School of Computer Science &

Information Technology

Department of Master of Computer Applications C# & .NET Technologies – 18MCA304

Unit V – **Web & ASP.NET**

Kathiresan V Assistant Professor

School of Computer Science & IT JGI Knowledge Campus Jayanagar 9th block

Bengaluru - 69

**Contact Ph.: Mail Id:**

**+91-9611677907**

**+91-8660161661** [**v.kathiresan@jainuniversity.ac.in**](mailto:v.kathiresan@jainuniversity.ac.in)

**global.asax**

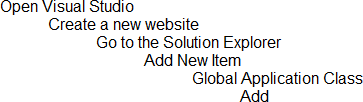
**IsPostBack**

Global.asax is an optional file which is used to handling higher level application events such as Application\_Start, Application\_End, Session\_Start, Session\_End etc. It is also popularly known as ASP.NET Application File. This file resides in the root directory of an ASP.NET-based application.

Global.asax contains a Class representing your application as a whole. At run time, this file is parsed and compiled into a dynamically generated .NET Framework class derived from the HttpApplication base class. You can deploy this file as an assembly in the \bin directory of an ASP.NET application. The Global.asax file itself is configured so that if a user requests the file, the request is rejected. External users cannot download or view the code written within it.

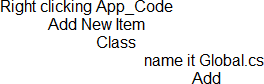
# How to careet a Global.asax file

Global.asax file don't create normally; you need to add it by yourself. How to ?



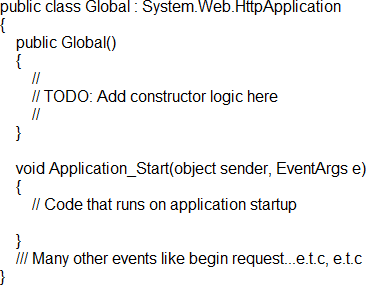
Your Global.asax file

After that you need to add a class in your project.



Inherit the newly generated by System.Web.HttpApplication and copy all the method created Global.asax to Global.cs and also add an inherit attribute to the Global.asax file

Now your Global.asax will look like following:



The Global.asax file is in the root application directory. While Visual Studio .NET automatically inserts it in all new ASP.NET projects, it's actually an optional file. It's okay to delete it?if you aren't using it. The .asax file extension signals that it's an application file rather than an ASP.NET file that uses aspx. The Global.asax file is configured so that any direct HTTP request (via URL) is rejected automatically, so users cannot download or view its contents. The ASP.NET page framework recognizes automatically any changes that are

made to the Global.asax file. The framework reboots the application, which includes closing all browser sessions, flushes all state information, and restarts the application domain.

Themes

A theme decides the look and feel of the website. It is a collection of files that define the looks of a page. It can include skin files, CSS files & images.

We define themes in a special App\_Themes folder. Inside this folder is one or more subfolders named Theme1, Theme2 etc. that define the actual themes. The theme property is applied late in the page's life cycle, effectively overriding any customization you may have for individual controls on your page.

# How to apply themes

There are 3 different options to apply themes to our website:

1. Setting the theme at the page level: the Theme attribute is added to the page directive of the page.

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inheri

ts="Default" Theme="Theme1"%>

1. Setting the theme at the site level: to set the theme for the entire website you can set the theme in the web.config of the website. Open the web.config file and locate the <pages> element and add the theme attribute to it:

<pages theme="Theme1">

....

....

</pages>

1. Setting the theme programmatically at runtime: here the theme is set at runtime through coding. It should be applied earlier in the page's life cycle ie. Page\_PreInit event should be handled for setting the theme. The better option is to apply this to the Base page class of the site as every page in the site inherits from this class.

Page.Theme = Theme1;

# Uses of Themes

1. Since themes can contain CSS files, images and skins, you can change colors, fonts, positioning and images simply by applying the desired themes.
2. You can have as many themes as you want and you can switch between them by setting a single attribute in the web.config file or an individual aspx page. Also you can switch between themes programmatically.
3. Setting the themes programmatically, you are offering your users a quick and easy way to change the page to their likings.
4. Themes allow you to improve the usability of your site by giving users with vision problems the option to select a high contrast theme with a large font size.

# Example

Create 2 themes for the page – one with red background (Theme1) and another with an image as a background (Theme2). When the user selects a particular theme from the ListBox then that theme should be applied dynamically for the page.



# Answer

1. Solution Explorer -> Right click -> Add ASP.NET folder -> Themes.

A new folder App\_Themes is added to the Solution Explorer and a new folder Theme1 is added inside it.

1. Theme1 -> Right click -> Add new item -> Stylesheet -> name it as Theme1.css
2. Inside Theme1.css body

{

background-color:Red;

}

1. Again add a new file.

App\_Themes -> Right click -> Add ASP.NET folder -> Themes A new folder Themes2 is created.

1. Themes2 -> Right click -> Add new item -> Stylesheet -> name it as Theme2.css
2. Create an Images folder inside Theme2 and add a picture file pic1.jpg inside this. File Theme2.css contains –

body

{

background-image:url(Images/pic1.jpg);

}

1. Now create a default.aspx as follows. Add a heading and a list box having AutoPostBack to True.

<body>

<form id="form1" runat="server">

<div>

<h3>Select your page Theme : </h3>

<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True" Height="41px" onselectedindexchanged= "ListBox1\_SelectedIndexChanged" Width="175px">

<asp:ListItem>Theme1</asp:ListItem>

<asp:ListItem>Theme2</asp:ListItem>

</asp:ListBox>

<br />

</div>

</form>

</body>

1. Inside the default.aspx.cs file a static variable themeValue is defined which saves the value of current theme. In the Page\_PreInit event of the page, selected theme from the ListBox is applied to the page and inside the constrctor of the page this PreInit event is provided a EventHandler.

public partial class default : System.Web.UI.Page

{

static string themeValue;

public ThemeTest()

{

this.PreInit+=new EventHandler(Page\_PreInit);

}

private void Page\_PreInit(object sender, EventArgs e)

{

Page.Theme = themeValue;

}

protected void ListBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

themeValue = ListBox1.SelectedItem.Value; Response.Redirect(Request.Url.ToString());

}

}

1. Now run the page. When we select the Theme from the ListBox, immediately the page is automatically applied with the theme.

**Note :** In the ListBox event Response.Redirect is used since the theme is applied in the PreInit event only. So when the Item is selected then PreInit is already over. So you need to resend the data to the server so that the PreInit event is handled there and the Theme is applied automatically.

What is Themes

Sometimes we need to change the layout of application for different-different user. For example if a single application is used by different-different client and every client has his own color schema, logo, font etc. requirements then we use themes **A theme is a collection of skin files, css, graphics, images etc**.

How to Apply Themes

To add theme we have to first create the theme folder using name App\_Themes.

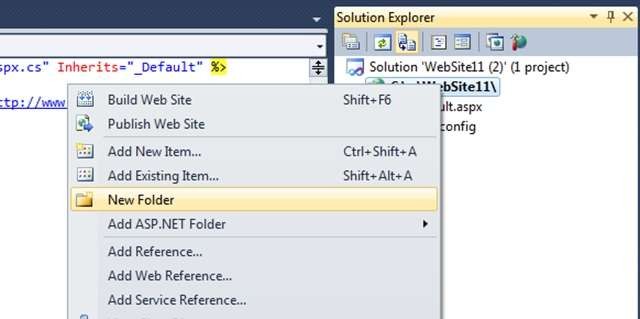
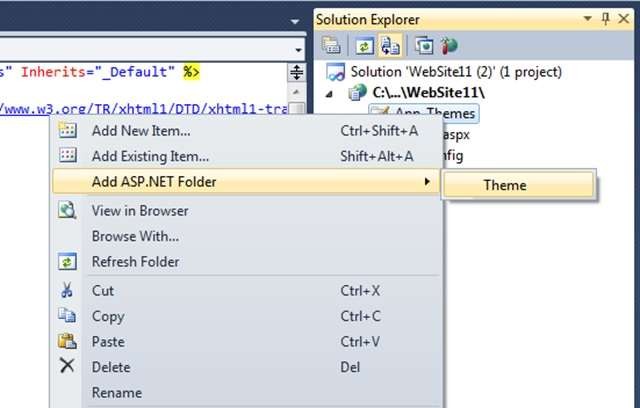


Figure 1

Now add theme in the App\_Themes folder by clicking on right on this folder and go to Add Asp.net folder and select theme. And give it proper name. I created two theme named Client1 and Client2.



# Figure 2

**How to Apply Theme at Compile time**

Now I will tell how to apply the theme at compile time. Later on in this article I will also tell how to add theme dynamically.

To add theme on page we need to set the Theme property at page directive.

<%@ Page Theme="Client1" Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>

This them will be applied only in this page. But if we want to apply theme on whole project then we need to add it at web config file. In the <system.web> tag we have to add the following tag:-

<pages theme="Client1"/>

Now this them will be applied to all pages in this project. Skin files in Themes

Skin files are those files in which we define the common property setting for asp.net controls like button, texbox, label etc. The extension of skin files is .skin. It comes under the themes.

Type of Skin

There are two type of skin in asp.net and these are as follows:- Default Skin

When we want to apply same property for all controls in the page then we use default skin. This skin automatically applies to the control which defined in the skin. This type of skin does not contain the SkinId property.

Named skin

A named skin is applied to those controls in which we pass skin id. A named skin contains the SkinId.

Example of Skin:-

Right click on the theme in which you want to add skin file and give it proper name.

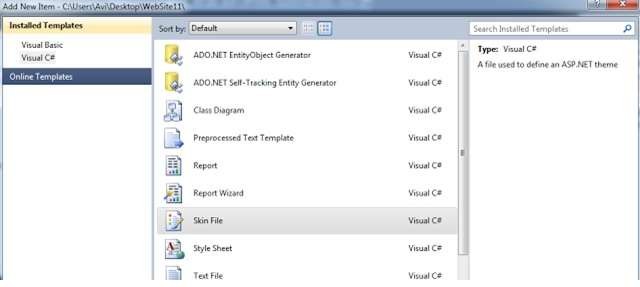


Figure 3

Empty the skin file and add your own code in this file. For example I added skin for button where I add the back color property which is as follows:-

<asp:Button runat="server" BackColor="Red" />

The output of this code as follows:-

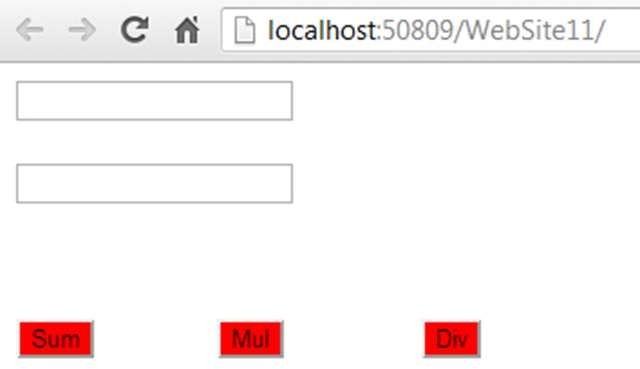


Figure 4

This is the example of Default skin file.

Now add the control property using skinId which is as follows:-

<asp:Button runat="server" BackColor="Red" skinid="sk1" />

<asp:Button runat="server" BackColor="Green" skinid="sk2" />

<asp:Button runat="server" BackColor="Pink" skinid="sk3" />

Now apply these skin ids to the buttons in the form which is as follows:-

<asp:Button ID="Button1" runat="server" Text="Sum" SkinID="sk1" />

<asp:Button ID="Button2" runat="server" Text="Mul" SkinID="sk2" />

<asp:Button ID="Button3" runat="server" Text="Div" SkinID="sk3" />

The output of this code as follows:-



Figure 4

Change theme dynamically

If we want to change the theme dynamically using programming then we have to change it in Page\_PreInit event. With the help of following code we can dynamically set the theme:-

Page.Theme = "Client2";

According to user status you can change them using this code. Similarly as we work with skin files we can add the CSS file too.

# Cookie

A cookie is a small bit of text that accompanies requests and pages as they go between the Web server and browser. The cookie contains information the Web application can read whenever the user visits the site.

In simple term, cookie is a small text file sent by web server and saved by web browser on client machine.

# Some point to understand cookies

It is a server side object.

Cookies are used in state management. HttpCookie class is there for cookie by .NET Class-> httpSessionState

SessionID is send to the client in the form of cookie by server if the request is generated for the first time.

# Session Cookie

Name -> ASP.NET\_SessionID Value -> will be alphanumeric value.

Name and value will be sent to client as session cookie (which is stored in cookie header). ASP.NET 2.0 it is also used for mobile development.

Session timeout is 20 minutes by default (it depends on sliding time request). In mobile development each request is treated as new request.

Earlier to 2.0 session was maintained but 2.0 onwards we have to explicitly maintain session (so we can use web and mobile development).

If the browser doesn't support cookie or cookies are off in browser, Cookie less session is maintained.

Cookies can only store string type of data.

You can only store 4kb (4096 bytes) of data in cookie. When browser goes out of scope cookies are expired.

Cookies can be stored in persist and non persist manner. By default cookies are non persistent. Persistent cookies are sorted on client browser.

Cookies data is not secured.

Cookies are identified on basis of there name. Single and multiple value cookies (on basis of key). Cookies are domain specific also.

IE support more than 100 cookies.

Cookies were introduced from Netscape browser.

You can set the expiration date of the cookie for some day in the future it will remain their until that day unless manually deleted by the user

I will explain you with the help of a program how to store cookies and how to retrieve the cookies.

# Code for Default.aspx design page

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits= "\_Default" %>

<!DOCTYPE html PUBLIC "-

//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-> transitional.dtd">

<html xmlns="[http://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

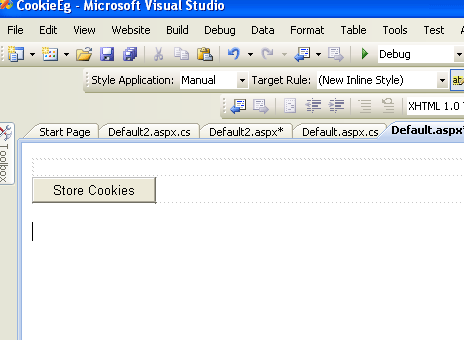
<asp:Button ID="Button1" runat="server" Text="Store Cookies" OnClick="Button1\_Clic k" />

</div>

</form>

</body>

</html>



# Code for Default.aspx.cs page

**using** System;

**using** System.Configuration;

**using** System.Data; **using** System.Linq; **using** System.Web;

**using** System.Web.Security;

**using** System.Web.UI;

**using** System.Web.UI.HtmlControls;

**using** System.Web.UI.WebControls;

**using** System.Web.UI.WebControls.WebParts;

**using** System.Xml.Linq;

**public** partial **class** \_Default : System.Web.UI.Page

{

**protected void** Page\_Load(**object** sender, EventArgs e)

{

}

HttpCookie myCookie;

**protected void** Button1\_Click(**object** sender, EventArgs e)

{

myCookie = **new** HttpCookie("test");

myCookie.Values["name"] = "Puran Singh Mehra";

myCookie.Values["age"] = "30+";

myCookie.Values["Profession"] = "Software";

myCookie.Values["Address"] = "India";

myCookie.Expires = DateTime.Now.AddHours(1);

Response.Cookies.Add(myCookie);

Response.Redirect("Default2.aspx");

}

}

# Output 1:



**Code for Default2.aspx design page**

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default2.aspx.cs" Inherits

="Default2" %>

<!DOCTYPE html PUBLIC "-

//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-> transitional.dtd">

<html xmlns="[http://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Style="top: 231px

;

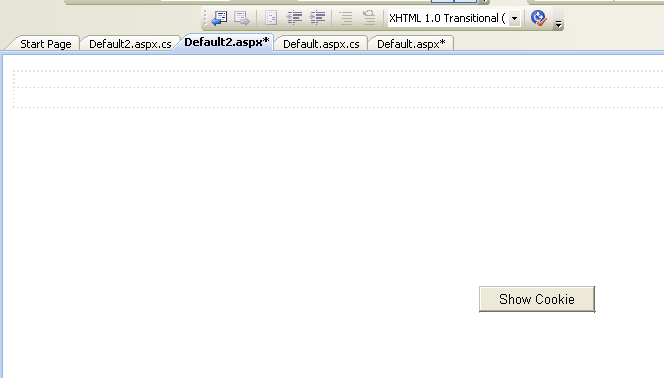
left: 476px; position: absolute; height: 26px; width: 116px" Text="Show Cookie" />

</form>

</html>

</body>

</div>



# Code for Default2.aspx.cs design page using System;

**using** System.Collections;

**using** System.Configuration;

**using** System.Data;

**using** System.Linq;

**using** System.Web;

**using** System.Web.Security;

**using** System.Web.UI;

**using** System.Web.UI.HtmlControls;

**using** System.Web.UI.WebControls;

**using** System.Web.UI.WebControls.WebParts;

**using** System.Xml.Linq;

**public** partial **class** Default2 : System.Web.UI.Page

{

**protected void** Page\_Load(**object** sender, EventArgs e)

{

**foreach** (**string** str **in** Request.Headers)

{

Response.Write(str + "=" + Request.Headers[str] + "<br>");

}

Response.Write(" " + "<br>"

);

}

**protected void** Button1\_Click(**object** sender, EventArgs e)

{

**foreach** (**string** str **in** Request.Cookies["test"].Values)

{

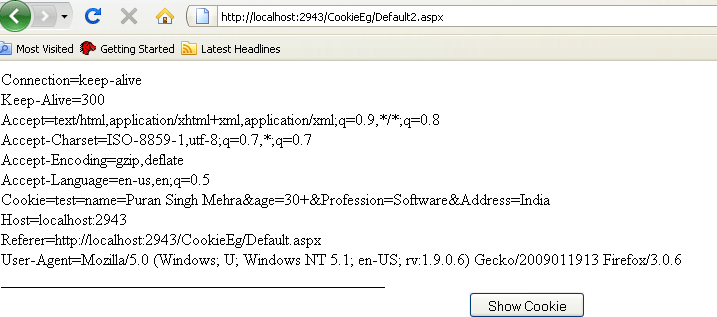
Response.Write(str + "=" + Request.Cookies["test"].Values[str] + "<br>");

}

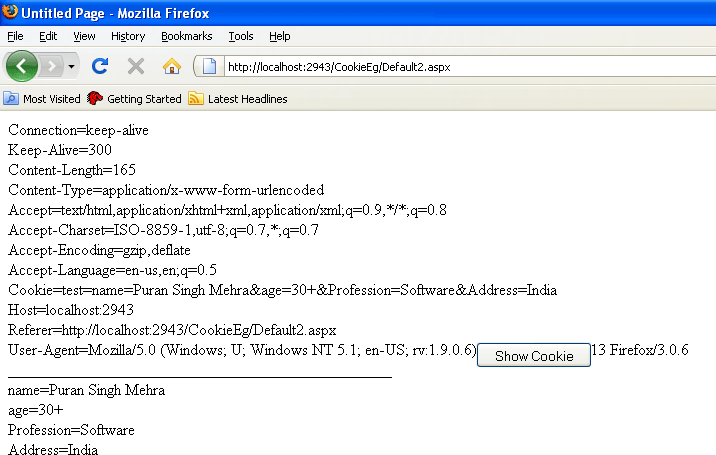
}

}

# Output 2:



**Output 3:**



**Cookies Merits**

Data is stored on client side, so it is faster. Cookies are best used to store small amounts of data,

for example it can store User ID and Password. This UserID and Password can then be used to identify the user and read user information from a database or other data store. It is recommended to encrypt a password before it is stored in a cookie as cookies are not secured.

# Cookies Demerits

Security problem as they are stored on client. Client can delete cookie any time.

Browsers also impose limitations on how many cookies your site can store on the user's computer.

Cookie can only store 4kb of data.

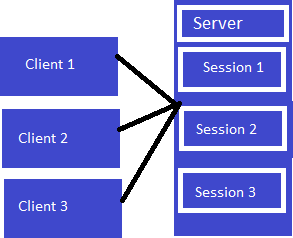
Data is not authenticated i.e. client can delete the cookie.

**Session**

We all know that the web uses the HTTP protocol and the HTTP protocol is a stateless protocol; in other words, when a client sends a request to the server, an instance of the page is created and the page is converted to HTML format and then the server provides the response and then the instance of the page and the value of the control are destroyed. So if we have a requirement to store the values of the controls and pass them into another web form then a State Management Technique is used.

Introduction

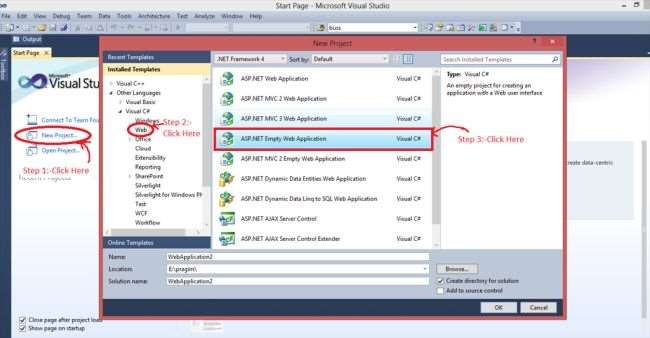
Session is a State Management Technique. A Session can store the value on the Server. It can support any type of object to be stored along with our own custom objects. A session is one of the best techniques for State Management because it stores the data as client-based, in other words the data is stored for every user separately and the data is secured also because it is on the server.’



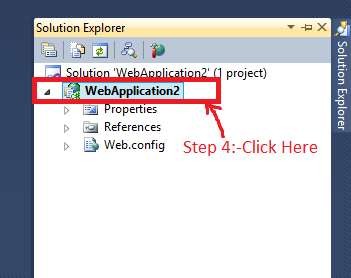
Now here I am explaining sessions with an example.

**Step 1:** Open Visual Studio 2010.

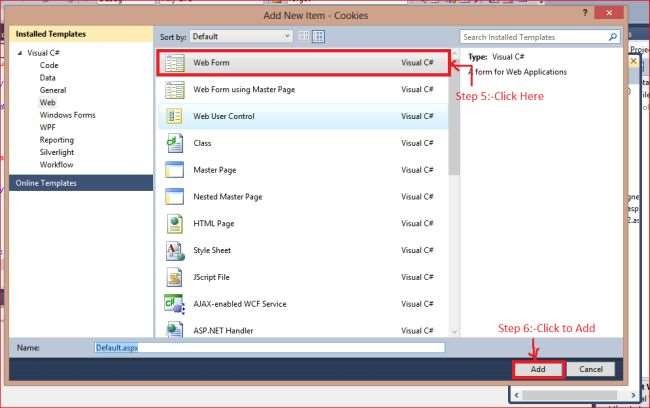
**Step 2:** Then Click on "New Project" -> "WEB" -> "ASP.NET Empty Web Application".



**Step 3:** Now click on Solution Explorer.



**Step 4:** Now right-click on the "Add" -> "New Item" -> "Web Form" and add the name of the web form and I had added 2 Web Form1.aspx and Web Form2.aspx.



**Step 5:** After adding the web form the following code is added to Web Form1.aspx.

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" In herits="Session.WebForm1" %>

<!DOCTYPE html PUBLIC "-

//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-> transitional.dtd">

<html xmlns="[http://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head id="Head1" runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

User Name:-<asp:TextBox ID="tbUserName" runat="server"></asp:TextBox>

<br />

<br />

Password:-<asp:TextBox ID="tbpwd" runat="server"></asp:TextBox>

<br />

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="Submit" />

</div>

</form>

</body>

</html>

And the code is:

**protected void** Button1\_Click(**object** sender, EventArgs e)

{

//textbox value is stored in Session

Session["UserName"] = tbUserName.Text;

Session["Pwd"] = tbpwd.Text;

Response.Redirect("WebForm2.aspx");

}

And then add the following code to Web Form2.aspx.

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm2.aspx.cs" In herits="Session.WebForm2" %>

<!DOCTYPE html PUBLIC "-

//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-> transitional.dtd">

<html xmlns="[http://www.w3.org/1999/xhtml"](http://www.w3.org/1999/xhtml)>

<head id="Head1" runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

User Name:-<asp:TextBox ID="tbUserName" runat="server"></asp:TextBox>

<br />

<br />

Password:-<asp:TextBox ID="tbpwd" runat="server"></asp:TextBox>

<br />

</div>

</form>

</body>

</html>

And the code of the code behind is:

**protected void** Page\_Load(**object** sender, EventArgs e)

{

//Session value is assign on the text box

**if** (Session["UserName"] != **null**)

{

tbUserName.Text = Session["UserName"].ToString();

}

**if** (Session["Pwd"] != **null**)

{

tbpwd.Text = Session["Pwd"].ToString();

}

}

Now to set the session we need to use a config file. We can set the session on one of the following 2 types of configuration files:

1. Machine Configuration file: Machine Configuration is applied for all application.
2. Application Configuration file: It's applied for only application by application basis.

The following is the configuration of the Web.config file:

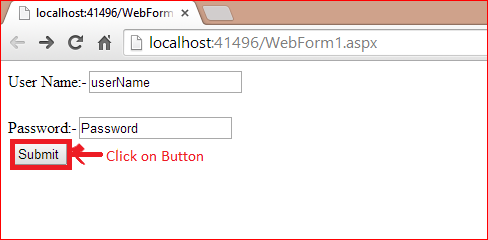
<system.web>

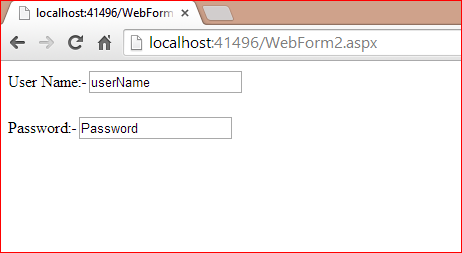
<sessionState mode="SQLServer" sqlConnectionString="Server=DIVS\SQLEXPRESS;Inte grated Security=true"></sessionState>

<compilation debug="true" targetFramework="4.0" />

</system.web>

# Output





**ASP.NET Session Events**

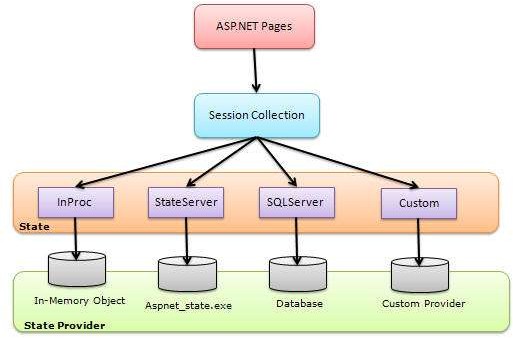
There are 2 types of events available in ASP.NET. We can handle both sessions in a global.asax file.

1. Session\_Start(): When the new session is initialized then the session\_start event is raised.
2. Session\_end(): When the session is Expires then the Session\_End event raised.

ASP.NET Session Mode

In ASP.NET there are 4 types of Session Mode.

**Off:** We can disable the session mode for the entire application using the off mode.



# ASP.NET Session Management [Example]

The HTTP protocol on which all web applications work is a stateless protocol. By stateless, it just means that information is not retained from one request to another.

For instance, if you had a login page which has 2 textboxes, one for the name and the other for the password. When you click the Login button on that page, the application needs to ensure that the username and password get passed onto the next page.

In ASP.Net, this is done in a variety of ways. The first way is via a concept called ViewState. This is wherein ASP.Net automatically stores the contents of all the controls. It also ensures this is passed onto the next page. This is done via a property called the ViewState.

It is not ideal for a developer to change anything in the view state. This is because it should be handled by ASP.Net only.

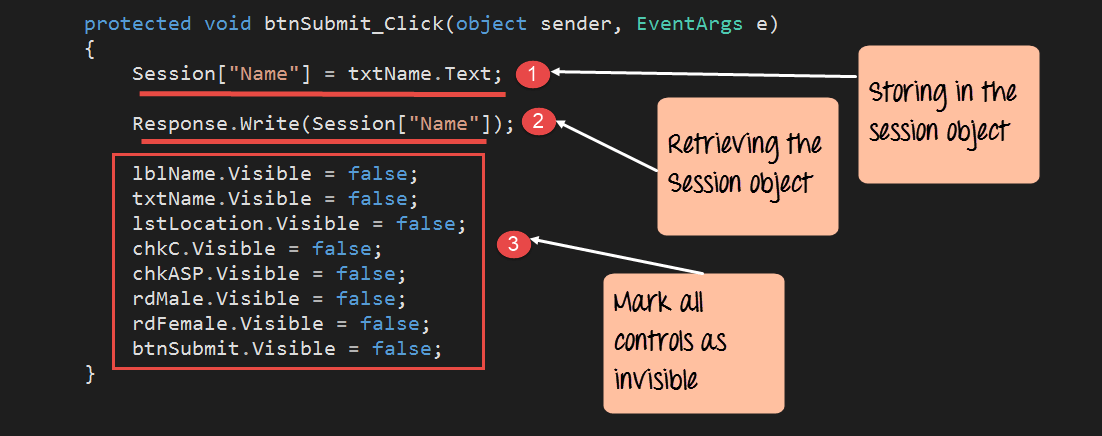
The other way is to use an object called a "Session Object." The Session object is available throughout the lifecycle of the application. You can store any number of key-value pairs in the Session object. So on any page, you can store a value in the Session object via the below line of code.

Session["Key"]=value

This stores the value in a Session object and the 'key' part is used to give the value a name. This allows the value to be retrieved at a later point in time. To retrieve a value, you can simply issue the below statement.

Session["Key"]

In our example, we are going to use the Session object to store the name entered in the name textbox field in the page. We are then going to retrieve that value and display it on the page accordingly. Let's add the below code to the Demo.aspx.cs file.



protected void btnSubmit\_Click(object sender,EventArgs e)

{

Session["Name"] = txtName.Text; Response.Write(Session["Name"]);

lblName.Visible = false; txtName.Visible = false; 1stLocation.Visible = false; chkC.Visible = false; chkASP.Visible = false; rdMale.Visible = false; rdFemale.Visible = false; btnSubmit.Visible = false;

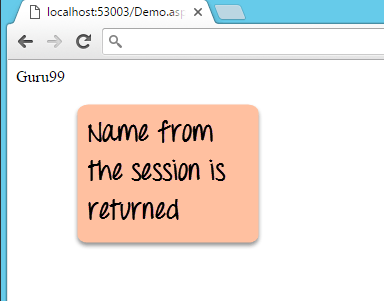
}

# Code Explanation:-

1. The first line of code takes the value of the Name textbox control and stores it in the Session object. By specifying the code of Session["Name"] , we are giving the property a name called "Name." By specifying a name for the property, it becomes easier to retrieve it at a later point in time.
2. The next line of code retrieves the stored value from the Session object. It then writes this value via the 'Response.Write' method back to the client.
3. Finally, we make all the controls on the form as invisible. If we don't do this, all the controls plus our response values will be displayed together.

Once you make the above changes, you will see the following output

# Output:



From the output, you can see that the Session value of name was retrieved and displayed in the browser.

# Session states

Let us start from the very beginning. Let"s first try to understand why we need to maintain the state of our application or why we need State Management. As we all know, our web is **"Stateless"**, in other words a new instance of a web page class, is recreated each time the page is posted to the server. HTTP is a stateless protocol and it can't hold the client information on the page. For example, if the user inserts some information on one page and then moves to the next page then that inserted data will be lost from the first page and moreover the user will not be able to retrieve that information.

So basically here we need someone to hold the state of our application. Here is the privitage role of our **"session state"**. Basically a session is a variable used between the client and the server that is stored on the server side. Now it can be stored either on an Internet Information Service (IIS) server that is by default our **"inproc"** mode or it can be stored in a state or SQL Server that is our **"outproc"** mode. We will discuss both, the inproc and outproc modes in detail later in the article.

So a session helps to maintain the user state and data all over the application by storing the information on the server memory. Also a session can store any kind of information or object on the server side and is accessible in the entire website.

Now let's discuss the entire scenario that happens when the user state and data is maintained using session state. First of all when the user requests a new application or page, first the "Application" start event fires in the get state and that application object is sharable in the entire website. After the application life cycle, the session start event fires for the specific user in the get state but when some other user again requests that page, no application start event will fire, only the session start event in the post state will fire for that specific user. Every object is stored in the application on the basis of the Key value. We can see both the application and session start up events by adding a **"Global.asax"** file in our project.

The process of maintaining the session state proceeds in the following manner. First the client hits the website and the information is stored in the session. Then a Session table will be made by default on the IIS server and in the session IDs of all the users visiting the website will be stored by the server. Now the next time the client requests some information with the unique session ID from the server, the server looks in the session providers and retrieves the serialized data from the state server and type casts the object.



FIG: PROCESS FOR MAINTAINING THE SESSION STATE IN THE APPLICATION