LAB -1: ASP.NET State Management

One of the most important facts with Web Applications is StateLess mode of behavior. This means that value which is posted by the end-user using browser towards the server is not maintain during the post-back. This is because the Http protocol using which when we make the request to the web server is stateless. So to maintain the state of the value state management techniques are provided as below:

- Client-Side State Management:
  - ViewState.
  - QueryString.
  - Cookies.
- Server-Side State Management:
  - Session.
  - Application.

Exercise 1: Using View State:

ViewState is an object used to store page specific data. Every web controls has 'EnableviewState' property. This object maintains the data in hidden fields on the client side. It is very important for the developer to make the decision on whether to enable viewstate of the control or not because of the following facts:

Advantages:

- Maintain the value (state) of the control during all consecutive post-backs of the same page.

Limitations:

- Excess use of the ViewState increases the rendered size of the page and may reduce the performance of the page.
- ViewState cannot be accessed across pages.
- Once we redirect from one page to another page, viewstate of the previous page is removed.

Task 1: Open VS2010/VS2008, create an ASP.NET Web Site. Name it as ASP_StateManagement Demos.

Task 2: Rename 'Default.aspx' to 'WebForm_ViewState.aspx'. Add the following code in the designer as below:

```
<asp:TextBox ID="txtName" runat="server"
Width="335px"></asp:TextBox>
```

```
        <asp:Button ID="btnSet" runat="server" 
onclick="btnSet_Click" 
            style="width: 34px" Text="Set" />
        <asp:Button ID="btnGet" runat="server" 
onclick="btnGet_Click" Text="Get" />
```

(Note: Ignore 'onClick' event here, we will use them in next tasks.)

Task 3: On the onClick event of both buttons write the following code:

```
protected void btnSet_Click(object sender, EventArgs e)
        {
            ViewState["Name"] = txtName.Text;
            txtName.Text = "";
        }

        protected void btnGet_Click(object sender, EventArgs e)
        {
            txtName.Text = ViewState["Name"].ToString() ;
        }
```

The above code define 'ViewState' object of 'StateBag' class. This class is responsible for managing the view state of all ASP.NET server controls. The code defines, 'ViewState["Name"]', object used to store the value entered in the TextBox 'txtName' on the click event of the 'Set' button. On the click event of the 'Get' button the value is retrieved back from the viewstate and redisplayed in the textbox.

Task 4: Right click on the 'WebForm_ViewState.aspx' and select 'View in Browser'.

Verification:

- Enter some text in the textbox and click on the 'Set' button.
- The TextBox now should be blank.
- Click on the 'Get' button, you should found the same text which you have entered in the textbox earlier.

Exercise 2: Using QueriString.

In Exercise 1, we have discussed advantages and limitations of ViewState. Now if we need to maintain the state of the values across the pages, e.g. based upon entries made on page one if we need to display some data on page two then we need to make use of QueriString. It is defined as the portion of the URL after question mark (?) in name/value pair.

E.g. : http://www.xyz.com?Name=Value

Following are the facts with QueryString:

Advantages:

- Used to pass values across the pages.

Limitations:

- Since values are displayed in URL, critically important value e.g. Account Number, UserName, Password or Credit Card number should not be passed using QueriString.
- Size of value should not be greater than 1024k.

Task 1: In the web site created in Exercise 1, add two new WebForm, name them as ‘frm_QueryStringSender.aspx’ and ‘frm_QueryStringReceiver.aspx’.

Task 2: On ‘frm_QueryStringSender.aspx’ add the below designer coed:

```
<asp:TextBox ID="txtfname" runat="server"></asp:TextBox>
        <asp:TextBox ID="txtlname" runat="server"></asp:TextBox>
        <asp:Button ID="btnSend" runat="server" onclick="btnSend_Click"
Text="Send" />
```

Task 3: Open ‘frm_QueryStringSender.aspx.cs’ and write the following code on ‘Send’ button:

```
protected void btnSend_Click(object sender, EventArgs e)
    {
        string pageUrl = "frm_QueryStringReceiver.aspx?Name=" +
txtfname.Text + " " +txtlname.Text;

        Response.Redirect(pageUrl);
    }
```

The above code defines a string variable named ‘pageUrl’. This contains value as the name of the receiver page and the name/value pair using ‘Name’ and textbox text properties. Using ‘Response.Redirect()’, the control will be shifted to the receiver page.

Task 4: Open ‘frm_QueryStringReceiver.aspx’ and add a label on it.

Task 5: Open ‘frm_QueryStringReceiver.aspx.cs’ and write the following code on ‘Page.Load’ event:

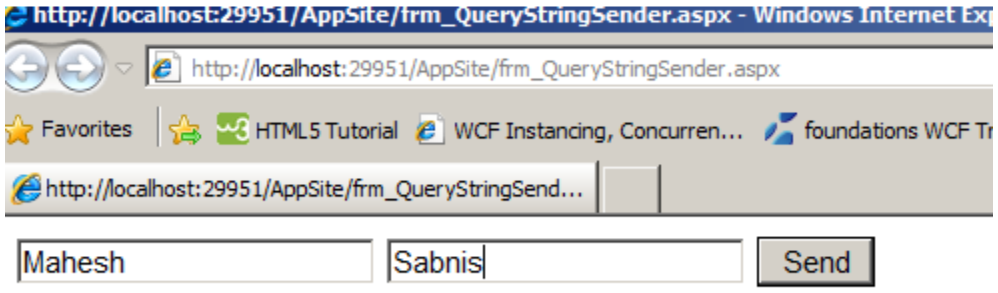
```
protected void Page_Load(object sender, EventArgs e)
    {
        string receivedValue = Request.QueryString["Name"];
        Label1.Text = receivedValue;
    }
```

The code above ‘Request.QueryString[“Name”]’, gets the HTTP QueriString variables and reads its value. This value will be further displayed the Label control.

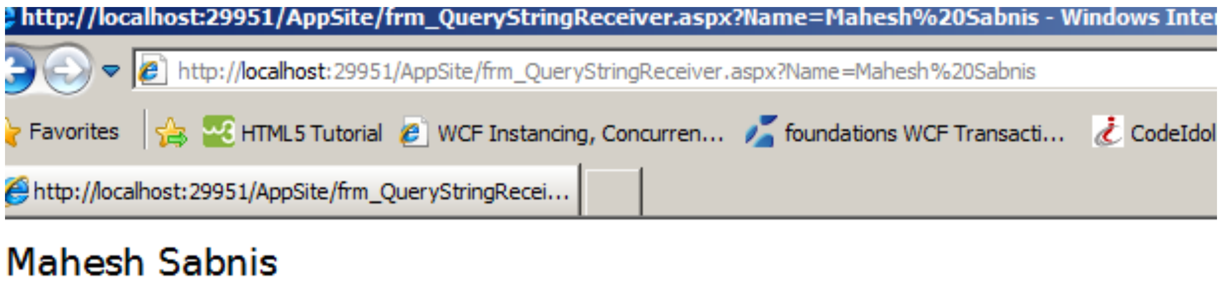
Task 6: Run the ‘frm_QueryStringSender.aspx’.

Verification:

- In the textboxes of the ‘frm_QueryStringSender.aspx’, type any values as below:


http://localhost:29951/AppSite/frm_QueryStringSender.aspx - Windows Internet Exp
http://localhost:29951/AppSite/frm_QueryStringSender.aspx
Favorites
HTML5 Tutorial
WCF Instancing, Concurren...
foundations WCF Tr
http://localhost:29951/AppSite/frm_QueryStringSend...
Mahesh
Sabnis
Send

- Click on the ‘Send’ button, the Url on the Open ‘frm_QueryStringReceiver.aspx’ will look as below with the values displayed in the label:


http://localhost:29951/AppSite/frm_QueryStringReceiver.aspx?Name=Mahesh%20Sabnis - Windows Inte
http://localhost:29951/AppSite/frm_QueryStringReceiver.aspx?Name=Mahesh%20Sabnis
Favorites
HTML5 Tutorial
WCF Instancing, Concurren...
foundations WCF Transacti...
CodeIdol
http://localhost:29951/AppSite/frm_QueryStringRecei...

Mahesh Sabnis

Exercise 3: Using Session.

After using the above two client side state management techniques, now it is the time to think about, the server side state management. The most important here is ‘Session’ state management techniques. It is more secure to use session state to maintain user critical information. The session objects are maintained by the web server in its metabase. Every Web Server have an algorithm to manage sessions. They are technically defined as Per User Request. Whenever any user makes first request to the web server the session is established and the entire information is managed on the server for that session. Following are some of the facts with session state:

- Class -> HttpSession.
- Properties:
    - IsNewSession.
    - SessionId.

- IsCookieLess.
- TimeOut.

Programmatically, the session information is stored in Session objects. These objects are created in 'Session_Start' event of the 'Global.asax' file of the web application.

Task 1: In the Web Application, created in Exercise 1, add two new WebForms, name them as 'frm_SessionSender.aspx' and 'frm_QueryStringReceiver.aspx'.

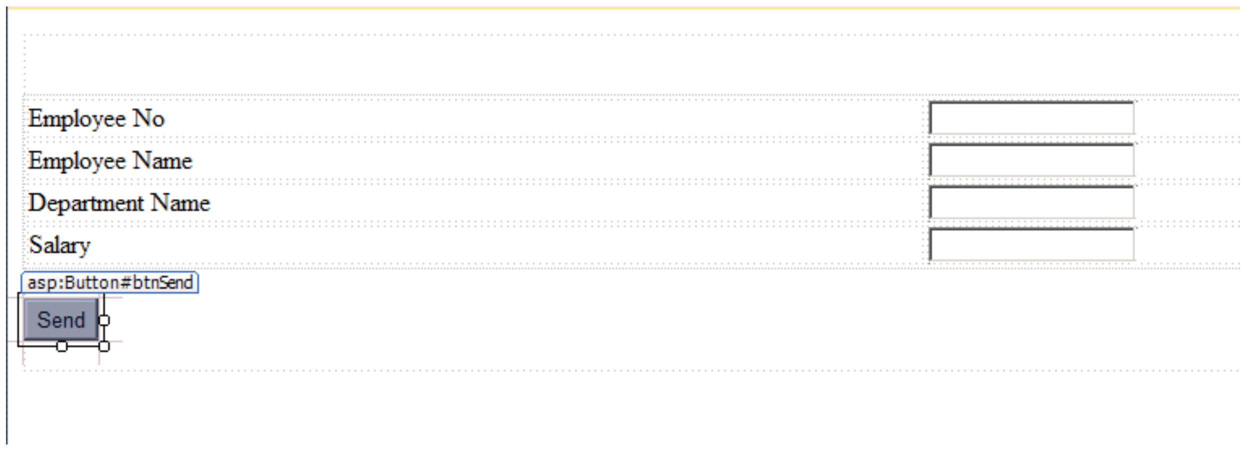
Task 2: Open 'Global.asax' file and add the following Session objects in 'Session_Start' event:

```
  void Session_Start(object sender, EventArgs e)
    {
        Session["Custom"] = "";
    }
```

Task 3: In the Web Application add a new class, name it as 'Employee' and add properties as below:

```
public class Employee
{
    public int EmpNo { get; set; }
    public string EmpName { get; set; }
    public string DeptName { get; set; }
    public double Salary { get; set; }
}
```

Task 4: Design 'frm_SessionSender.aspx' as below screen shot:



Task 5: Open 'frm_SessionSender.aspx.cs' and write the following code in click of 'Send' button:

```
protected void btnSend_Click(object sender, EventArgs e)
    {
        Employee objEmp = new Employee()
        {
             EmpNo=Convert.ToInt32(txteno.Text) ,
             EmpName= txtename.Text,
              DeptName= txtdname.Text,
               Salary = Convert.ToInt32(txtsal.Text)
        };
        Session["Custom"] = objEmp;
        Response.Redirect("frm_SessionReceiver.aspx");
    }
```

The code above defines an instance of ‘Employee’ class and set its property values using values entered in textboxes. The Employee object is then stored in Session object defined in Globakl.asax and the control is redirected to ‘frm_SessionReceiver.aspx’.

Task 6: On ‘frm_SessionReceiver.aspx’, add a Label control and open ‘frm_SessionReceiver.aspx.cs’. In the Page.Load event add the following code:

```
protected void Page_Load(object sender, EventArgs e)
    {
       Employee objEmp = (Employee)Session["Custom"];
       Label1.Text = objEmp.EmpNo.ToString()  + " " + objEmp.EmpName +
" " + objEmp.DeptName + " " + objEmp.Salary.ToString();
    }
```

The code above retrieve Employee information from the ‘Session[“Custom”]’ object and display values in Label control.

Task 7: Run the ‘frm_SessionSender.aspx’ and enter values in Textboxes and click on ‘Send’ button. The information entered on textboxes should be displayed on ‘frm_SessionReceiver.aspx’.

Exercise 4: Using Cookies

Cookies are the files created by the server on the client’s machine over the session. These are the files used for storing personalization information for the site with respect to the session user. These files are used in every post back from client to web server for data communication. If cookies are disabled the communication takes place using QueriString. But by considering limitations of QueriString, it is strongly recommended that cookies should not be disabled. HttpCookie class is used to create cookies on the client. It has ‘Name’, ‘Path’, ‘Value’ and ‘Expires’ etc properties. Following are facts about cookies:

Advantages:

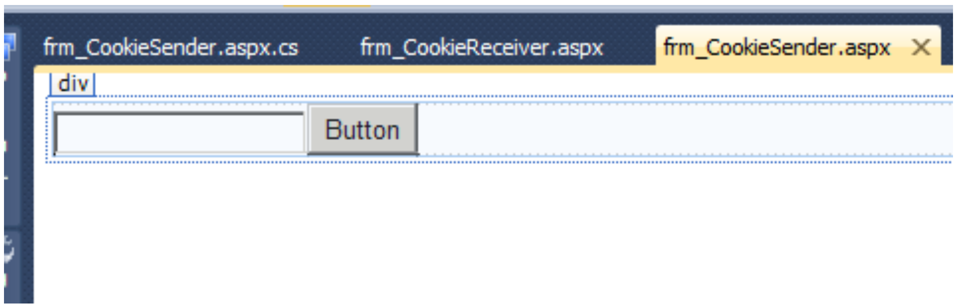
- Contains session specific information in the file created on end-user’s machine.
- Can contain more data as compare to QueriString.

Limitations:

- Cannot be transferred across machines.

Task 1: In the Web Application in the Exercise 1, add two new WebForms, name them as ‘frm_CookieSender.aspx’ and ‘frm_CookieReceiver.aspx’.

Task 2: Design ‘frm_CookieSender.aspx’ as below:



Task 3: Open ‘frm_CookieSender.aspx.cs’ and write the following code in ‘Send’ button click event:

```
protected void Button1_Click1(object sender, EventArgs e)
    {
        HttpCookie cok = new HttpCookie("MyCookie");

        cok.Path = "/";

        cok.Expires = DateTime.Now.AddMinutes(30);

        cok.Value = TextBox1.Text;

        Response.Cookies.Add(cok);

        Response.Redirect("frm_CookieReceiver.aspx");
    }
```

The code above defines an object of ‘HttpCookie’ class with properties like, ‘Name’, ’Expires’ and ‘Value’. Using ‘Response.Cookies.Add’ the cookie object is added in the cookie collection and then the control is transferred to ‘from_CookieReceiver.aspx’ page.

Task 4: Open ‘from_CookieReceiver.aspx.cs’ and the page loaded event add the following code:

```
protected void Page_Load(object sender, EventArgs e)
    {
        HttpCookie cok = Request.Cookies["MyCookie"];

        Response.Write(cok.Value);
    }
```

The code above request the cookie objects using its name and reads value stored in it.

Task 5: Run the ‘frm_CookieSender.aspx’, enter some text in TextBox and click on the ‘Send’ button, you will find value from the cookie displayed on the ‘frm_CookieReceiver.aspx’.

Case Study:

Create an ASP.NET Web application for Storing Personal Information as below:

- Create a Class Person with Following Properties with validation:
  - PersonId -> Must be Numeric.
  - PersonName -> Must be Character.
  - Age -> Must be Numeric.
  - Gender -> Must be Character (Male/Female)
  - Address -> Can be Alpha Numeric.
  - StateName
  - CityName
- Create another class PersonBizAction with following method(s):
  - CreatePerson
    - Return Type -> void.
    - Input Params -> Person object.
  - GetAllPersons()
    - Return Type: -> List<Person>
    - Input Params -> None.
  - GetAllPersonByState()
    - Return Type -> List<Person>
    - Input Params -> string stateName
  - GetAllPersonByCity()
    - Return Type -> List<Person>
    - Input Params -> string cityName
  - GetAllPersonByGender()
    - Return Type -> List<Person>
    - Input Params -> string cityName

- Add Web Forms as below:
  - Frm_RegistrationPerson.
  - Frm_PersonDetails
- Create ‘frm_RegisterPerson.apsx’ web form with following design:
  - Textboxes for following properties:
    - PersonId, PersonName, Age.
  - DropDownList for following properties:

        - State, Cities, Gender.
    - Multi-Line TextBox
        - Address
    - Buttons for following:
        - Save.
        - Display.
    - Validation controls according property validations.
- Functionality on ‘Frm_RegisterPerson.apsx’:
    - State and City DropDownLists must be filled with respective State and City Names.
    - When user selects any StateName from drop down list, the city drop down list must be filled with respective city names. (Define logic (methods) accordingly in ‘PersonBizAction’ class.)
    - After user enter valid data in respective controls and click ‘Save’ button, the data must be added in List<Person> and all textbox controls must be made blank, and lists must be reseted.
    - After clicking on ‘Display’ button, control must be redirected to ‘Frm_PersonDetails.aspx’ page. This will Table control where all the PersonDetails will be displayed from the List<Person>. Based upon properties in the person class and number of person records in List<Persobn> Columns and Rows must be added in the table control on this page. (Write logic accrodingle. Hint: TableRow and TableCell class.)
    - On ‘Frm_RegisterPerson.apsx’, when user selects ‘State’ and click on the ‘Display’ button the control must be redirected to ‘Frm_PersonDetails.aspx’ where statewise person details must be displayed in Table control. Repeat the same for City and gender wise person details.
- Optianal:
    - Try to provide ‘Edit’ and ‘Delete’ functionality.