

```
<div>
<h3>Session State – 1 </h3>
<asp:Label ID="Label1" Text="Enter a string" runat="server" />
<asp:TextBox ID="txtInput" runat="server" />
<asp:Button ID="btnSend" runat="server" Text="Send"/>
</div>
```

- There are two label a button on the page to display the info and input message .
The markup file code is as follows (Session02.aspx):

```
<div>
<h3> Session State – 2 </h3>
<asp:Label ID="Label2" Text="Session value:" runat="server" />
<asp:Label ID="lblInputMessage" runat="server" />
</div>
```

8. Switch to Codebehind
- *Session01.aspx*

```
protected void btnSend_Click(object sender, EventArgs e)
{
    Session["message"] = txtInput.Text;
    Response.Redirect("Session02.aspx"); //redirect to Session02.aspx
}
```

- *Session02.aspx*

```
protected void Page_Load(object sender, EventArgs e)
{
    lblInputMessage.Text = (string)Session["message"];
}
```

Procedure 02: Passing value between pages using **QueryString**

*(open two new blank web page with the name **QueryStringPass.aspx** and **QueryStringRequest.aspx**)*

7. To demonstrates the concept of passing value by **querystring**. Let's create two .aspx page and pass the value one page to another.
- There is a button on the page, two text box to enter string and a label to display the info text
The markup file code is as follows (QueryStringPass.aspx):

```
<div>
<h3> QueryString variable passing</h3>
<asp:Label ID="Label1" Text="Enter a username and password" runat="server" />
<asp:TextBox ID="txtUsername" runat="server" />
<asp:TextBox ID="txtPassword" runat="server" />
<asp:Button ID="btnSend" runat="server" Text="Send"/>
</div>
```

- There are two label a button on the page to display the requested text.
The markup file code is as follows (QueryStringRequest.aspx):

```
<div>
<h3> QueryString variable requesting </h3>
<asp:Label ID="Label2" Text="Querystring value:" runat="server" />
<asp:Label ID="lblInputMessage" runat="server" />
</div>
```

8. Switch to Codebehind
- *QueryStringPass.aspx*

```
protected void btnSend_Click(object sender, EventArgs e)
{
Response.Redirect("QueryStringRequest.aspx?Username=" + txtUsername.Text +
"&Password=" + txtPassword.Text);
}
```

- *QueryStringRequest.aspx*

```
protected void Page_Load(object sender, EventArgs e)
{
    string uname= Request.QueryString["Username"]; //You can also retrieve this
values using their position in the querystring Eg. Request.QueryString[0]
    string upass= Request.QueryString["Password"];

    lblInputMessage.Text = "your username and password "+ uname+ "- " + upass;

}
```

IMPORTANT NOTE: Before starting Procedure 03, you need to run the script that's given (SchoolDBScript.sql). All procedures are applied by using this database.

Setting-up DB script: Setting up **SchoolDB** Database

1. Download the DB Script file named "SchoolDBScript.sql" from course web site.
2. Open Microsoft SQL Server **Management Studio**.
3. Click on **New Query** after user authentication.
4. **Drag and drop** the DB Script file onto **New Query** window in MS SQL Server.
5. Then click **Execute** button on the upper menu.
6. To verify the success of deploying database, **Right-click** to the **Databases** option in the Object Explorer which is located left hand side and click **Refresh** button to update list.
7. Check the presence of database **SchoolDB**

Procedure 03: Retrieving data from database and binding in a GridView object by Trusted_Connection. *(open a new blank web page with the name **SelectADO.aspx**)*

7. Drag and drop a GridView control to your page for displaying all Courses

```
<asp:GridView ID="gvCourses" runat="server"></asp:GridView>
```

8. Switch to Codebehind

   - *Add the following lines to the Namespace*

```
using System.Data;//contain classes for accessing and managing data from diverse
sources
using System.Configuration;//contains the types that provide the programming model
for handling configuration data
using System.Data.SqlClient; //Data Provider for SQL Server
```

-write a method to get all courses from database and call this method in page load

```
   protected void Page_Load(object sender, EventArgs e)
    {
        gvCourses.DataSource = GetCourses();
        gvCourses.DataBind();
    }
    private DataTable GetCourses()
    {
        string connectionString =
"server=servername;database=SchoolDB;Trusted_Connection=True;";
        SqlConnection connection = new SqlConnection(connectionString);
        connection.Open();
        DataTable dtCourse = new DataTable();
```

```
        if (connection.State == ConnectionState.Open)
        {
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT
CourseName,Location FROM Course", connection);
            adapter.Fill(dtCourse);
        } return dtCourse;
      }
```

Output:

CourseName	Location
CSE5041 Database Design and Development	AK3B0810
MCB1007 Intr. to Probability and Statistics	AK01
CSE5001 Web Programming	AK5C0810
CSE5031 Operating Systems	AK2B16
CSE7093 Engineering Economics	AK4B0810
CSE0463 Software Quality & Testing	AK4C0406

Procedure 04: Inserting data to the database by using user credentials in the connection string.
*(open a new blank web page with the name **InsertADO.aspx**)*

7. To take user input insert a textbox and button to the page

Course Name:

```
<asp:TextBox ID="txtCourse" runat="server" />
<asp:Button ID ="btnSave" Text="Save" runat="server"/>
<asp:Label ID="lblMessage" runat="server" />
```

8. Switch to Codebehind

- *Add the following lines to the Namespace*

```
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
```

-write a method to insert course to database and call this method in button click

```
protected void btnSend_Click(object sender, EventArgs e)
{
        int result = InsertCourse(txtCourse.Text);
        if (result > 0)
            lblMessage.Text = "The record is added successfully";
        else lblMessage.Text = "The record is failed.Please try again!";
}
    private int InsertCourse(string pCourseName)
    {
        int result = 0;
string connectionString="Server=servername;Database=SchoolDB;uid=sa;pwd= ikucse;";
        SqlConnection connection = new SqlConnection(connectionString);
        connection.Open();

        if (connection.State == ConnectionState.Open)
        {
            SqlCommand command = new SqlCommand("INSERT INTO Course (CourseName,
Location) VALUES (‘”+pCourseName +”’, NULL)", connection);
            result = command.ExecuteNonQuery();
        }
        return result;
    }
```

9. Run the project (F6), and insert data.

Procedure 05: Creating Registration Form with validation
*(open a new blank web page with the name **Registration.aspx**)*

7. To create a registration form insert a html table within;
   - to take user input 5 text box, to send inputs a button and to display message a literal,
   - to control all your input insert 4 **RequiredFieldValidator**, to compare password a **CompareValidator** and to check email format a **RegularExpressionValidator**,
   the following code between the form tags.

```
Registration
Student Name: <asp:TextBox ID="txtName" runat="server" />
<asp:RequiredFieldValidator ErrorMessage="Required" ForeColor="Red"
ControlToValidate="txtName" runat="server" />

Username:
<asp:TextBox ID="txtUsername" runat="server" />
<asp:RequiredFieldValidator ErrorMessage="Required" ForeColor="Red"
ControlToValidate="txtUsername" runat="server" />

Password: <asp:TextBox ID="txtPassword" runat="server" TextMode="Password" />
<asp:RequiredFieldValidator ErrorMessage="Required" ForeColor="Red"
ControlToValidate="txtPassword" runat="server" />

Confirm Password:<asp:TextBox ID="txtConfirmPassword" runat="server"
TextMode="Password"/>
<asp:CompareValidator ErrorMessage="Passwords do not match." ForeColor="Red"
ControlToCompare="txtPassword" ControlToValidate="txtConfirmPassword"
runat="server" />

Email:<asp:TextBox ID="txtEmail" runat="server" />
<asp:RequiredFieldValidator ErrorMessage="Required" Display="Dynamic"
ForeColor="Red" ControlToValidate="txtEmail" runat="server" />
<asp:RegularExpressionValidator runat="server" Display="Dynamic"
ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
ControlToValidate="txtEmail" ForeColor="Red" ErrorMessage="Invalid email address."
/>

<asp:Button ID ="btnSubmit" Text="Submit" runat="server"/>
<asp:Literal ID="LiteralText" runat="server" Text=" "></asp:Literal>
```

8. Double Click the Submit button, and you will be taken to the *CodeBehind* file of our page. Add the following code to your page.
   - *Add the following lines to the Namespace*

```
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
```

- To insert a new student to the database add the following code to the Button click event

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
        int userID = 0;
 string constr = "server= servername;database=SchoolDB;Trusted_Connection=True;";
        using (SqlConnection con = new SqlConnection(constr))
        {
```

```
            using (SqlCommand cmd = new SqlCommand("INSERT INTO
Student(StudentName,StudentUsername, StudentPassword, StudentEmail) VALUES(@sname,
@uname, @pass, @email); Select @@Identity"))
            {
                using (SqlDataAdapter sda = new SqlDataAdapter())
                {
                  cmd.Parameters.AddWithValue("@sname ", txtName.Text.Trim());
                  cmd.Parameters.AddWithValue("@uname ", txtUsername.Text.Trim());
                  cmd.Parameters.AddWithValue("@pass ", txtPassword.Text.Trim());
                  cmd.Parameters.AddWithValue("@email ", txtEmail.Text.Trim());
                  cmd.Connection = con;
                  con.Open();
                  //int result= cmd.ExecuteNonQuery(); //Returns the count of rows
effected by the Query
                  userID = Convert.ToInt32(cmd.ExecuteScalar()); //Returns the
first column of the first row
                  con.Close();
                }
            }
            string message = string.Empty;
            if (userID > 0)
                message = "Registration successful ";
            else message = "Sorry, your registration attempt was unsuccessful.
Please try again";

            LiteralText.Text = message;
      }
}
```

9. Run the project (F6), and have a look.

Now you can start your lab assignment!

Take Home Exercises

1. Write a program to insert two/more rows and two columns to a DataTable
2. Write a generic method for adding Data to a DataTable
3. Write a generic method for adding Data to a DataSet
4. Write a program to create link in data list and redirect you to the related page.
5. Write a program to keep/retrieve password in cookie.

USEFUL INFORMATION

Working with Views and Windows Data Providers

Provider Name	API prefix	Data Source Description
ODBC Data Provider	Odbc	Data Sources with an ODBC interface. Normally older data bases.
OleDb Data Provider	OleDb	Data Sources that expose an OleDb interface, i.e. Access or Excel.
Oracle Data Provider	Oracle	For Oracle Databases.
SQL Data Provider	Sql	For interacting with Microsoft SQL Server.
Borland Data Provider	Bdp	Generic access to many databases such as Interbase, SQL Server, IBM DB2, and Oracle.

References

1. Learn ASP.NET Web Application Framework, Tutorials Point, 2014
2. Michaelis, Mark, and Eric Lippert. Essential C# 6.0. Addison-Wesley Professional, 2015.