Class: BCA Semester - V
Subject: Programming with Asp.Net
UNIT-2

33.List state management methods and explain any one with

There are different methods which are used for State Management. They are

1. View State
2. Cookies
3. Hidden Form Fields
4. Query Strings
5. Session State
6. Application State

Hidden Form Fields

This is client side methods for state management. HTTP is state less protocol. You can save the state of the data by using Hidden Form Fields. A hidden field stores text data with the HTML <input type="hidden"> or by using <asp:HiddenField>.

A hidden form field is not visible in the browser but you can set its properties like other controls. When page is posted to the server ,the content of the hidden form field is sent in the HTTP Form collection along with the values of other controls. In order to make available hidden form field value during page processing ,you must submit the page using HTTP post method.

That means you can not use hidden form field,if the page is processed in response to a link or HTTP GET method.

ASP.NET provides <asp:HiddenField> for state management. <asp:HiddenField> supports following properties.

Properties

Property	Description
Value	Sets the value which is stored in HiddenField
Visible	Sets Hidden Form Field is visible or not.

Example

```
<asp:HiddenField ID="HiddenField1" runat="Server" Value="This is the Value of Hidden field" />
```

```
<asp:Button ID="Button1" runat="Server" Text="Change Label Text to Hiidden Field Value" onclick="Button1_Click" /> <br />
<asp:Label ID="Label1" runat="server" Text="Before Submit Form"></asp:Label>


        // CODE BEHIND
        // Fires when Button is clicked
  protected void Button1_Click(object sender, EventArgs e)
   {
      Label1.Text = HiddenField1.Value;


   }
```

34.Explain server side and client side state management methods.

Client Side State Management Methods

1. View State
2. Cookies
3. Hidden Form Fields
4. Query Strings

Server Side State Management Methods

1. Session State
2. Application State

Note:-Explain these methods in further answers.

35.Explain View State in detail.

Understanding View State

- The HTTP protocol, the fundamental protocol of the World Wide Web, is a stateless protocol. Each time you request a web page from a website, from the website's perspective, you are a completely new person.
- The ASP.NET Framework, however, manages to transcend this limitation of the HTTP protocol. For example, if you assign a value to a Label control's Text property, the Label control retains this value across multiple page requests.

- This page contains a Button control and a Label control. Each time you click the Button control, the value displayed by the Label control is incremented by 1. How does the Label control preserve its value across postbacks to the web server?

```
<%@ Page Language=”C#” %><!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<script runat=”server”>
protected void btnAdd_Click(object sender, EventArgs e)
{
    lblCounter.Text = (Int32.Parse(lblCounter.Text) + 1).ToString();
}
</script>
<html xmlns=”http://www.w3.org/1999/xhtml” >
<head id=”Head1” runat=”server”>
<title>Show View State</title>
</head>
<body>
<form id=”form1” runat=”server”>
<div>
<asp:Button id=”btnAdd” Text=”Add” OnClick=”btnAdd_Click” runat=”server”/>
<asp:Label id=”lblCounter” Text=”0” runat=”server” />
</div></form></body></html>
```

- ✓ The ASP.NET Framework uses a trick called View State. If you open the page in your browser and select View Source, you’ll notice that the page includes a hidden form field named __VIEWSTATE that looks like this:

~~~~~~
~~~~~~

<input type=”hidden” name=”__VIEWSTATE” id=”__ VIEWSTATE” value=”/wEPDwUKLTc2ODE1OTYxNw9kFgICBA9kFgIC Aw8PFgIeBFRleHQFATFkZGT3tMnThg9KZpGak55p367vfInj1w==” />

~~~~~~

- ✓ This hidden form field contains the value of the Label control’s Text property (and the values of any other control properties that are stored in View State).
- ✓ When the page is posted back to the server, the ASP.NET Framework rips apart this string and re-creates the values of all the properties stored in View State. In this way, the ASP.NET Framework preserves the state of control properties across postbacks to the web server.
- ✓ By default, View State is enabled for every control in the ASP.NET Framework. If you change the background color of a Calendar control, the new background color is remembered across postbacks. If you change the selected item in a DropDownList, the selected item is remembered across postbacks. The values of these properties are automatically stored in View State.
- ✓ View State is a good thing, but sometimes it can be too much of a good thing. The __VIEWSTATE hidden form field can become very large. Stuffing too much data into View State can slow down the rendering of a page because the contents of the hidden field must be pushed back and forth between the web server and web browser

Disabling View State

- ✓ In certain circumstances, you might want to disable view state for an individual control or for an ASP.NET page as a whole. For example, you might have a control that contains a lot of data (imagine a RadioButtonList control with 1,000 options).
- ✓ You might not want to load the data into the hidden __VIEWSTATE form field if you are worried that the form data would significantly slow down the rendering of the page.
- ✓ You also might want to disable view state if you need to automatically reinitialize a control every time a page is loaded.
- ✓ You can disable view state for an individual control by modifying the EnableViewState property. You can use this property to disable view state for both HTML and Web controls.

- ✓ Instead of disabling view state control by control, you also can disable view state for the whole page. To disable view state for an entire page, modify the EnableViewState attribute of the Page directive.

```
<%@ Page EnableViewState="False" %>
```

Adding Values to View State

- ✓ You can take advantage of view state within your code by adding values to the state bag class. If you add values to this class, they are automatically added to the hidden VIEWSTATE form variable; therefore, they are available across multiple form posts.
- ✓ To add a value to the state bag, use a statement such as the following:
- ✓ **ViewState("SomeItem") = "Some Value"**
~~~~~~

- ✓ This statement adds a new item to the state bag class named SomeItem with the value Some Value.

Example

```
protected void btnCounter_Click(object sender, EventArgs e)
  {
    int c;
    if (ViewState["counter"] == null)
    {
      c = 1;
      ViewState["counter"] = c;
    }
    else
    {
      c = Convert.ToInt32(ViewState["counter"]) + 1;
      ViewState["counter"] = c;
    }
    lblCounter.Text = "Counter:" + c.ToString();
  }
```

36.Explain QueryString in details.

- ✓ Often you need to pass variable content between your html pages or aspx webforms in context of **Asp.Net**. For example in first page you collect information about your client, her name and last name and use this information in your second page.
- ✓ For passing variables content between pages **ASP.NET** gives us several choices. One choice is using **QueryString** property of Request Object. QueryString is used to add some additional information on URL.You can add some additional information after ? on URL and pass it along with URL.
- ✓ You should not pass sensitive information with querystring.For example you should not pass password information with URL.Consider following example.

https://www.xyz.com/welcome.html?name=Vijay

This html addresses use **QueryString** property to pass values between pages. In this address you send 2 information.

1. *Welcome.html* this is the page your browser will go.
2. name=Vijay you send a name variable which is set to Vijay

As you have guessed ? starts your **QueryString**, and & is used between variables. Building such a query string in **Asp.Net** is very easy.

Advantages of this approach

1. It is very easy.

Disadvantages of this approach

1. **QueryString** have a max length, If you have to send a lot information this approach does not work.
2. **QueryString** is visible in your address part of your browser so you should not use it with sensitive information.
3. **QueryString** can not be used to send & and space characters.

37.Explain Cookies in detail.

- ✓ Cookie is a small text file which is stored on client's machine. Cookie is used to store some small information on the client.

  Cookie is consist of two parts.

  1.key➔Which is the name of the cookie.

  2.value➔Which is the information stored in the cookie.

- ✓ You can also specify the life of cookies. If you do not specify they will last on client's machine till you do not clean it. You can also specify particular date and time when the cookie should be automatically deleted from client's machine.
- ✓ Cookies are used to transfer some information from one page to another page. client can disable cookies on its browser. So you should not store important information on cookie.
- ✓ You need to use HttpCookie class to create or use cookies.

  HttpCookie v=new HttpCookie("User","ABC");

- ✓ In above example "v" is the cookie object."User" is the name of cookie and "ABC" is the value of cookie.

You can do the same thing as follows:

HttpCookie v=new HttpCookie();

v.Name="User";

v.Value=”ABC”;

You can also specify expiration time.

v.Expires=DateTime.Now.AddDays(30);

It means cookie will remain stored for 30 days from the date of creation.

Property	Description
Name	It specifies the name of cookie.
Value	It holds the value of cookie.
Expires	Get or Set expiration time for cookie.
Domain	If you want to store cookie in a specific folder,you can specify Domain.
Path	Get or set path of cookie to be stored.
TimeStamp	Gives you date and time of cookie creataion.

After creation of cookie you have to add cookie to the response object as follows:

HttpCookie v=new HttpCookie(“User”,”ABC”);

Response.Cookies.Add(v);

To retrieve cookie information

String user=Request.Cookies[“User”].Value;

~~~~~~

Example

```
protected void btnCookie_Click(object sender, EventArgs e)
    {
        HttpCookie cok = new HttpCookie("User");
        cok.Value = "abc";
        cok.Expires = DateTime.Now.AddDays(30);
        Response.Cookies.Add(cok);
        Response.Redirect(@"~\ReadCookie.aspx");
    }
```

~~~~~~

When you click on button User named cookie create with abc value, we read this cookie in ReadCookie.aspx page.

```
protected void Page_Load(object sender, EventArgs e)
    {
      HttpCookie cok=Request.Cookies["User"];
      Response.Write(cok.Value);
    }
```

38.Exaplian ApplicationState in details.

Using Application State

- ✓ You can store variables and objects directly in application state. An item that has been added to application state is available within any ASP.NET page executing in the application. Items stored in application state function like global variables for an application.
- ✓ The following statement, for example, creates a new item named myItem with the value Hello! in application state:
  Application["myItem"] = "Hello!";
- ✓ After this statement is executed in an ASP.NET page, the value of myItem can be retrieved within any other ASP.NET page contained in the same application. To read the value of an application variable, you can use a statement like this:

  Response.Write(Application["myItem"].ToString());
- ✓ This statement displays the value of the application item named myItem.
- ✓ You can add more complex objects, such as collections and DataSets, in an application state. For example, you can add an existing DataSet to application state like this:

  Application["DataSet"] = dstDataSet;

Methods of the Application Sate are as follows:

Property	Description
Add	It is used to add application variable.
Clear	It is used to clear all application state key and value.
GetKey	It is used to get the key value of application state.
Lock	The Lock method locks the application state object so that it can be accessed only by the current thread. Typically, this means that only the current page can access objects in application state.

Unlock	The Unlock method releases the lock on application state. It enables other pages to access values in application state again.
Remove	It is used to remove application value.
RemoveAll	It is used remove all application values.
RemoveAt	It is used to remove particular application value.

Example

Global.asax page

```
void Application_Start(object sender, EventArgs e)
    {
        Application["visitor"] = 0;
    }
void Session_Start(object sender, EventArgs e)
    {
        int counter = Convert.ToInt32(Application["visitor"]) + 1;
        Application["visitor"] = counter;
    }
```

First of all create application variable named visitor in Application_Start event and then after increament it by one in Session_Start event.

```
protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Visitor No." + Application["visitor"].ToString());
    }
```

Here in page load event we print visitor using application variable.

39.Explain session state in detail.

- ✓ When user requests any web page from web site ,Web server assigns **Session ID**
- ✓ and gives each client memory portion to store information. Each client gets different sessionId.

- ✓ When you close your browser the session is lost and all the information stored under SessionID is cleared from the server memory. Client does not have any any authority to stop session handling .Any information stored under session is global to application. But session data is different for different user.

Methods of Session object are as follows:

Property	Description
Add	It is used to add session variable.
Abandon	It is used to terminated all session variables.
Clear	It is used to clear key and value of session.
Remove	It is used to remove specific session value.
RemoveAll	It is used remove all session values.
RemoveAt	It is used to remove particular session value.

Example

Here first we create session called user and then read it in ReadSession.aspx page.

~~~~~~

```
protected void btnCreateSession_Click(object sender, EventArgs e)
   {
      Session["user"] = "ABC";
      Response.Redirect(@"~\ReadSession.aspx");
   }
```

~~~~~~

~~~~~~

ReadSession.aspx

```
protected void Page_Load(object sender, EventArgs e)
   {
      Response.Write("Welcome " + Session["user"].ToString());
   }
```
~~~~~~

HARIVANDANA
EDUCATION TRUST