Agenda

- *Static Members*
- *View State*
- *Hidden Field in Form*
- *Query String*
- *HttpContext*
- *Cookies - HttpCookie*
- *Sessions - HttpSessionState*
- *Application – HttpApplicationState*
- *Summary of All Features.*

Static member in WebForm Class

It can used for Sharing data between requests by any user but its scope is local to the WebForm in which it is declared.

Using Static/Shared member

```
public partial class Default : System.Web.UI.Page
{
   public static int SharedCounter;
   public int InstanceCounter;
   protected void Page_Load(object sender, EventArgs e)
   {
      SharedCounter += 1;
      InstanceCounter += 1;
      Response.Write("Press F5 and refresh the page<br>");
      Response.Write("Instance Counter : " + InstanceCounter + "<br>");
      Response.Write("Shared Counter : " + SharedCounter + "<br>");
   }
}
```

In this example the value of **InstanceConter** is always "1" because with every request a new instance of the webform is created and InstanceCounter always by default gets intialized to "0".

The value of the **SharedCounter** increments with every request (irrespective of the client or browser instance) because all the instances of the webform share a single copy of SharedCounter because it's a Shared/Static member of the class.

Static Members in static class in App_Code folder:

We can write a class in **App_Code** folder and it can be shared by all the webforms in the web application. By making the members of the class as shared/static we can store data in that member in one webform and retrieve that in another webform.

The data in such variables is shared by all the clients.

Step1: Add a class to App_Code folder: /App_Code/GlobalSettings.cs

GlobalSettings class

```
public static class GlobalSettings
{
   public static string Data;
}
```

Step 2: Add to the project "**Page1.aspx**"

Page1.aspx

```
<asp:TextBox ID="txtDemo" runat="server"></asp:TextBox>
<asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
<a href="Page2.aspx">Page2.aspx</a>
```

Handling btnSubmit click event

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
   GlobalSettings.Data = txtDemo.Text;
}
```

Step 3: Add to the project "**Page2.aspx**" with following entered inside form tag.

```
<div><%= GlobalSettings.Data %></div>
```

Step 4: Make Page1.aspx as Startup Page.

Step 5: Run the WebApplication

Step 6: Enter Data in textbox of Page1 and Submit, on server the data is stored in static variable of the class in App_Code folder.

Step 7: Navigate to Page2.aspx to see that the value entered in Page1 is retrieved here from the static variable of the class in App_Code folder.

ViewState

- ViewState is a **collection** of **Key-Value** pairs, where **Key** is of type **String** and **Value** is of type **Object**. But only those objects which are marked as **Serializable** can be stored in ViewState.
- It's of type **StateBag** and is used for storing state/data to be used in **postback** to the same webform.
- ViewState being hidden element in form tag rendered, its different for different clients & is submitted to server only if form is submitted or posted back.
- ViewState cannot be programmed on client in javascript because the value of it is **encrypted** before it is rendered to the browser.
- If ViewState of the page is disabled (<%@**Page EnableViewState**="**False**") then anything added to it is not retained.
- **ViewStateMode**: This property can take 3 values,
  1. Enabled
  2. Disabled
  3. Inherit

Example 1:

In ViewStateDemo.aspx:

```
<asp:Button ID="btnNew" runat="server" Text="New" onclick="btnNew_Click" />
```

```
<asp:Button ID="btnEdit" runat="server" Text="Edit" onclick="btnEdit_Click" />
<asp:Button ID="btnSave" runat="server" Text="Save" Visible="False"
onclick="btnSave_Click"/>
```

ViewStateDemo.aspx.cs

```
protected void btnNew_Click(object sender, EventArgs e)
{
   btnNew.Visible = false;
   btnEdit.Visible = false;
   btnSave.Visible = true;
   ViewState["state"] = "new";
}
protected void btnEdit_Click(object sender, EventArgs e)
{
   btnNew.Visible = false;
   btnEdit.Visible = false;
   btnSave.Visible = true;
   ViewState["state"] = "edit";
}
protected void btnSave_Click(object sender, EventArgs e)
{
   if (ViewState["state"].ToString() == "new")
      Response.Write("New record inserted");
   else
      Response.Write("Existing record updated");
}
```

Steps for running the execution:

1. Open ViewStateDemo.aspx in two browser instances
2. In one browser instance click on “New” Button and in other click on “Edit” button.
3. In both browsers click on “Save” Button
4. In First browser we get the o/p: New record inserted
5. In Second browser we get the o/p: Existing record updated

Explanation: When “New” was clicked in first browser, the value of key (state) in ViewState collection is set to “new” and the same is rendered to the browser in the form of hidden element. Later when “Save” was clicked, along with rest of the form elements the value of hidden element “ViewState” is also submitted and thus the value of key (state) in ViewState is found to be “new”.

Example2: Storing Integer in PropertyBag

```
protected void Page_Load(object sender, EventArgs e)
{
   int n;
   if (ViewState["cnt"] == null)
      n = 0;
   else
      n = (int)ViewState["cnt"];
   n++;
   ViewState["cnt"] = n;
   Response.Write(n.ToString());
}
```

Example 3: Storing object of type Demo in ViewState

```
//Storing a serializable object (Demo) in PropertyBag
protected void Page_Load(object sender, EventArgs e)
{
   Demo d;
   if (ViewState["do"] == null)
   {
      d = new Demo();
      d.N = 1;
   }
   else
   {
      d = (Demo)ViewState["do"]; //Deserialzation
      d.N++;
   }
   ViewState["do"] = d; //Serialization
   Response.Write(d.N.ToString());
}
```

HiddenField

- Like ViewState, HiddenField is also used for managing state of client in round trips to the same webform.
- ViewState is managed by Page framework but HiddenField has to be coded by us.
- It can store data which is of type **string** and cannot store serializable custom objects unlike ViewState
- The value of HiddenField can be programatically set on client using Javascript, the same can be retrieved on server from the HiddenField object (Example 2).

Example 1: (Similar to ViewState Example2)

Add a HiddenField and Button to the WebForm.

Put the following code in Page_Load:

Page Load event handler

```
protected void Page_Load(object sender, EventArgs
{
   if (string.IsNullOrEmpty(HiddenField1.Value))
      HiddenField1.Value = "0";
   int n = int.Parse(HiddenField1.Value);
   n++;
   HiddenField1.Value = n.ToString();
   Response.Write(n.ToString());
}
```

Example 2: To get Datetime of the client when the form was posted

In a Form add the button and HiddenField:

Using Hidden field

```
<asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClientClick="SetClientTime()" />
<asp:HiddenField ID="hdnClientDateTime" runat="server" />
```

In <Head> add the script:

JavaScript

```
<script language="javascript">
   function SetClientTime() {
      document.getElementById("hdnClientDateTime").value = new Date(); //Current Date & time of Client
   }
</script>
```

In CS File add the following:

Page Load event handler

```
protected void Page_Load(object sender, EventArgs e)
{
   if (IsPostBack)
      Response.Write("Client Date/Time: " + hdnClientDateTime.Value.ToString());
}
```

Note: Both ViewState and Hidden element can be used only for managing state in round trips to same webform but they cannot be used for managing state between different webforms of the web application.

QueryString

- QueryString is the string in URL after "?".
- It's a collection of Key-Value pairs and both are of datatype string.

- It is ideal for transfering of data from one page to other when using either **HyperLink** or **Response.Redirect**.
- The query string should not have special characters like "&", "+", "?" etc used in either parameter or its value.
- **The value of QueryString parameter must be encoded using Server.UrlEncode. But it's not required to decode because Request.QueryString automatically decodes it.**

In QSPage1.aspx: (Startup Page)

Adding controls

```
Faculty Name:   <asp:TextBox ID="txtFaculty" runat="server"></asp:TextBox>
<asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click" Text="Submit" /><br />
<asp:HyperLink ID="hlnkPage2" runat="server">Page2</asp:HyperLink>
```

In **Page_Load** Event Handler

Page Load event handler

```
protected void Page_Load(object sender, EventArgs e)
{
   hlnkPage2.NavigateUrl = "QSPage2.aspx?Faculty=" + Server.UrlEncode("F1+F2") + "&Timings=7.30AM";
}
protected void btnSubmit_Click(object sender, EventArgs e)
{
   string url = "QSPage2.aspx?Faculty=" + Server.UrlEncode(txtFaculty.Text) + "&Timings=7.30AM";
   Response.Redirect(url);
}
```

In QSPage2.aspx

In **Page_Load** Event handler

Page Load event handler

```
protected void Page_Load(object sender, EventArgs e)
{
   Response.Write(Request.QueryString["Faculty"] + "<br>");
   Response.Write(Request.QueryString["Timings"] + "<br>");
}
protected void btnSubmit_Click(object sender, EventArgs e)
{
   string url = "QSPage2.aspx?Faculty=" + Server.UrlEncode(txtFaculty.Text) + "&Timings=7.30AM";
   Response.Redirect(url);
}
```

Note: Server.HtmlEncode can be used for encoding characters like <, >, &, etc in Html Body of a web page.

<%= **Server.HtmlEncode(**"a<b><>&><"**) +** "   Demo" %>

Improvements in Code Expressions in .NET 4.0

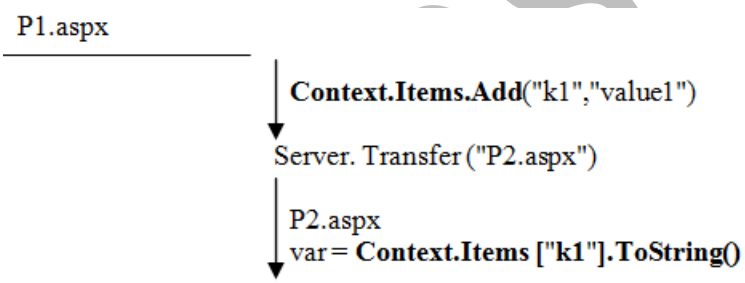
There is a new code expression releasing with ASP.Net 4.0 version that will html encode the value.

<%: HTMLOutput %> is same as <%= HttpUtility.HtmlEncode(HTMLOutput) %>

HttpContext (Context)

1. Its an object created on server for **every request** and thus its life time is same as the life time of the request.
2. It can be used for managing state in a given request so that code in different files of the web application including master pages, webform, user control, custom classes etc can share the same state.
3. If a Key-Value pair is added to the context of a request, it can be retrieved back from the context anytime during that requests lifetime.
4. It can be used for managing state while switching from one page to another using **Server.Transfer**.

Here Key and value both are of type object. If Key used is of type "string" then it is case sensitive.



In Page1.aspx

In Button Click

Page1.aspx.cs

```
Context.Items.Add("k1","value1"); //Add a key value pair to the context.
Server.Transfer("Page2.aspx");
```

In Page2.aspx

In Page Load:

Page2.aspx.cs

```
Response.Write(Context.Items["k1"].ToString()); //Retrieve from context the Value based on the key.
```

Note:

Using *HttpContext.Current* we can get the reference to the context of the current request in any part of the web application.

We can get reference to Request, Response, Server objects from the current context of the request. i.e. HttpContext.Current.Request

Working with Cookies

1. Cookie is a Name-Value pair which on behalf of server is saved on client machine (either in memory or in file)
2. Once saved, web browser automatically includes this name-value pair along with every request it submits to that server.
3. Cookies are used for sharing data across webforms requested by same user/client.
4. Common Examples of Cookies
    - Authenticated User information
    - User specific settings on the site
    - Shopping Cart
5. Cookies generated on server (added to the response) are rendered to the browser through a **HTTP Response Header** called "**Set-Cookie**".

    Set-Cookie: ***name***=*VALUE*; **expires**=*DATE*;**path**=*PATH*; **domain**=*DOMAIN_NAME*; **secure=**lsHTTPS

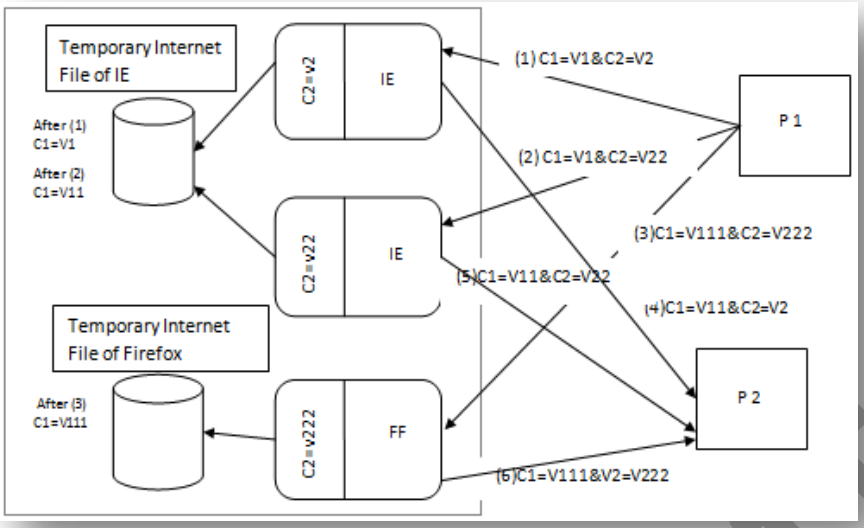
    These attributes are used by browser for deciding whether to include the cookie in the request or not (based on the URL of the request).
6. With every request the browser automatically includes all the cookies appropriate to that request.The cookie Name-Value pairs are submitted in the form of **HTTP Request Header** called "**Cookie**" and they are saved on the server in the form of CGI environmental variable (Server Variable) called as **HTTP_COOKIE**.

    Cookie: N1=v1;N2=v2;N3=v3.

Note: Domain, Path, Expires and Secure attributes are not included with a cookie in the request.

Types of Cookies:

1. **Persistent Cookie**: Cookies which are permanantly stored on the client machine in a file. These cookie values are shared by all the browser instances of a particular type running on that machine i.e Cookies of IE cannot be used in Firefox browser but all the instances of IE will share a common set of Persistent cookies.
2. **Non-Persistent Cookie** These Cookies which are temporarily stored in the browser's memory. These are not shared across browser instance and are also called as **Session Cookie**. Every instance of IE will have a different set of Non-Persistent cookies.



Programming Cookies in ASP.NET

Request.Cookies is a collection of **HttpCookie** objects contructed for every cookie included in the request.

Response.Cookies is a collection of **HttpCookie** objects created for every cookie to be included in the response so that these cookies can be appended to the existing list of cookies on the client machine.

- If a cookie added to the response is already existing on the client and also if its domain and path attributes match to the new cookie, the old cookie is overwritten with the new cookie.
- The only way of destroying the cookie already with client / browser is to add to the response a cookie with same name and path properties but an **expired date**.

Default values for the properties of Cookie

- **Value** = " "
- **Path** = "/"
- **Expires** = (If not set then it's a Non persistent cookie)
- **Domain** = "<Current domain of the website>"
- **Secure** = False

Example 1: Default.aspx

Default.aspx

```
Name:  <asp:TextBox ID="txtKey" runat="server"></asp:TextBox>  <br />
Value: <asp:TextBox ID="txtValue" runat="server"></asp:TextBox><br />
<asp:CheckBox ID="chkPersistance" runat="server" Text="Is Persistance" /><br />
<asp:CheckBox ID="chkSecured" runat="server" Text="Is Secured" /><br />
<asp:Button ID="btnSetCookie" runat="server" OnClick="btnSetCookie_Click" Text="Set Cookie" />
<asp:Button ID="btnGetCookie" runat="server" OnClick="btnGetCookie_Click" Text="Get Cookie" />
<asp:Button ID="btnDestroyCookie" runat="server" OnClick="btnDestroyCookie_Click" Text="Destroy Cookie" />
```

```
<asp:Button ID="btnGetAllCookies" runat="server" OnClick="btnGetAllCookies_Click" Text="Get All Cookies" />
<asp:Label ID="lblValue" runat="server" /></asp:Label>
```

Default.aspx.cs

```
protected void btnSetCookie_Click(object sender, EventArgs e)
{
   HttpCookie c = new HttpCookie(txtKey.Text);
   c.Value = txtValue.Text;
   c.Path = Request.ApplicationPath;
   if (chkPersistance.Checked)
      c.Expires = DateTime.MaxValue;
   if (chkSecured.Checked)
      c.Secure = true;
   Response.Cookies.Add(c);
}
protected void btnGetCookie_Click(object sender, EventArgs e)
{
   HttpCookie c = Request.Cookies[txtKey.Text];
   if (c == null)
      txtValue.Text = "Missing";
   else
      txtValue.Text = c.Value;
}
protected void btnDestroyCookie_Click(object sender, EventArgs e)
{
   HttpCookie c = new HttpCookie(txtKey.Text);
   c.Expires = DateTime.Now.AddDays(-1);
   c.Path = Request.ApplicationPath;
   Response.Cookies.Add(c);
}
protected void btnGetAllCookies_Click(object sender, EventArgs e)
{
   lblValue.Text = "";
   foreach (String key in Request.Cookies)
      lblValue.Text += key + " " + Request.Cookies[key].Value + "<br>";
}
```

Persistent cookies are stored on client in Temporary Internet Folder

To View them: Tools → Internet Options → General → Settings → View Files…(Sort by Last Accessed Date) → The cookie file name will be based on Application Name (provided the cookie path is set).

Note: The Persistent cookie value can be modified on the client machine. If the cookie value / file size are not changed the browser uses the modified value (if closed and restarted) otherwise the browser automatically destroys all the cookies in the cookie file.

Example 2:

Step 1: Add to the web application project Page1.aspx and make it as Startup Page.

Step 2: Add to it a TextBox (txtName) and Button (btnContinue)

Adding controls to Page1.aspx

```
Name: <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
<asp:Button ID="btnContinue" runat="server" Text="Continue"  onclick="btnContinue_Click" />
```

Step 3: Add Page2.aspx to the web application

Step 4: Add to it a TextBox (txtName) and Button (btnContinue)

Adding controls to Page2.aspx

```
Age: <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
<asp:Button ID="btnContinue" runat="server" Text="Continue"  onclick="btnContinue_Click" />
```

Step 5: Add Page3.aspx to the web application:

Step 6: Add two Literal Controls to it.

Adding controls to Page3.aspx

```
Age: <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
<asp:Button ID="btnContinue" runat="server" Text="Continue"  onclick="btnContinue_Click" />
```

Step 7: In Page1.aspx handle Click event of btnContinue and write the following code in it:

Code behind: Page1.aspx.cs

```
HttpCookie cookie = new HttpCookie("name");
cookie.Value = txtName.Text;
Response.Cookies.Add(cookie);
Response.Redirect("Page2.aspx");
```

Step 8: In Page2.aspx handle Click event of btnContinue and write the following code in it:

Code behind: Page2.aspx.cs

```
HttpCookie cookie = new HttpCookie("age");
cookie.Value = txtAge.Text;
```

```
Response.Cookies.Add(cookie);
Response.Redirect("Page3.aspx");
```

Step 9: In Page3.aspx handle Load event of Page and write the following code in it:

Code behind: Page3.aspx.cs

```
lcName.Text = Request.Cookies["name"].Value;
lcAge.Text = Request.Cookies["age"].Value;
```

Example 3: Login Demo using Cookies:

Login.aspx

```
<asp:Label ID="lblError" runat="server" Text=""></asp:Label> </div>
Username:<asp:TextBox ID="txtUsername" runat="server"></asp:TextBox> <br />
Password:<asp:TextBox ID="txtPassword" runat="server"></asp:TextBox>  <br />
<asp:CheckBox ID="chkRemember" runat="server" Text="Remember my u/p on this machine" /><br />
<asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin_Click" />
```

Login.aspx.cs

```
protected void btnLogin_Click(object sender, EventArgs e)
{
   string un = txtUsername.Text;
   string pwd = txtPassword.Text;
   if (un == pwd)
   {
      //Valid user
      HttpCookie cookie = new HttpCookie("AuthCookie");
      cookie.Value = un;
      if (chkRemember.Checked)
         cookie.Expires = DateTime.MaxValue;
      Response.Cookies.Add(cookie);
      Response.Redirect("~/default.aspx");
   }
   else
      lblError.Text = "Invalid u/p";
}
```

Default.aspx

```
<asp:LinkButton ID="lnkLogout" runat="server" Visible="False" OnClick="lnkLogout_Click"> Logout</asp:LinkButton>
```

Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
   HttpCookie cookie = Request.Cookies["AuthCookie"];
   lnkLogout.Visible = (cookie != null);
   if (cookie != null)
      Response.Write("Hello " + cookie.Value); //Client is Authenticated
   else
     Response.Redirect("Login.aspx") //Client is not Logged-In.
}

protected void lnkLogout_Click(object sender, EventArgs e)
{
   HttpCookie cookie = new HttpCookie("AuthCookie");
   cookie.Expires = DateTime.Now.AddDays(-1);
   Response.Cookies.Add(cookie);
   Response.Redirect("LoginForm.aspx");
}
```

CookieDictionary

Limitations of Cookies:

There are limitations on the number of cookies that a client can store at any one time.

- 300 total cookies
- 4 kilobytes per cookie
- 20 cookies per server or domain

A cookie which has many SubKey - Value pairs is called as "**CookieDictionary**".

Note: While using CookieDictionary don't use character "&" in its SubKey or Value.

Using CookieDictionary

Cookie Dictionary Name: `<asp:TextBox ID="txtDictionaryName" runat="server"/><br />`

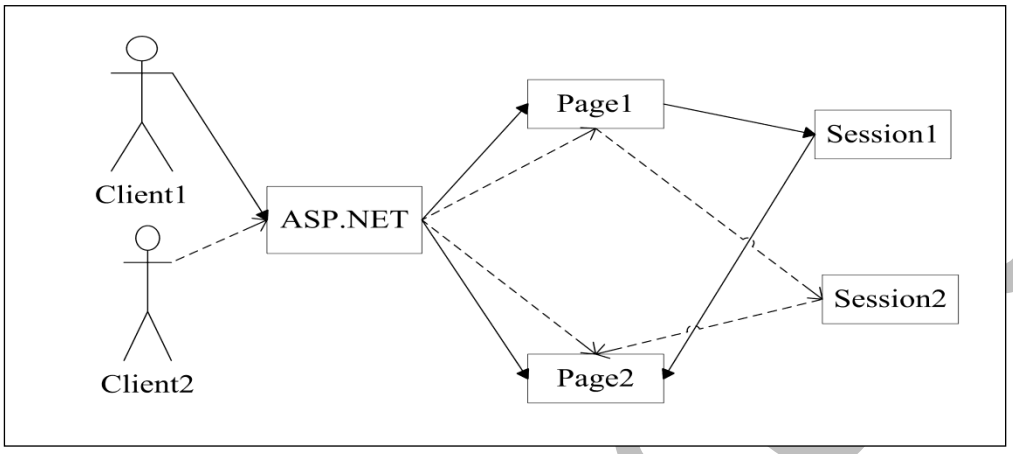
Sub Key: `<asp:TextBox ID="txtSubKey" runat="server" /><br />`

Value: `<asp:TextBox ID="txtValue" runat="server" /><br />`

```
<asp:Button ID="btnAddToDictionary" runat="server" OnClick="btnAddToDictionary_Click" Text="Add To
Dictionary"/>
<asp:Button ID="btnShowDictionary" runat="server" OnClick="btnShowDictionary_Click" Text="Show Dictionary" />
<asp:Label ID="lblValue" runat="server"></asp:Label>
```

Session State Management

- A session is an **object (of class HttpSessionState) created on server** on behalf of a given client.
- This object is created upon sever receiving the first request from the client and the same object is reused for all subsequent requests by that client.
- It has key-value pairs representing the state of the client on behalf of whom it is created.



- The key-value pair is of type string-object and hence in session we can store any type of data unlike cookies where we can only store only strings.
- Also sessions can store any amount of data and also we can have unlimited number of key-value pairs.
- When a new session is created a unique **ID** is assigned to it and the same is rendered in the response as a **non-persistent cookie.** With all subsequent requests, the browser includes this cookie and asp.net uses it for deciding which session object to be used for that client's request.
- Time out period is defined for every session. The default value is 20 minutes. If the time after the last request by the client exceeds the timeout period then the session corresponding to that client is automatically destroyed on the server.
- A Session is not immediately destroyed if the browser instance is closed on the client. Because the browser doesn't update the server when it is closed thus ASP.NET server keeps the session for the Timeout period even though it is not anymore useful.
- Programmatically a session can be destroyed using **Session.Abandon()** but the session i.e. variables and its data would be destroyed after the response of the current page is rendered (but not immediately when that line of code is executed).

  Session.RemoveAll() removes all keys in the session but the session object is not destroyed.

Example1: SessionDemo.aspx.cs

Page_Load event

```
protected void Page_Load(object sender, EventArgs e)
{
   if (Session["cnt"] == null)
      Session["cnt"] = 0;
```

```
    Session["cnt"] = (int)Session["cnt"] + 1;
    lblSession.Text = Session["cnt"].ToString();
}
```

- If this page is requested from two different browser instances, the value of Session["cnt"] will be independent/different in both the browser instances.
- But from an already existing browser, if a New Window is opened (Ctrl + N), as both the browser windows belong to the same instance they will use the same value for Session["cnt"].

Example2:

Login.aspx

```
<div><asp:Label ID="lblError" runat="server" Text=""/></div>
Username:<asp:TextBox ID="txtUsername" runat="server"></asp:TextBox> <br />
Password:<asp:TextBox ID="txtPassword" runat="server"></asp:TextBox>  <br />
<asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin_Click" />
```

Login.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Session["Username"] != null)
            Response.Redirect("Default.aspx");
    }
}
protected void btnLogin_Click(object sender, EventArgs e)
{
    string un = txtUsername.Text;
    string pwd = txtPassword.Text;
    if (un == pwd) //if u/p are same then its valid u/p other its invalid
    {
        Session["Username"] = un;
        Response.Redirect("default.aspx");
    }
    else
        lblError.Text = "Invalid u/p";
}
```

Default.aspx

Default.aspx

```
This is Default page of the WEB APPLICATION.<br />
Hello <asp:Label ID="lblUserName" runat="server"></asp:Label><br /><br />
<asp:LinkButton ID="lnkLogin" runat="server" OnClick="lnkLogin_Click"></asp:LinkButton>
```

Default.aspx.cs

Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
   if (Session["Username"] != null)
   {
      //User has logged in.
      lblUserName.Text = Session["Username "].ToString();
      lnkLogin.Text = "Logout";
   }
   else
   {
      //Not logged in yet
      lblUserName.Text = "Anonymous (Not logged in)";
      lnkLogin.Text = "Login";
   }
}
protected void lnkLogin_Click(object sender, EventArgs e)
{
   if (Session["Username"] == null)
      Response.Redirect("Login.aspx");
   else
   {
      Session.Remove("Username"); //Logout by removing the key from session.
      Server.Transfer(Request.Path); //Transer to current page so that Page_Load is executed again.
   }
}
```

Code: 8.20 C#

In some cases we may not want to use sessions in the web page then we set the page directive attribute **EnableSessionState** to false. <%@ Page …. EnableSessionState="false" %>. This improves the performance of that page as session object of page need not be initialized.

EnableSessionState = True / False / Readonly.

Session.IsReadOnly: This property is used to check if the session is read only or not.

Session.IsNewSession: This property is used to check if it is the first request from the client.

Session.Timeout = 30 (default is 20 mins)

Configuring Session In web.config (Write the following after the section <System.Web>:

```
<sessionState timeout="30" cookieless="UseUri" cookieName="Demo"/>
```

Cookieless Session: If the Request from the client is received and if the URL doesn't have any SessionID then a new Session is created for the client and the request is redirected to the URL of current page with Session included.

Ex: http://localhost/WebApp/(S(4c31np554tn4hv55tt0zlc3m))/Default.aspx

> Note: While giving link to other pages the URL must be framed using "~", this includes the application path and the session id.

If the session is managed using URI, url must be framed as below:

```
string url = "http://" + Request.Url.Host + ":" + Request.Url.Port +
Response.ApplyAppPathModifier(Request.ApplicationPath) + "Demo.aspx";
```

WebGarden & WebFarm

- If a single instance of a web application spans over the multiple instances of the webserver running on the same machine is then it is called as **WebGarden**,
- If a single instance of a web application spans over the multiple instances of the webserver running on the **different** machine is then it is called as **WebFarm**,

  ```
  <sessionState mode="[InProc]/StateServer/SqlServer" stateConnectionString = "tcpip=localhost:42424" />
  ```
- If mode="InProc", Session variables are stored in the AppDomain (of the web application) of the worker process (aspnet_wp.exe)
- If session is used for a web application on webfarm or webgarden, the sessionState mode must be set to either **StateServer** or **SqlServer**.
- State server is a windows service and can be started using ControlPanel → AdminstrativeTools → Services → ASP.NET StateService (right click and select start).
- The port number on which the state service is running can be changed in windows registry at the following location:

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\aspnet_state\Parameters\Port

Compressing Session Values (4.0 feature)

ASP.NET session out-of-process state values are saved in a database or on the server. These are saved in a serialized format. Bigger session values consume more resources to be sent to the server. Now, those can be compressed with a **new** built-in property **compressionEnabled**. This attribute for the sessionState element can be mentioned in the *web.config*, like this:

```
<sessionState mode="SQLServer" stateConnectionString=". . ." compressionEnabled="true"/>
```

HttpApplicationState

DataType of **Application** is **HttpApplicationState**.

An Application object is used for sharing name-value pairs across different web pages irrespective of the client requesting them.

Using "Application" object

```
if (Application["cnt"] == null)
   Application["cnt"] = 0;
Application.Lock();
Application["cnt"] = (int)Application["cnt"] + 1;
Application.UnLock();
lblApplication.Text = Application["cnt"].ToString();
```

Code: 8.21 C#

Because the datatype of the value in application variables is object, any type of data can be stored in the application variables. This makes it as NOT type safe. Instead the same state can be managed using Static members in a class under APP_CODE folder.

Summary of StateManageTechniques:

	Location	Secured	Is State Shared Across Pages	Across Clients
Static Class	Server	Yes	Yes	Yes
Application	Server	Yes	Yes	Yes
ViewState	Client	Yes	No	No
HiddenField	Client	No	No	No
QueryString	Client	No	Two Pages	No
Context	Server	Yes	No (Request Only)	No
Cookies	Client	No	Yes	No
Session	Server	Yes	Yes	No