

## What is ASP.NET

ASP.NET is a Web application framework developed by Microsoft to build dynamic data driven Web applications and Web services.

1. ASP.NET is a subset of .NET framework. In simple terms a framework is a collection of classes.
2. ASP.NET is the successor to classic ASP (Active Server Pages).

## What other technologies can be used to build web applications

1. PHP
2. Java
3. CGI
4. Ruby on Rails
5. Perl

## What is a Web Application?

A web application is an application that is accessed by users using a web browser. Examples of web browsers include:

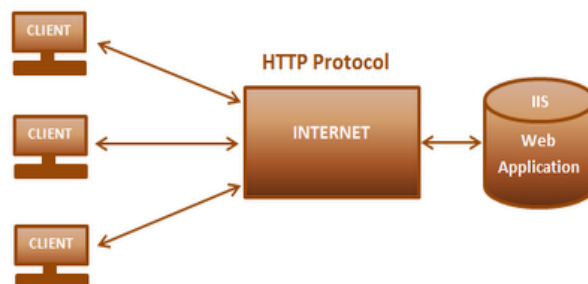
1. Microsoft Internet Explorer
2. Google Chrome
3. Mozilla FireFox
4. Apple Safari
5. Netscape Navigator

## What are the advantages of Web applications?

1. Web Applications just need to be installed only on the web server, where as desktop applications need to be installed on every computer, where you want to access them.
2. Maintenance, support and patches are easier to provide.
3. Only a browser is required on the client machine to access a web application.
4. Accessible from anywhere, provided there is internet.
5. Cross Platform

## How Web applications work?

1. Web applications work on client/server architecture
2. On the client all you need is a browser, that can understand HTML
3. On the server side, the Web application runs under Microsoft Internet Information Services (IIS)



When the client enters the URL of the web application in the browser, and submits the request. The web server which hosts the web application, receives the request. The request is then processed by the application. The application generates, the HTML and hands it over to the IIS (web server). Finally, IIS sends the generated HTML to the client, who made the initial request. The client browser will the interpret the HTML and displays the user interface. All this communication, happens over the internet using HTTP protocol. HTTP stands for Hyper Text Transfer Protocol. A protocol is a set of rules that govern how two or more items communicate.

## Creating ASP.NET website

### Start Page in Visual Studio:

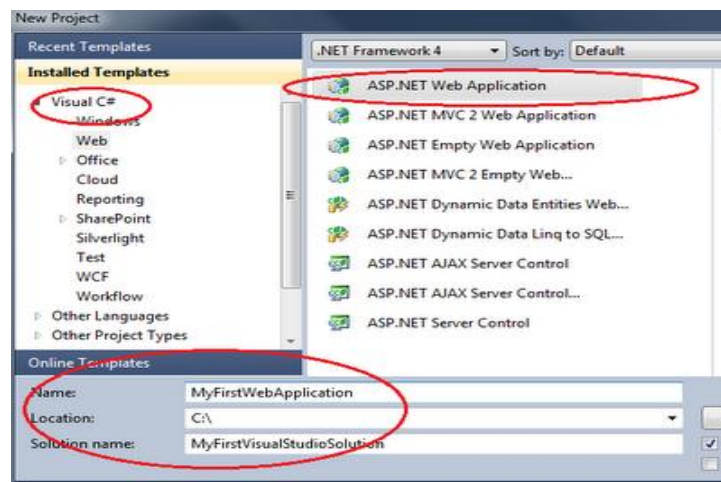
When you first run visual studio, you will see the start page. The start page contains latest news related to .NET development, learning and community resources. If you are using visual studio 2010, at the bottom of the start page, you will notice the following 2 options.

1. **Close page after project load** - Select this option if you want to close the start page, as soon as you open and load a project.
2. **Show page on startup** - Uncheck this option, if you don't want the start page to be shown, when you start visual studio.

If you have closed the start page, and later, if you want to see it again, select START PAGE from the VIEW menu.

### Creating your first ASP.NET web application:

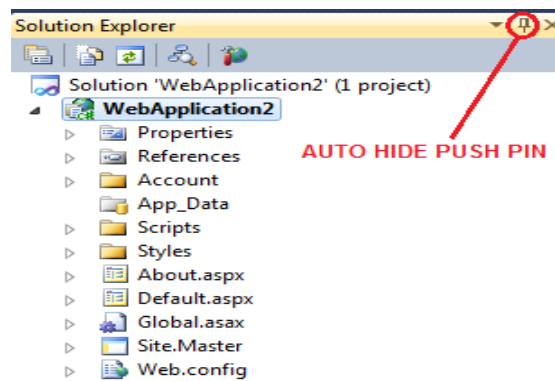
1. Select **File => New Project**
2. Select the **Programming language** you want to use from **Installed Templates** section, in the **New Project dialog box**. Out of the box, you can either use **C#** or **Visual Basic** to develop ASP.NET web applications.
3. Now, Select **ASP.NET Web Application**, from the middle section of the **New Project dialog box**.
4. Give your **project** and **solution** a meaningful name.
5. Select the **location**, where you want the solution to be created.
6. Finally click **OK**.



### Different windows in visual studio:

At this point, you should have your first web application created. Now, let's shift our focus, to learn more about the windows that we see in visual studio.

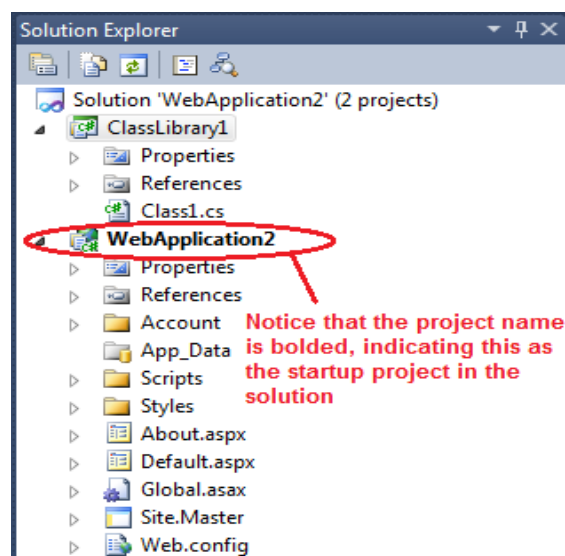
**Solution Explorer:** To view the solution explorer window, from the VIEW menu, select SOLUTION EXPLORER. Or you can also use keyboard short cut, CTRL + W, S. On the solution explorer, use the AUTO-HIDE push pin button, to either show or hide solution explorer.



Visual Studio organizes applications into projects and solutions. A solution is a collection of projects. In our example, we have **WebApplication2** solution. This solution has only one project - **WebApplication2**. If you want to add another project to the solution, simply right click the solution, and Select **Add => New Project**. For example, to add a class library project, select **CLASS LIBRARY** from the **New Project** dialog box and click **OK**.

At this point, your solution should contain 2 projects :

1. A web Application Project - WebApplication2
2. A class library project - ClassLibrary1



Notice that, in the solution explorer, WebApplication2 project is **bolded**, indicating that this is the start up project. You can only have one start up project in a solution. If you want to change your start up project, **RIGHT CLICK** the project, and select "SET AS STARTUP PROJECT". The start-up project is the project that runs when you click Start in Visual Studio .NET. When you're developing multiple projects as part of a single solution, the start-up project usually calls the other projects in the solution.

**The solution file will have a .sln extension** and the project file will have **.csproj** (if c# is the programming language) or **.vbproj** (if visual basic is the programming language)

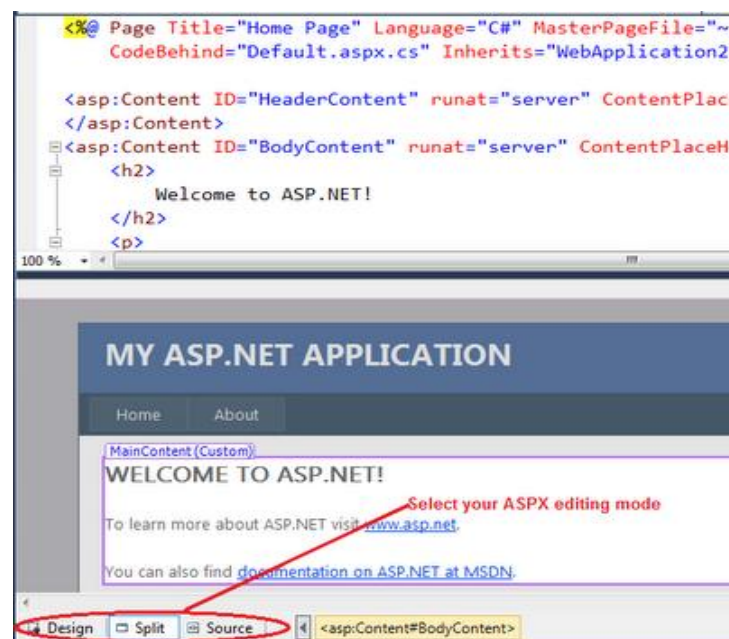
**Tool Box:** To view the TOOL BOX, Select TOOL BOX from the VIEW menu, or use the keyboard short cut, CTRL + W, X. Just like, solution explorer, tool box can be auto hidden using the AUTO-

HIDE PUSH PIN. Toolbox displays the controls and components that can be used on a web form.

Web Forms: WebForms has the extension of .aspx. A web form also has a code behind and designer files. Code behind files has the extension of .aspx.cs (if c# is the programming language) or .aspx.vb (if vb is the programming language). Designer files contains the extension of .aspx.designer.cs (if c# is the programming language) or .aspx.designer.vb (if visual basic is the programming language). Code behind files contain the code that user writes, where as the designer file contains the auto generated code. You **shouldn't** change the code in the designer file, because that code might later be modified by Visual Studio and your changes could be overwritten. A Web form is associated with its code file using the @Page directive found in the Web form's HTML.

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication2._Default" %>
```

A webform's HTML can be edited either in Source or Design mode. You can also choose SPLIT mode, which shows both the DESIGN and the SOURCE at the same time.



Properties Window: Used to change property of a webform or a control on a webform. To view the Properties window, select PROPERTIES WINDOW from the VIEW menu, or use keyboard short cut CTRL + W, P.

## What is viewstate in asp.net

Web Applications work on HTTP protocol. HTTP protocol is a **stateless protocol**, meaning it **does not retain state between user requests**. Let's understand the stateless nature of the HTTP protocol with an example.

**Create a new asp.net web application project.** Drag and drop a TextBox and a Button control onto the webform. Change the Text property of the Button control to Click Me.

**At this point, double click the button control**, which should generate the event handler in the code behind file. Modify the code behind file, so the code in WebForm1 class looks as shown below.

1. In the scope of WebForm1 class, we are creating an integer variable ClicksCount which is initialized to ZERO.
2. On the **Page\_Load()** event handler, we are setting the Text property of TextBox1 to ZERO. We do this initialization, only, when the request is an initial GET request.
3. In the **Button1\_Click()** event, we are incrementing the value of the ClicksCount by 1, and then assigning the value to the Text property of TextBox1.

```
public partial class WebForm1 : System.Web.UI.Page
{
    int ClicksCount = 0;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            TextBox1.Text = "0";
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        ClicksCount = ClicksCount + 1;
        TextBox1.Text = ClicksCount.ToString();
    }
}
```

With this code in place, run the application, and click the Button. We expect the count to be increased every time we click the button. When you click it the first time, it gets incremented to 1. After that, no matter how many times you click it, the value stays at 1. This is because of the stateless nature of the web applications that work on HTTP protocol.

**So what actually happens when you make a GET request for this WebForm1?**

When we compile this project an assembly is generated. Since the name of the project is ViewStateDemo, the name of the assembly will be ViewStateDemo.dll. So when a request is made for WebForm1, The application's assembly(ViewStateDemo.dll) creates an instance (object), of WebForm1, initializes ClicksCount to ZERO, and set's the TextBox1.Text to ZERO. As this is the initial GET request, the Button1\_Click() event will not be executed. At this point the web server, generates the HTML to respond to the request, and posts that response back to the browser. **It then immediately destroys the instance of the WebForm1.**

The browser receives the HTML, and we should now see textbox set to ZERO.

### What happens when we click the Button on WebForm1?

When we click the Button, the WebForm1 gets posted to the server. This is aPostBack request, NOT A GET REQUEST. So, when the webform is posted back, a new instance of this webform is created again, initializing the ClicksCount variable to ZERO. This time, the code that is wrapped between IF(!ISPOSTBACK) block is not executed. Button1\_Click() event gets executed as this is aPostBack event. ClicksCount is incremented from 0 to 1. The value is then assigned to the Text Property of TextBox1. Generates the HTML, sends it to client and destroys the webform.

At this Point, we should see the value increased to 1.

### What happens when we click the Button on WebForm1 again?

When you click the button for the second time, the webform gets posted back again. A new instance of WebForm1 is created. ClicksCount initialized to ZERO. In the Button1\_Click() event, the value gets incremented to 1 and assigned to TextBox1. HTML gets generated and sends it to client and destroys the webform.

So, no matter how many times you click the Button, the value of the TextBox, will not move **beyond 1**.

Now, let's see, how to preserve the state between requests using ViewState variables. Re-write the code in WebForm1, as shown below.

```
public partial class WebForm1 : System.Web.UI.Page
{
    int ClicksCount = 1;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            TextBox1.Text = "0";
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (ViewState["Clicks"] != null)
        {
            ClicksCount = (int)ViewState["Clicks"] + 1;
        }
        TextBox1.Text = ClicksCount.ToString(); ;
        ViewState["Clicks"] = ClicksCount;
    }
}
```

Click the Button now, and the value gets incremented every time we click. So how is this possible now. It's possible because, we are using the ViewState variable Clicks to preserve the data between requests. The ViewState data, travels with every request and response between the client and the web server.

Now, let's try to achieve the same behaviour, without explicitly storing data in a ViewState variable. Modify the WebForm1 code as shown below.

```
public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            TextBox1.Text = "0";
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        int ClicksCount = Convert.ToInt32(TextBox1.Text) + 1;
        TextBox1.Text = ClicksCount.ToString();
    }
}
```

Upon clicking the Button, the value gets incremented correctly as expected. This is possible because, TextBox1 is an asp.net server control, that uses viewstate internally, to preserve data across postbacks.

Because Web forms have very short lifetimes, ASP.NET takes special steps to preserve the data entered in the controls on a Web form. Data entered in controls is sent with each request and restored to controls in Page\_Init. The data in these controls is then available in the Page\_Load(), Button\_Click(), and many more events, that occur after Page\_Init() event.

On the other hand the HTML controls, **do not retain state across post backs**. Only ASP.NET server controls retains state. To prove this:

1. Add a new webform to the web application project
2. Drag and Drop Input(Text) control from the HTML tab, in the ToolBox
3. Drag and Drop TextBox control from the Standard tab, in the ToolBox
4. Finally drag and drop a button
5. Set the newly added webform as the start page by right clicking on it, in the solution explorer
6. Run the project, by pressing CTRL + F5
7. Type "TEST" into both the controls (ASP.NET TextBox and the HTML TextBox), and press the button
8. You should see that, the value in the ASP.NET TextBox is preserved across postback, but not the value in the standard HTML textbox

An HTML control can be converted in ASP.NET server control, by adding **runat="server"** attribute in the HTML source as shown below.

```
<input id="Text1" runat="server" type="text" />
```

Now, if you type TEST and click the button, both controls now retain state across postback.



ViewState data is serialized into base64-encoded strings, and is stored in Hidden input field `__ViewState`. To view this hidden input field, right click on the browser and select "View Page Source" for google chrome. In internet explorer, right click and select "View Source"

## Events in the life cycle of a web application

Before understanding Application level events, let's talk about Session State and Application State variables. By default, ViewState of one web form is not available in another web form.

For example, if you define `ViewState["MyData"] = "View State Example"` in `WebForm1`. `ViewState["MyData"]` is only available in `WebForm1`. `ViewState["MyData"]` will be null on any other web form in the application.

If you want to make your data available on multiple web forms, there are several techniques in ASP.NET, as listed below:

1. Query Strings
2. Cookies
3. Session State
4. Application State

Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data. Only the current session has access to its Session state.

Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data. All sessions can read and write Application State variables.

In an ASP.NET web application, `Global.asax` file contains the application level events.

```
void Application_Start(object sender, EventArgs e)
{
    // Code that runs on application startup
}
```

```
void Application_End(object sender, EventArgs e)
{
    // Code that runs on application shutdown
}
```

```
void Application_Error(object sender, EventArgs e)
{
    // Code that runs when an unhandled error occurs
}
```

```
void Session_Start(object sender, EventArgs e)
{
    // Code that runs when a new session is started
}
```

```
void Session_End(object sender, EventArgs e)
{
    // Code that runs when a session ends.
    // Note: The Session_End event is raised only when the sessionstate mode
    // is set to InProc in the Web.config file. If session mode is set to StateServer
```

```
// or SQLServer, the event is not raised.  
}
```

In general, Application events are used to initialize data that needs to be available to all the current sessions of the application. Where as Session events are used to initialize data that needs to be available only for a given individual session, but not between multiple sessions.

Now, let's write a simple application, using session and application level events. Create a new asp.net web application, and copy paste the following code in Global.asax file:

1. Application\_Start() event gets fired, when a first request is made, and if the application is not already running.
2. Session\_Start() event is fired every time a new browser instance, with a different session-id, visits the application.
3. Session\_End() event is fired when the user session times out. The default is 20 minutes. This can be configured in the web.config file.

```
void Application_Start(object sender, EventArgs e)  
{  
    // Create Application state variables  
    Application["TotalApplications"] = 0;  
    Application["TotalUserSessions"] = 0;  
    // Increment TotalApplications by 1  
    Application["TotalApplications"] = (int)Application["TotalApplications"] + 1;  
}  
void Session_Start(object sender, EventArgs e)  
{  
    // Increment TotalUserSessions by 1  
    Application["TotalUserSessions"] = (int)Application["TotalUserSessions"] + 1;  
}  
void Session_End(object sender, EventArgs e)  
{  
    // Decrement TotalUserSessions by 1  
    Application["TotalUserSessions"] = (int)Application["TotalUserSessions"] - 1;  
}
```

Copy and paste the following code in WebForm1.aspx.

```
protected void Page_Load(object sender, EventArgs e)  
{  
    Response.Write("Number of Applications: " + Application["TotalApplications"]);  
    Response.Write("<br/>");  
    Response.Write("Number of Users Online: " + Application["TotalUserSessions"]);  
}
```

Now, when you run the application, you get the following output:

Number of Applications: 1  
Number of Users Online: 1

Copy the URL and open a new instance of the browser. Paste the URL and press enter. In the new instance of the browser, we still see the same output.

We expected the Number of Users Online to be 2. The new instance of the browser, is treated as part of the same session, because, by default the browser uses cookies to store session id. The session id is read from the same cookie when you opened the new browser window. Hence, Number of Users Online is not incremented.

### **How to get a new session-id and force the Session\_Start() event to execute?**

1. Close the browser: Close the existing browser window, which automatically deletes the session cookie. Now, open a new browser instance. Since, the existing session cookie associated with the previous browser instance is deleted. The new instance of the browser, will get a new session-id and a session cookie. Now, if you navigate to WebForm1.aspx, Session\_Start() event gets fired and Number of Users Online is incremented to 2.

2. Open a new instance of a different browser: For example, if you first visited the application with Google Chrome, now try accessing the same page with internet explorer, Session\_Start() event gets fired and Number of Users Online is incremented to 2.

3. Use Cookie-less Sessions: To use cookie-less sessions set the cookieless attribute to true in web.config as shown below.

```
<sessionState mode="InProc" cookieless="false"></sessionState>
```

### **What is a Session, in a web application?**

A session is a unique instance of the browser. A single user can have multiple sessions, by visiting your application, with multiple instances of the browser running with a different session-id on his machine.

## Difference between ViewState, Session State and Application State in asp.net

### ViewState:

Add a new WebForm, to the project and name it ViewState1.aspx. Drag and drop a button and a text box control onto the webform. Double click the button control on the webform. This automatically generates the event handler, for the button control.

Now add another webform, to the project, with name ViewState2.aspx. Just like you have done for ViewState1.aspx, drag and drop a TextBox and a Button control onto this webform as well.

Now, copy and paste the following code in ViewState1.aspx.cs and ViewState2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (ViewState["Clicks"] == null)
        {
            ViewState["Clicks"] = 0;
        }
        TextBox1.Text = ViewState["Clicks"].ToString();
    }
}
```

```
protected void Button1_Click(object sender, EventArgs e)
{
    int ClicksCount = (int)ViewState["Clicks"] + 1;
    TextBox1.Text = ClicksCount.ToString();
    ViewState["Clicks"] = ClicksCount;
}
```

Now, run the application, and navigate to ViewState1.aspx. Click the button control. Everytime, you click the button, the clicks count get incremented and is displayed in the TextBox, as expected.

Now, navigate to ViewState2.aspx. Click the button, on this page. Notice, that the value starts from ZERO, indicating that, each page has it's own ViewState["Clicks"].

So, the conclusion is that, ViewState of a webform is available only with in that webform, by default.

So, where does this viewstate, gets stored - On the client or on the server? ViewState is stored on the page using a hidden field called \_ViewState. So, ViewState travels along with the page, between the client and the server, with each request and response.

ASP.NET uses viewstate, to retain the values a user types into controls on the webform, across postbacks.

### SessionState:

Add a new webform with name SessionState1.aspx. Drag and drop a button and a text box control onto SessionState1.aspx. Do the same thing by adding a page with name SessionState2.aspx.

Copy and paste the following code in SessionState1.aspx.cs and SessionState2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Session["Clicks"] == null)
        {
            Session["Clicks"] = 0;
        }
        TextBox1.Text = Session["Clicks"].ToString();
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    int ClicksCount = (int)Session["Clicks"] + 1;
    TextBox1.Text = ClicksCount.ToString();
    Session["Clicks"] = ClicksCount;
}
```

Add the following sessionstate element to your web.config file, under system.web. This setting, specifies the web application to use cookieless sessions.

```
<sessionState mode="InProc" cookieless="true"></sessionState>
```

Run the application and navigate to SessionState1.aspx. Click the button 3 times, and notice that, the value 3 is displayed in the TextBox. Now, navigate to SessionState2.aspx. The value 3 is displayed in the TextBox on SessionState2.aspx. Now, click twice, the value is incremented to 5. Now, navigate back to SessionState1.aspx, and you should see the value 5. This proves that a session state variable is accessible across all pages in a web application.

Now, open a new browser window and navigate to SessionState1.aspx (Make sure you have a different session-id). Notice that, the value in the textbox is ZERO. So, this proves that, Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data. Only the current session has access to its Session state.

### Application State:

Add a new webform with name ApplicationState1.aspx. Drag and drop a button and a text box control onto ApplicationState1.aspx. Do the same thing by adding a page with name ApplicationState2.aspx.

Copy and paste the following code in ApplicationState1.aspx.cs and ApplicationState2.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Application["Clicks"] == null)
        {
```

```

        Application["Clicks"] = 0;
    }
    TextBox1.Text = Application["Clicks"].ToString();
}
}

```

```

protected void Button1_Click(object sender, EventArgs e)
{
    int ClicksCount = (int)Application["Clicks"] + 1;
    TextBox1.Text = ClicksCount.ToString();
    Application["Clicks"] = ClicksCount;
}

```

Run the application and navigate to ApplicationState1.aspx. Click the button 3 times, and notice that, the value 3 is displayed in the TextBox. Now, navigate to ApplicationState2.aspx. The value 3 is displayed in the TextBox on ApplicationState2.aspx. Now, click twice, the value is incremented to 5. Now, navigate back to ApplicationState1.aspx, and you should see the value 5. This proves that an application state variable is accessible across all pages in a web application.

Now, open a new browser window and navigate to ApplicationState1.aspx. Notice that, the value in the textbox is still 5. So, this proves that, Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data. All sessions can read and write Application State variables.

So, in short, the differences are as follows:

#### **ViewState:**

1. ViewState of a webform is available only within that webform
2. ViewState is stored on the page in a hidden field called `_ViewState`. Because of this, the ViewState, will be lost, if you navigate away from the page, or if the browser is closed.
3. ViewState is used by all asp.net controls to retain their state across postback

#### **Session State:**

1. Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data.
2. Session state variables are stored on the web server.
3. SessionState variables are cleared, when the user session times out. The default is 20 minutes. This is configurable in web.config

#### **Application State:**

1. Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data.
2. Application State variables are stored on the web server.
3. Application State variables are cleared, when the process hosting the application is restarted.

## ASP.NET Page Life Cycle Events

Events can occur at 3 levels in an asp.net web application:

1. At the application level. (Example - Session\_Start event in global.asax)
2. At the Page or web form level (Example - Page\_Load)
3. At the control level(Example - Selected Index changed event of a dropdownlist)

web applications work on a stateless protocol. Every time a request is made for a webform, the following sequence of events occur:

1. Web Application creates an instance of the requested webform.
2. Processes the events of the webform.
3. Generates the HTML, and sends the HTML back to the requested client.
4. The webform gets destroyed and removed from the memory.

The following are some of the commonly used events in the life cycle of an asp.net webform. These events are shown in order of occurrence, except for, Error event, which occurs only if there is an unhandled exception.

**PreInit** - As the name suggests, this event happens just before page initialization event starts. IsPostBack, IsCallback and IsCrossPagePostBack properties are set at this stage. This event allows us to set the master page and theme of a web application dynamically. PreInit is extensively used when working with dynamic controls.

**Init** - Page Init, event occurs after the Init event, of all the individual controls on the webform. Use this event to read or initialize control properties. The server controls are loaded and initialized from the Web form's view state.

**InitComplete** - As the name says, this event gets raised immediately after page initialization.

**PreLoad** - Happens just before the Page Load event.

**Load** - Page Load event, occurs before the load event of all the individual controls on that webform.

**Control Events** - After the Page load event, the control events like button's click, dropdownlist's selected index changed events are raised.

**Load Complete** - This event is raised after the control events are handled.

**PreRender** - This event is raised just before the rendering stage of the page.

**PreRenderComplete** - Raised immediately after the PreRender event.

**Unload** - Raised for each control and then for the page. At this stage the page is, unloaded from memory.

**Error** - This event occurs only if there is an unhandled exception.

To see the events execution order, create a new asp.net web project, and copy the following code.



```

protected void Page_PreInit(object sender, EventArgs e)
{ Response.Write("Page_PreInit" + "<br/>"); }

protected void Page_Init(object sender, EventArgs e)
{ Response.Write("Page_Init" + "<br/>"); }

protected void Page_InitComplete(object sender, EventArgs e)
{ Response.Write("Page_InitComplete" + "<br/>"); }

protected void Page_PreLoad(object sender, EventArgs e)
{ Response.Write("Page_PreLoad" + "<br/>"); }

protected void Page_LoadComplete(object sender, EventArgs e)
{ Response.Write("Page_LoadComplete" + "<br/>"); }

protected void Page_PreRender(object sender, EventArgs e)
{ Response.Write("Page_PreRender" + "<br/>"); }

protected void Page_PreRenderComplete(object sender, EventArgs e)
{ Response.Write("Page_PreRenderComplete" + "<br/>"); }

protected void Page_Unload(object sender, EventArgs e)
{
    //Response.Write("Page_Unload" + "<br/>");
}

```

Run the project and you should see the following output.

```

Page_PreInit
Page_Init
Page_InitComplete
Page_PreLoad
Page_LoadComplete
Page_PreRender
Page_PreRenderComplete

```

Note: If you uncomment, `Response.Write()` method call in `Page_Unload()` event, you get **System.Web.HttpException** stating - **Response is not available in this context**. This makes sense because, the Unload event is raised after the page has been fully rendered, and the HTML is already sent to the client. At this stage, the webform instance is ready to be discarded. So, at this point, page properties such as `Response` and `Request` are unloaded and clean up is performed.

## ASP.NET Server control events

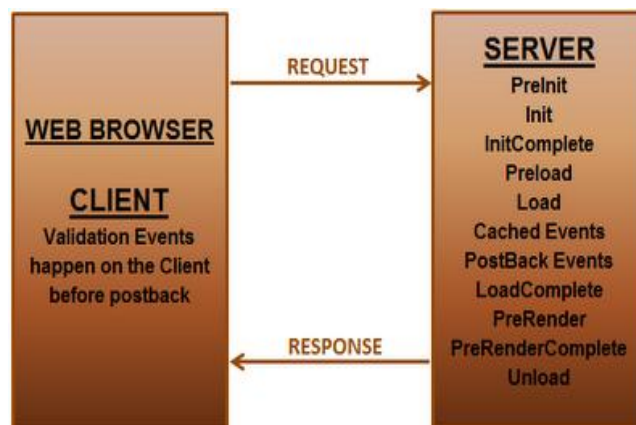
ASP.NET server controls, such as TextBox, Button, and DropDownList has their own events. For example, Button has a click event. TextBox has TextChanged event, and DropDownList has SelectedIndexChanged event. We have a set of asp.net validation controls, that has validation events. The events that all these controls expose, can be broadly divided into 3 categories:

**PostBack events** - These events submit the Web page, immediately to the server for processing. Click event of a button control is an example for PostBack event.

**Cached events** - These events are saved in the page's view state to be processed when a postback event occurs. TextChanged event of TextBox control, and SelectedIndexChanged event of a DropDownList control are examples of cached events. Cached events can be converted into postback events, by setting the AutoPostBack property of the control to true.

**Validation events** - These events occur on the client, before the page is posted back to the server. All validation controls use these type of events.

we have understood that control events are processed after the PageLoad event. The picture below depicts the same. Among the control events, Cached events happen before PostBack events.



To understand the order in which Page and Server control events execute, add a Web form with a TextBox, RequiredFieldValidator, and a Button control. You can find RequiredFieldValidator under Validation tab, in the ToolBox. Double click the TextBox control on the WebForm, and the event handler TextBox1\_TextChanged() will be automatically generated. Along the same lines, double click the Button control, Button1\_Click() event handler will be generated. Right Click the RequiredFieldValidator control on the webform and select Properties. From the Properties window, Set ControlToValidate property to TextBox1. At this stage copy and paste the following code in WebForm1.aspx.cs.

```
protected void Page_PreInit(object sender, EventArgs e)
{
    Response.Write("Page_PreInit" + "<br/>");
}
protected void Page_Init(object sender, EventArgs e)
{
    Response.Write("Page_Init" + "<br/>");
}
protected void Page_InitComplete(object sender, EventArgs e)
```

```

{
    Response.Write("Page_InitComplete" + "<br/>");
}
protected void Page_PreLoad(object sender, EventArgs e)
{
    Response.Write("Page_PreLoad" + "<br/>");
}
protected void Page_LoadComplete(object sender, EventArgs e)
{
    Response.Write("Page_LoadComplete" + "<br/>");
}
protected void Page_PreRender(object sender, EventArgs e)
{
    Response.Write("Page_PreRender" + "<br/>");
}
protected void Page_PreRenderComplete(object sender, EventArgs e)
{
    Response.Write("Page_PreRenderComplete" + "<br/>");
}
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    Response.Write("Text Changed Event" + "<br/>");
}
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("Button Click" + "<br/>");
}

```

**Now, run the project, and when the webform renders,** the page level events occur in the following order. Notice that, TextChanged and ButtonClick events are not fired.

```

Page_PreInit
Page_Init
Page_InitComplete
Page_PreLoad
Page_LoadComplete
Page_PreRender
Page_PreRenderComplete

```

Don't type anything in the TextBox, and click the button control. The RequiredFieldValidator message is displayed. No other events get processed and the page is not posted back to the server.

**Now, enter some text, into the TextBox and Click the button.** Notice that, Text Changed Event and Button Click, execute after Page Load and Before Page Load Complete events. Among the control events, TextChanged event is fired before the click event. The execution order is shown below.

```

Page_PreInit
Page_Init
Page_InitComplete
Page_PreLoad
Text Changed Event

```

Button Click

Page\_LoadComplete

Page\_PreRender

Page\_PreRenderComplete

## IsPostBack in asp.net

IsPostBack is a Page level property, that can be used to determine whether the page is being loaded in response to a client postback, or if it is being loaded and accessed for the first time.

In real time there are many situations where IsPostBack property is used. For example, consider the webform used to register employee details. A sample form that we will use for this example is shown below. The form has First Name, Last Name and City fields.

**Employee Details Form**

First Name:

Last Name:

City:

If you want to follow along with me, copy and paste the following HTML in a web form.

```
<table style="font-family: Arial">
<tr>
<td colspan = "2"><b>Employee Details Form</b></td>
</tr>
<tr>
<td>First Name: </td>
<td> <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox> </td>
</tr>
<tr>
<td>Last Name: </td>
<td> <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox> </td>
</tr>
<tr>
<td>City:</td>
<td>
<asp:DropDownList ID="ddlCity" runat="server">
</asp:DropDownList>
</td>
</tr>
<tr>
<td></td>
<td>
<asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Register Employee" />
</td>
</tr>
</table>
```

Copy and Paste the following code in the code behind file of the web form.

```
protected void Page_Load(object sender, EventArgs e)
{
    LoadCityDropDownList();
}
public void LoadCityDropDownList()
{
```

```

ListItem li1 = new ListItem("London");
ddlCity.Items.Add(li1);

ListItem li2 = new ListItem("Sydney");
ddlCity.Items.Add(li2);

ListItem li3 = new ListItem("Mumbai");
ddlCity.Items.Add(li3);
}
protected void Button1_Click(object sender, EventArgs e)
{
}

```

Now run the application. Look at the City DropDownList. The cities, (London, Sydney and Mumbai) are correctly shown as expected. Just **click the button once**. Notice, that the city names in the **DropDownList are duplicated**. So, every time you click the button, the city names are again added to the DropDownList.

### Let's now understand the cause for this duplication.

We know that all ASP.NET server controls retain their state across postback. These controls make use of ViewState. So, the first time, when the webform load. the cities get correctly added to the DropDownList and sent back to the client.

Now, when the client clicks the button control, and the webform is posted back to the server for processing. During the Page initialization, ViewState restoration happens. During this stage, the city names are retrieved from the viewstate and added to the DropDownList. PageLoad event happens later in the life cycle of the webform. During page load we are again adding another set of cities. Hence, the duplication.

### How to solve the DropDownList items duplication

There are several ways to solve this. **One of the best ways to do this, is to use IsPostBack** property. So, in the Page\_Load, call LoadCityDropDownList() method, if the request, is not a postback request. That is, only if the webform is being loaded and accessed for the first time.

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        LoadCityDropDownList();
    }
}

```

Another way to solve, this problem is to simply disable the ViewState of the DropDownList control. To disable the viewstate, right click the DropDownList control, and set EnableViewState property to false. Now run the project, and the cities duplication issue is gone.

But the problem, with this approach is that, the DropDownList list, does not remember your selection across postback. That is, Select "Mumabi" as the city, and submit the form. When the page rerenders, observer that selection is set back to "London". Another problem with, disabling the viewstate is that, the DropDownList events may not work correctly as expected.

Another way to solve this, is to clear all the DropDownList items, before calling LoadCityDropDownList() method. **But this not efficient from a performance perspective.** The modified code is shown below.

```
protected void Page_Load(object sender, EventArgs e)
{
    ddlCity.Items.Clear();
    LoadCityDropDownList();
}
```

## IIS - Internet Information Services and ASP.NET

### What is a web server?

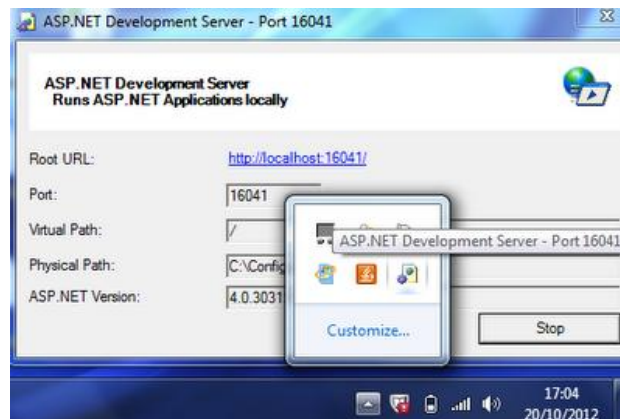
In simple terms, a web server is software that is used to deliver web pages to clients using the Hypertext Transfer Protocol (HTTP). For example, IIS is a web server that can be used to run asp.net web applications.

### Do you need IIS to develop and test asp.net web applications?

No, Visual Studio ships with a built-in web server. If you want to just build and test web applications on your machine, you don't need an IIS. Keep in mind, Built-in web server will not serve requests to another computer. By default, visual studio uses the built-in web server.

Create a new asp.net web application and run it by pressing CTRL + F5. Notice the URL of the page in the browser. A random port number is used. On my machine it was using port number 16041.  
`http://localhost:16041/WebForm1.aspx`.

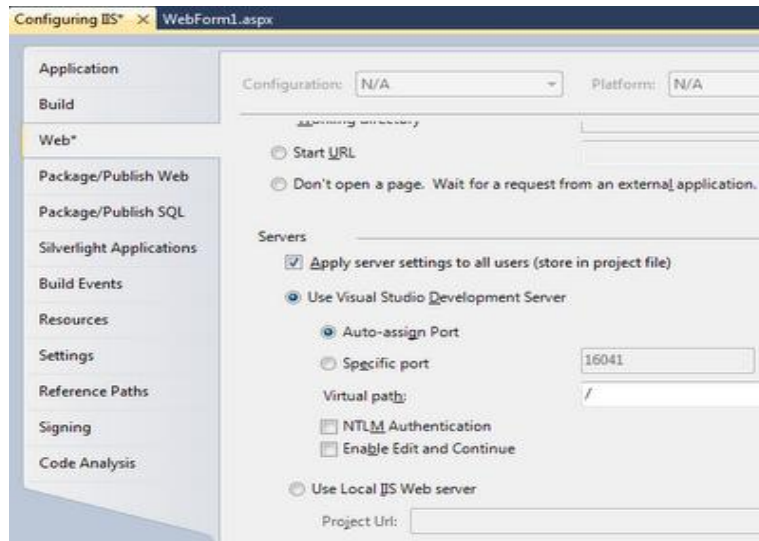
To confirm, if this is the built-in visual studio development server, check the notifications area on the task bar, and you should see ASP.NET Development Server running. Please refer to the image below.



Another way to check, if visual studio is using, built-in development server:

1. Right click on the web application project in solution explorer and select Properties.
2. On the Project properties window, click on the Web tab.
3. Scroll down to servers section - Notice that "Use visual studio development server" is selected.
4. By default, visual studio auto assigns an available port. If you want to assign a specific port, you can do so, by selecting Specific Port radio button.





### How to check if IIS is installed?

1. Click on the windows Start button
2. Type INETMGR in the Run window.
3. Click OK.
4. If you get IIS manager window, **it is installed**, **otherwise not installed**.

### How to install IIS?

1. Click on the start button and select ControlPanel
2. Click on Programs
3. Click on, Turn windows features on or Off, under Programs and features option
4. In the windows features
5. Select Internet Information Services and related services, and click OK

### To configure a virtual directory in IIS to run asp.net web applications :

1. In the IIS Manager window, double click on the iis server name in the connections section.
2. Expand sites
3. Right click on Default Web Site, and Select Add Application.
4. Give an alias name. This is the name you will use in the URL, when connecting to your web application.
5. Click the button next to the textbox under physical path. Select the physical web application folder.

### You can also create the virtual directory from visual studio, on the project properties window:

1. Select Use Local IIS Web Server
2. Project URL will be populated automatically. You can change the name of the virtual directory if you wish to do so.
3. Click Create Virtual Directory button.
4. After a few seconds the virtual directory was successfully created message will appear.
5. Click OK

## ASP.NET TextBox Control

The TextBox control is used to get the input from the user of the web application. An asp.net textbox has several properties, that we need to be aware of as a developer.

### Properties of a TextBox control:

**1. TextMode Property** - SingleLine, MultiLine and Password.

When you set the TextMode to MultiLine, use Rows property to control the number of lines to display for a MultiLine TextBox.

**2. Text** - Use this property to set or get the Text from the TextBox.

**3. MaxLength** - The maximum number of characters that a user can enter.

**4. ReadOnly** - Set this property to true if you don't want the user to change the text in the TextBox.

**5. ToolTip** - The tooltip is displayed when the mouse is over the control.

**6. Columns** - Use this property to specify the width of the TextBox in characters

**7. Height** - Set the height

**8. Width** - Set the width

**9. AutoPostBack** - By default, the TextChanged event of a TextBox control is cached in the viewstate, and is executed when the webform is submitted thru a postback by clicking the button control. If you want to change this behaviour, and post the webform immediately when the Text is changed, set AutoPostBack to true. Setting this property to true, will convert the cached event into a postback event.

### Events of TextBox:

**TextChanged** - This event is fired, when the text is changed.

### Methods of a TextBox:

**Focus** - Set input focus onto the control.

To view the properties of the TextBox, Right click on the control, and select Properties. In the properties window, you can also find the events supported by the control.

All these properties can be set at the design time, or at runtime using code.

## ASP.NET Radio Button Control

Radio Button control is used, when you want the user to select only one option from the available choices. For example, the gender of a person. A person can be Male or Female. He cannot be both. So, if the user has first selected Male, and if tries to select Female, the initial Male selection he made should automatically get de-selected. Another example, would be when you want the user to select his or her favourite colour.

In short, if you want to provide the user with mutually exclusive options, then choose a Radio Button Control.

### Important Properties of the Radio Button Control:

**Checked** - This is a boolean property, that is used to check if the button is checked or not.

**Text** - This is string property used to get or set the text associated with the radio button control

**TextAlign** - right or left. On which side of the radio button the text should appear

**AutoPostBack** - Set this property to true, if you want the webform to be posted immediately when the checked status of the radio button changes.

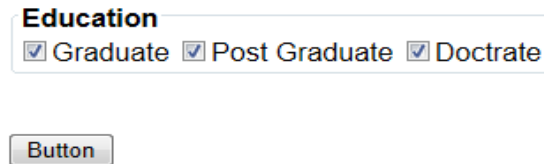
**Group Name** - By default, the individual radio button selections, are not mutually exclusive. If you have a group of radio buttons, and if you want the selections among the group to be mutually exclusive, then use the same group name for all the radio button controls.

### Events:

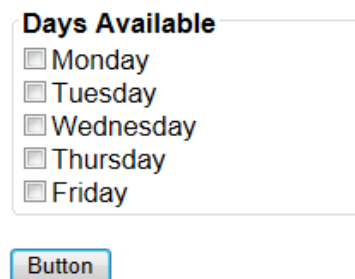
**CheckedChanged** - This event is fired when the checked status of the radio button control is changed.

## ASP.NET CheckBox Control

CheckBox Control is used, when you want the user to select more than one option from the available choices. For example, the education of a person. A person can have a graduate degree, post graduate degree and a doctrate. In this case the user selects all the 3 checkboxes. Where as a person, may just have a graduate degree, in which case he only selects, the graduate checkbox.



Another example would be when you want the user to select the days of his availability.



In short, if you want to provide the user with more than one option to select from, then choose a check box Control.

### Important Properties of the CheckBox Control:

**Checked** - This is a boolean property, that is used to check if the check box is checked or not.

**Text** - This is a string property used to get or set the text associated with the check box control

**TextAlign** - right or left. On which side of the check box the text should appear

**AutoPostBack** - Set this property to true, if you want the webform to be posted immediately when the checked status of the check box changes.

### Methods:

**Focus()** - Just like TextBox, checkbox also supports, Focus() method. If you want to set the input focus, to a specific checkbox, Call this method for that check box control.

### Events:

**CheckedChanged** - This event is fired when the checked status of the check button control is changed.

HTML of the ASPX page, we used in the example

```
<div style="font-family:Arial">
  <fieldset style="width:350px">
    <legend><b>Education</b></legend>
    <asp:CheckBox ID="GraduateCheckBox" Checked="true" Text="Graduate" runat="server"
      oncheckedchanged="GraduateCheckBox_CheckedChanged" />
```

```

        <asp:CheckBox ID="PostGraduateCheckBox" Text="Post Graduate" runat="server" />
        <asp:CheckBox ID="DoctradeCheckBox" Text="Doctrade" runat="server" />
    </fieldset>&nbsp;
    <br /><br />
    <asp:Button ID="Button1" runat="server" Text="Submit" onclick="Button1_Click" />
</div>

```

Code from the CodeBehind file

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        PostGraduateCheckBox.Focus();
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    StringBuilder sbUserChoices = new StringBuilder();
    if (GraduateCheckBox.Checked)
    {
        sbUserChoices.Append(GraduateCheckBox.Text);
    }
    if (PostGraduateCheckBox.Checked)
    {
        sbUserChoices.Append(", " + PostGraduateCheckBox.Text);
    }
    if (DoctradeCheckBox.Checked)
    {
        sbUserChoices.Append(", " + DoctradeCheckBox.Text);
    }
    Response.Write("Your Selections: " + sbUserChoices.ToString());
}

protected void GraduateCheckBox_CheckedChanged(object sender, EventArgs e)
{
    Response.Write("Graduate Checkbox Selection changed");
}

```

## **ASP.NET Hyperlink control**

The ASP.NET Hyperlink control is used to create a link to another Web page.

### **Properties:**

**Text** - The link text that will be shown to the user

**Navigate URL** - The URL of the page to which the user will be sent

**ImageURL** - The URL of the image, that will be displayed for the link. If you specify both the Text and ImageUrl, the image will be displayed instead of the text. If for some reason, the image is not unavailable, the text will be displayed.

**Target** - If target is not specified, the web page to which the hyperlink is linked, will be displayed in the same window. If you set the Target to \_blank, the web page will be opened in a new window.

### **Methods:**

**Focus()** - Call this method to Set the input focus when the page loads.

### **Events:**

No HyperLink control specific events

## ASP.NET Button, LinkButton and ImageButton Controls

The Button, LinkButton and ImageButton controls in ASP.NET are used to post a page to the server:

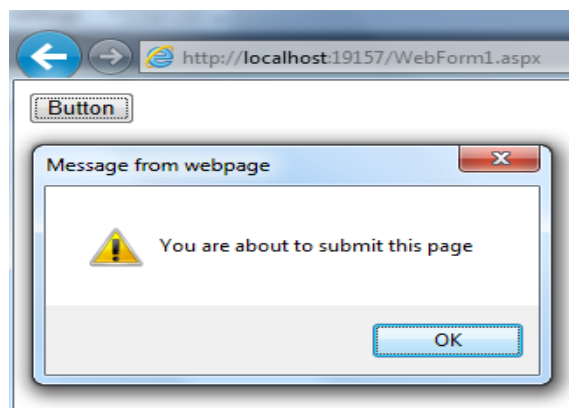
1. **Button** - The Button control is used to display a push button. Use the Text property to change the Text on the Button control.
2. **LinkButton** - LinkButton displays the button like a HyperLink. Use the Text property to change the LinkText.
3. **ImageButton** - ImageButton provides the flexibility of associating an Image with the button, using the ImageURL property.

All the 3 button controls support CommandName and CommandArgument properties. These 3 button controls also support CausesValidation and ValidationGroup properties. We will discuss about these properties, when we talk about validation controls in asp.net. We will discuss aboutPostBackURL property, when we talk about cross page post back.

All the 3 button controls, exposes **client side click event** and **server side click event**. You can associate the javascript, that you want to run in response to the click event on the client side using OnClientClick property as shown below.

```
<asp:Button ID="Button1" runat="server"
OnClientClick="alert('You are about to submit this page')"
Text="Button" />
```

When you click this button, you will get a popup as shown below. Once you click OK, the webform will be submitted to the server for processing server side click event.



In the above example we are using javascript, alert() function. The client side alert message box, can be used to communicate important information to the user. For example messages like:

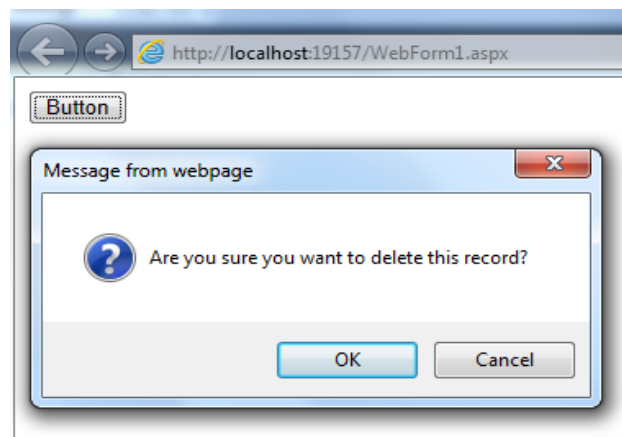
1. You are about to place an order
2. You are about to leave this website

Sometimes, we may accidentally click on a **delete** button, which deletes the record permanently. So, whenever, we do things like this, we want to be double sure, if the user really wants to delete the record. The javascript confirm(), function can be used for this purpose.

```
<asp:Button ID="Button1" runat="server"
OnClientClick="return confirm('Are you sure you want to delete this record?')"
```

```
Text="Button" />
```

When you click the button now, the user will be shown a confirmation box, as shown below.



If you click cancel, the `confirm()` function returns false and the webform will not be submitted. If you click OK, the `confirm()` function returns true, and the webform will be posted to the server.

So, far we have associated the javascript, to the client click event of a button control at design time. It is also, possible, to do the same at runtime using the Button controls attribute collection as shown below.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Button1.Attributes.Add("onclick", "return confirm('Do you want to delete the record?');");
    }
}
```



## Command Event of an asp.net button control

ASP.NET button control exposes 2 events - Click and Command events. When the Button is clicked, both the events are raised. **Click event happens before the Command event.**

To prove this drag and drop a button control onto the webform:

1. Double click the Button control. This will automatically generate the click event handler in the code behind file
2. To generate the command event handler, right click the button control and select properties. Click the events icon, in the properties window. Double click on the command event. The event handler for the command event should now be generated.

If you are following along with me. At this stage the HTML for the button control in the aspx page, should look as shown below.

```
<asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click"
  CommandName="Button1" oncommand="Button1_Command" />
```

Please copy and paste the following code in the code behind file.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("Button1 Click event handled <br/>");
}
protected void Button1_Command(object sender, CommandEventArgs e)
{
    Response.Write("Button1 Command event handled <br/>");
}
```

When you click the Button now, you should see the following output. This proves that when a button is clicked, first the Click event and then the Command event is fired:

1. Button1 Click event handled
2. Button1 Command event handled

The click event handler and the command event handlers, are attached to the respective **Click** and **Command** events in the HTML using **onclick** and **oncommand** attributes. The event handlers can also be attached programatically as shown below.

```
protected void Page_Load(object sender, EventArgs e)
{
    Button1.Click += new EventHandler(Button1_Click);
    Button1.Command += new CommandEventHandler(Button1_Command);
}
```

Note: Eventhandlers can be associated to the events of a control in 2 ways:

1. Declaratively at design time in the HTML
2. Programatically using delegates

If you have multiple button controls on a webform, and if you want to programmatically determine which Button control is clicked, we can make use of [Command](#) event, along with [CommandName](#) and [CommandArgument](#) properties. Command event, makes it possible to have a single event handler method responding to the click event of multiple buttons. The command event, CommandName and CommandArgument properties are extremely useful when working with data-bound controls like Repeater, GridView, DataList.

Let's understand this with an example. Consider the HTML below. Here we have 4 buttons. Notice that all the button controls have the same command event handler method - `oncommand="CommandButton_Click"`. Also, notice the CommandName and CommandArgument properties. We will later use these properties, in the code behind to determine which button is clicked.

```
<asp:Button ID="PrintButton" runat="server" Text="Print" oncommand="CommandButton_Click"
CommandName="Print"/>
```

```
<asp:Button ID="DeletButton" runat="server" Text="Delete"
oncommand="CommandButton_Click" CommandName="Delete"/>
```

```
<asp:Button ID="Top10Button" runat="server" Text="Show Top 10 Employees"
oncommand="CommandButton_Click"
    CommandName="Show" CommandArgument="Top10"/>
```

```
<asp:Button ID="Bottom10Button" runat="server" Text="Show Bottom 10 Employees"
oncommand="CommandButton_Click"
    CommandName="Show" CommandArgument="Bottom10"/>
```

```
<asp:Label ID="OutputLabel" runat="server"></asp:Label>
```

Copy and Paste the following code in your code behind file. The CommandEventArgs object, has the CommandName and CommandArgument properties, that are used to programmatically determine which button the user has clicked.

```
protected void CommandButton_Click(object sender, CommandEventArgs e)
{
    switch (e.CommandName)
    {
        case "Print":
            OutputLabel.Text = "You clicked Print Button";
            break;
        case "Delete":
            OutputLabel.Text = "You clicked Delete Button";
            break;
        case "Show":
            if (e.CommandArgument.ToString() == "Top10")
            {
                OutputLabel.Text = "You clicked Show Top 10 Employees Button";
            }
            else
            {
                OutputLabel.Text = "You clicked Show Bottom 10 Employees Button";
            }
        }
    }
```

```
    }  
    break;  
default:  
    OutputLabel.Text = "We don't know which button you clicked";  
    break;  
}  
}
```

Note: All the 3 button controls - Button, LinkButton and ImageButton, expose Command event, the CommandName and CommandArgument properties.

## Dropdownlist in asp.net

Drag and drop a DropDownList control onto the webform.

To add items to the DropDownList at design time:

1. Right click on the DropDownList control and select properties.
2. In the properties, click on the ellipsis button next to Items property.
3. In the ListItem Collection Editor window, click the Add button
4. Set the Text to Male and Value to 1.
5. Click the Add button again, which will add another ListItem object
6. Set the Text to Female and Value to 2.
7. Finally click OK

Now switch the webform to source mode. Notice that in the HTML it has added ListItem object, as shown below.

```
<asp:DropDownList ID="DropDownList1" runat="server">
  <asp:ListItem Value="1">Male</asp:ListItem>
  <asp:ListItem Value="1">Female</asp:ListItem>
</asp:DropDownList>
```

If you run the web application now, you should see that Male and Female items shown in the DropDownList.

If you want a specific listitem to be selected in the dropdownlist, when the page loads, then set the Selected property of the ListItem object to true. This can be done in 2 ways:

1. Using the ListItem Collection Editor window or
2. In the HTML of the webform

The HTML with the Selected property is shown below.

```
<asp:DropDownList ID="DropDownList1" runat="server">
  <asp:ListItem Value="1">Male</asp:ListItem>
  <asp:ListItem Value="1" Selected="True">Female</asp:ListItem>
</asp:DropDownList>
```

To hide a ListItem in the DropDownList, set the Enabled property to False.  
To add items to the DropDownList at run time using code

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ListItem maleListItem = new ListItem("Male", "1");
        ListItem femaleListItem = new ListItem("Female", "2");

        DropDownList1.Items.Add(maleListItem);
        DropDownList1.Items.Add(femaleListItem);
    }
}
```

If you are adding listitem objects, to the DropDownList in the Page\_Load event, make sure you do only when the page is loaded for the first time. Otherwise, every time, you post the page back, by clicking a button, the list items will be added again causing duplication.

A DropDownList is a collection of ListItem objects. Along the same lines, the following controls are also a collection of ListItem objects. So, adding items to these controls is also very similar to DropDownList:

1. CheckBoxList
2. RadioButtonList
3. BulletedList
4. ListBox

## Data bind asp.net dropdownlist with data from the database

Binding data from a database table, to a dropdownlist. We will be using tblCity table for this demo. Please find the script below, to create and populate the table.

Create table tblCity

```
(  
    CityId int primary key,  
    CityName nvarchar(50),  
    Country nvarchar(50)  
)
```

Insert into tblCity values(101, 'Delhi', 'India')

Insert into tblCity values(102, 'London', 'UK')

Insert into tblCity values(103, 'New York', 'US')

Insert into tblCity values(104, 'Tokyo', 'Japan')

Create an ASP.NET web application. Drag and drop a DropDownList control onto the webform. Copy and paste the following code in the code behind page.

```
protected void Page_Load(object sender, EventArgs e)  
{  
    if (!IsPostBack)  
    {  
        string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
        using (SqlConnection con = new SqlConnection(CS))  
        {  
            SqlCommand cmd = new SqlCommand("Select CityId, CityName, Country from tblCity",  
con);  
            con.Open();  
            SqlDataReader rdr = cmd.ExecuteReader();  
            DropDownList1.DataSource = rdr;  
            DropDownList1.DataBind();  
        }  
    }  
}
```

Run the application. Notice that, the DropDownList displays, System.Data.Common.DataRecordInternal instead of the City names. This is because, we haven't specified the DataTextField and DataValueField properties of the DropDownList. The code below specifies both the properties.

```
protected void Page_Load(object sender, EventArgs e)  
{  
    if (!IsPostBack)  
    {  
        string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;  
        using (SqlConnection con = new SqlConnection(CS))  
        {  
            SqlCommand cmd = new SqlCommand("Select CityId, CityName, Country from tblCity",  
con);  
            con.Open();
```

```
        SqlDataReader rdr = cmd.ExecuteReader();
        DropDownList1.DataTextField = "CityName";
        DropDownList1.DataValueField = "CityId";
        DropDownList1.DataSource = rdr;
        DropDownList1.DataBind();
    }
}
```

Run the application now. The city names are displayed as expected. But make sure to set the properties(DataTextField, DataValueField) before calling DataBind() method. Also, note that, these properties can be set in the HTML of the aspx page as well.

```
<asp:DropDownList ID="DropDownList1" DataTextField="CityName"
    DataValueField="CityId" runat="server">
</asp:DropDownList>
```

## Binding an asp.net dropdownlist with an XML file

Binding a DropDownList to data that is retrieved from a database. We will learn about binding the DropDownList to Data from an XML file.

First add an XML file, to the web application project. To do this:

1. Right click on the web application project, and select Add => New Item.
2. In the Add New Item dialog box, select XML File.
3. Give the XML file a meaningful name. In our case let's name it Countries.xml and click Add.
4. In the Countries.xml file, copy and paste the following

```
<?xml version="1.0" encoding="utf-8" ?>
<Countries>
  <Country>
    <CountryId>101</CountryId>
    <CountryName>India</CountryName>
  </Country>
  <Country>
    <CountryId>102</CountryId>
    <CountryName>US</CountryName>
  </Country>
  <Country>
    <CountryId>103</CountryId>
    <CountryName>Australia</CountryName>
  </Country>
  <Country>
    <CountryId>104</CountryId>
    <CountryName>UK</CountryName>
  </Country>
</Countries>
```

Drag and drop a DropDownList on the webform. Copy and paste the following code in the code behind page.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //Create a new DataSet
        DataSet DS = new DataSet();
        //Read the xml data from the XML file using ReadXml() method
        DS.ReadXml(Server.MapPath("Countries.xml"));
        DropDownList1.DataTextField = "CountryName";
        DropDownList1.DataValueField = "CountryId";
        DropDownList1.DataSource = DS;
        DropDownList1.DataBind();
        ListItem li = new ListItem("Select", "-1");
        DropDownList1.Items.Insert(0, li);
    }
}
```



The important thing to notice here is that, we are using `ReadXml()` method of the `DataSet` object, to read the data from the `Countries.xml` file into a `DataSet`. `Server.MapPath()` method returns the physical path of the file from the provided virtual path.

To insert a `ListItem` at a specific location use the `Insert()` method specifying the index of the location where you want to insert, and the `listitem` object.

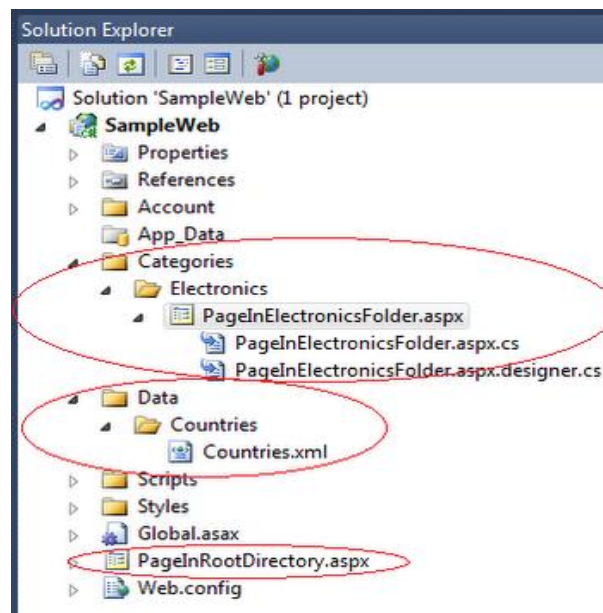
## Mapping virtual path to physical path using Server.MapPath method

This method returns the physical path for a given virtual path. This method can be used in several different ways, depending on the characters that we use in the virtual path. Let's understand this with an example.

1. Create an asp.net web application in C:\ and name it SampleWeb.
2. Right click on the SampleWeb project in solution explorer and add a new webform and name it PageInRootDirectory.aspx
3. Add a new folder to the project and name it Categories.
4. Right click on the Categories folder, and add another folder. name it Electronics
5. Add a webform to the Electronics folder and name it PageInElectronicsFolder.aspx
6. At this point, right click on the web application project and add a new folder. Name it Data.
7. Add a sub folder to Data, and name it Countries
8. Right click on the Countries folder and add an XML file. Name it Countries.xml.
9. Copy and paste the following in Countries.xml file.

```
<?xml version="1.0" encoding="utf-8" ?>
<Countries>
  <Country>
    <CountryId>101</CountryId>
    <CountryName>India</CountryName>
  </Country>
  <Country>
    <CountryId>102</CountryId>
    <CountryName>US</CountryName>
  </Country>
  <Country>
    <CountryId>103</CountryId>
    <CountryName>Australia</CountryName>
  </Country>
  <Country>
    <CountryId>104</CountryId>
    <CountryName>UK</CountryName>
  </Country>
</Countries>
```

If you are following along with me, at this stage, the solution explorer should look as shown below.



Copy and paste the following code in PageInElectronicsFolder.aspx.cs

```
Response.Write(". returns " + Server.MapPath(".") + "<br/>");  
Response.Write(".. returns " + Server.MapPath("..") + "<br/>");  
Response.Write("~ returns " + Server.MapPath("~/") + "<br/>");
```

Running this page would produce the following output.

```
. returns C:\SampleWeb\SampleWeb\Categories\Electronics  
.. returns C:\SampleWeb\SampleWeb\Categories  
~ returns C:\SampleWeb\SampleWeb
```

From the output, it should be clear that

Server.MapPath(".") returns the current physical directory of the page that you are running  
Server.MapPath("..") returns the parent physical directory of the page that you are running  
Server.MapPath("~/") returns the physical path of the root directory of the application

## Mapping virtual path to physical path using Server.MapPath method

Drag and drop a DropDownList control onto PageInElectronicsFolder.aspx webform. Copy and paste the following code in the code behind file.

```
DataSet DS = new DataSet();
DS.ReadXml(Server.MapPath("../Data/Countries/Countries.xml"));
DropDownList1.DataTextField = "CountryName";
DropDownList1.DataValueField = "CountryId";
DropDownList1.DataSource = DS;
DropDownList1.DataBind();
```

C:\SampleWeb\SampleWeb is the root directory for the sample asp.net web application that we used in the Demo. To get to this root directory we are passing ../ to the Server.MapPath() method as shown below.

```
DS.ReadXml(Server.MapPath("../Data/Countries/Countries.xml"));
```

The number of double dots, that you use, depends on how deep you are in the folder hierarchy. To avoid confusion, if any time you want to navigate to the root directory of the application, it is better to use ~(tilde) character as shown below.

```
DS.ReadXml(Server.MapPath("~/Data/Countries/Countries.xml"));
```

Tilde(~) symbol resolves to the root application directory, no matter how deep you are in the folder hierarchy. This is the advantage of using ~(tilde), over ../(2 Dots). The following code would work from any folder in our application.

```
DS.ReadXml(Server.MapPath("~/Data/Countries/Countries.xml"));
```

Where as, the following code will only work from folders, that are 2 levels deeper relative to the root directory of the application.

```
DS.ReadXml(Server.MapPath("../Data/Countries/Countries.xml"));
```

## Retrieving selected item text, value and index of an asp.net dropdownlist

we will discuss about retrieving the selected item text, index and value from a dropdownlist. Let's first add some items to the dropdownlist. To do this:

1. Drag and drop a dropdownlist and a button control onto the webform
2. Copy and paste the following HTML on the webform.

```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem Text="Select Continent" Value="-1"></asp:ListItem>
    <asp:ListItem Text="Asia" Value="1"></asp:ListItem>
    <asp:ListItem Text="Europe" Value="2"></asp:ListItem>
    <asp:ListItem Text="Africa" Value="3"></asp:ListItem>
    <asp:ListItem Text="North America" Value="4"></asp:ListItem>
    <asp:ListItem Text="South America" Value="5"></asp:ListItem>
</asp:DropDownList>
<br /><br />
<asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
```

Copy and paste the following code in the code behind file.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (DropDownList1.SelectedValue != "-1")
    {
        Response.Write("Selected Item Text = " + DropDownList1.SelectedItem.Text + "<br/>");
        Response.Write("Selected Item Value = " + DropDownList1.SelectedItem.Value + "<br/>");
        Response.Write("Selected Item Index = " + DropDownList1.SelectedIndex.ToString());
    }
    else
    {
        Response.Write("Please select the continent");
    }
}
```

Explanation of the code:

1. In the button click event, we first check if the user has made a selection. If the DropDownList1.SelectedValue is -1, then we know that user has not made a choice.  
`if (DropDownList1.SelectedValue != "-1")`
2. If the user has made a selection in the DropDownList, then we retrieve the Text of the Selected ListItem object as shown below.  
`DropDownList1.SelectedItem.Text`
3. To retrieve the Selected item value from the DropDownList, we can use  
`DropDownList1.SelectedItem.Value`  
OR  
`DropDownList1.SelectedValue`
4. To get the index of the Selected ListItem object of the dropdownlist, we use SelectedIndex property as shown below.

### DropDownList1.SelectedIndex

The SelectedIndex and SelectedValue properties of the DropDownList can also be used to have a list

item selected in the DropDownList. For example, the following code would default the selection to Asia when the page loads for the first time.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        DropDownList1.SelectedValue = "1";
    }
}
```

The same thing can also be achieved using the SelectedIndex property as shown below.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        DropDownList1.SelectedIndex = 1;
    }
}
```

## Cascading dropdown in asp.net

First create the required tables and populate them, with some sample data using the SQL script below.

Create Table tblContinents

```
(  
  ContinentId int identity primary key,  
  ContinentName nvarchar(50)  
)
```

Insert into tblContinents values ('Asia')

Insert into tblContinents values ('Europe')

Insert into tblContinents values ('South America')

Create Table tblCountries

```
(  
  CountryId int identity primary key,  
  CountryName nvarchar(50),  
  ContinentId int foreign key references dbo.tblContinents(ContinentId)  
)
```

Insert into tblCountries values ('India', 1)

Insert into tblCountries values ('Japan', 1)

Insert into tblCountries values ('Malaysia', 1)

Insert into tblCountries values ('United Kingdom', 2)

Insert into tblCountries values ('France', 2)

Insert into tblCountries values ('Germany', 2)

Insert into tblCountries values ('Argentina', 3)

Insert into tblCountries values ('Brazil', 3)

Insert into tblCountries values ('Colombia', 3)

Create Table tblCities

```
(  
  CityId int identity primary key,  
  CityName nvarchar(50),  
  CountryId int foreign key references dbo.tblCountries(CountryId)  
)
```

Insert into tblCities values ('Bangalore', 1)

Insert into tblCities values ('Chennai', 1)

Insert into tblCities values ('Mumbai', 1)

Insert into tblCities values ('Tokyo', 2)

Insert into tblCities values ('Hiroshima', 2)

Insert into tblCities values ('Saku', 2)

Insert into tblCities values ('Kuala Lumpur', 3)

Insert into tblCities values ('Ipoh', 3)

Insert into tblCities values ('Tawau', 3)

Insert into tblCities values ('London', 4)

Insert into tblCities values ('Manchester', 4)  
Insert into tblCities values ('Birmingham', 4)

Insert into tblCities values ('Paris', 5)  
Insert into tblCities values ('Cannes', 5)  
Insert into tblCities values ('Nice', 5)

Insert into tblCities values ('Frankfurt', 6)  
Insert into tblCities values ('Eutin', 6)  
Insert into tblCities values ('Alsfeld', 6)

Insert into tblCities values ('Rosario', 7)  
Insert into tblCities values ('Salta', 7)  
Insert into tblCities values ('Corrientes', 7)

Insert into tblCities values ('Rio de Janeiro', 8)  
Insert into tblCities values ('Salvador', 8)  
Insert into tblCities values ('Brasília', 8)

Insert into tblCities values ('Cali', 9)  
Insert into tblCities values ('Montería', 9)  
Insert into tblCities values ('Bello', 9)

```
Create procedure spGetContinents
as
Begin
    Select ContinentId, ContinentName from tblContinents
End
```

```
Create procedure spGetCountriesByContinentId
@ContinentId int
as
Begin
    Select CountryId, CountryName from tblCountries
    where ContinentId = @ContinentId
End
```

```
Create procedure spGetCitiesByCountryId
@CountryId int
as
Begin
    Select CityId, CityName from tblCities
    where CountryId = @CountryId
End
```

Let's understand cascading dropdownlists with an example. The following are the 3 dropdownlist controls, that we will have in our asp.net web application:

1. Continents DropDownList
2. Countries DropDownList
3. Cities DropDownList



When the webform first loads, only the continents dropdownlist should be populated. Countries and Cities dropdownlist should be disabled and should not allow the user to select anything from these 2 dropdownlists. Once, the user makes a selection in the continents dropdownlist, and then Countries dropdownlist should be enabled and populated with the countries that belong to the selected continent. The same logic applies for the cities dropdownlist.

To achieve this drag and drop 3 dropdownlist controls onto the webform. The HTML of the Webform should be as shown below.

```
<asp:DropDownList ID="ddlContinents" Width="200px" DataTextField="ContinentName"
    DataValueField="ContinentId" runat="server" AutoPostBack="True"
    onselectedindexchanged="ddlContinents_SelectedIndexChanged">
</asp:DropDownList>
<br /><br />
<asp:DropDownList ID="ddlCountries" DataValueField="CountryId"
    DataTextField="CountryName" Width="200px" runat="server" AutoPostBack="True"
    onselectedindexchanged="ddlCountries_SelectedIndexChanged">
</asp:DropDownList>
<br /><br />
<asp:DropDownList ID="ddlCities" Width="200px" DataTextField="CityName"
    DataValueField="CityId" runat="server">
</asp:DropDownList>
```

Copy and paste the following code in the code behind page

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        PopulateContinentsDropDownList();
    }
}

private void PopulateContinentsDropDownList()
{
    ddlContinents.DataSource = GetData("spGetContinents", null);
    ddlContinents.DataBind();

    ListItem liContinent = new ListItem("Select Continent", "-1");
    ddlContinents.Items.Insert(0, liContinent);

    ListItem liCountry = new ListItem("Select Country", "-1");
    ddlCountries.Items.Insert(0, liCountry);

    ListItem liCity = new ListItem("Select City", "-1");
    ddlCities.Items.Insert(0, liCity);

    ddlCountries.Enabled = false;
    ddlCities.Enabled = false;
}

private DataSet GetData(string SPName, SqlParameter SPParameter)
```

```

{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    SqlConnection con = new SqlConnection(CS);
    SqlDataAdapter da = new SqlDataAdapter(SPName, con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    if (SPParameter != null)
    {
        da.SelectCommand.Parameters.Add(SPParameter);
    }
    DataSet DS = new DataSet();
    da.Fill(DS);
    return DS;
}

```

```

protected void ddlContinents_SelectedIndexChanged(object sender, EventArgs e)
{
    if (ddlContinents.SelectedValue == "-1")
    {
        ddlCities.SelectedIndex = 0;
        ddlCountries.SelectedIndex = 0;
        ddlCities.Enabled = false;
        ddlCountries.Enabled = false;
    }
    else
    {
        ddlCountries.Enabled = true;

        SqlParameter parameter = new SqlParameter();
        parameter.ParameterName = "@ContinentId";
        parameter.Value = ddlContinents.SelectedValue;

        ddlCountries.DataSource = GetData("spGetCountriesByContinentId", parameter);
        ddlCountries.DataBind();

        ListItem liCountry = new ListItem("Select Country", "-1");
        ddlCountries.Items.Insert(0, liCountry);

        ddlCities.SelectedIndex = 0;
        ddlCities.Enabled = false;
    }
}

```

```

protected void ddlCountries_SelectedIndexChanged(object sender, EventArgs e)
{
    if (ddlCountries.SelectedValue == "-1")
    {
        ddlCities.SelectedIndex = 0;
        ddlCities.Enabled = false;
    }
    else
    {
        ddlCities.Enabled = true;
    }
}

```

```
SqlParameter parameter = new SqlParameter();
parameter.ParameterName = "@CountryId";
parameter.Value = ddlCountries.SelectedValue;

ddlCities.DataSource = GetData("spGetCitiesByCountryId", parameter);
ddlCities.DataBind();

ListItem liCity = new ListItem("Select City", "-1");
ddlCities.Items.Insert(0, liCity);
}
}
```

## Asp.net checkboxlist control

We will learn about asp.net checkboxlist control. Just like DropDownList:

1. CheckBoxLayout is collection of ListItem objects.
2. Items can be added to the CheckBoxLayout in the HTML source or in the code behind file
3. CheckBoxLayout can be bound to a database table or an xml file

DropDownList is generally used, when you want to present the user with multiple choices, from which you want him to select only one option. Where as if you want the user to select more than one option, then a CheckBoxLayout control can be used.

Create an asp.net web application. Copy and paste the following HTML

```
<asp:CheckBoxLayout ID="checkboxBoxLayoutEducation" runat="server"
    RepeatDirection="Horizontal">
    <asp:ListItem Text="Diploma" Value="1"></asp:ListItem>
    <asp:ListItem Text="Graduate" Value="2"></asp:ListItem>
    <asp:ListItem Text="Post Graduate" Value="3"></asp:ListItem>
    <asp:ListItem Text="Doctrate" Value="4"></asp:ListItem>
</asp:CheckBoxLayout>
<br />
<asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
```

Copy and paste the following code in Button1\_Click event. The sample code prints the Text, Value and Index of the selected list item object.

```
// Loop thru each list item in the checkboxlist
foreach (ListItem li in checkboxBoxLayoutEducation.Items)
{
    // If the list item is selected
    if (li.Selected)
    {
        // Retrieve the text of the selected list item
        Response.Write("Text = " + li.Text + ", ");
        // Retrieve the value of the selected list item
        Response.Write("Value = " + li.Value + ", ");
        // Retrieve the index of the selected list item
        Response.Write("Index = " + checkboxBoxLayoutEducation.Items.IndexOf(li).ToString());
        Response.Write("<br/>");
    }
}
```

By default, the ListItem objects are laid out in vertical direction. If you want to change the direction, use RepeatDirection property

```
<asp:CheckBoxLayout ID="checkboxBoxLayoutEducation" runat="server" RepeatDirection="Horizontal">
```

RepeatColumns property specifies the number of columns used to lay out the items.

Set the Enabled property of the ListItem object to false, to disable the selection, in the CheckBoxLayout control.

SelectedIndex property of the CheckBoxList control can also be used to get the index of the selected item in the checkboxlist. But this property, returns only one selected item, and that too, the item with the lowest index. SelectedIndex property returns -1, if nothing is selected.

SelectedValue property returns the selected Item's value, but only for one selected item. If no item is selected this property returns empty string.

To retrieve the Text of the selected item, SelectedItem.Text property can be used. SelectedItem will be NULL, if nothing is selected, and hence, calling Text and Value properties may cause NullReferenceException. Hence, it is important to check for null, when using SelectedItem property of a CheckBoxList control.

```
if (checkboxListEducation.SelectedItem != null)
{
    Response.Write(checkboxListEducation.SelectedItem.Text);
}
```

## Asp.net checkboxlist, select or deselect all list items

1. Selecting a specific ListItem with-in a CheckBoxList control using SelectedValue and SelectedIndex properties
2. Selecting or De-Selecting all ListItems of a CheckBoxList control

To have a ListItem pre-selected, when the page renders, we can do that in the HTML by setting the Selected property to True as shown below.

```
<asp:ListItem Text="Diploma" Selected="True" Value="1"></asp:ListItem>
```

This can be programmatically done, using the SelectedValue property as shown below.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        CheckBoxList1.SelectedValue = "1";
    }
}
```

SelectedIndex property can also be used.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        CheckBoxList1.SelectedIndex = 2;
    }
}
```

Copy and paste the following HTML and Code in the ASPX and the code behind page

```
<asp:Button ID="buttonSelectAll" runat="server" Text="Select All"
    onclick="buttonSelectAll_Click" />
 
<asp:Button ID="buttonDeselectAll" runat="server" Text="De-Select All"
    onclick="buttonDeselectAll_Click" />
<br /><br />
<asp:CheckBoxList ID="CheckBoxList1" runat="server"
    RepeatDirection="Horizontal">
    <asp:ListItem Text="Diploma" Value="1"></asp:ListItem>
    <asp:ListItem Text="Graduate" Value="2"></asp:ListItem>
    <asp:ListItem Text="Post Graduate" Value="3"></asp:ListItem>
    <asp:ListItem Text="Doctrate" Value="4"></asp:ListItem>
</asp:CheckBoxList>
```

```
protected void buttonSelectAll_Click(object sender, EventArgs e)
{
    foreach (ListItem li in CheckBoxList1.Items)
    {
        li.Selected = true;
    }
}
```

```
}
```

```
protected void buttonDeselectAll_Click(object sender, EventArgs e)
```

```
{
```

```
    foreach (ListItem li in CheckBoxList1.Items)
```

```
    {
```

```
        li.Selected = false;
```

```
    }
```

```
}
```

## Asp.net ListBox control

We will discuss about ListBox control. Just like DropDownList and CheckBoxLayout, ListBox control is also a collection of ListItem objects. Working with ListBox control is very similar to DropDownList and CheckBoxLayout. Adding items and binding to a datasource is exactly similar. let's discuss about the properties that are specific to the ListBox control.

### Properties:

**Rows** : The number of visible rows in the Listbox. A scrollbar is automatically generated, if the total number of item are greater than the number of visible rows in the listbox.

**SelectionMode** : SelectionMode can be Single or Muiltiple. By default, this property value is Single, meaning when the listbox renders, the user can select only one item from the listbox. Set this property to Muiltiple, to enable multiple item selections. To select, multiple items from the listbox, hold-down the CTRL key, while the listitem's are selected.

Please note that, it is not possible to set the selected property of more than one ListItem object to true, if the SelectionMode of the listbox is to Single.

Retrieving the selected item's Text, Value and Index is similar to DropDownList and CheckBoxLayout

### ASPX Page code:

```
<asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
  <asp:ListItem Text="Diploma" Value="1"></asp:ListItem>
  <asp:ListItem Text="Graduate" Value="2"></asp:ListItem>
  <asp:ListItem Text="Post Graduate" Value="3"></asp:ListItem>
  <asp:ListItem Text="Doctrate" Value="4"></asp:ListItem>
</asp:ListBox>
<br />
<br />
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
```

### Code Behind Code:

```
protected void Button1_Click(object sender, EventArgs e)
{
    foreach (ListItem li in ListBox1.Items)
    {
        if (li.Selected)
        {
            Response.Write("Text = " + li.Text + ", ");
            Response.Write("Value = " + li.Value + ", ");
            Response.Write("Index = " + ListBox1.Items.IndexOf(li).ToString());
            Response.Write("<br/>");
        }
    }
}
```

### ASP.NET ListBox Example:



Text = Diploma, Value = 1, Index = 0

Text = Post Graduate, Value = 3, Index = 2

Text = Doctrate, Value = 4, Index = 3

Diploma
Graduate
Post Graduate
Doctrate

Button
--------

## ASP.NET CheckBoxList and ListBox real time example

We will discuss about a simple real time example using asp.net checkboxlist and listbox.

Copy and Paste the following HTML on the ASPX page

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server"
    RepeatDirection="Horizontal" AutoPostBack="True"
    onselectedindexchanged="CheckBoxList1_SelectedIndexChanged">
    <asp:ListItem Text="Diploma" Value="1"></asp:ListItem>
    <asp:ListItem Text="Graduate" Value="2"></asp:ListItem>
    <asp:ListItem Text="Post Graduate" Value="3"></asp:ListItem>
    <asp:ListItem Text="Doctrate" Value="4"></asp:ListItem>
</asp:CheckBoxList>
<br />
<asp:ListBox ID="ListBox1" runat="server" Height="78px" Width="127px">
</asp:ListBox>
<br /><br />
<asp:Label ID="lblMessage" runat="server" Font-Bold="true"></asp:Label>
```

Copy and Paste the following code in the code behind page

```
protected void CheckBoxList1_SelectedIndexChanged(object sender, EventArgs e)
{
    // Everytime the selection changes, clear the items in the listbox
    ListBox1.Items.Clear();
    // Loop thru each listitem in the checkboxlist
    foreach (ListItem li in CheckBoxList1.Items)
    {
        // If the listitem is selected
        if (li.Selected)
        {
            // Add the listitem text to the listbox
            ListBox1.Items.Add(li.Text);

            // Add the listitem as an object. This ensures the listitem is
            // selected in the listbox. For this to work, listbox,
            // SelectionMode must be set to Multiple. The SelectionMode
            // Property can be set in the HTML source also.
            // ListBox1.SelectionMode = ListSelectionMode.Multiple
            // ListBox1.Items.Add(li);
        }
    }
    // If nothing is selected from the checkboxlist
    if (CheckBoxList1.SelectedIndex == -1)
    {
        // Set the label ForeColor to Red
        lblMessage.ForeColor = System.Drawing.Color.Red;
    }
    // If atleast one listitem is selected
    else
    {

```

```
// Set the label forecolor to black
lblMessage.ForeColor = System.Drawing.Color.Black;
}
// Display the total number of items selected from the checkboxlist
lblMessage.Text = ListBox1.Items.Count.ToString() + " item(s) selected";
}
```

## ASP.NET RadioButtonList Control

In ASP.NET there are several list controls, like

1. DropDownList
2. CheckBoxLayout
3. BulletedList
4. ListBox
5. RadioButtonList

we will learn about asp.net RadioButtonList control. Just like every other list control:

1. RadioButtonList is also a collection of ListItem objects.
2. Items can be added to the RadioButtonList in the HTML source or in the code behind file
3. RadioButtonList like any other list control supports databinding. For example, RadioButtonList can be bound to a database table or an xml file

CheckBoxLayout is generally used, when you want to present the user with multiple choices, from which you want him to select one or more options. Where as if you want the user to select only one option, then a RadioButtonList control can be used, i.e RadioButtonList is commonly used to present mutually exclusive choices.

Create an asp.net web application. Copy and paste the following HTML

```
<asp:RadioButtonList ID="ColorRadioButtonList" runat="server"
    RepeatDirection="Horizontal">
    <asp:ListItem Text="Red" Value="1"></asp:ListItem>
    <asp:ListItem Text="Green" Value="2"></asp:ListItem>
    <asp:ListItem Text="Blue" Value="3"></asp:ListItem>
    <asp:ListItem Text="Orange" Value="4"></asp:ListItem>
</asp:RadioButtonList>
<br />
<asp:Button ID="btnSubmit" runat="server" Text="Submit"
    onclick="btnSubmit_Click"/>&nbsp;
<asp:Button ID="btnClearSelection" runat="server" Text="Clear Selection"
    onclick="btnClearSelection_Click"/>
```

Copy and paste the following code in your code-behind page

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    // If the user has made a choice
    if (ColorRadioButtonList.SelectedIndex != -1)
    {
        Response.Write("Text = " + ColorRadioButtonList.SelectedItem.Text + "<br/>");
        Response.Write("Value = " + ColorRadioButtonList.SelectedItem.Value + "<br/>");
        Response.Write("Index = " + ColorRadioButtonList.SelectedIndex.ToString());
    }
    // If the user has not selected anything
    else
    {
        Response.Write("Please select your favourite color");
    }
}
```

```

    }
}

protected void btnClearSelection_Click(object sender, EventArgs e)
{
    // Clear the user selection
    ColorRadioButtonList.SelectedIndex = -1;
}

```

### RadioButtonList Example

Text = Green  
 Value = 2  
 Index = 1

☐ Red ☒ Green ☐ Blue ☐ Orange

By default, the ListItem objects are laid out in vertical direction. If you want to change the direction, use RepeatDirection property

```
<asp:RadioButtonList ID="ColorRadioButtonList" runat="server" RepeatDirection="Horizontal">
```

RepeatColumns property specifies the number of columns used to lay out the items.

RepeatLayout property, specifies the layout to be used by each list item. The following are the values allowed by RepeatLayout property:

1. Table
2. Flow
3. OrderedList
4. UnorderedList

Please note that the, OrderedList and UnorderedList layouts are only supported, if the RepeatDirection is vertical.

Set the Enabled property of the ListItem object to false, to disable the selection, in the RadioButtonList control.

To retrieve the Text of the selected item, SelectedItem.Text property can be used. SelectedItem will be NULL, if nothing is selected, and hence, calling Text and Value properties may cause **NullReferenceException**. Hence, it is important to check for null, when using SelectedItem property of a RadioButtonList control.

```

if (ColorRadioButtonList.SelectedItem != null)
{
    Response.Write(ColorRadioButtonList.SelectedItem.Text);
}

```

NullReferenceException can also be avoided, using the SelectedIndex property

```
if (ColorRadioButtonList.SelectedIndex != -1)
{
    Response.Write(ColorRadioButtonList.SelectedItem.Text);
}
```

## Bulleted list in asp.net

In ASP.NET there are several list controls, like

1. DropDownList
2. CheckBoxLayout
3. RadioButtonList
4. ListBox
5. BulletedList

Just like every other list control

1. BulletedList is also a collection of ListItem objects.
2. Items can be added to the BulletedList in the HTML source or in the code behind file
3. BulletedList like any other list control supports databinding. For example, BulletedList can be bound to a database table or an xml file

### Properties of BulletedList:

**BulletStyle** - This property, is used to customize the bullets style. If CustomImage is specified as the BulletStyle, then BulletImageUrl, also needs to be specified.

```
<asp:BulletedList ID="CountriesBulletedList" runat="server" BulletStyle="Numbered">
  <asp:ListItem Text="India" Value="1"></asp:ListItem>
  <asp:ListItem Text="US" Value="2"></asp:ListItem>
  <asp:ListItem Text="UK" Value="3"></asp:ListItem>
  <asp:ListItem Text="France" Value="4"></asp:ListItem>
</asp:BulletedList>
```

**FirstBulletNumber** - The Number at which the ordered list starts.

**DisplayMode** - Can be Text, HyperLink or LinkButton. The default is Text.

The following HTML, sets the DisplayMode="HyperLink". By default, the target page is displayed in the same browser window. Set the Target property of the BulletedList to \_blank, to open the target page in it's own window.

```
<asp:BulletedList ID="CountriesBulletedList" runat="server"
  BulletStyle="Numbered" DisplayMode="HyperLink">
  <asp:ListItem Text="Google" Value="http://google.com"></asp:ListItem>
  <asp:ListItem Text="Youtube" Value="http://Youtube.com"></asp:ListItem>
  <asp:ListItem Text="Blogger" Value="http://Blogger.com"></asp:ListItem>
  <asp:ListItem Text="Gmail" Value="http://Gmail.com"></asp:ListItem>
</asp:BulletedList>
```

The following HTML, sets DisplayMode="LinkButton" and onclick="CountriesBulletedList\_Click"

```
<asp:BulletedList ID="CountriesBulletedList" runat="server"
  DisplayMode="LinkButton" onclick="CountriesBulletedList_Click">
  <asp:ListItem Text="Google" Value="1"></asp:ListItem>
  <asp:ListItem Text="Microsoft" Value="2"></asp:ListItem>
  <asp:ListItem Text="Dell" Value="3"></asp:ListItem>
  <asp:ListItem Text="IBM" Value="4"></asp:ListItem>
</asp:BulletedList>
```

## Code behind code

```
protected void CountriesBulletdList_Click(object sender, BulletedListEventArgs e)
{
    ListItem li = CountriesBulletdList.Items[e.Index];
    Response.Write("Text = " + li.Text + "<br/>");
    Response.Write("Value = " + li.Value + "<br/>");
    Response.Write("Index = " + e.Index.ToString());
}
```



## List controls in asp.net

In ASP.NET there are several list controls, like

1. DropDownList
2. CheckBoxLayout
3. RadioButtonList
4. ListBox
5. BulletedList

All these controls are:

1. Collection of ListItem objects
2. ListItems can be added in the HTML source or in the code behind file
3. Supports Databinding

The only difference here is the tag name, otherwise adding ListItems is very identical.

```
<asp:DropDownList ID="DropDownList1" runat="server">
  <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
  <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
  <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
</asp:DropDownList>
<br />
<asp:CheckBoxLayout ID="CheckBoxLayout1" runat="server">
  <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
  <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
  <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
</asp:CheckBoxLayout>
<br />
<asp:RadioButtonList ID="RadioButtonList1" runat="server">
  <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
  <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
  <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
</asp:RadioButtonList>
<br />
<asp:ListBox ID="ListBox1" runat="server">
  <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
  <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
  <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
</asp:ListBox>
<br />
<asp:BulletedList ID="BulletedList1" runat="server">
  <asp:ListItem Text="Item1" Value="1"></asp:ListItem>
  <asp:ListItem Text="Item2" Value="2"></asp:ListItem>
  <asp:ListItem Text="Item3" Value="3"></asp:ListItem>
</asp:BulletedList>
```

Adding ListItems using code. Since all the list controls inherit from ListControl class, AddListItems() method can be used to add listitems to any list control. A parent class reference variable can point to a derived class object.

This fact allows us to pass any list control into the AddListItems() method as a parameter.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        AddListItems(DropDownList1);
        AddListItems(CheckBoxList1);
        AddListItems(RadioButtonList1);
        AddListItems(ListBox1);
        AddListItems(BulletedList1);
    }
}

private void AddListItems(ListControl listControl)
{
    ListItem li1 = new ListItem("Item1", "1");
    ListItem li2 = new ListItem("Item2", "2");
    ListItem li3 = new ListItem("Item3", "3");

    listControl.Items.Add(li1);
    listControl.Items.Add(li2);
    listControl.Items.Add(li3);
}
```

ListBox (If SelectionMode=Multiple) and CheckBoxList allows user to select multiple items. So, to retrieve all the selected listitem's Text, Value and Index use a foreach loop.

Reusable method that can be used with any control that derives from ListControl class, but works best with controls that allows multiple selections.

```
private void RetrieveMultipleSelections(ListControl listControl)
{
    foreach (ListItem li in listControl.Items)
    {
        if (li.Selected)
        {
            Response.Write("Text = " + li.Text + ", Value = " + li.Value +
                ", Index = " + listControl.Items.IndexOf(li).ToString() + "<br/>");
        }
    }
}
```

ListBox (If SelectionMode=Single), RadioButtonList and DropDownList allows user to select only one item. So, use SelectedIndex and SelectedItem properties to retrieve the Text, Value and Index of the selected listitem.

Reusable method that can be used with any control that derives from ListControl class, but works best with controls that allows single selection.

```
private void RetrieveSelectedItemTextValueandIndex(ListControl listControl)
{

```

```

if (listControl.SelectedIndex != -1)
{
    Response.Write("Text = " + listControl.SelectedItem.Text + "<br/>");
    Response.Write("Value = " + listControl.SelectedItem.Value + "<br/>");
    Response.Write("Index = " + listControl.SelectedIndex.ToString());
}
}

```

BulletedList(If DisplayMode=LinkButton) - In the click event handler, use the BulletedListEventArgs parameter object to retrieve the Text, Value and Index of the listitem, the user has clicked.

```

protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
{
    ListItem li = BulletedList1.Items[e.Index];
    Response.Write("Text = " + li.Text + "<br/>");
    Response.Write("Value = " + li.Value + "<br/>");
    Response.Write("Index = " + e.Index.ToString());
}

```

## Fileupload control in asp.net

We will discuss about fileupload control. FileUpload control is a combination of a text box and a browse button that enable users to select a file to upload to the server.

Create an asp.net web application project. Drag and drop the FileUpload control on the webform. The fileUpload control only allows the user to select the file. To upload the selected file, drag and drop a button control. Change the ID of the button to btnUpload and the Text to Upload File. Also drag and drop a label control, and change the ID of the label to lblMessage. At this stage the HTML of the webform should be as shown below.

```
<asp:FileUpload ID="FileUpload1" runat="server" />
  

<asp:Button ID="btnUpload" runat="server" Text="Upload File"
  onclick="btnUpload_Click" />
<br />
<asp:Label ID="lblMessage" Font-Bold="true" runat="server">
</asp:Label>
```

Right click on the web application project and add a folder with name Uploads. This folder is going to store all the uploaded files.

Copy and paste the following code in btnUpload\_Click() event handler

```
// If the user has selected a file
if (FileUpload1.HasFile)
{
    // Get the file extension
    string fileExtension = System.IO.Path.GetExtension(FileUpload1.FileName);

    if (fileExtension.ToLower() != ".doc" && fileExtension.ToUpper() != ".docx")
    {
        lblMessage.ForeColor = System.Drawing.Color.Red;
        lblMessage.Text = "Only files with .doc and .docx extension are allowed";
    }
    else
    {
        // Get the file size
        int fileSize = FileUpload1.PostedFile.ContentLength;
        // If file size is greater than 2 MB
        if (fileSize > 2097152)
        {
            lblMessage.ForeColor = System.Drawing.Color.Red;
            lblMessage.Text = "File size cannot be greater than 2 MB";
        }
        else
        {
            // Upload the file
            FileUpload1.SaveAs(Server.MapPath("~/Uploads/" + FileUpload1.FileName));
            lblMessage.ForeColor = System.Drawing.Color.Green;
            lblMessage.Text = "File uploaded successfully";
        }
    }
}
```

```
    }  
}  
else  
{  
    lblMessage.ForeColor = System.Drawing.Color.Red;  
    lblMessage.Text = "Please select a file";  
}
```

## Adrotator control in asp.net

We will learn about using the asp.net adrotator control. Adrotator control is used to display random ads. The ads information can be stored in an xml file or in a database table. We will discuss about using an XML file.

### XML file attributes:

**ImageUrl** - The URL of the image to display

**NavigateUrl** - The URL to navigate to, when the ad is clicked

**AlternateText** - The text to use if the image is missing

**Keyword** - Used by the adrotator control to filter ads

**Impressions** - A numeric value (a weighting number) that indicates the likelihood of how often the ad is displayed.

Create an asp.net web application project, and add an XML file. Name the XML file as AdsData.xml. Copy and paste the following in the XML file.

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>~/Images/Google.png</ImageUrl>
    <NavigateUrl>http://google.com</NavigateUrl>
    <AlternateText>Please visit http://www.Google.com</AlternateText>
    <Impressions>10</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>~/Images/Pragim.png</ImageUrl>
    <NavigateUrl>http://pragimtech.com</NavigateUrl>
    <AlternateText>Please visit http://www.pragimtech.com</AlternateText>
    <Impressions>20</Impressions>
  </Ad>
  <Ad>
    <ImageUrl>~/Images/Youtube.png</ImageUrl>
    <NavigateUrl>http://Youtube.com</NavigateUrl>
    <AlternateText>Please visit http://www.Youtube.com</AlternateText>
    <Impressions>40</Impressions>
  </Ad>
</Advertisements>
```

Create an Images folder in the project, and add the following images.





YOUTUBE.COM

Drag and Drop the AdRotator control on the webform. Set AdvertisementFile="~/AdsData.xml".

```
<asp:AdRotator AdvertisementFile="~/AdsData.xml" ID="AdRotator1" runat="server" />
```

To open the target web page in a separate browser window, set Target="\_blank"

Use KeyWord attribute to filter ads.

The KeywordFilter and AdvertisementFile properties can be changed at runtime also. Changing the KeywordFilter at runtime could be very useful. For example, when the AdRotator control is on a master page, and if you want to change the KeywordFilter on each content page based on the keyword density, so that, only the ads targeting the page content can be displayed.

## Asp.net calendar control

We will learn about the asp.net calendar control. Any time you want your users of the application, to provide a date, it is better to provide a calendar control from which they can select the date. In this session we will see how to do it.

Drag and drop a TextBox, ImageButton and a Calendar control on the webform. Create an Image folder and add the following Calendar.png to the Images folder.



Set the ImageUrl property of the image button to Calendar.png

```
ImageUrl="~/Images/Calendar.png"
```

### HTML of the ASPX page

```
<asp:TextBox ID="TextBox1" runat="server" Width="115px"></asp:TextBox>
<asp:ImageButton ID="ImageButton1" runat="server"
    ImageUrl="~/Images/Calendar.png" onclick="ImageButton1_Click" />
<asp:Calendar ID="Calendar1" runat="server" ondayrender="Calendar1_DayRender"
    onselectionchanged="Calendar1_SelectionChanged"></asp:Calendar>
```

### Code-Behind page code

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Calendar1.Visible = false;
    }
}
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    if (Calendar1.Visible)
    {
        Calendar1.Visible = false;
    }
    else
    {
        Calendar1.Visible = true;
    }
}
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    TextBox1.Text = Calendar1.SelectedDate.ToShortDateString();
    Calendar1.Visible = false;
}
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    if (e.Day.IsWeekend || e.Day.IsOtherMonth)
    {

```



```

        e.Day.IsSelectable = false;
        e.Cell.BackColor = System.Drawing.Color.LightGray;
    }
}

```

## Date Conversion methods

```

Response.Write("ToString() - " + DateTime.Now.ToString() + "<br/>");
Response.Write("ToLongDateString() - " + DateTime.Now.ToLongDateString() + "<br/>");
Response.Write("ToShortDateString() - " + DateTime.Now.ToShortDateString() + "<br/>");
Response.Write("ToLongTimeString() - " + DateTime.Now.ToLongTimeString() + "<br/>");
Response.Write("ToShortTimeString() - " + DateTime.Now.ToShortTimeString() + "<br/>");

```

## Output:

```

ToString() - 17/11/2012 19:32:49
ToLongDateString() - 17 November 2012
ToShortDateString() - 17/11/2012
ToLongTimeString() - 19:32:49
ToShortTimeString() - 19:32

```

## DateTime format strings

```

Response.Write("d - " + DateTime.Now.ToString("d") + "<br/>");
Response.Write("D - " + DateTime.Now.ToString("D") + "<br/>");
Response.Write("dd/MM/yyyy - " + DateTime.Now.ToString("dd/MM/yyyy") + "<br/>");
Response.Write("dd/MMMM/yyyy - " + DateTime.Now.ToString("dd/MMMM/yyyy") + "<br/>");
Response.Write("dd/MMMM/yy - " + DateTime.Now.ToString("dd/MMMM/yy") + "<br/>");
Response.Write("MM/dd/yy - " + DateTime.Now.ToString("MM/dd/yy") + "<br/>");

```

## Output

```

d - 17/11/2012
D - 17 November 2012
dd/MM/yyyy - 17/11/2012
dd/MMMM/yyyy - 17/November/2012
dd/MMMM/yy - 17/November/12
MM/dd/yy - 11/17/12

```

## Asp.net calendar control properties and events

### Useful Properties of the Calendar control:

**Caption** - This is a string read/write property.

**CaptionAlign** - Used to align the caption.

**DayHeaderStyle** - Style properties that can be used to customize the look and feel of the day header in the calendar

**DayNameFormat** - Can be Full, Short, FirstLetter, FirstTwoLetters, Shortest

**DayStyle** - Style properties that can be used to customize the look and feel of the day in the calendar

**FirstDayOfWeek** - Which day of the week is displayed first

**NextPrevFormat** - Can be ShortMonth, FullMonth, CustomText

**NextMonthText** - The text to use for the next month button.

**PrevMonthText** - The text to use for the previous month button.

**SelectionMode** - Can be Day, DayWeek, DayWeekMonth. Determines if Days, Weeks and Months are selectable.

If the SelectionMode is set to Day, then the user can select only one day. In this case to retrieve the selected date, we use SelectedDate property of the calendar as shown below.

```
Response.Write(Calendar1.SelectedDate.ToShortDateString());
```

However, if the SelectionMode is set to DayWeek or DayWeekMonth. In this case if you want to retrieve all the selected dates within the selected week or month, then SelectedDates property can be used as shown below. Using SelectedDate, property returns only the first selected date of the week or month.

```
foreach (DateTime selectedDate in Calendar1.SelectedDates)
{
    Response.Write(selectedDate.ToShortDateString() + "<br/>");
}
```

### Events:

SelectionChanged - Fires when the date, week or Month selection is changed, by the user.

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    foreach (DateTime selectedDate in Calendar1.SelectedDates)
    {
        Response.Write(selectedDate.ToShortDateString() + "<br/>");
    }
}
```

DayRender - Fires as a day in the calendar control is being rendered.

```
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    if (!e.Day.IsOtherMonth && e.Day.Date.Day % 2 == 0)
    {
        e.Cell.Text = "x";
        e.Cell.Font.Bold = true;
    }
}
```

```

        e.Cell.ForeColor = System.Drawing.Color.White;
        e.Cell.BackColor = System.Drawing.Color.Red;
        e.Cell.ToolTip = "Booked";
    }
    else
    {
        e.Cell.ToolTip = "Available";
    }
}

```

VisibleMonthChanged - Fires when the visible month is changed by the user

```

protected void Calendar1_VisibleMonthChanged(object sender, MonthChangedEventArgs e)
{
    Response.Write("Month changed from ");
    Response.Write(GetMonthName(e.PreviousDate.Month));
    Response.Write(" to ");
    Response.Write(GetMonthName(e.NewDate.Month));
}
private string GetMonthName(int MonthNumber)
{
    switch (MonthNumber)
    {
        case 1:
            return "Jan";
        case 2:
            return "Feb";
        case 3:
            return "Mar";
        case 4:
            return "Apr";
        case 5:
            return "May";
        case 6:
            return "Jun";
        case 7:
            return "Jul";
        case 8:
            return "Aug";
        case 9:
            return "Sep";
        case 10:
            return "Oct";
        case 11:
            return "Nov";
        case 12:
            return "Dec";
        default:
            return "Invalid Month";
    }
}

```

## Hidden field in asp.net

The HiddenField control is used to store a value that needs to be persisted across posts to the server, but you don't want the control or its value visible to the user. For example, when editing and updating an employee record, we don't want the user to see the EmployeeId. So, we will store the EmployeeId in a HiddenField, so that it can then be used on the server to update the correct employee's record.

### SQL Script

Create Table tblEmployees

```
(  
  Id int Primary Key,  
  Name nvarchar(50),  
  Gender nvarchar(10),  
  DeptName nvarchar(10)  
)
```

Insert into tblEmployees values(201, 'Mark', 'Male', 'IT')

Insert into tblEmployees values(202, 'Steve', 'Male', 'Payroll')

Insert into tblEmployees values(203, 'John', 'Male', 'HR')

### HTML of the ASPX Page

```
<asp:HiddenField ID="HiddenField1" runat="server" />  
<table>  
  <tr>  
    <td>Name:</td>  
    <td>  
      <asp:TextBox ID="txtName" runat="server"></asp:TextBox>  
    </td>  
  </tr>  
  <tr>  
    <td>Gender:</td>  
    <td>  
      <asp:TextBox ID="txtGender" runat="server"></asp:TextBox>  
    </td>  
  </tr>  
  <tr>  
    <td>Department:</td>  
    <td>  
      <asp:TextBox ID="txtDept" runat="server"></asp:TextBox>  
    </td>  
  </tr>  
</table>  
<asp:Button ID="Button1" runat="server" Text="Update Employee"  
  onclick="Button1_Click" />&nbsp; <br>  
<asp:Button ID="Button2" runat="server" onclick="Button2_Click"  
  Text="Load Employee" />
```

## Code-Behind code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        LoadEmployee();
    }
}

private void LoadEmployee()
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(CS))
    {
        string sqlQuery = "Select Id, Name, Gender, DeptName from tblEmployees where Id=202";
        SqlCommand cmd = new SqlCommand(sqlQuery, con);
        con.Open();
        using (SqlDataReader rdr = cmd.ExecuteReader())
        {
            while (rdr.Read())
            {
                txtName.Text = rdr["Name"].ToString();
                txtGender.Text = rdr["Gender"].ToString();
                txtDept.Text = rdr["DeptName"].ToString();
                HiddenField1.Value = rdr["Id"].ToString();
            }
        }
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(CS))
    {
        string sqlQuery = "Update tblEmployees set Name=@Name, Gender=@Gender,
DeptName=@ DeptName where Id=@Id";
        SqlCommand cmd = new SqlCommand(sqlQuery, con);

        cmd.Parameters.AddWithValue("@Name", txtName.Text);
        cmd.Parameters.AddWithValue("@Gender", txtGender.Text);
        cmd.Parameters.AddWithValue("@DeptName", txtDept.Text);
        cmd.Parameters.AddWithValue("@Id", HiddenField1.Value);
        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();

        txtName.Text = "";
        txtDept.Text = "";
        txtGender.Text = "";
    }
}
```

```

}

protected void Button2_Click(object sender, EventArgs e)
{
    LoadEmployee();
}

```

### **HiddenField:**

1. Value property of the HiddenField is used to Get or set the value.
2. The value is stored as string
3. ViewState uses HiddenField to maintain state across postback
4. HiddenField is rendered as an <input type= "hidden"/> element

### **Alternatives for HiddenField:**

View state, QueryStrings, session state, and cookies can be used as an alternative for HiddenField. Session state and cookies will be accessible from other pages as well, and will be available until their timeout has reached. Whereas ViewState and HiddenField data, is available only on that page and the data is lost when you navigate away from the page.

### **Advantages of HiddenField:**

HiddenField data is lost when you navigate away from the page. Doesn't require any explicit cleanup task.

HiddenField is accessible to client-side scripts

```

<script type="text/javascript">
    function GetHiddenFieldValue()
    {
        alert(document.getElementById('HiddenField1').value);
    }
</script>

```

### **Disadvantage of HiddenField:**

Hidden field data can be seen, by viewing the page source. Never, use HiddenField to store confidential data

## Multiview control in asp.net

As the name states, a multiview is made up of multiple view controls, and each view control in turn can have controls inside it.

The HTML below shows a multiview, with 3 views:

```
<asp:MultiView ID="MultiView1" runat="server">
  <asp:View ID="View1" runat="server">
  </asp:View>
  <asp:View ID="View2" runat="server">
  </asp:View>
  <asp:View ID="View3" runat="server">
  </asp:View>
</asp:MultiView>
```

Let's create a simple example, where we want to capture employee information on a step by step basis.

1. First capture Employee Personal details
2. Next capture Employee contact details
3. Show summary for confirmation. Upon confirmation, save the data to a database table

### HTML of the aspx page

```
<div style="font-family: Arial">
  <asp:MultiView ID="multiViewEmployee" runat="server">
    <asp:View ID="viewPersonalDetails" runat="server">
      <table style="border:1px solid black">
        <tr>
          <td colspan="2">
            <h2>Step 1 - Personal Details</h2>
          </td>
        </tr>
        <tr>
          <td>First Name</td>
          <td>
            <asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>
          </td>
        </tr>
        <tr>
          <td>Last Name</td>
          <td>
            <asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>
          </td>
        </tr>
        <tr>
          <td>Gender</td>
          <td>
            <asp:DropDownList ID="ddlGender" runat="server">
              <asp:ListItem Text="Male" Value="Male"></asp:ListItem>
              <asp:ListItem Text="Female" Value="Female"></asp:ListItem>
            </asp:DropDownList>
          </td>
        </tr>
      </table>
    </asp:View>
  </asp:MultiView>
</div>
```

```

        </asp:DropDownList>
    </td>
</tr>
<tr>
    <td colspan="2" style="text-align:right">
        <asp:Button ID="btnGoToStep2" runat="server"
            Text="Step 2 >>" onclick="btnGoToStep2_Click" />
    </td>
</tr>
</table>
</asp:View>
<asp:View ID="viewContactDetails" runat="server">
    <table style="border:1px solid black">
        <tr>
            <td colspan="2">
                <h2>Step 2 - Contact Details</h2>
            </td>
        </tr>
        <tr>
            <td>Email Address</td>
            <td>
                <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>Mobile Number</td>
            <td>
                <asp:TextBox ID="txtMobile" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>
                <asp:Button ID="btnBackToStep1" runat="server" Text="<< Step 1 "
                    onclick="btnBackToStep1_Click" />
            </td>
            <td style="text-align:right">
                <asp:Button ID="btnGoToStep3" runat="server" Text="Step 3 >>"
                    onclick="btnGoToStep3_Click" />
            </td>
        </tr>
    </table>
</asp:View>
<asp:View ID="viewSummary" runat="server">
    <table style="border:1px solid black">
        <tr>
            <td colspan="2"><h2>Step 3 - Summary</h2></td>
        </tr>
        <tr>
            <td colspan="2"><h3>Personal Details</h3></td>
        </tr>
        <tr>
            <td>First Name</td>

```



```

        <td>
            :<asp:Label ID="lblFirstName" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td>Last Name</td>
        <td>
            :<asp:Label ID="lblLastName" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td>Gender</td>
        <td>
            :<asp:Label ID="lblGender" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td colspan="2"><h3>Contact Details</h3></td>
    </tr>
    <tr>
        <td>Email Address</td>
        <td>
            :<asp:Label ID="lblEmail" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td>Mobile Number</td>
        <td>
            :<asp:Label ID="lblMobile" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Button ID="Button1" runat="server" Text="<< Step 2"
            onclick="Button1_Click" />
        </td>
        <td style="text-align:right">
            <asp:Button ID="Button2" runat="server" Text="Submit >>"
            onclick="Button2_Click" />
        </td>
    </tr>
</table>
</asp:View>
</asp:MultiView>
</div>

```

### Code-Behind Page:

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {

```

```

        multiViewEmployee.ActiveViewIndex = 0;
    }
}

protected void btnGoToStep2_Click(object sender, EventArgs e)
{
    multiViewEmployee.ActiveViewIndex = 1;
}

protected void btnBackToStep1_Click(object sender, EventArgs e)
{
    multiViewEmployee.ActiveViewIndex = 0;
}

protected void btnGoToStep3_Click(object sender, EventArgs e)
{
    lblFirstName.Text = txtFirstName.Text;
    lblLastName.Text = txtLastName.Text;
    lblGender.Text = ddlGender.SelectedValue;

    lblEmail.Text = txtEmail.Text;
    lblMobile.Text = txtMobile.Text;

    multiViewEmployee.ActiveViewIndex = 2;
}

protected void Button1_Click(object sender, EventArgs e)
{
    multiViewEmployee.ActiveViewIndex = 1;
}

protected void Button2_Click(object sender, EventArgs e)
{
    // Write ado.net code to save data to a database table
    Response.Redirect("~/Confirmation.aspx");
}

```

ActiveViewIndex property of the Multiview control is used to determine, the view that is visible or active.

This can also be achieved using the wizard control. Another way to achieve this, is by creating multiple webforms and passing data between webforms using:

1. Cookies
2. Query Strings
3. Session variables

## Wizard control in asp.net

Wizard control enables creation of multi-step user interface. Wizard control provides with built-in previous/next functionality.

Let's create a simple example, where we want to capture employee information on a step by step basis.

1. First capture Employee Personal details
2. Next capture Employee contact details
3. Show summary for confirmation. Upon confirmation, save the data to a database table

A wizard is a collection of WizardSteps. The StepType property of WizardStep determines the correct previous/next buttons to show. we will discuss a simple example using wizard control.

### HTML of the aspx page:

```
<div>
  <asp:Wizard ID="Wizard1" runat="server"
    onfinishbuttonclick="Wizard1_FinishButtonClick"
    onnextbuttonclick="Wizard1_NextButtonClick">
    <SideBarStyle HorizontalAlign="Left" VerticalAlign="Top" />
    <WizardSteps>
      <asp:WizardStep ID="WizardStep1" runat="server" StepType="Start" Title="Step 1 -
Personal Details">
        <table style="border: 1px solid black">
          <tr>
            <td>
              First Name
            </td>
            <td>
              <asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>
            </td>
          </tr>
          <tr>
            <td>
              Last Name
            </td>
            <td>
              <asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>
            </td>
          </tr>
          <tr>
            <td>
              Gender
            </td>
            <td>
              <asp:DropDownList ID="ddlGender" runat="server">
                <asp:ListItem Text="Male" Value="Male"></asp:ListItem>
                <asp:ListItem Text="Female" Value="Female"></asp:ListItem>
              </asp:DropDownList>
            </td>
          </tr>
        </table>
      </asp:WizardStep>
    </WizardSteps>
  </asp:Wizard>
</div>
```

```

        </td>
    </tr>
</table>
</asp:WizardStep>
<asp:WizardStep ID="WizardStep3" runat="server" StepType="Step" Title="Step 2 -
Contact Details">
    <table style="border: 1px solid black">
        <tr>
            <td>
                Email Address
            </td>
            <td>
                <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td>
                Mobile Number
            </td>
            <td>
                <asp:TextBox ID="txtMobile" runat="server"></asp:TextBox>
            </td>
        </tr>
    </table>
</asp:WizardStep>
<asp:WizardStep ID="WizardStep2" runat="server" StepType="Finish" Title="Step 3 -
Summary">
    <table style="border: 1px solid black">
        <tr>
            <td colspan="2">
                <h3>
                    Personal Details</h3>
                </td>
        </tr>
        <tr>
            <td>
                First Name
            </td>
            <td>
                <asp:Label ID="lblFirstName" runat="server"></asp:Label>
            </td>
        </tr>
        <tr>
            <td>
                Last Name
            </td>
            <td>
                <asp:Label ID="lblLastName" runat="server"></asp:Label>
            </td>
        </tr>
        <tr>
            <td>

```

```

        Gender
    </td>
    <td>
        <asp:Label ID="lblGender" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td colspan="2">
        <h3>
            Contact Details</h3>
        </td>
    </tr>
    <tr>
    <td>
        Email Address
    </td>
    <td>
        <asp:Label ID="lblEmail" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td>
        Mobile Number
    </td>
    <td>
        <asp:Label ID="lblMobile" runat="server"></asp:Label>
    </td>
</tr>
</table>
</asp:WizardStep>
</WizardSteps>
</asp:Wizard>
</div>

```

#### Code Behind:

```

protected void Wizard1_NextButtonClick(object sender, WizardNavigationEventArgs e)
{
    if (e.NextStepIndex == 2)
    {
        lblFirstName.Text = txtFirstName.Text;
        lblLastName.Text = txtLastName.Text;
        lblGender.Text = ddlGender.SelectedValue;

        lblEmail.Text = txtEmail.Text;
        lblMobile.Text = txtMobile.Text;
    }
}

protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{

```

```
Response.Redirect("~/Confirmation.aspx");  
}
```

## Asp.net Wizard control properties

### Properties:

**ActiveStepIndex** - Used to set or get the ActiveStepIndex of the wizard control

**DisplayCancelButton** - Determines the visibility of the cancel button in the wizard control

**CancelButtonImageUrl** - If the cancel button type is set to Image button. Then set this property to specify the image.

**CancelButtonStyle** - The style properties to customize the cancel button

**CancelButtonText** - If the cancel button type is set to Link or Button, then set this property to specify the Text of the button.

**CancelButtonType** - Use to Specify the type of cancel button. This can be Button, Image or Link.

**CancelDestinationPageUrl** - The destination page to redirect to, when the cancel button is clicked. This can be a page with in the application or an external website.

**DisplaySideBar** - Determines if the wizard sidebar should be displayed or not

**FinishCompleteButtonType** - The button type of the finish step's finish button

**FinishPreviousButtonType** - The button type of the finish step's previous button

**HeaderStyle** - The style properties to customize the wizard header

**HeaderText** - The header text of the wizard control

```
protected void Page_PreRender(object sender, EventArgs e)
{
    if (Wizard1.ActiveStepIndex == 1)
    {
        Wizard1.HeaderText = "Contact Details";
    }
    else if (Wizard1.ActiveStepIndex == 2)
    {
        Wizard1.HeaderText = "Summary";
    }
}
```

**NavigationButtonStyle** - The style properties to customize the wizard navigation buttons

**NavigationStyle** - The style properties to customize the navigation area that holds the navigation buttons

**SideBarButtonStyle** - The style properties to customize the wizard sidebar buttons

**SideBarStyle** - The style properties to customize the wizard sidebar

**StartNextButtonType** - The type of the start step's next button

**StepNextButtonType** - The button type of next step button

**StepPreviousButtonType** - The button type of previous step button

**StepStyle** - The style properties to customize the wizard steps

Note: WizardSteps can be added in the HTML source or using the WizardSteps collection editor.

## Asp.net Wizard control events

Drag and drop a wizard control on the webform. Right click on the wizard control and select the properties. Click on the events icon. This displays all the events of the wizard control.

To generate the event handler method for ActiveStepChanged event, double click on the textbox next to the event. Follow the same process to generate the event handler methods for the rest of the events of the wizard control.

### HTML of the aspx page:

```
<div style="font-family: Arial">
  <asp:Wizard ID="Wizard1" runat="server"
    onactivestepchanged="Wizard1_ActiveStepChanged"
    oncancebuttonclick="Wizard1_CancelButtonClick"
    onnextbuttonclick="Wizard1_NextButtonClick"
    onfinishbuttonclick="Wizard1_FinishButtonClick"
    onpreviousbuttonclick="Wizard1_PreviousButtonClick"
    onsidebarbuttonclick="Wizard1_SideBarButtonClick">
    <SideBarStyle VerticalAlign="Top" />
    <WizardSteps>
      <asp:WizardStep runat="server" title="Step 1">
        <asp:CheckBox ID="chkBoxCancel" Text="Cancel Navigation" runat="server" />
      </asp:WizardStep>
      <asp:WizardStep runat="server" title="Step 2">
      </asp:WizardStep>
      <asp:WizardStep runat="server" title="Step 3">
      </asp:WizardStep>
    </WizardSteps>
  </asp:Wizard>
</div>
```

### Code-Behind page code:

```
// ActiveStepChanged - Fires when the active step of the index is changed.
protected void Wizard1_ActiveStepChanged(object sender, EventArgs e)
{
    Response.Write("Active Step Changed to " + Wizard1.ActiveStepIndex.ToString() + "<br/>");
}
// CancelButtonClick - Fires when the cancel button of the wizard control is clicked.
// To display the cancel button, set DisplayCancelButton=True.
protected void Wizard1_CancelButtonClick(object sender, EventArgs e)
{
    Response.Redirect("Cancel Button Clicked");
}
// NextButtonClick - Fires when the next button of the wizard control is clicked.
protected void Wizard1_NextButtonClick(object sender, WizardNavigationEventArgs e)
{
    Response.Write("Current Step Index = " + e.CurrentStepIndex.ToString() + "<br/>");
    Response.Write("Next Step Index = " + e.NextStepIndex.ToString() + "<br/>");
    if (chkBoxCancel.Checked)
    {

```



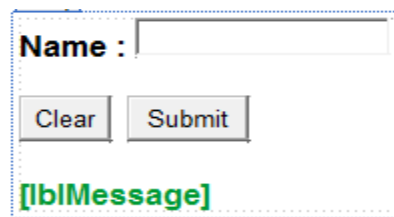
```

        Response.Write("Navigation to next step will be cancelled");
        e.Cancel = true;
    }
}
// FinishButtonClick - Fires when the finish button is clicked
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    Response.Write("Finish button clicked <br/>");
    Response.Write("Current Step Index = " + e.CurrentStepIndex.ToString() + "<br/>");
    Response.Write("Next Step Index = " + e.NextStepIndex.ToString());
}
// PreviousButtonClick - Fires when the previous button is clicked
protected void Wizard1_PreviousButtonClick(object sender, WizardNavigationEventArgs e)
{
    Response.Write("Previous button clicked<br/>");
}
// SideBarButtonClick - Fires when the sidebar button is clicked
protected void Wizard1_SideBarButtonClick(object sender, WizardNavigationEventArgs e)
{
    Response.Write("Sidebar button clicked<br/>");
}

```

## UseSubmitBehavior property of the Button control

Design a webform with a TextBox, Label and 2 Button controls as shown in the image below.

A screenshot of a web form. It features a label 'Name : ' followed by a text input field. Below the input field are two buttons: 'Clear' and 'Submit'. At the bottom, there is a green label with the text '[lblMessage]'.

For your convenience, I have included the HTML of the aspx page.

```
<div style="font-family: Arial">
    <strong>Name : </strong>
    <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
    <br />
    <br />
    <asp:Button ID="btnClear" runat="server" onclick="btnClear_Click" Text="Clear"/>
    &nbsp;
    <asp:Button ID="btnSubmit" runat="server" onclick="btnSubmit_Click" Text="Submit" />
    <br />
    <br />
    <asp:Label ID="lblMessage" runat="server" Font-Bold="True"
ForeColor="#009933"></asp:Label>
</div>
```

Double click the Clear and Submit buttons, to generate the event handlers, and paste the following code in the code behind page.

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    lblMessage.Text = "You entered: " + txtName.Text;
}

protected void btnClear_Click(object sender, EventArgs e)
{
    txtName.Text = "";
}
```

Now, run the web application project. Enter a name in the textbox and hit the enter key on the keyboard. Notice that the cancel button has the submit behaviour.

In the HTML of the aspx page, set UseSubmitBehavior="false" for the clear button.

```
<asp:Button ID="btnClear" UseSubmitBehavior="false" runat="server"
onclick="btnClear_Click" Text="Clear"/>
```

Now, run the web application project again. Enter a name in the textbox and hit the enter key on the keyboard. Notice that the submit button has the submit behaviour, as expected.

The UseSubmitBehavior property specifies if the Button control uses the browser's built-in submit

function or the ASP.NET postback mechanism.

This property is TRUE by default. When set to FALSE, ASP.NET adds a client-side script to post the form. To view the client side script added by the ASP.NET, right click on the browser and view source.

## Asp.net wizard control templates

We will look at the advanced features of the wizard control like :

1. Set focus to the first control in the wizard step when the page loads, so that the user can start typing into the textbox directly.
2. Attach javascript to the Next, Previous and Finish buttons, to display a confirmation dialog box before the user moves to the next step.
3. Setting UseSubmitBehavior="true" for the Previous button in the wizard control.

In the HTML below, we have a wizard control with 3 steps. Each step has a TextBox control.

```
<asp:Wizard ID="Wizard1" runat="server">
  <WizardSteps>
    <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
      <asp:TextBox ID="Step1TextBox" runat="server"></asp:TextBox>
    </asp:WizardStep>
    <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2">
      <asp:TextBox ID="Step2TextBox" runat="server"></asp:TextBox>
    </asp:WizardStep>
    <asp:WizardStep ID="WizardStep3" runat="server" Title="Step 3">
      <asp:TextBox ID="Step3TextBox" runat="server"></asp:TextBox>
    </asp:WizardStep>
  </WizardSteps>
</asp:Wizard>
```

The following code, sets the focus to the correct textbox, based on the ActiveStepIndex. Make sure you have this code in the Page\_PreRender event. Copying this code in the Page\_Load() event will not work correctly. This is because the ActiveStepIndex is changed on the Button click event which happens after the Page\_Load() event. As the Page\_PreRender() events occurs after the Button\_Click event, the code works correctly as expected.

```
protected void Page_PreRender(object sender, EventArgs e)
{
    if (Wizard1.ActiveStepIndex == 0)
    {
        Step1TextBox.Focus();
    }
    else if (Wizard1.ActiveStepIndex == 1)
    {
        Step2TextBox.Focus();
    }
    else if (Wizard1.ActiveStepIndex == 2)
    {
        Step3TextBox.Focus();
    }
}
```

To attach javascript to the buttons in the navigation bar(next, previous, Finish), we need to use Navigation Templates. By default, the wizard control generates these buttons automatically. To make the wizard control use navigation templates and attach javascript

1. Right click on the wizard control and select "Show smart tag"
2. Click on "Convert To Start Navigation Template".
3. Now in the HTML source, specify the javascript that needs to be executed in response to the OnClientClick event.

```
<asp:Wizard ID="Wizard1" runat="server">
  <StartNavigationTemplate>
    <asp:Button ID="StartNextButton" runat="server" CommandName="MoveNext" Text="Next"
      OnClientClick="return confirm('Are you sure you want to go to next step');" />
  </StartNavigationTemplate>
  <WizardSteps>
    <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
      <asp:TextBox ID="Step1TextBox" runat="server"></asp:TextBox>
    </asp:WizardStep>
    <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2">
      <asp:TextBox ID="Step2TextBox" runat="server"></asp:TextBox>
    </asp:WizardStep>
    <asp:WizardStep ID="WizardStep3" runat="server" Title="Step 3">
      <asp:TextBox ID="Step3TextBox" runat="server"></asp:TextBox>
    </asp:WizardStep>
  </WizardSteps>
</asp:Wizard>
```

Use StepNavigationTemplate, to attach javascript to the Next and Previous buttons on all wizard steps, where StepType="Step"

Use FinishNavigationTemplate, to attach javascript to the Finish button on the last step of the wizard control.

It is also possible to add javascript in code. To add the javascript using code for the Next button, in a wizard step, where StepType="Step"

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Button btnNext =
        (Button)Wizard1.FindControl("StepNavigationTemplateContainerID").FindControl("StepNextButton");
        btnNext.Attributes.Add("onclick", "return confirm('Are you sure you want to move to the next step');");
    }
}
```

To make the next button of the wizard control the default button, set UseSubmitBehavior=False for the Previous button. So that, when the user hits the Enter key, after entering the required data, the user moves to the next step and not the previous step.

## Literal control in asp.net

1. In many ways a Literal control is similar to a Label control. Both of these controls are used to display Text on a webform. The Text property can be set in the HTML or in the code-behind.
2. Label control wraps the text in a span tag when rendered. Any style that is applied to the Label control, will be rendered using the style property of the span tag.

### For example, the following HTML

```
<asp:Label ID="Label1" runat="server" Text="Lable Text"
ForeColor="Red" Font-Bold="true" ></asp:Labe>
```

### Will be rendered as

```
<span id="Label1" style="color:Red;font-weight:bold;">Lable Text</span>
```

3. A literal control, doesn't output any surrounding tags. The Text is displayed as is.

### For example, the following HTML

```
<asp:Literal ID="Literal1" runat="server"
Text="Literal Control Text"></asp:Literal>
```

### will be rendered as

Literal Control Text

4. If you want to apply any styles to a literal control, include them in the Text of the literal control. For example, the following HTML sets the font to red color and bold.

```
<asp:Literal ID="Literal1" runat="server"
Text="<b><font color='Red'>Literal Control Text</font></b>">
</asp:Literal>
```

The above HTML will be rendered as

```
<b><font color='Red'>Literal Control Text</font></b>
```

5. So, if you just want the text to be displayed without any styles, then use Literal control, else use Label control.

6. By default, both the Label and Literal Control's does not encode the text they display. For example, the following HTML displays a javascript alert

```
<asp:Label ID="Label" runat="server"
Text="<script>alert('Lable Text');</script>">
</asp:Label>
<br />
<asp:Literal ID="Literal1" runat="server"
Text="<script>alert('Literal Text');</script>">
</asp:Literal>
```

and will be rendered as

```
<span id="Label"><script>alert('Lable Text');</script></span>
<br />
<script>alert('Literal Text');</script>
```

7. To HTML encode the Label Text, Server.HtmlEncode() method can be used, and for Literal control, Mode property can be used.

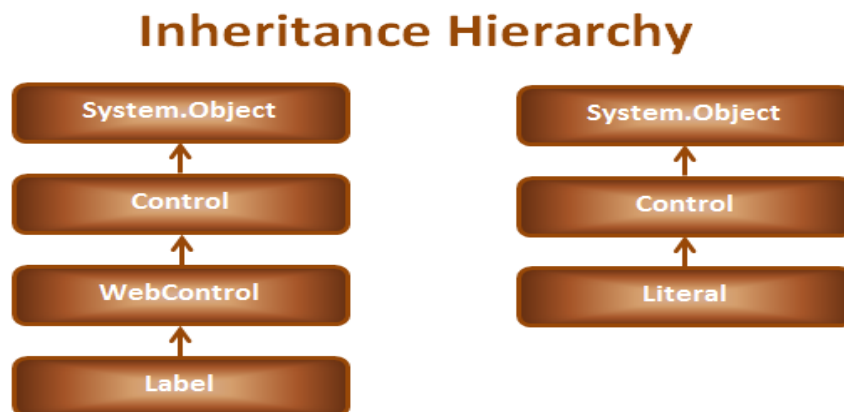
```
<asp:Label ID="Label1" runat="server">
<%=Server.HtmlEncode("<script>alert('Lable Text');</script>")%>
</asp:Label>
<br />
<asp:Literal ID="Literal1" runat="server"
Text="<script>alert('Literal Text');</script>"
Mode="Encode"></asp:Literal>
```

The above HTML will be rendered as

```
<span id="Label1">&lt;script&gt;alert(&#39;Lable Text&#39;);&lt;/script&gt;</span>
<br />
&lt;script&gt;alert(&#39;Literal Text&#39;);&lt;/script&gt;
```

8. Literal control is a light weight control, when compared with the Label control.

9. The inheritance hierarchy for Literal control class is (Object => Control => Literal), where as for the Lable control, the hierarchy is (Object => Control => WebControl => Label)



## Asp.net panel control

The panel control is used as a container for other controls. A panel control is very handy, when you want to group controls, and then show or hide, all the controls in the group. Panel control, is also very useful, when adding controls to the webform dynamically.

We will discuss about, using the Panel control to group controls, and then toggle their visibility, using the Panel control's Visible property.

The following webform is used by both, an Admin and Non-Admin user. When the Admin user is selected from the dropdownlist, we want to show the controls that are relevant to the Admin user. When the Non-Admin user is selected, only the Non-Admin specific content and controls should be shown.

HTML of the ASPX page. At the moment we are not using Panel control.

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="true"
    onselectedindexchanged="DropDownList1_SelectedIndexChanged">
    <asp:ListItem Text="Select User" Value="-1"></asp:ListItem>
    <asp:ListItem Text="Admin" Value="Admin"></asp:ListItem>
    <asp:ListItem Text="Non-Admin" Value="Non-Admin"></asp:ListItem>
</asp:DropDownList>
<table>
    <tr>
        <td colspan="2">
            <asp:Label ID="AdminGreeting" runat="server" Font-Size="XX-Large"
                Text="You are logged in as an administrator">
            </asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="AdminNameLabel" runat="server" Text="Admin Name:">
            </asp:Label>
        </td>
        <td>
            <asp:TextBox ID="AdminNameTextBox" runat="server" Text="Tom">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="AdminRegionLabel" runat="server" Text="Admin Region:">
            </asp:Label>
        </td>
        <td>
            <asp:TextBox ID="AdminRegionTextBox" runat="server" Text="Asia">
            </asp:TextBox>
        </td>
    </tr>
</table>
```



```

<td>
    <asp:Label ID="AdminActionsLabel" runat="server" Text="Actions:">
    </asp:Label>
</td>
<td>
    <asp:TextBox ID="AdminActionsTextBox" runat="server" Font-Size="Medium"
TextMode="MultiLine"

        Text="There are 4 user queries to be answered by the end of Dcemeber 25th 2013."
        Font-Bold="True" ></asp:TextBox>
</td>
</tr>
</table>
<table>
<tr>
<td colspan="2">
    <asp:Label ID="NonAdminGreeting" runat="server" Font-Size="XX-Large"
    Text="Welcome Tom!">
    </asp:Label>
</td>
</tr>
<tr>
<td>
    <asp:Label ID="NonAdminNameLabel" runat="server" Text="User Name:">
    </asp:Label>
</td>
<td>
    <asp:TextBox ID="NonAdminNameTextBox" runat="server" Text="Mike">
    </asp:TextBox>
</td>
</tr>
<tr>
<td>
    <asp:Label ID="NonAdminRegionLabel" runat="server" Text="User Region:">
    </asp:Label>
</td>
<td>
    <asp:TextBox ID="NonAdminRegionTextBox" runat="server" Text="United Kingdom">
    </asp:TextBox>
</td>
</tr>
<tr>
<td>
    <asp:Label ID="NonAdminCityLabel" runat="server" Text="City:">
    </asp:Label>
</td>
<td>
    <asp:TextBox ID="NonAdminCityTextBox" runat="server" Text="London">
    </asp:TextBox>
</td>
</tr>
</table>

```

Code-Behind code. Since we are not using the panel control, each control's visible property need to be changed depending on the selection in the dropdownlist.

```
protected void Page_Load(object sender, EventArgs e)
{
    // When the page first loads, hide all admin and non admin controls
    if (!IsPostBack)
    {
        HideAdminControls();
        HideNonAdminControls();
    }
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (DropDownList1.SelectedValue == "Admin")
    {
        ShowAdminControls();
        HideNonAdminControls();
    }
    else if (DropDownList1.SelectedValue == "Non-Admin")
    {
        ShowNonAdminControls();
        HideAdminControls();
    }
    else
    {
        HideAdminControls();
        HideNonAdminControls();
    }
}

private void HideAdminControls()
{
    AdminGreeting.Visible = false;
    AdminNameLabel.Visible = false;
    AdminNameTextBox.Visible = false;
    AdminRegionLabel.Visible = false;
    AdminRegionTextBox.Visible = false;
    AdminActionsLabel.Visible = false;
    AdminActionsTextBox.Visible = false;
}

private void ShowAdminControls()
{
    AdminGreeting.Visible = true;
    AdminNameLabel.Visible = true;
    AdminNameTextBox.Visible = true;
    AdminRegionLabel.Visible = true;
    AdminRegionTextBox.Visible = true;
    AdminActionsLabel.Visible = true;
    AdminActionsTextBox.Visible = true;
}
```

```

private void HideNonAdminControls()
{
    NonAdminGreeting.Visible = false;
    NonAdminNameLabel.Visible = false;
    NonAdminNameTextBox.Visible = false;
    NonAdminRegionLabel.Visible = false;
    NonAdminRegionTextBox.Visible = false;
    NonAdminCityLabel.Visible = false;
    NonAdminCityTextBox.Visible = false;
}
private void ShowNonAdminControls()
{
    NonAdminGreeting.Visible = true;
    NonAdminNameLabel.Visible = true;
    NonAdminNameTextBox.Visible = true;
    NonAdminRegionLabel.Visible = true;
    NonAdminRegionTextBox.Visible = true;
    NonAdminCityLabel.Visible = true;
    NonAdminCityTextBox.Visible = true;
}

```

HTML of the ASPX page. The panel control is used to group the controls.

```

<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="true"
    onselectedindexchanged="DropDownList1_SelectedIndexChanged">
    <asp:ListItem Text="Select User" Value="-1"></asp:ListItem>
    <asp:ListItem Text="Admin" Value="Admin"></asp:ListItem>
    <asp:ListItem Text="Non-Admin" Value="Non-Admin"></asp:ListItem>
</asp:DropDownList>
<asp:Panel ID="AdminPanel" runat="server">
<table>
    <tr>
        <td colspan="2">
            <asp:Label ID="AdminGreeting" runat="server" Font-Size="XX-Large"
                Text="You are logged in as an administrator">
            </asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="AdminNameLabel" runat="server" Text="Admin Name:">
            </asp:Label>
        </td>
        <td>
            <asp:TextBox ID="AdminNameTextBox" runat="server" Text="Tom">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="AdminRegionLabel" runat="server" Text="Admin Region:">
            </asp:Label>

```

```

        </td>
        <td>
            <asp:TextBox ID="AdminRegionTextBox" runat="server" Text="Asia">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="AdminActionsLabel" runat="server" Text="Actions:">
            </asp:Label>
        </td>
        <td>
            <asp:TextBox ID="AdminActionsTextBox" runat="server" Font-Size="Medium"
TextMode="MultiLine"

                Text="There are 4 user queries to be answered by the end of Dcemeber 25th 2013."
                Font-Bold="True" ></asp:TextBox>
        </td>
    </tr>
</table>
</asp:Panel>
<asp:Panel ID="NonAdminPanel" runat="server">
<table>
    <tr>
        <td colspan="2">
            <asp:Label ID="NonAdminGreeting" runat="server" Font-Size="XX-Large"
                Text="Welcome Tom!">
            </asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="NonAdminNameLabel" runat="server" Text="User Name:">
            </asp:Label>
        </td>
        <td>
            <asp:TextBox ID="NonAdminNameTextBox" runat="server" Text="Mike">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <asp:Label ID="NonAdminRegionLabel" runat="server" Text="User Region:">
            </asp:Label>
        </td>
        <td>
            <asp:TextBox ID="NonAdminRegionTextBox" runat="server" Text="United Kingdom">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>

```

```

        <asp:Label ID="NonAdminCityLabel" runat="server" Text="City:">
        </asp:Label>
    </td>
    <td>
        <asp:TextBox ID="NonAdminCityTextBox" runat="server" Text="London">
        </asp:TextBox>
    </td>
</tr>
</table>
</asp:Panel>

```

Code-Behind code:

```

protected void Page_Load(object sender, EventArgs e)
{
    // When the page first loads, hide admin and non admin panels
    if (!IsPostBack)
    {
        AdminPanel.Visible = false;
        NonAdminPanel.Visible = false;
    }
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (DropDownList1.SelectedValue == "Admin")
    {
        AdminPanel.Visible = true;
        NonAdminPanel.Visible = false;
    }
    else if (DropDownList1.SelectedValue == "Non-Admin")
    {
        AdminPanel.Visible = false;
        NonAdminPanel.Visible = true;
    }
    else
    {
        AdminPanel.Visible = false;
        NonAdminPanel.Visible = false;
    }
}

```

## Creating controls dynamically using asp.net panel control

The panel control is used as a container for other controls. A panel control is very handy, when you want to group controls, and then show or hide, all the controls in the group.

Panel control, is also very useful, when adding controls to the webform dynamically. We will discuss this in this session.

**Control Type** ☐ Label ☒ TextBox ☒ Button **How Many**

**Label Controls**

**TextBox Controls**

TextBox - 1    TextBox - 2    TextBox - 3

**Button Controls**

Button - 1    Button - 2    Button - 3

HTML of the ASPX page

```
<div style="font-family: Arial">
  <table>
    <tr>
      <td><b>Control Type</b></td>
      <td>
        <asp:CheckBoxList ID="chkBoxListControlType" runat="server"
RepeatDirection="Horizontal">
          <asp:ListItem Text="Label" Value="Label"></asp:ListItem>
          <asp:ListItem Text="TextBox" Value="TextBox"></asp:ListItem>
          <asp:ListItem Text="Button" Value="Button"></asp:ListItem>
        </asp:CheckBoxList>
      </td>
      <td><b>How Many</b></td>
      <td>
        <asp:TextBox ID="txtControlsCount" runat="server" Width="40px"></asp:TextBox>
      </td>
      <td>
        <asp:Button ID="btnGenerateControl" runat="server" Text="Generate Controls"
onclick="btnGenerateControl_Click" />
      </td>
    </tr>
    <tr>
      <td colspan="5">
        <h3>Label Controls</h3>
      </td>
    </tr>
    <tr>
      <td colspan="5" id="tdLabels" runat="server">
        <asp:Panel ID="pnlLabels" runat="server">
          </asp:Panel>
        </td>
      </tr>
  </table>
</div>
```

```

        <%--<asp:PlaceHolder ID="phLabels" runat="server">
        </asp:PlaceHolder>--%>
    </td>
</tr>
<tr>
    <td colspan="5">
        <h3>TextBox Controls</h3>
    </td>
</tr>
<tr>
    <td colspan="5" id="tdTextBoxes" runat="server">
        <asp:Panel ID="pnlTextBoxes" runat="server">
            </asp:Panel>
            <%--<asp:PlaceHolder ID="phTextBoxes" runat="server">
            </asp:PlaceHolder>--%>
        </td>
</tr>
<tr>
    <td colspan="5">
        <h3>Button Controls</h3>
    </td>
</tr>
<tr>
    <td colspan="5" id="tdButtons" runat="server">
        <asp:Panel ID="pnlButtons" runat="server">
            </asp:Panel>
            <%--<asp:PlaceHolder ID="phButtons" runat="server">
            </asp:PlaceHolder>--%>
        </td>
</tr>
</table>
</div>

```

Code-Behind Code:

```

protected void btnGenerateControl_Click(object sender, EventArgs e)
{
    // Retrieve the count of the controls to generate
    int Count = Convert.ToInt16(txtControlsCount.Text);
    // Loop thru each list item in the CheckBoxList
    foreach (ListItem li in chkBoxListControlType.Items)
    {
        if (li.Selected)
        {
            // Generate Lable Controls
            if (li.Value == "Label")
            {
                for (int i = 1; i <= Count; i++)
                {
                    Label lbl = new Label();
                    lbl.Text = "Label - " + i.ToString();
                    //phLabels.Controls.Add(lbl);
                }
            }
        }
    }
}

```

```

        //tdLabels.Controls.Add(lbl);
        pnlLabels.Controls.Add(lbl);
    }
}
// Generate TextBox controls
else if (li.Value == "TextBox")
{
    for (int i = 1; i <= Count; i++)
    {
        TextBox txtBox = new TextBox();
        txtBox.Text = "TextBox - " + i.ToString();
        //phTextBoxes.Controls.Add(txtBox);
        //tdTextBoxes.Controls.Add(txtBox);
        pnlTextBoxes.Controls.Add(txtBox);
    }
}
// Generate Button Controls
else
{
    for (int i = 1; i <= Count; i++)
    {
        Button btn = new Button();
        btn.Text = "Button - " + i.ToString();
        //phButtons.Controls.Add(btn);
        //tdButtons.Controls.Add(btn);
        pnlButtons.Controls.Add(btn);
    }
}
}
}
}

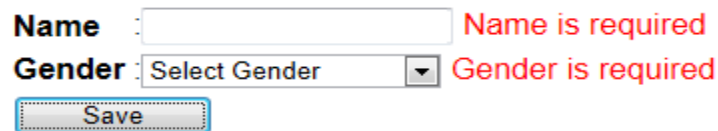
```



## RequiredField validator control in asp.net

Validation controls are used to ensure if, the data, entered by the user is valid. Microsoft asp.net framework, provides 6 built-in validation controls.

1. RequiredFieldValidator
2. RangeValidator
3. RegularExpressionValidator
4. CompareValidator
5. CustomValidator
6. ValidationSummary



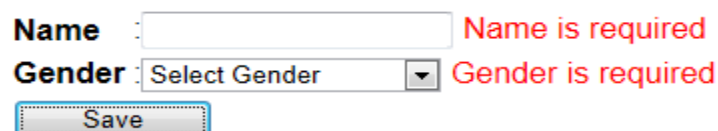
A screenshot of a web form. It contains two input fields: a text box labeled 'Name' and a dropdown menu labeled 'Gender'. To the right of the 'Name' text box, the text 'Name is required' is displayed in red. To the right of the 'Gender' dropdown menu, the text 'Gender is required' is displayed in red. Below these fields is a 'Save' button.

These validation controls can be used to perform both client side and server side validation.

Browsers understand only client scripts and HTML. In the past to perform client side validation, developers had to write themselves the required javascript code. With validation controls, we don't have to write javascript, we can use the built-in validation controls, which will generate the required javascript for us.

**Client scripts can spread viruses and cause security concerns.** Because of this, users may disable JavaScript on their browser. If this happens, client side validation is skipped. That is why, it is always a good practice to have server side validation as well.

The validation controls also perform server side validation. Server side validation is always performed, irrespective of whether the client side validation is done or not. This control, ensures that the required field is entered by the user. Let us understand with an example.



A screenshot of a web form, identical to the one above. It shows a 'Name' text box and a 'Gender' dropdown menu, both with red error messages 'Name is required' and 'Gender is required' respectively. A 'Save' button is located below the fields.

Consider the HTML below. TextBox with ID="txtName" expects the user to enter their name. This is a required field. Next to this textbox is a RequiredFieldValidator control, which is used to ensure that the user has entered his name. ControlToValidate property specifies the control to validate. ErrorMessage is the message that is displayed when the validation fails. To validate ddlGender DropDownList, we have RequiredFieldValidatorGender. We have specified InitialValue="-1". This will ensure that, if the user has selected, "Select Gender" option from the DropDownList, the control will still fail the validation.

```
<table>
  <tr>
    <td>
```

```

        <b>Name</b>
    </td>
    <td>
        <:asp:TextBox ID="txtName" runat="server" Width="150px">
        </asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidatorName" runat="server"
        ErrorMessage="Name is required" ForeColor="Red"
        ControlToValidate="txtName" Display="Dynamic" >
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td>
        <b>Gender</b>
    </td>
    <td>
        <:asp:DropDownList ID="ddlGender" runat="server" Width="150px">
        <asp:ListItem Text="Select Gender" Value="-1"></asp:ListItem>
        <asp:ListItem Text="Male" Value="Male"></asp:ListItem>
        <asp:ListItem Text="Female" Value="Female"></asp:ListItem>
        </asp:DropDownList>
        <asp:RequiredFieldValidator ID="RequiredFieldValidatorGender" runat="server"
        ErrorMessage="Gender is required" ForeColor="Red" InitialValue="-1"
        ControlToValidate="ddlGender" Display="Dynamic" >
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Button ID="btnSave" runat="server" Text="Save" Width="100px"
        onclick="btnSave_Click"/>
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Label ID="lblStatus" runat="server" Font-Bold="true">
        </asp:Label>
    </td>
</tr>
</table>

```

Code-Behind code: Page.IsValid is a read-only property. This property returns true if all the validation controls has passed validation. Use this property to perform server side validation.

```

protected void btnSave_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblStatus.ForeColor = System.Drawing.Color.Green;
        lblStatus.Text = "Data saved successfully!";
    }
}

```

```
else
{
    lblStatus.ForeColor = System.Drawing.Color.Red;
    lblStatus.Text = "Data not valid and not saved!";
}
}
```

Run the project, and notice that the client side validation is performed. Now, disable Javascript. To disable javascript in, internet explorer

1. Click on Tools on the Menu Bar. If the Menu bar is not visible, press ALT+T.
2. From the Tools menu, select Internet Options
3. Click on the Security tab, on the internet options window
4. Select Local Intranet icon and click on Custom Level button
5. Scroll down in the Security Settings window, and find Scripting option in the list.
6. Click Disable radiobutton under active scripting
7. Click OK and click Yes on the warning.

The above steps should disable javascript. Run the project now, and click Save button, without entering any data. Notice, that the client side validation is skipped, but the server side validation is performed.

## Rangevalidator control in asp.net

This control is used to check if the value is within a specified range of values. For example, Rangevalidator can be used to check if the age falls between 1 and 100.

Age	<input type="text" value="121"/>	Age must be between 1 & 100
Date Available	<input type="text" value="20/01/2020"/>	Date must be between 01/01/2012 & 31/12/2012
<input type="button" value="Save"/>		

In the HTML below, TextBox txtAge captures age of the person. If the user enters any number that is not between 1 & 100 the **validation fails**. The minimum and maximum value for the age is specified by MinimumValue and MaximumValue properties. Since, age is an integer; the Type is specified as integer.

```
<asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ErrorMessage="Age must be between 1 & 100"
    MinimumValue="1" MaximumValue="100"
    ControlToValidate="txtAge" Type="Integer" >
</asp:RangeValidator>
```

Properties specific to Rangevalidator control:

Type - This property, specifies the data type of the value to check. Data types supported include - Currency, Date, Double, Integer, String.

MinimumValue - The minimum value allowed

MaximumValue - The maximum value allowed

Complete HTML of the aspx page used in the Demo: Rangevalidator only checks if the entered data is within the allowed range. If you want to check for a required field, use RequiredFieldValidator. For the age field, we are using both RequiredFieldValidator and RangeValidator. Also notice that, in this example we are using the Display property. If the Display property is not set, or, if it is set to static, then the error message will be rendered, with style visibility:hidden. Because of this, the error message will always occupy the space on the screen even if the validation passes. This pushes "Age is Required" error message to the right. To correct this we have set Display="Dynamic". This renders the error message with style display:none. If a tag has this style, it will not occupy space when not visible.

```
<table>
<tr>
<td>
<b>Age</b>
</td>
<td>
:<asp:TextBox ID="txtAge" runat="server" Width="150px">
</asp:TextBox>
<asp:RangeValidator ID="RangeValidatorAge" runat="server"
    ErrorMessage="Age must be between 1 & 100"
    MinimumValue="1" MaximumValue="100"
    ControlToValidate="txtAge" Type="Integer"
    ForeColor="Red" Display="Dynamic">
</asp:RangeValidator>
```

```

        <asp:RequiredFieldValidator ID="RequiredFieldValidatorAge"
        runat="server" ErrorMessage="Age is required"
        ControlToValidate="txtAge" ForeColor="Red"
        Display="Dynamic" >
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td>
        <b>Date Available</b>
    </td>
    <td>
        :<asp:TextBox ID="txtDateAvailable" runat="server" Width="150px">
        </asp:TextBox>
        <asp:RangeValidator ID="RangeValidatorDateAvailable" runat="server"
        ErrorMessage="Date must be between 01/01/2012 & 31/12/2012"
        MinimumValue="01/01/2012" MaximumValue="31/12/2012"
        ControlToValidate="txtDateAvailable" Type="Date"
        ForeColor="Red">
        </asp:RangeValidator>
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Button ID="btnSave" runat="server" Text="Save" Width="100px"
        onclick="btnSave_Click" />
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Label ID="lblStatus" runat="server" Font-Bold="true">
        </asp:Label>
    </td>
</tr>
</table>

```

Code-Behind page code:

```

protected void btnSave_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblStatus.ForeColor = System.Drawing.Color.Green;
        lblStatus.Text = "Data Saved successfully";
    }
    else
    {
        lblStatus.ForeColor = System.Drawing.Color.Red;
        lblStatus.Text = "Validation Failed! Data not saved";
    }
}

```

**Display property is supported by all validation controls.**

**None** - Error message not rendered and displayed next to the control. Used to show the error message only in the ValidationSummary control

**Static** - The error message is displayed next to the control if validation fails. Space is reserved on the page for the message even if validation succeeds. The span tag is rendered with style visibility:hidden

**Dynamic** - The error message is displayed next to the control if validation fails. Space is not reserved on the page for the message if the validation succeeds. The span tag is rendered with style display:none.

## CompareValidator control in asp.net

CompareValidator control is used to compare the value of one control with the value of another control or a constant value. The comparison operation can be any of the following.

1. Equal
2. GreaterThan
3. GreaterThanEqual
4. LessThan
5. LessThanEqual
6. NotEqual
7. DataTypeCheck

<b>Password</b>	<input type="password" value="...."/>	
<b>Retype Password</b>	<input type="password" value="...."/>	Password and Retype Password must match
<b>Date of application</b>	<input type="text" value="01/01/2010"/>	Date of application must be greater than 01/01/2012
<b>Age</b>	<input type="text" value="TEN"/>	Age must be a number
<input type="button" value="Save"/>		

CompareValidator can also be used for DataType checking.

The following are the properties that are specific to the compare validator

1. ControlToCompare - The control with which to compare
2. Type - The DataType of the value to compare. String, Integer etc.
3. Operator = The comparison operator. Equal, NotEqual etc.
4. ValueToCompare - The constant value to compare with.

SetFocusOnError property is supported by all validation controls. If this property is set to true, then the control will automatically receive focus, when the validation fails.

**Using CompareValidator to compare the value of one control with the value of another control.**

```
<table>
  <tr>
    <td>
      <b>Password</b>
    </td>
    <td>
      <asp:TextBox ID="txtPassword" runat="server" Width="150px"
TextMode="Password"></asp:TextBox>
    </td>
  </tr>
  <tr>
    <td>
      <strong>Retype Password</strong>
    </td>
    <td>
      <asp:TextBox ID="txtRetypePassword" runat="server"
Width="150px" TextMode="Password"></asp:TextBox>
      <asp:CompareValidator ID="CompareValidatorPassword" runat="server"
```

```

        ErrorMessage="Password and Retype Password must match"
        ControlToValidate="txtRetypePassword" ControlToCompare="txtPassword"
        Type="String" Operator="Equal" ForeColor="Red">
    </asp:CompareValidator>
</td>
</tr>
</table>

```

### **Using CompareValidator to compare the value of one control with a constant value.**

```

<asp:TextBox ID="txtDateofapplication" runat="server" Width="150px">
</asp:TextBox>
<asp:CompareValidator ID="CompareValidatorDateofbirth" runat="server"
ErrorMessage="Date of application must be greater than 01/01/2012"
ControlToValidate="txtDateofapplication" ValueToCompare="01/01/2012"
Type="Date" Operator="GreaterThan" ForeColor="Red"
SetFocusOnError="true"></asp:CompareValidator>

```

### **Using CompareValidator to check DataType**

```

<asp:TextBox ID="txtAge" runat="server" Width="150px"></asp:TextBox>
<asp:CompareValidator ID="CompareValidatorAge" runat="server"
ErrorMessage="Age must be a number" ControlToValidate="txtAge"
Operator="DataTypeCheck" Type="Integer" ForeColor="Red"
SetFocusOnError="true"></asp:CompareValidator>

```



## RegularExpressionValidator control in asp.net

This is a very powerful validation control. This control is used to check if the value of an associated input control matches the pattern specified by a regular expression. The only property that is specific to the RegularExpressionValidator is ValidationExpression.



This example, checks if the input matches the pattern of an email address

```
<asp:TextBox ID="txtEmail" runat="server" Width="150px">
</asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionValidatorEmail" runat="server"
ControlToValidate="txtEmail" ValidationExpression="\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-
.])\\w+)*"
ErrorMessage="Invalid Email" ForeColor="Red"></asp:RegularExpressionValidator>
```

By default client side validation is turned on. To disable client side validation set EnableClientScript to false. This property is supported by all validation controls.

To disable validation control set Enabled property to false.

## CustomValidator control in asp.net

Please enter a positive even number  Not an even number

CustomValidator control allows us to write a method with a custom logic to handle the validation of the value entered. If none of the other validation controls, serve our purpose, then the CustomValidator can be used.

Just like any other validation control, CustomValidator can be used for both client and server side validation. Client side and server side validation functions, needs to be written by the developer, and associate them to the custom validator control.

The following are the properties that are specific to the CustomValidator control:

**ClientValidationFunction** - Specifies the name of the client side validation method.

**ValidateEmptyText** - Specifies whether the validator validates the control, when the text of the control is empty. By default this property is false, and both the client side and server side validation functions will not be invoked, if the associated input control is empty.

Events specific to the CustomValidator control:

**OnServerValidate** - Specifies the name of the server side validation method.

HTML of the aspx page: This example checks if the entered number is even. ValidateEmptyText property of the CustomValidator control, is set to true. So the validation is also triggered when the textbox is empty.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
  <script language="javascript" type="text/javascript">
    // Client side validation function to check if the number is even.
    function IsEven(source, args)
    {
      if (args.Value == "")
      {
        args.IsValid = false;
      }
      else
      {
        if (args.Value % 2 == 0)
        {
          args.IsValid = true;
        }
        else
        {
          args.IsValid = false;
        }
      }
    }
  </script>
</head>
<body>
  <div>
    Please enter a positive even number
    <input type="text" value="79"/>
    <span style="color: red; font-weight: bold;">Not an even number
  </div>
  <input type="button" value="Save"/>
</body>
</html>
```

```

    }
</script>
</head>
<body>
    <form id="form1" runat="server">
    <div style="font-family: Arial">
        <table>
            <tr>
                <td>
                    <b>Please enter a positive even number</b>
                </td>
                <td>
                    <asp:TextBox ID="txtEvenNumber" runat="server"></asp:TextBox>
                    <asp:CustomValidator ID="CustomValidatorEvenNumber" runat="server"
                        ForeColor="Red"
                        ErrorMessage="Not an even number"
                        OnServerValidate="CustomValidatorEvenNumber_ServerValidate"
                        ControlToValidate="txtEvenNumber"
                        ClientValidationFunction="IsEven"
                        ValidateEmptyText="true">
                    </asp:CustomValidator>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <asp:Button ID="btnSubmit" runat="server" Text="Save"
                        onclick="btnSubmit_Click" />
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <asp:Label ID="lblStatus" runat="server" Font-Bold="true"></asp:Label>
                </td>
            </tr>
        </table>
    </div>
    </form>
</body>
</html>

```

Code-Behind page code: Set EnableClientScript to true, to test the server side validation method.

```

protected void btnSubmit_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblStatus.Text = "Data Saved";
        lblStatus.ForeColor = System.Drawing.Color.Green;
    }
    else
    {
        lblStatus.Text = "Validation Error! Data Not Saved";
    }
}

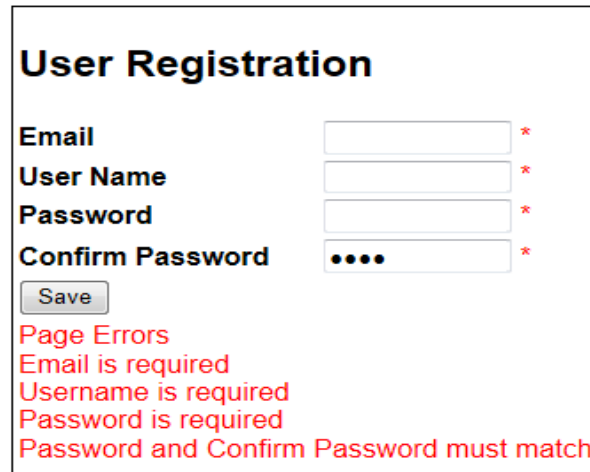
```

```
        lblStatus.ForeColor = System.Drawing.Color.Red;
    }
}
```

```
protected void CustomValidatorEvenNumber_ServerValidate(object source,
ServerValidateEventArgs args)
{
    if (args.Value == "")
    {
        args.IsValid = false;
    }
    else
    {
        int Number;
        bool isNumber = int.TryParse(args.Value, out Number);
        if (isNumber && Number >= 0 && (Number % 2) == 0)
        {
            args.IsValid = true;
        }
        else
        {
            args.IsValid = false;
        }
    }
}
```

## ValidationSummary control in asp.net

ValidationSummary control is used to display a summary of all validation errors occurred in a Web page, at one place. In general, in a real time applications, it is very common to display a red star symbol next to the input control where the error occurred, and then the detailed error message in the validation summary control as shown below.



The screenshot shows a web form titled "User Registration". It contains four input fields: "Email", "User Name", "Password", and "Confirm Password". Each field has a red asterisk (\*) to its right, indicating a validation error. Below the fields is a "Save" button. At the bottom of the form, there is a section titled "Page Errors" in red, followed by four error messages in red text: "Email is required", "Username is required", "Password is required", and "Password and Confirm Password must match".

Properties specific to the validation summary control:

**HeaderText** - The header text for the validation summary control

**ShowSummary** - Whether to display the summary text of all the validation errors

**ShowMessageBox** - Whether to display a message box with all the validation errors

**DisplayMode** - Display format for the summary.

DisplayMode can be List, BulletList, SingleParagraph

HTML of the ASPX Page:

```
<div style="font-family: Arial">
  <table style="border: 1px solid black">
    <tr>
      <td colspan="2">
        <h2>User Registration</h2>
      </td>
    </tr>
    <tr>
      <td>
        <b>Email</b>
      </td>
      <td>
        <asp:TextBox ID="txtEmail" runat="server" Width="100px">
        </asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidatorEmail"
runat="server" ForeColor="Red"
ErrorMessage="Email is required"
ControlToValidate="txtEmail" Display="Dynamic" Text="*">
        </asp:RequiredFieldValidator>
      </td>
    </tr>
  </table>
</div>
```

```

        <asp:RegularExpressionValidator ID="RegularExpressionValidatorEmail"
runat="server" ErrorMessage="Invalid Email Format"
ControlToValidate="txtEmail" ForeColor="Red"
ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*"
Text="*">
        </asp:RegularExpressionValidator>
    </td>
</tr>
<tr>
    <td>
        <b>User Name</b>
    </td>
    <td>
        <asp:TextBox ID="txtUserName" runat="server" Width="100px">
        </asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidatorUserName"
runat="server" ForeColor="Red"
ErrorMessage="Username is required"
ControlToValidate="txtUserName" Display="Dynamic" Text="*">
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td>
        <b>Password</b>
    </td>
    <td>
        <asp:TextBox ID="txtPassword" runat="server" Width="100px"
TextMode="Password"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidatorPassword"
runat="server" ForeColor="Red"
ErrorMessage="Password is required"
ControlToValidate="txtPassword" Display="Dynamic" Text="*">
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td>
        <b>Confirm Password</b>
    </td>
    <td>
        <asp:TextBox ID="txtConfirmPassword" runat="server" TextMode="Password"
Width="100px"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidatorPassword" runat="server"
ErrorMessage="Password and Confirm Password must match"
ControlToValidate="txtConfirmPassword" ControlToCompare="txtPassword"
Operator="Equal" Type="String" ForeColor="Red" Text="*">
        </asp:CompareValidator>
    </td>
</tr>
<tr>
    <td colspan="2">

```

```
<asp:Button ID="btnSubmit" runat="server" Text="Save"
    onclick="btnSubmit_Click" />
</td>
</tr>
<tr>
    <td colspan="2">
        <asp:ValidationSummary ID="ValidationSummary1" runat="server"
            ForeColor="Red" HeaderText="Page Errors" ShowMessageBox="True"
            ShowSummary="true" DisplayMode="List"
        />
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Label ID="lblStatus" runat="server" Font-Bold="true"></asp:Label>
    </td>
</tr>
</table>
</div>
```

## ValidationGroups in asp.net

Consider the image below.

The image shows two ASP.NET web forms side-by-side. The left form is titled 'Login' and contains two text boxes labeled 'User Name' and 'Password'. Both text boxes have a red asterisk to their right. Below the text boxes is a 'Login' button. Underneath the button, there are three lines of red text: 'Page Errors', 'Username is required', and 'Password is required'. The right form is titled 'User Registration' and contains four text boxes labeled 'Email', 'User Name', 'Password', and 'Confirm Password'. Below these text boxes are two buttons: 'Register' and 'Clear'. There are no error messages visible on this form.

The first problem here is that, when I click the Clear button, Form validation still happens. When I click the clear button, I just want to clear the textboxes in the Registration section. Validations doesn't make any sense here. **So, how do I prevent validation from happening?**

Just, set the CausesValidation property of the button control to false.

The second problem, is that when I click the Login button, only fields in the Login section (UserName and Password) needs to be validated. Along the same lines when I click the "Register" button, only fields in the Registration section (Email, UserName, Password and ConfirmPassword) needs to be validated. If we don't use validation groups, then by default, whenever, you click a button, all the validation controls on the page get validated.

So, when you click the login button, and if you want only, the fields in the Login section (UserName and Password) to be validated, then set, the ValidationGroup property of the validation controls and the login button control to the same group name. Use a different group name for the validation controls and register button, in the registration section.

### HTML of the aspx page:

```
<div style="font-family: Arial">
<table>
  <tr>
    <td style="vertical-align: top">
      <table style="border: 1px solid black">
        <tr>
          <td colspan="2">
            <h2>
              Login</h2>
            </td>
          </tr>
        </tr>
        <tr>
          <td>
            <b>User Name</b>
          </td>
          <td>
            <asp:TextBox ID="txtUN" runat="server" Width="100px">
            </asp:TextBox>
          </td>
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td colspan="2">
      <table style="border: 1px solid black">
        <tr>
          <td colspan="2">
            <h2>
              User Registration
            </h2>
          </td>
        </tr>
        <tr>
          <td>
            <b>Email</b>
          </td>
          <td>
            <asp:TextBox ID="txtEmail" runat="server" Width="100px">
            </asp:TextBox>
          </td>
        </tr>
        <tr>
          <td>
            <b>User Name</b>
          </td>
          <td>
            <asp:TextBox ID="txtUN2" runat="server" Width="100px">
            </asp:TextBox>
          </td>
        </tr>
        <tr>
          <td>
            <b>Password</b>
          </td>
          <td>
            <asp:TextBox ID="txtPW" runat="server" Width="100px">
            </asp:TextBox>
          </td>
        </tr>
        <tr>
          <td>
            <b>Confirm Password</b>
          </td>
          <td>
            <asp:TextBox ID="txtCPW" runat="server" Width="100px">
            </asp:TextBox>
          </td>
        </tr>
        <tr>
          <td colspan="2">
            <asp:Button ID="btnRegister" runat="server" Text="Register" CausesValidation="false" ValidationGroup="Register">
            </asp:Button>
            <asp:Button ID="btnClear" runat="server" Text="Clear" CausesValidation="false" ValidationGroup="Register">
            </asp:Button>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
</div>
```



[illegible]

```

<tr>
  <td colspan="2">
    <h2>
      User Registration</h2>
    </td>
  </tr>
</tr>
<tr>
  <td>
    <b>Email</b>
  </td>
  <td>
    <asp:TextBox ID="txtEmail" runat="server" Width="100px">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidatorEmail"
      runat="server" ForeColor="Red"
      ErrorMessage="Email is required"
      ControlToValidate="txtEmail" Display="Dynamic"
      Text="*" ValidationGroup="Registration">
    </asp:RequiredFieldValidator>
    <asp:RegularExpressionValidator
      ID="RegularExpressionValidatorEmail"
      runat="server"
      ErrorMessage="Invalid Email Format"
      ControlToValidate="txtEmail" ForeColor="Red"
      ValidationExpression="\w+([-+.']\w+)*@\w+([-.']\w+)*\.\w+([-.']\w+)*"
      Text="*"
      ValidationGroup="Registration">
    </asp:RegularExpressionValidator>
  </td>
</tr>
</tr>
<tr>
  <td>
    <b>User Name</b>
  </td>
  <td>
    <asp:TextBox ID="txtUserName" runat="server" Width="100px">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidatorUserName"
      runat="server" ForeColor="Red"
      ErrorMessage="Username is required"
      ControlToValidate="txtUserName" Display="Dynamic"
      Text="*" ValidationGroup="Registration">
    </asp:RequiredFieldValidator>
  </td>
</tr>
</tr>
<tr>
  <td>
    <b>Password</b>
  </td>
  <td>
    <asp:TextBox ID="txtPassword" runat="server"
      Width="100px" TextMode="Password"></asp:TextBox>
  </td>
</tr>

```

```

        <asp:RequiredFieldValidator ID="RequiredFieldValidatorPassword"
            runat="server" ForeColor="Red"
            ErrorMessage="Password is required"
            ControlToValidate="txtPassword" Display="Dynamic"
            Text="*" ValidationGroup="Registration">
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
<td>
        <b>Confirm Password</b>
    </td>
<td>
        <asp:TextBox ID="txtConfirmPassword" runat="server"
            TextMode="Password" Width="100px"></asp:TextBox>
        <asp:CompareValidator ID="CompareValidatorPassword"
            runat="server"
            ErrorMessage="Password and Confirm Password must match"
            ControlToValidate="txtConfirmPassword"
            ControlToCompare="txtPassword" Operator="Equal"
            Type="String" ForeColor="Red" Text="*"
            ValidationGroup="Registration">
        </asp:CompareValidator>
    </td>
</tr>
<tr>
<td colspan="2">
        <asp:Button ID="btnSubmit" runat="server" Text="Register"
            OnClick="btnSubmit_Click" ValidationGroup="Registration"/>
        &nbsp;
        <asp:Button ID="btnClear" runat="server" onclick="btnClear_Click"
            CausesValidation="false" Text="Clear" />
    </td>
</tr>
<tr>
<td colspan="2">
        <asp:ValidationSummary ID="ValidationSummary1" runat="server"
            ForeColor="Red" HeaderText="Page Errors" ShowMessageBox="True"
            ShowSummary="true" DisplayMode="List"
            ValidationGroup="Registration"/>
    </td>
</tr>
<tr>
<td colspan="2">
        <asp:Label ID="lblStatus" runat="server"
            Font-Bold="true"></asp:Label>
    </td>
</tr>
</table>
</td>
</tr>
</table>

```

</div>

### Code-Behind:

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblStatus.Text = "No registration validation Errors";
        lblStatus.ForeColor = System.Drawing.Color.Green;
    }
    else
    {
        lblStatus.Text = "Registration validation Errors";
        lblStatus.ForeColor = System.Drawing.Color.Red;
    }
}
```

```
protected void btnClear_Click(object sender, EventArgs e)
{
    txtEmail.Text = "";
    txtUserName.Text = "";
    txtPassword.Text = "";
    txtConfirmPassword.Text = "";
}
```

```
protected void btnLogin_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        lblLoginStatus.Text = "No Login Validation Errors";
        lblStatus.ForeColor = System.Drawing.Color.Green;
    }
    else
    {
        lblStatus.Text = "Login validation errors";
        lblStatus.ForeColor = System.Drawing.Color.Red;
    }
}
```

## **Different page navigation techniques in asp.net**

**What are the different page navigation techniques in asp.net?**

**OR**

**How do you move from one webform to another webform in asp.net?**

**OR**

**How do you link pages in an application?**

This is a very common interview question in asp.net. There are several techniques to navigate between webforms in asp.net as listed below.

1. Hyperlink control - Is used to navigate to another page. The page you want to navigate to is specified by the NavigateURL property. Using hyperlink, you can navigate to another page with in the same application or to an external web site. The hyperlink control is rendered as an HTML anchor tag.
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

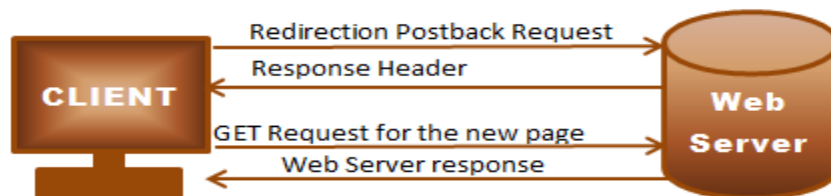
## Response.Redirect in asp.net

The following are the different page navigation techniques in asp.net:

1. Hyperlink control
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

Response.Redirect is similar to clicking on a hyperlink. The Hyperlink control does not expose any server side events. So when the user clicks on a hyperlink, there is no server side event to intercept the click.

So, if you want to intercept a click event in code, use the Button, LinkButton or the ImageButton server control. In the button click event, call Response.Redirect() method. When the user clicks the button, the web server receives, a request for redirection. The server then sends a response header to the client. The client then automatically issues a new GET request to the web server. The web server will then serve the new page. So, in short, Response.Redirect causes 2 request/response cycles.



Also, note that when Response.Redirect is used the URL in the address bar changes and the browser history is maintained.

Response.Redirect() can be used to navigate pages/websites on the same web server or on a different web server.

## Server.Transfer in asp.net

The following are the different page navigation techniques in asp.net:

1. Hyperlink control
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

Create an asp.net web application and add 2 webforms. Copy and paste the following HTML on WebForm1.aspx

```
<div style="font-family: Arial">
  <table>
    <tr>
      <td colspan="2">
        <h1>
          This is WebForm1</h1>
        </td>
      </tr>
      <tr>
        <td>
          <b>Name</b>
        </td>
        <td>
          :<asp:TextBox ID="txtName" runat="server">
          </asp:TextBox>
        </td>
      </tr>
      <tr>
        <td>
          <b>Email</b>
        </td>
        <td>
          :<asp:TextBox ID="txtEmail" runat="server">
          </asp:TextBox>
        </td>
      </tr>
      <tr>
        <td colspan="2">
          <asp:Button ID="btnTransfer" runat="server"
            Text="Transfer to WebForm2" Width="250px"
            OnClick="btnTransfer_Click"/>
        </td>
      </tr>
      <tr>
        <td colspan="2">
          <asp:Button ID="btnTransferToExternalWebsite"
            runat="server" Width="250px"
            OnClick="btnTransferToExternalWebsite_Click">

```

```

        Text="Transfer to External WebSite"/>
    </td>
</tr>
</table>
</div>

```

Code-Behind code for WebForm1.aspx.cs

```

protected void btnTransfer_Click(object sender, EventArgs e)
{
    //Send the user to webform2 using Server.Transfer
    //Set the boolean parameter preserveForm=true
    //This ensures that the posted form values can be retrieved
    //Since the default value for this parameter is true, the
    //form values are preserved, even if this parameter is not used.
    Server.Transfer("~/WebForm2.aspx", true);
}
protected void btnTransferToExternalWebsite_Click(object sender, EventArgs e)
{
    //Transfer to websites/pages on a different web server causes
    //runtime error
    Server.Transfer("http://pragimtech.com/home.aspx");
}

```

WebForm2.aspx code:

```

<div>
    <table>
        <tr>
            <td>
                <b>Name</b>
            </td>
            <td>
                <asp:Label ID="lblName" runat="server">
                </asp:Label>
            </td>
        </tr>
        <tr>
            <td>
                <b>Email</b>
            </td>
            <td>
                <asp:Label ID="lblEmail" runat="server">
                </asp:Label>
            </td>
        </tr>
    </table>
</div>

```

WebForm2.aspx.cs code

```

protected void Page_Load(object sender, EventArgs e)

```



```

{
    //Get the form values from the previous page
    System.Collections.Specialized.NameValueCollection nameValueCollection =
        Request.Form;

    lblName.Text = nameValueCollection["txtName"];
    lblEmail.Text = nameValueCollection["txtEmail"];

    //Page previousPage = this.Page.PreviousPage;
    //if (previousPage != null)
    //{
    //    TextBox previousPageNameTextBox = (TextBox)previousPage.FindControl("txtName");
    //    lblName.Text = previousPageNameTextBox.Text;

    //    TextBox previousPageEmailTextBox = (TextBox)previousPage.FindControl("txtEmail");
    //    lblEmail.Text = previousPageEmailTextBox.Text;
    //}
}

```

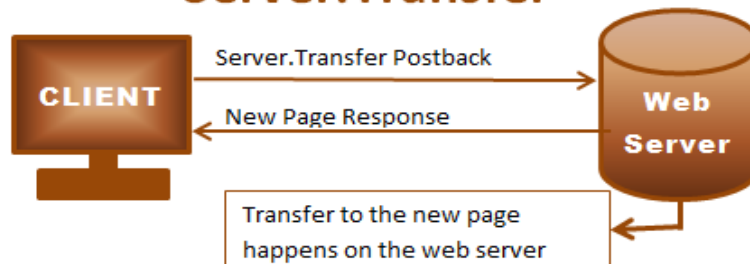
The following are the differences between Server.Transfer and Response.Redirect:

1. Just like hyperlink and Response.Redirect, Server.Transfer is used to navigate to other pages/sites running on the same web server.
2. Server.Transfer cannot be used to navigate to sites/pages on a different web server.
3. Server.Transfer does not change the URL in the address bar
4. Server.Transfer is faster than Response.Redirect as the redirection happens on the server in one Request/Response cycle. Response.Redirect() involves 2 Request/Response cycles.
5. With Server.Transfer the Form Variables from the original request are preserved.

## Response.Redirect



## Server.Transfer



## Server.execute in asp.net

The following are the different page navigation techniques in asp.net:

1. Hyperlink control
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

Server.Transfer and Server.Execute are similar in many ways:

1. The URL in the browser remains the first page URL.
2. Server.Transfer and Server.Execute can only be used to navigate to sites/pages on the same web server. Trying to navigate to sites/pages on a different web server, causes runtime exception.
3. Server.Transfer and Server.Execute preserves the Form Variables from the original request.

The major difference between Server.Transfer and Server.Execute is that, Server.Transfer terminates the execution of the current page and starts the execution of the new page, whereas Server.Execute process the second Web form without leaving the first Web form. After completing the execution of the first webform, the control returns to the second webform.

WebForm1.aspx code:

```
<div style="font-family: Arial">
  <table>
    <tr>
      <td colspan="2">
        <h1>
          This is WebForm1</h1>
        </td>
      </tr>
      <tr>
        <td>
          <b>Name</b>
        </td>
        <td>
          :<asp:TextBox ID="txtName" runat="server">
            </asp:TextBox>
          </td>
        </tr>
      <tr>
        <td>
          <b>Email</b>
        </td>
        <td>
          :<asp:TextBox ID="txtEmail" runat="server">
            </asp:TextBox>
          </td>
        </tr>
      <tr>
        <td>
```

```

        <td colspan="2">
            <asp:Button ID="btnExecute" runat="server"
                Text="Server.Execute - WebForm2"
                Width="250px" onclick="btnExecute_Click"/>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Button ID="btnExecuteToExternalWebsite"
                runat="server" Width="250px"
                Text="Server.Execute - External WebSite"
                onclick="btnExecuteToExternalWebsite_Click" />
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Label ID="lblStatus" ForeColor="Green"
                Font-Bold="true" runat="server"></asp:Label>
        </td>
    </tr>
</table>
</div>

```

WebForm1.aspx.cs code:

```

protected void btnExecute_Click(object sender, EventArgs e)
{
    Server.Execute("~/WebForm2.aspx", true);
    lblStatus.Text = "The call returned after processing the second webform";
}
protected void btnExecuteToExternalWebsite_Click(object sender, EventArgs e)
{
    Server.Execute("http://pragimtech.com/home.aspx");
}

```

WebForm2.aspx code:

```

<div style="font-family: Arial">
    <table>
        <tr>
            <td colspan="2">
                <h1>This is WebForm2</h1>
            </td>
        </tr>
        <tr>
            <td>
                <b>Name</b>
            </td>
            <td>
                <asp:Label ID="lblName" runat="server">
                </asp:Label>
            </td>
        </tr>
    </table>

```

```

</tr>
<tr>
  <td>
    <b>Email</b>
  </td>
  <td>
    :<asp:Label ID="lblEmail" runat="server">
    </asp:Label>
  </td>
</tr>
<tr>
  <td colspan="2">
    <asp:Button ID="btnPostBack" runat="server"
    Text="Simply Post Back" />
  </td>
</tr>
</table>
</div>

```

WebForm2.aspx.cs code:

```

protected void Page_Load(object sender, EventArgs e)
{
    System.Collections.Specialized.NameValueCollection previousFormcollection = Request.Form;
    lblName.Text = previousFormcollection["txtName"];
    lblEmail.Text = previousFormcollection["txtEmail"];

    //Page previousPage = Page.PreviousPage;
    //if (previousPage != null)
    //{
    //    lblName.Text = ((TextBox)previousPage.FindControl("txtName")).Text;
    //    lblEmail.Text = ((TextBox)previousPage.FindControl("txtEmail")).Text;
    //}
}

```

## Cross page posting in asp.net

The following are the different page navigation techniques in asp.net:

1. Hyperlink control
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

Cross page posting allows to post one page to another page. By default, when you click a button, the webform posts to itself. If you want to post to another webform on a button click, set the PostBackUrl of the button, to the page that you want to post to.

WebForm1.aspx code: Notice that, the PostBackUrl property of the button with ID=btnCrossPagePostback is set to WebForm2.aspx. When this button is clicked WebForm1.aspx gets posted to WebForm2.aspx.

```
<div style="font-family: Arial">
  <table>
    <tr>
      <td colspan="2">
        <h1>
          This is WebForm1</h1>
        </td>
      </tr>
      <tr>
        <td>
          <b>Name</b>
        </td>
        <td>
          :<asp:TextBox ID="txtName" runat="server">
          </asp:TextBox>
        </td>
      </tr>
      <tr>
        <td>
          <b>Email</b>
        </td>
        <td>
          :<asp:TextBox ID="txtEmail" runat="server">
          </asp:TextBox>
        </td>
      </tr>
      <tr>
        <td colspan="2">
          <asp:Button ID="btnCrossPagePostback" runat="server"
            Text="Cross Page Postback - WebForm2"
            Width="250px" PostBackUrl="~/WebForm2.aspx"/>
        </td>
      </tr>
    </table>
  </div>
```

```

        <tr>
        <td colspan="2">
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
                Text="Server.Transfer - WebForm2" Width="250px" />
        </td>
        </tr>
    </table>
</div>

```

WebForm1.aspx.cs code:

```

protected void Button1_Click(object sender, EventArgs e)
{
    Server.Transfer("~/WebForm2.aspx");
}

```

WebForm2.aspx code:

```

<div style="font-family: Arial">
    <table>
    <tr>
    <td colspan="2">
        <h1>This is WebForm2</h1>
    </td>
    </tr>
    <tr>
    <td>
        <b>Name</b>
    </td>
    <td>
        <asp:Label ID="lblName" runat="server">
        </asp:Label>
    </td>
    </tr>
    <tr>
    <td>
        <b>Email</b>
    </td>
    <td>
        <asp:Label ID="lblEmail" runat="server">
        </asp:Label>
    </td>
    </tr>
    <tr>
    <td colspan="2">
        <asp:Label ID="lblStatus" runat="server"
            ForeColor="Red" Font-Bold="true"></asp:Label>
    </td>
    </tr>
    </table>
</div>

```

WebForm2.aspx.cs code: Page.IsCrossPagePostBack Property is used to indicate whether the page is involved in a cross-page postback.

```
protected void Page_Load(object sender, EventArgs e)
{
    Page previousPage = Page.PreviousPage;
    if (previousPage != null && previousPage.IsCrossPagePostBack)
    {
        lblName.Text = ((TextBox)previousPage.FindControl("txtName")).Text;
        lblEmail.Text = ((TextBox)previousPage.FindControl("txtEmail")).Text;
    }
    else
    {
        lblStatus.Text = "Landed on this page using a technique other than cross page post back";
    }
}
```

The problem with FindControl() method is that, if you mis-spell the ControlID, we could get a **runtime NullReferenceException**. we will discuss about obtaining a strongly typed reference to the previous page, which can avoid NullReferenceExceptions.

## Cross page postback strongly typed reference

The following are the different page navigation techniques in asp.net:

1. Hyperlink control
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

we used FindControl() method to get a reference to the TextBox on the previous page. The problem with FindControl() method is that, if we mis-spell the ID of the control, we don't get any compile time errors, but may cause runtime nullreference exceptions.

There are 2 ways to obtain a strongly typed reference:

**The first step in obtaining a strongly typed reference, is to create public properties.** We want to convert the values of TextBox controls(txtName and txtEmail) into properties(Name and Email) respectively. The Name and Email properties are created as Read-Only properties, as we just need to read the values on the destination page.

```
//Name - read only property
public string Name
{
    get
    {
        return txtName.Text;
    }
}
//Email - read only property
public string Email
{
    get
    {
        return txtEmail.Text;
    }
}
```

**The next step is to obtain a strongly typed reference to the previous page and access the public propertie as shown below.** This code must be in the Page\_Load event on WebForm2.aspx.cs. If Name or Email properties are mis-spelled, we get an immediate compile time error. Hence, strongly typed references can eliminate runtime nullreference exceptions.

```
protected void Page_Load(object sender, EventArgs e)
{
    //Type cast PreviousPage to WebForm1
    WebForm1 previousPage = (WebForm1)Page.PreviousPage;
    if (previousPage != null && previousPage.IsCrossPagePostBack)
    {
        //Access the Name and Email public properties
        lblName.Text = previousPage.Name;
    }
}
```



```

        lblEmail.Text = previousPage.Email;
    }
    else
    {
        lblStatus.Text = "You landed on this page using a technique other than cross page post back";
    }
}

```

**PreviousPageType** directive can also be used to obtain, a strongly typed reference to the previous page. In our example, for WebForm2, the previous page is WebForm1. So, in the HTML source of WebForm2.aspx paste the line below, after the Page directive.

```
<%@ PreviousPageType VirtualPath="~/WebForm1.aspx" %>
```

In the code behind file, this.PreviousPage property or PreviousPage(without any prefix), returns a strongly typed reference to WebForm1. Please note, that Page.PreviousPage property, still returns the loosely typed Page object.

```

protected void Page_Load(object sender, EventArgs e)
{
    //this.PreviousPage returns a strongly typed reference
    //WebForm1 previousPage = this.PreviousPage;
    //PreviousPage also returns a strongly typed reference
    WebForm1 previousPage = PreviousPage;
    //Page.PreviousPage returns loosely typed reference
    //Page previousPage = Page.PreviousPage;
    if (previousPage != null && previousPage.IsCrossPagePostBack)
    {
        //Access the Name and Email public properties
        lblName.Text = previousPage.Name;
        lblEmail.Text = previousPage.Email;
    }
    else
    {
        lblStatus.Text = "You landed on this page using a technique other than cross page post back";
    }
}

```

So in short to obtain a strongly typed reference, there are 2 very simple steps:

First Step – Create Public Properties (Read-Only is sufficient)

Second Step – Obtain a strongly typed reference by TypeCasting or by using the PreviousPageType directive.

## Opening new window using javascript in asp.net

The following are the different page navigation techniques in asp.net:

1. Hyperlink control
2. Response.Redirect
3. Server.Transfer
4. Server.Execute
5. Cross-Page postback
6. Window.Open

we will discuss about, opening a new window using javascript method window.open(). First the syntax of the window.open() method.

window.open(URL, name, features, replace)

Parameter Name	Description
URL (Optional)	The URL of the page to open. If URL is not specified, a new window with about:blank is opened.
Name (Optional)	Specifies the target attribute or the name of the window. name - The name of the window _blank - Opens in a new window. Default, if nothing is specified. _self - Opens in the same page _parent - Loaded into the parent frame _top - URL replaces any framesets that may be loaded
Features (optional)	A comma-separated list of items. resizable = yes no or 0 1 scrollbars = yes no or 0 1 toolbar = yes no or 0 1 location = yes no or 0 1 (Specifies whether to display the Navigation Bar. The default is yes) status = yes no or 0 1 menubar = yes no or 0 1 left = yes no or pixels top = yes no or pixels width = yes no or pixels height = yes no or pixels
Replace(Optional)	A boolean parameter that specifies whether the url creates a new entry or replaces the current entry in the window's history list. This parameter only takes effect if the url is loaded into the same window. true - url replaces the current document in the history list. false - url creates a new entry in the history list.

HTML of WebForm1.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
  <title>Training Demo</title>
```

```

<script type="text/javascript">
    // Javascript function to open the new window
    function OpenNewWindow()
    {
        var Name = document.getElementById('txtName').value;
        var Email = document.getElementById('txtEmail').value;
        window.open('WebForm2.aspx?Name=' + Name + '&Email=' + Email, '_blank', 'toolbar=no,
location=no, resizable=yes,

width=500px, height=500px', true);
    }
</script>
</head>
<body>
    <form id="form1" runat="server">
<div style="font-family: Arial">
    <table>
        <tr>
            <td colspan="2">
                <h1>
                    This is WebForm1</h1>
                </td>
            </tr>
            <tr>
                <td>
                    <b>Name</b>
                </td>
                <td>
                    :<asp:TextBox ID="txtName" runat="server">
                    </asp:TextBox>
                </td>
            </tr>
            <tr>
                <td>
                    <b>Email</b>
                </td>
                <td>
                    :<asp:TextBox ID="txtEmail" runat="server">
                    </asp:TextBox>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <input id="Button1" type="button" value="HTML Input Button - Window.Open"
                        onclick="OpenNewWindow()" style="width: 300px"
                    />
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <asp:Button ID="Button2" runat="server"
                        Text="ASP.NET Button - Window.Open()" onclick="Button2_Click"

```

```

        Width="300px" />
    </td>
</tr>
</table>
</div>
</form>
</body>
</html>

```

Code-Behind code for WebForm1.aspx.cs

```

protected void Button2_Click(object sender, EventArgs e)
{
    string strJavascript = "<script type='text/javascript'>window.open('WebForm2.aspx?Name=";
    strJavascript += txtName.Text + "&Email=" + txtEmail.Text + "','_blank');</script>";
    Response.Write(strJavascript);
}

```

HTML of WebForm2.aspx

```

<div style="font-family: Arial">
    <table>
        <tr>
            <td colspan="2">
                <h1>This is WebForm2</h1>
            </td>
        </tr>
        <tr>
            <td>
                <b>Name</b>
            </td>
            <td>
                <asp:Label ID="lblName" runat="server">
                </asp:Label>
            </td>
        </tr>
        <tr>
            <td>
                <b>Email</b>
            </td>
            <td>
                <asp:Label ID="lblEmail" runat="server">
                </asp:Label>
            </td>
        </tr>
    </table>
</div>

```

Code-Behind code for WebForm2.aspx.cs

```

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

```

```
lblEmail.Text = Request.QueryString["Email"];  
}
```

## Techniques to send data from one webform to another in in asp.net

Different techniques to move data from one webform to another:

1. Cross Page Postback
2. Context.Handler object

The following concepts will be discussed in the subsequent sessions:

3. Query strings
4. Cookies
5. Session state
6. Application state

In general, the members of one Web form are unavailable from a subsequently displayed Web form. However, when navigating between Web forms using the Transfer or Execute method, data can be retrieve from the previous Web form using Context.Handler object.

Points to remember Context.Handler object:

1. Context.Handler returns WebForm1 as the previous page, only the first time when you land on WebForm2 from WebForm1. If there is a button on WebForm2, and if you click the button, the page will postback, and Context.Handler will return WebForm2 instead of WebForm1.
2. For the Context.Handler to return WebForm1 as the previous page, you should have landed on WebForm2, using Server.Transfer or Server.Execute method from WebForm1.
3. The control values from the previous page, can be accessed using FindControl() method or using public properties. The problem with FindControl() method is that, if you mis-spell the ControlID, we could get a runtime NullRefernceException. Using public properties, instead of FindControl() method, can eliminate runtime NullRefernceExceptions.

WebForm1.aspx HTML source:

```
<div style="font-family: Arial">
<table>
  <tr>
    <td colspan="2">
      <h1>
        This is WebForm1</h1>
      </td>
    </tr>
    <tr>
      <td>
        <b>Name</b>
      </td>
      <td>
        :<asp:TextBox ID="txtName" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td>
        <b>Email</b>
      </td>
    </tr>
```

```

        </td>
        <td>
            <asp:TextBox ID="txtEmail" runat="server">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Button ID="Button1" runat="server"
            Text="Go to WebForm2" onclick="Button1_Click" />
        </td>
    </tr>
</table>
</div>

```

WebForm1.aspx.cs Code:

```

protected void Button1_Click(object sender, EventArgs e)
{
    Server.Transfer("~/WebForm2.aspx");
}

public string Name
{
    get
    {
        return txtName.Text;
    }
}

public string Email
{
    get
    {
        return txtEmail.Text;
    }
}

```

WebForm2.aspx HTML source:

```

<div style="font-family: Arial">
<table>
    <tr>
        <td colspan="2">
            <h1>This is WebForm2</h1>
        </td>
    </tr>
    <tr>
        <td>
            <b>Name</b>
        </td>
        <td>

```

```

        :<asp:Label ID="lblName" runat="server">
        </asp:Label>
    </td>
</tr>
<tr>
    <td>
        <b>Email</b>
    </td>
    <td>
        :<asp:Label ID="lblEmail" runat="server">
        </asp:Label>
    </td>
</tr>
</table>
</div>

```

WebForm2.aspx.cs Code:

```

//On postback Context.Handler returns WebForm2
if (!IsPostBack)
{
    Page lastpage = (Page)Context.Handler;
    if (lastpage is WebForm1)
    {
        //Use FindControl() if public properties does not exist on the
        //previous page(WebForm1). FindControl() may cause
        //NullReferenceExceptions due to mis-spelled control Id's

        //lblName.Text = ((TextBox)lastpage.FindControl("txtName")).Text;
        //lblEmail.Text = ((TextBox)lastpage.FindControl("txtEmail")).Text;

        //Using public properties can eliminate NullReferenceExceptions
        lblName.Text = ((WebForm1)lastpage).Name;
        lblEmail.Text = ((WebForm1)lastpage).Email;
    }
}

```



## QueryString in asp.net

Different techniques to move data from one webform to another:

1. Cross Page Postback
2. Context.Handler object
3. Query strings

The following concepts will be discussed in the subsequent sessions :

4. Cookies
5. Session state
6. Application state

Points to remember about query strings:

1. Querystrings are name/value collection pairs
2. Using querystrings, is a very common way to send data from one webform to another.
3. Query strings are appended to the page URL.
4. ?(Question Mark), indicates the beginning of a query string and its value.
5. It is possible to use more than one query string. The first query string is specified using the ?(question mark). Subsequent query strings can be appended to the URL using the &(ampersand) symbol.
6. There is a limit on the Query string length. Hence, Query strings cannot be used to send very long data.
7. Query strings are visible to the user, hence should not be used to send sensitive information, unless encrypted.
8. To read the query string value, use Request object's QueryString property.
9. &(ampersand) is used to concatenate query strings, so if you want to send &, as value for the query string there are 2 ways, as shown below:

### Using Server.UrlEncode() method

```
Response.Redirect("WebForm2.aspx?UserName=" + Server.UrlEncode(txtName.Text) +  
"&UserEmail=" + Server.UrlEncode(txtEmail.Text));
```

**Or**

**&(ampersand) is encoded as %26, so use, Replace() function to replace & with %26**

```
Response.Redirect("WebForm2.aspx?UserName=" + txtName.Text.Replace("&", "%26") +  
"&UserEmail=" + txtEmail.Text.Replace("&", "%26"));
```

WebForm1.aspx HTML: We want to send Name and Email, that user enters on WebForm1.aspx to WebForm2.aspx using query strings.

```
<div style="font-family: Arial">  
<table>  
  <tr>  
    <td colspan="2">  
      <h1>  
        This is WebForm1</h1>  
      </td>  
    </tr>  
  </table>
```

```

        <td>
            <b>Name</b>
        </td>
        <td>
            <:asp:TextBox ID="txtName" runat="server">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td>
            <b>Email</b>
        </td>
        <td>
            <:asp:TextBox ID="txtEmail" runat="server">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Button ID="btnSendData" runat="server"
            Text="Go to WebForm2" onclick="btnSendData_Click" />
        </td>
    </tr>
</table>
</div>

```

WebForm1.aspx.cs

```

protected void btnSendData_Click(object sender, EventArgs e)
{
    //Using Server.UrlEncode to encode &(ampersand)
    //Response.Redirect("WebForm2.aspx?UserName=" + Server.UrlEncode(txtName.Text) +
    //    "&UserEmail=" + Server.UrlEncode(txtEmail.Text));

    //Using String.Replace() function to replace &(ampersand) with %26
    Response.Redirect("WebForm2.aspx?UserName=" + txtName.Text.Replace("&", "%26") +
        "&UserEmail=" + txtEmail.Text.Replace("&", "%26"));
}

```

WebForm2.aspx HTML:

```

<div style="font-family: Arial">
<table>
    <tr>
        <td colspan="2">
            <h1>This is WebForm2</h1>
        </td>
    </tr>
    <tr>
        <td>
            <b>Name</b>
        </td>
    </tr>

```

```

        <td>
            <:asp:Label ID="lblName" runat="server">
            </asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <b>Email</b>
        </td>
        <td>
            <:asp:Label ID="lblEmail" runat="server">
            </asp:Label>
        </td>
    </tr>
</table>
</div>

```

WebForm2.aspx.cs Code:

```

protected void Page_Load(object sender, EventArgs e)
{
    // Read the QueryString values
    lblName.Text = Request.QueryString["UserName"];
    lblEmail.Text = Request.QueryString["UserEmail"];
}

```

## Cookies in asp.net

Different techniques to send data from one webform to another:

1. Cross Page Postback
2. Context.Handler object
3. Query strings
4. Cookies

The following concepts will be discussed in the subsequent sessions :

5. Session state
6. Application state

Just like QueryStrings, Cookies can also be used to send data from one webform to another. In general, web sites use cookies to store user preferences or other information that is client-specific. Cookies store small amounts of information on the client's machine.

Cookies can be broadly classified into 2 types:

- 1. Persistent cookies** - Remain on the client computer, even after the browser is closed. You can configure how long the cookies remain using the expires property of the HttpCookie object.
- 2. Non-Persistent cookies** - If you don't set the Expires property, then the cookie is called as a Non-Persistent cookie. Non-Persistent cookies only remain in memory until the browser is closed.

On WebForm1.aspx, user enters Name and Email. Let's write these values on to the client's computer using cookies. Finally read the values from the cookie and display them in WebForm2.aspx.

### WebForm1.aspx HTML source:

```
<div style="font-family: Arial">
<table>
  <tr>
    <td colspan="2">
      <h1>
        This is WebForm1</h1>
      </td>
    </tr>
    <tr>
      <td>
        <b>Name</b>
      </td>
      <td>
        <asp:TextBox ID="txtName" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td>
        <b>Email</b>
      </td>
```

```

        <td>
            <asp:TextBox ID="txtEmail" runat="server">
            </asp:TextBox>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Button ID="btnSendData" runat="server"
            Text="Go to WebForm2" onclick="btnSendData_Click" />
        </td>
    </tr>
</table>
</div>

```

### WebForm1.aspx.cs code:

```

protected void btnSendData_Click(object sender, EventArgs e)
{
    // Create the cookie object
    HttpCookie cookie = new HttpCookie("UserDetails");
    cookie["Name"] = txtName.Text;
    cookie["Email"] = txtEmail.Text;
    // Cookie will be persisted for 30 days
    cookie.Expires = DateTime.Now.AddDays(30);
    // Add the cookie to the client machine
    Response.Cookies.Add(cookie);

    Response.Redirect("WebForm2.aspx");
}

```

### WebForm2.aspx HTML Source:

```

<div style="font-family: Arial">
<table>
    <tr>
        <td colspan="2">
            <h1>This is WebForm2</h1>
        </td>
    </tr>
    <tr>
        <td>
            <b>Name</b>
        </td>
        <td>
            <asp:Label ID="lblName" runat="server">
            </asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <b>Email</b>
        </td>
        <td>

```

```
        :<asp:Label ID="lblEmail" runat="server">
        </asp:Label>
    </td>
</tr>
</table>
</div>
```

#### WebForm2.aspx.cs Code:

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie = Request.Cookies["UserDetails"];
    if (cookie != null)
    {
        lblName.Text = cookie["Name"];
        lblEmail.Text = cookie["Email"];
    }
}
```

## How to Check if cookies are enabled or disabled

Different techniques to send data from one webform to another:

1. Cross Page Postback
2. Context.Handler object
3. Query strings
4. Cookies

How to Check if cookies are enabled or disabled in asp.net? Most of the articles on the internet, states we can use Request.Browser.Cookies property to check, if the cookies are enabled or disabled.

**This is incorrect.**

```
if (Request.Browser.Cookies)
{
    //Cookies Enabled
}
else
{
    //Cookies Disabled
}
```

Request.Browser.Cookies property is used to check, if the browser supports cookies. Most modern browsers, support cookies. Irrespective of whether, the cookies are enabled or disabled, if the browser supports cookies, Request.Browser.Cookies always returns true. So use this property to check if the browser supports cookies and not to check if the cookies are enabled or disabled.

```
if (Request.Browser.Cookies)
{
    //Browser supports cookies
}
else
{
    //Browser does not supports cookies
}
```

So, the next question is, how do we check, if cookies are enabled or disabled?

1. Write a Test Cookie
2. Redirect to the same page
3. Read the Test Cookie
4. If Cookies present - **Cookies are enabled**
5. Else - **Cookies are disabled.**

To disable cookies in Internet Explorer(IE 9):

1. Click on Tools
2. Select Internet Options
3. Click on the Privacy tab
4. Click the Advanced button, under settings

5. Check Override automatics cookie handling check box
6. Select Block radio button under First Party cookies and Third Party Cookie

The above steps disable cookies, only for the internet zone. If you are testing code on your local machine, and to disable cookies for localhost:

1. Run the application
2. Press F12, to open developer tools
3. Then Select, Cache - Disable Cookies

### WebForm1.aspx HTML source:

```
<div style="font-family: Arial">
<table>
  <tr>
    <td colspan="2">
      <h1>
        This is WebForm1</h1>
      </td>
    </tr>
    <tr>
      <td>
        <b>Name</b>
      </td>
      <td>
        <:asp:TextBox ID="txtName" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td>
        <b>Email</b>
      </td>
      <td>
        <:asp:TextBox ID="txtEmail" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:Button ID="btnSendData" runat="server"
          Text="Go to WebForm2" onclick="btnSendData_Click" />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:Label ID="lblMessage" runat="server"
          ForeColor="Red" Font-Bold="true">
        </asp:Label>
      </td>
    </tr>
  </table>
```



</div>

### WebForm1.aspx.cs code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        // Check if the browser supports cookies
        if (Request.Browser.Cookies)
        {
            if (Request.QueryString["CheckCookie"] == null)
            {
                // Create the test cookie object
                HttpCookie cookie = new HttpCookie("TestCookie", "1");
                Response.Cookies.Add(cookie);
                // Redirect to the same webform
                Response.Redirect("WebForm1.aspx?CheckCookie=1");
            }
            else
            {
                //Check the existence of the test cookie
                HttpCookie cookie = Request.Cookies["TestCookie"];
                if (cookie == null)
                {
                    lblMessage.Text = "We have detected that, the cookies are disabled on your browser.  
Please enable cookies.";
                }
            }
        }
        else
        {
            lblMessage.Text = "Browser doesn't support cookies. Please install one of the modern  
browser's that support cookies.";
        }
    }
}

protected void btnSendData_Click(object sender, EventArgs e)
{
    // Create the cookie object
    HttpCookie cookie = new HttpCookie("UserDetails");
    cookie["Name"] = txtName.Text;
    cookie["Email"] = txtEmail.Text;
    // Cookie will be persisted for 30 days
    //cookie.Expires = DateTime.Now.AddDays(30);
    // Add the cookie to the client machine
    Response.Cookies.Add(cookie);

    Response.Redirect("WebForm2.aspx");
}
```

### WebForm2.aspx HTML Source:

```
<div style="font-family: Arial">
<table>
  <tr>
    <td colspan="2">
      <h1>This is WebForm2</h1>
    </td>
  </tr>
  <tr>
    <td>
      <b>Name</b>
    </td>
    <td>
      <asp:Label ID="lblName" runat="server">
        </asp:Label>
      </td>
  </tr>
  <tr>
    <td>
      <b>Email</b>
    </td>
    <td>
      <asp:Label ID="lblEmail" runat="server">
        </asp:Label>
      </td>
  </tr>
</table>
</div>
```

### WebForm2.aspx.cs Code

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie = Request.Cookies["UserDetails"];
    if (cookie != null)
    {
        lblName.Text = cookie["Name"];
        lblEmail.Text = cookie["Email"];
    }
}
```

## Asp.net session state

Just like Query strings, Session State variables can also be used to send data from one webform to another.

Points to remember about session state variables:

1. Session state variables are stored on the web server by default, and are kept for the life time of a session.
2. The default session state mode is InProc
3. The life time of a session is determined by the timeout value in web.config file. The default is 20 minutes. The time-out value can be adjusted according, to your application requirements.  
<sessionState mode="InProc" timeout="30"></sessionState>
4. Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data.
5. It is always a good practice to check, if a session state variable is null before calling any of its methods, such as ToString(). Otherwise, we may run into runtime NullReferenceExceptions.

```
if (Session["Name"] != null)
{
    lblName.Text = Session["Name"].ToString();
}
```

6. Application performance can be improved by disabling session state, if it's not required. Session state can be turned off at the page or application level.

To turn of the session state at the page level, set EnableSessionState="False" in the page directive  
<% @ Page Language="C#" EnableSessionState="False" AutoEventWireup="true"  
CodeBehind="WebForm1.aspx.cs" Inherits="AdoDemo.WebForm1" %>

To turn of the session state at the application level, set SessionState mode=false in web.config file.

```
<sessionState mode="Off"></sessionState>
```

### WebForm1.aspx HTML source

```
<div style="font-family: Arial">
<table>
  <tr>
    <td colspan="2">
      <h1>
        This is WebForm1</h1>
      </td>
    </tr>
    <tr>
      <td>
        <b>Name</b>
      </td>
      <td>
        <asp:TextBox ID="txtName" runat="server">
        </asp:TextBox>
      </td>
    </tr>
  </table>
```

```

</tr>
<tr>
    <td>
        <b>Email</b>
    </td>
    <td>
        :<asp:TextBox ID="txtEmail" runat="server">
        </asp:TextBox>
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Button ID="btnSendData" runat="server"
        Text="Go to WebForm2" onclick="btnSendData_Click" />
    </td>
</tr>
</table>
</div>

```

### WebForm1.aspx.cs code:

```

protected void btnSendData_Click(object sender, EventArgs e)
{
    Session["Name"] = txtName.Text;
    Session["Email"] = txtEmail.Text;
    Response.Redirect("WebForm2.aspx");
}

```

### WebForm2.aspx HTML Source:

```

<div style="font-family: Arial">
<table>
    <tr>
        <td colspan="2">
            <h1>This is WebForm2</h1>
        </td>
    </tr>
    <tr>
        <td>
            <b>Name</b>
        </td>
        <td>
            :<asp:Label ID="lblName" runat="server">
            </asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            <b>Email</b>
        </td>
        <td>
            :<asp:Label ID="lblEmail" runat="server">

```

```
        </asp:Label>
    </td>
</tr>
</table>
</div>
```

### WebForm2.aspx.cs code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["Name"] != null)
    {
        lblName.Text = Session["Name"].ToString();
    }
    if (Session["Email"] != null)
    {
        lblEmail.Text = Session["Email"].ToString();
    }
}
```

## Cookie less sessions in asp.net

By default sessions use cookies. The session-id is stored as a cookie on the client computer. This session-id, is then, used by the web-server to identify if the request is coming from the same user or a different user.

To demonstrate cookieless sessions. Run the application, and navigate to webform1.aspx. I am using google chrome as my default browser for visual studio. To set google chrome as default browser:

1. Right click on WebForm1.aspx in the solution explorer
2. Select Browse with option
3. From the list select Google chrome
4. Click "Set as Default" button
5. Finally click browse.

At this point, webform1.aspx will be opened using google chrome. Fill in the details for Name and Email fields, and click "Go To WebForm2" button. To view the session cookie:

1. Right click on the browser, and select "Inspect Element"
2. Click on the "Resources" button
3. Expand cookies in the "Resources"
4. Finally, select localhost
5. You should now see a cookie with ASP.NET\_SessionId

Now, let's disable cookies. To disable cookies in chrome:

1. Click on the Button, next to the address bar, in the browser and select "Settings"
2. In the "Search Setting" text box, type cookies.
3. In the search results, click "content settings" button under privacy
4. Under "cookies", select "Block sites from setting any data" and click OK.

So, the cookies are disabled now. Run the application, and navigate to WebForm1.aspx. Fill name and email fields and navigate to WebForm2.aspx. Notice that the Name and Email fields are not displayed. This is because, cookies are disabled. When cookies are disabled, the session-id is not sent to the server. So the server has no way to figure out, if the request for WebForm2.aspx has come from the same user. That is the reason why these fields are not displayed on WebForm2.aspx.

Some of the users, does not like websites writing information to their computers. So it is very common for, users to disable cookies. If that is the case, then websites using cookies, to manage sessions may not work as expected. However, to overcome this problem, cookieless sessions can be enabled. To enable cookieless sessions, set cookieless="true" in web.config as shown below.

```
<sessionState mode="InProc" cookieless="true"></sessionState>
```

With this change, navigate to WebForm1.aspx, fill in Name and Email fields, and then navigate to WebForm2.aspx, and notice that, the Name and Email, fields are displayed as expected. Notice, that the session-id is now part of the URL. This session-id is sent back and forth between the client and the web server, with every request and response. The web server, uses the session-id from the URL, to identify if the request has come from the same user or a different user.

For cookieless sessions to work correctly, relative URL's must be used in the application, when redirecting users to different webforms. For example, if you are on <http://google.com/WebForm1.aspx> and if you want to navigate to WebForm2.aspx, then use

`Response.Redirect("~/WebForm2.aspx")` - Relative URL  
and not

`Response.Redirect("http://google.com/WebForm2.aspx")` - Absolute URL (or Complete Path)

## Inproc asp.net session state mode management

Asp.net session state mode can have any of the following 4 values. Asp.net session state mode is set in web.config file:

1. Off - Disables session state for the entire application.
2. InProc

**InProc session state mode:** When the session state mode is set to InProc, the session state variables are stored on the web server memory inside the asp.net worker process. This is the default session state mode.

### Advantages of InProc session state mode:

1. Very easy to implement. All that is required is, to set, the session state mode=InProc in web.config file.
2. Will perform best because the session state memory is kept on the webserver, within the ASP.NET worker process(w3wp.exe).
3. Suitable for web applications hosted on a single server.
4. Objects can be added without serialization

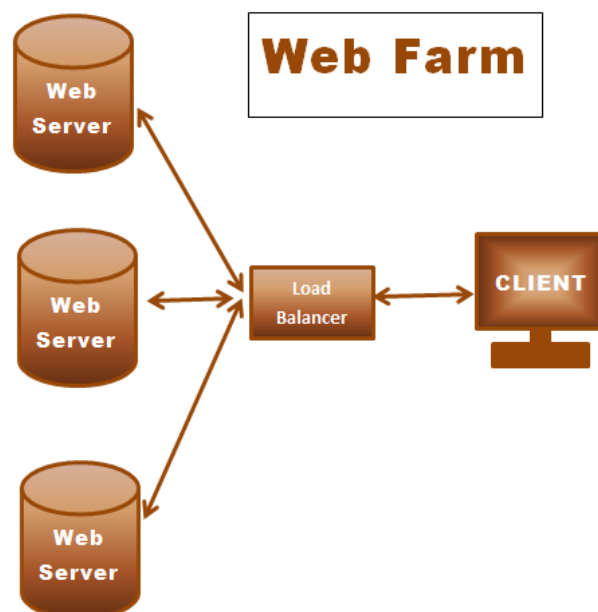
### Dis-advantages of InProc session state mode:

1. Session state data is lost, when the worker process or application pool is recycled.
2. Not suitable for web farms and web gardens.
3. Scalability could be an issue.

### Note:

Web Garden - Web application deployed on a server with multiple processors

Web Farm - Web application deployed on multiple server.





## **StateServer asp.net session state mode management**

Asp.net session state mode can have any of the following 4 values. Asp.net session state mode is set in web.config file:

1. Off - Disables session state for the entire application.
2. InProc
3. StateServer

When the session state mode is set to StateServer, the session state variables are stored in a process, called as asp.net state service. This process is different from the asp.net worker process. The asp.net state service can be present on a web server or a dedicated machine.

Steps to follow, to configure asp.net web application to use StateServer:

1. Start the ASP.NET state Service. To start the asp.net state service
  - a) Click Start > Type Run > Press Enter
  - b) In the run window, type services.msc and click OK.
  - c) In the services window, right click on ASP.NET State Service and select Start.
2. In web.config set sessionState mode="StateServer"
3. Set stateConnectionString="tcpip=StateServer:42424"  
Example: `<sessionState mode="StateServer" stateConnectionString="tcpip=localhost:42424" timeout="20"></sessionState>`

### **Advantages of using StateServer session state mode:**

1. ASP.NET worker process independent. Survives worker process restart.
2. Can be used with web farms and web gardens.
3. State server offers more scalability than InProc.

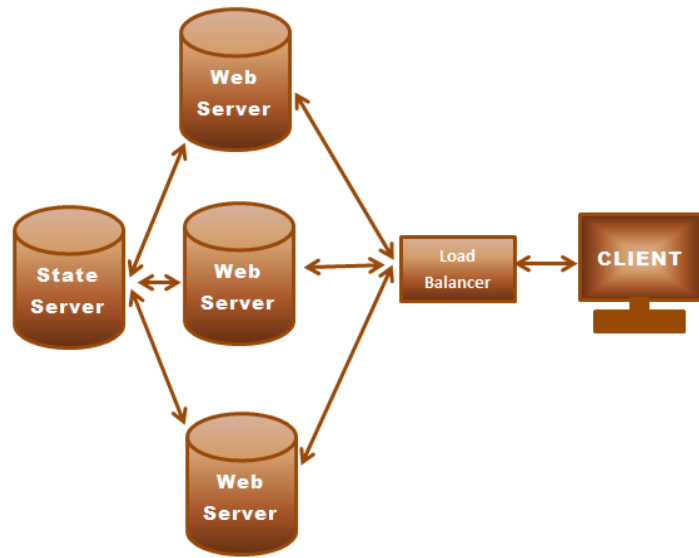
### **Dis-advantages of using StateServer session state mode:**

1. StateServer is slower than InProc
2. Complex objects, need to be serialized and deserialized
3. If the StateServer, is on a dedicated machine, and if the server goes down all the sessions are lost.

### **Note:**

Web Garden - Web application deployed on a server with multiple processors

Web Farm - Web application deployed on multiple server.



## SQLServer asp.net session state mode management

Asp.net session state mode can have any of the following 4 values. Asp.net session state mode is set in web.config file:

1. Off - Disables session state for the entire application.
2. InProc
3. StateServer
4. SQLServer
5. Custom - Enables you to build your own Session State provider. For example, Oracle.

When the session state mode is set to SQLServer, the session state variables are stored in a SQLServer database.

Steps to follow, to configure asp.net web application to use SQLServer:

1. Create the ASPState database using aspnet\_regsql.exe tool. There are several versions of this tool. I am running .NET version 4.0, on a 64 bit operating system. So I will use the version that is present in C:\Windows\Microsoft.NET\Framework64\v4.0.30319.

- a) click Start > Type Run > Press Enter
- b) Type cmd > Press Enter
- c) In the command prompt type - cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319
- d) Press Enter
- e) Type - aspnet\_regsql.exe -S SQLServerName -E -ssadd -sstype p
- f) Press Enter. At this point you should have ASPState Database added.
- g) For help running this tool, please refer to the following MSDN article  
[http://msdn.microsoft.com/en-us/library/ms229862\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/ms229862(v=vs.100).aspx)

2. Set the Session state mode=SQLServer and sqlConnectionString

If you want to use windows authentication

```
<sessionState mode="SQLServer"
sqlConnectionString="data source=SQLServerName; integrated security=SSPI"
timeout="20"></sessionState>
```

If you want to use sql server authentication

```
<sessionState mode="SQLServer"
sqlConnectionString="data source=SQLServerName; user id=sa; password=pass"
timeout="20"></sessionState>
```

Note: If you use integrated security(windows authentication), you might get an error stating "Failed to login to session state SQL server for user 'IIS APPPOOL\ASP.NET v4.0'.". To resolve this error:

- a) click Start > Type Run > Press Enter
- b) Type inetmgr > Press Enter
- c) Expand IIS and Click on Application Pools.
- d) Right click on ASP.NET v4.0 and select Advanced settings
- e) Change Process Model > Identity to LocalSystem and Click OK

### Advantages of using SQLServer session state mode:

1. SQLServer is the most reliable option. Survives worker process recycling and SQL Server

restarts.

2. Can be used with web farms and web gardens.
3. More scalable than State server and InProc session state modes.

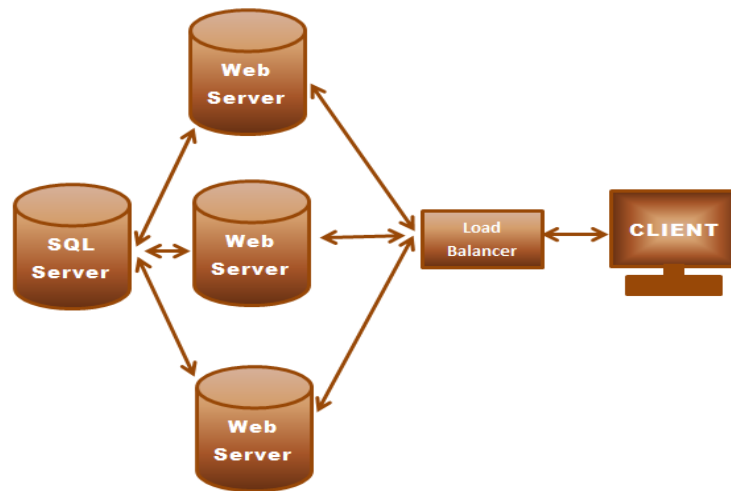
**Dis-advantages of using StateServer session state mode:**

1. Slower than StateServer and InProc session state modes
2. Complex objects, need to be serialized and deserialized

**Note:**

Web Garden - Web application deployed on a server with multiple processors

Web Farm - Web application deployed on multiple server.



## Asp.net Application state

1. Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data.
2. Application State variables are stored on the web server.
3. Application State variables are cleared, only when the process hosting the application is restarted, that is when the application ends.
4. Application State variables are not shared across a Web Farm or a Web Garden.
5. Application state variables are not thread safe. Lock and Unlock methods of the application class must be used to protect against race conditions, deadlocks, and access violations.

```
Application.Lock();  
Application["GlobalVariable"] = (int)Application["GlobalVariable"] + 1;  
Application.Unlock();
```

**Please Note:** In this example, we are using application state variables to send data from one web form to another. If the requirement, is just to send data from webform to another, you should consider other alternatives.

6. Use application state variables only, when the variables need to have global access and when you need them for entire time, during the life time of an application. Cache object, can be used, as an alternative, if you need to have global access for certain duration.

## Asp.net application state real time example

Application state variables are global, and all sessions have access to them. So, these variables can be used to track the number of users online. Every time a new user connects to your application, we want to increase the number of users online by 1. Along, the same lines, when ever a user session ends, then we need to decrease the number of users online by 1. But how do we know, when a new user connects to our application. Session\_Start() event is fired when ever a new session is established. When the session ends, Session\_End() event is fired. The event handlers are in global.asax file.

### Global.asax code:

```
public class Global : System.Web.HttpApplication
{
    void Application_Start(object sender, EventArgs e)
    {
        // Code that runs when the application starts
        Application["UsersOnline"] = 0;
    }
    void Session_Start(object sender, EventArgs e)
    {
        // Code that runs when a new user session is started
        Application.Lock();
        Application["UsersOnline"] = (int)Application["UsersOnline"] + 1;
        Application.Unlock();
    }
    void Session_End(object sender, EventArgs e)
    {
        // Code that runs when an existing user session ends.
        Application.Lock();
        Application["UsersOnline"] = (int)Application["UsersOnline"] - 1;
        Application.Unlock();
    }
}
```

The application state variable is accessible across the entire application. Copy and paste the following code in the Page\_Load() event of any webform.

```
if (Application["UsersOnline"] != null)
{
    Response.Write("Number of Users Online = " +
        Application["UsersOnline"].ToString());
}
```

By default, the browser instances share the session cookie. To have a new session id assigned, when a new browser instance requests the webform, set cookieless="true" for the sessionstate element in web.config. Now run the application. The following message should be displayed.

Number of Users Online = 1

Open a new browser window, copy and paste the URL from the other window. Make sure to delete the session-id, so the web server, assigns a new session-id to the second request. At this point, Number of Users Online should be incremented to 2.

## Exception handling in asp.net

Exceptions are unforeseen errors that happen within the logic of an application. For example, when reading a file, a number of exception conditions can occur:

1. The file might not exist
2. You may not have permissions to access the file

When an exception occurs and if it is not handled, then, that exception is called as an, unhandled exception. An unhandled exception is displayed to the user using an "yellow screen of death".

Displaying the screen of death is bad for 2 reasons:

1. The error messages are cryptic and may not make any sense to the end user
2. The exception information may be useful for a hacker, to hack into your application

Exception handling using - try-catch

**try** - Wrap the code in a try block that could possibly cause an exception. If a statement in the try block causes an exception, the control will be immediately transferred to the catch block.

**catch** - catches the exception and tries to correct the error and/or handles the exception

**finally** - Used to free resources. Finally block is guaranteed to execute irrespective of whether an exception has occurred or not

**throw** - used to raise an exception

The base class for all exceptions is the Exception class. Specific exceptions should be caught, before catching the general parent exception.

Create an asp.net web application, and add an xml file, Countries.xml. Copy and paste the following XML.

```
<?xml version="1.0" encoding="utf-8" ?>
<Countries>
  <Country>
    <Id>101</Id>
    <Name>India</Name>
    <Continent>Asia</Continent>
  </Country>
  <Country>
    <Id>102</Id>
    <Name>UK</Name>
    <Continent>Europe</Continent>
  </Country>
  <Country>
    <Id>103</Id>
    <Name>US</Name>
    <Continent>North America</Continent>
  </Country>
  <Country>
```

```

<Id>104</Id>
<Name>France</Name>
<Continent>Europe</Continent>
</Country>
</Countries>

```

Drag and drop, gridview and a label control on WebForm1.aspx. HTML of the aspx page.

```

<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
<br />
<asp:Label ID="lblError" Font-Bold="true" ForeColor="Red" runat="server"></asp:Label>

```

### WebForm1.aspx.cs Code:

```

public partial class WebForm1 : System.Web.UI.Page
{
    // Uncomment this line to print the name of the account
    // used to run the application code
    // Response.Write(System.Security.Principal.WindowsIdentity.GetCurrent().Name);
    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            DataSet ds = new DataSet();
            ds.ReadXml(Server.MapPath("~/Countries.xml"));
            GridView1.DataSource = ds;
            GridView1.DataBind();
        }
        // Catch specific exceptions first
        catch (System.UnauthorizedAccessException unauthorizedAccessException)
        {
            //Log the exception information
            lblError.Text = "Access to the file denied";
        }
        catch (System.IO.FileNotFoundException fileNotFoundException)
        {
            //Log the exception information
            lblError.Text = "File does not exist";
        }
        catch (Exception ex)
        {
            //Log the exception information
            lblError.Text = "There is an unknown problem. IT team is working on this issue. Please check back after some time";
        }
        finally
        {
            // Code to clean up resources like closing file handles
            // and database connection objects
        }
    }
}

```



}  
}

## Error events in asp.net

Handling exceptions using try/catch blocks is commonly termed as structured exception handling. Asp.net provide 2 error events:

**Page\_Error** - This event is raised at the page level, when there is an unhandled exception on the page. The event handler resides on the page.

**Application\_Error** - This event is raised at the application level, when there is an unhandled exception at an application level. The event handler resides in Global.asax file

These error events can be used as a substitute or supplemental to structured exceptional handling. Create an asp.net web application. Add a webform with name Errors.aspx. Copy and paste the following HTML in Errors.aspx.

```
<div style="font-family: Arial">
  <table style="border:1px solid black">
    <tr>
      <td style="color:Red">
        <h2>Application Error</h2>
      </td>
    </tr>
    <tr>
      <td>
        <h3>
          An unkown error has occured. We are aware of it and the IT team is currently working
          on this issue. Sorry for the inconvinience caused.</h3>
        </td>
      </tr>
    <tr>
      <td>
        <h5>
          If you need further assistance, please contact our helpdesk at
          helpdesk@companyhelpdesk.com
        </h5>
      </td>
    </tr>
  </table>
</div>
```

Add WebForm1.aspx to the project. Drag and drop a gridview control. Copy and paste the following code in Webform1.aspx.cs.

```
protected void Page_Load(object sender, EventArgs e)
{
    DataSet ds = new DataSet();
    ds.ReadXml(Server.MapPath("~/Data/Countries.xml"));

    GridView1.DataSource = ds;
    GridView1.DataBind();
}

protected void Page_Error(object sender, EventArgs e)
```

```

{
    // Get the exception details and log it in the database or event viewer
    Exception ex = Server.GetLastError();
    // Clear the exception
    Server.ClearError();
    // Redirect user to Error page
    Response.Redirect("Errors.aspx");
}

```

The code tries to read xml data from Countries.xml file. At the moment, the file is not present and we get a FileNotFoundException. Since this exception is not handled using a try/catch block in the Page\_Load event, the error get to the page level and is handled by Page\_Error event handler. In Page\_Error event handler:

1. We get the exception information using Server.GetLastError() method.
2. Do something with the exception, such as redirect the user to a generic error page, display an error message on the same page which caused the exception, try to correct the problem, or log the exception to a database table or event viewer and notify the development team.
3. We then clear, the exception, so that it is not get propagated to the application level.
4. Finally we redirect the user to a generic error page, Errors.aspx

Please note that:

1. If the exception is not cleared in the Page\_Error event, it gets propagated to the application level, and Application\_Error event handler gets executed. If we are not clearing the exception at the application level, the application crashes with the "Yellow Screen of Death".
2. If the exception is cleared and redirection to Errors.aspx is not done, then a blank page is displayed. This is because web form processing is immediately stopped when an exception occurs.

If an exception is not handled at the page level using Page\_Error event, it get's to the application level and can be handled using the Application\_Error event handler in Global.asax and can be used as a single, centralized location for error handling.

## Custom errors in asp.net

If there is an unhandled exception, by default, the generic yellow screen of death is displayed. Instead, custom error pages can be displayed. Custom error pages can be defined at 2 levels:

1. Application Level - In the web.config file using "customErrors" element.
  2. Page Level - In the Page directive, using "ErrorPage" attribute.
- Page level custom error pages takes precedence over application level custom error pages.

Custom error pages provide the flexibility of displaying a specific page in response to one or more of the available HTTP status codes. For a list of all the available HTTP status please visit the following Article.

[http://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](http://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

To specify the custom error pages at an application level, use customErrors element in web.config

```
<customErrors mode="On" defaultRedirect="DefaultErrorPage.aspx">
  <error statusCode="401" redirect="UnauthorizedErrorPage.aspx" />
  <error statusCode="404" redirect="PageNotFoundErrorPage.aspx" />
  <error statusCode="500" redirect="InternalServerErrorPage.aspx" />
</customErrors>
```

The mode attribute determines when a custom error page is displayed over the yellow screen of death, exception page. Mode attribute can have On, Off, or RemoteOnly. Default is RemoteOnly.

**On** - Custom error pages are displayed both on local and remote machines

**Off** - Custom error pages are not displayed anywhere

**RemoteOnly** - Custom error pages are displayed on remote machines, and exception page on local machine

If the redirection is done in Application\_Error() event handler in Global.asax, custom error pages will have no effect.

In your application, if you have to display specific custom error pages for specific http status codes, then use custom errors. If you just have one generic error page, then Global.asax can be used.

Please note that, the exception object needs to be retrieved, before the user is redirected to a custom error page. Because a custom error page is displayed through redirection, the context for the error is lost and Server.GetLastError returns nothing from the target custom error page.

## Windows event viewer

Exceptions in an asp.net web application can be logged to the event viewer. First let us discuss about the event viewer and create custom event log and event source. To access the event viewer:

1. Click on Start
2. Type "Run" and press enter
3. In the "Run" window type "eventvwr" and press enter
4. This should open event viewer

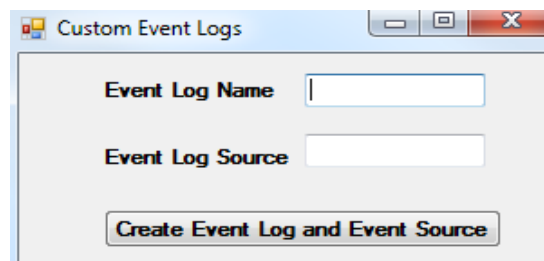
Under windows logs, you should see:

1. Application - Logs information from applications like MS Office, SQL Server, Visual Studio etc.
2. Security - Logs information related to security like user sign-ons, access checks etc
3. System - Logs information related to driver, system service failures etc.

Let us now create a windows application that can be used to create a custom windows event log and event source. The Event Source is the name of the application that logged the event. The event source is displayed in the details pane, when an event is selected in the event viewer.

Steps to create the windows application to create the custom windows event log and event source:

1. Drag and drop two textboxes, two labels, and a button control on to the windows form
2. Arrange the controls, so the form looks as shown below.



3. Double click the button control to generate the event handler
4. Copy and paste the following code in the click event handler

```
private void CreateEventLogButton_Click(object sender, EventArgs e)
{
    if (EventLogNameTextBox.Text != string.Empty && EventLogSourceTextBox.Text !=
string.Empty)
    {
        System.Diagnostics.EventLog.CreateEventSource
            (EventLogSourceTextBox.Text, EventLogNameTextBox.Text);
        MessageBox.Show("Event Log and Source Created");
    }
    else
    {
        MessageBox.Show("Event Log and Source are required");
    }
}
```

5. Run the application
6. Enter the name and source for the event log.
7. Click "Create Event Log and Event Source Button" button

Open the event viewer. The newly created "event log" should be under Applications and Service Logs. If you are not able to locate them, restart your machine.

To delete the custom event log, use the Delete() method:

```
System.Diagnostics.EventLog.Delete("EventLogNameToDelete")
```

## Logging exceptions to the windows eventviewer

Create an asp.net web application project. Add WebForm1.aspx. Drag and drop gridview control.

Copy and paste the following code.

```
//try
//{
// DataSet is System.Data namespace
DataSet ds = new DataSet();
// This line throws FileNotFoundException
ds.ReadXml(Server.MapPath("~/Data/Countries.xml"));
```

```
GridView1.DataSource = ds;
```

```
GridView1.DataBind();
```

```
//}
```

```
//catch (Exception ex)
```

```
//{
```

```
//  Logger.Log(ex);
```

```
//}
```

Add a webform with name - Errors.aspx.

```
<div style="font-family: Arial">
```

```
  <table style="border:1px solid black">
```

```
    <tr>
```

```
      <td style="color:Red">
```

```
        <h2>Application Error</h2>
```

```
      </td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>
```

```
        <h3>
```

```
          An unkown error has occured. We are aware of it and the IT team is currently working
          on this issue. Sorry for the inconvenience caused.</h3>
```

```
        </td>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>
```

```
        <h5>
```

```
          If you need further assistance, please contact our helpdesk at
          helpdesk@companyhelpdesk.com
```

```
        </h5>
```

```
      </td>
```

```
    </tr>
```

```
  </table>
```

```
</div>
```

Add a class file with name Logger.cs

```
public class Logger
```

```
{
```

```
  public static void Log(Exception exception)
```

```
  {
```

```

// Create an instance of StringBuilder. This class is in System.Text namespace
StringBuilder sbExceptionMessage = new StringBuilder();
sbExceptionMessage.Append("Exception Type" + Environment.NewLine);
// Get the exception type
sbExceptionMessage.Append(exception.GetType().Name);
// Environment.NewLine writes new line character - \n
sbExceptionMessage.Append(Environment.NewLine + Environment.NewLine);
sbExceptionMessage.Append("Message" + Environment.NewLine);
// Get the exception message
sbExceptionMessage.Append(exception.Message + Environment.NewLine +
Environment.NewLine);
sbExceptionMessage.Append("Stack Trace" + Environment.NewLine);
// Get the exception stack trace
sbExceptionMessage.Append(exception.StackTrace + Environment.NewLine +
Environment.NewLine);

// Retrieve inner exception if any
Exception innerException = exception.InnerException;
// If inner exception exists
while (innerException != null)
{
    sbExceptionMessage.Append("Exception Type" + Environment.NewLine);
    sbExceptionMessage.Append(innerException.GetType().Name);
    sbExceptionMessage.Append(Environment.NewLine + Environment.NewLine);
    sbExceptionMessage.Append("Message" + Environment.NewLine);
    sbExceptionMessage.Append(innerException.Message + Environment.NewLine +
Environment.NewLine);
    sbExceptionMessage.Append("Stack Trace" + Environment.NewLine);
    sbExceptionMessage.Append(innerException.StackTrace + Environment.NewLine +
Environment.NewLine);

    // Retrieve inner exception if any
    innerException = innerException.InnerException;
}

// If the Event log source exists
if (EventLog.SourceExists("PragimTech.com"))
{
    // Create an instance of the eventlog
    EventLog log = new EventLog("PragimTech");
    // set the source for the eventlog
    log.Source = "PragimTech.com";
    // Write the exception details to the event log as an error
    log.WriteEntry(sbExceptionMessage.ToString(), EventLogEntryType.Error);
}
}
}

```

Copy and pste the following code, in Global.asax, in Application\_Error() event handler

```

if (Server.GetLastError() != null)
{

```



```
// Get and Log the exception
Logger.Log(Server.GetLastError());
// Clear the exception
Server.ClearError();
// Transfer the user to Errors.aspx page
Server.Transfer("Errors.aspx");
}
```

Run the application. WebForm1.aspx throws an exception. Since this exception is not handled anywhere it gets propagated till the application level, and Application\_Error() event handler will be executed. This should log the exception to the windows eventviewer.

## Logging exceptions as information entry type in windows eventviewer

Create an asp.net web application and add a class file with name Logger.cs.

```
public class Logger
{
    public static void Log(Exception exception)
    {
        Log(exception, EventLogEntryType.Error);
    }
    public static void Log(Exception exception, EventLogEntryType eventLogEntryType)
    {
        do
        {
            sbExceptionMessage.Append("Exception Type" + Environment.NewLine);
            sbExceptionMessage.Append(exception.GetType().Name);
            sbExceptionMessage.Append(Environment.NewLine + Environment.NewLine);
            sbExceptionMessage.Append("Message" + Environment.NewLine);
            sbExceptionMessage.Append(exception.Message + Environment.NewLine +
Environment.NewLine);
            sbExceptionMessage.Append("Stack Trace" + Environment.NewLine);
            sbExceptionMessage.Append(exception.StackTrace + Environment.NewLine +
Environment.NewLine);

            exception = exception.InnerException;
        }
        while (exception != null);

        // If the Event log source exists
        if (EventLog.SourceExists("PragimTech.com"))
        {
            // Create an instance of the eventlog
            EventLog log = new EventLog("PragimTech");
            // set the source for the eventlog
            log.Source = "PragimTech.com";
            // Write the exception details to the event log as an error
            log.WriteEntry(sbExceptionMessage.ToString(), eventLogEntryType);
        }
    }
}
```

Add a webform, with name Calculator.aspx and copy the following HTML.

```
<div style="font-family: Arial">
    <table style="border: 1px solid black">
        <tr>
            <td><b>First Number</b></td>
            <td>
                <asp:TextBox ID="txtFirstNumber" runat="server"></asp:TextBox>
            </td>
        </tr>
    </table>
```

```

        <td><b>Second Number</b></td>
        <td>
            <asp:TextBox ID="txtSecondNumber" runat="server"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Button ID="btnDivide" runat="server" Text="Divide" onclick="btnDivide_Click"
            />
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <asp:Label ID="lblMessage" runat="server" Font-Bold="true"></asp:Label>
        </td>
    </tr>
</table>
</div>

```

### Calculator.aspx.cs code:

```

protected void btnDivide_Click(object sender, EventArgs e)
{
    try
    {
        int FirstNumber = Convert.ToInt32(txtFirstNumber.Text);
        int SecondNumber = Convert.ToInt32(txtSecondNumber.Text);

        lblMessage.ForeColor = System.Drawing.Color.Navy;
        int Result = FirstNumber / SecondNumber;
        lblMessage.Text = Result.ToString();
    }
    catch (FormatException formatException)
    {
        Logger.Log(formatException, EventLogEntryType.Information);
        lblMessage.ForeColor = System.Drawing.Color.Red;
        lblMessage.Text = "Only numbers are allowed";
    }
    catch (OverflowException overflowException)
    {
        Logger.Log(overflowException, EventLogEntryType.Information);
        lblMessage.ForeColor = System.Drawing.Color.Red;
        lblMessage.Text = "Numbers must be between " + Int32.MinValue.ToString() + " and " +
Int32.MaxValue.ToString();
    }
    catch (DivideByZeroException divideByZeroException)
    {
        Logger.Log(divideByZeroException, EventLogEntryType.Information);
        lblMessage.ForeColor = System.Drawing.Color.Red;
        lblMessage.Text = "Denominator cannot be ZERO";
    }
    catch (Exception exception)

```

```
{  
    Logger.Log(exception);  
    lblMessage.ForeColor = System.Drawing.Color.Red;  
    lblMessage.Text = "An unknown problem has occurred. Please try later";  
}  
}
```

FormatException, OverflowException, DivideByZeroException should now be logged as "Information" entry type instead of "Error" entry type in the event viewer.

## Logging exception to database

we will discuss about logging exceptions to a database table. The first step is to create the required table, to log the exceptions.

### Script to create table – tblLog

```
Create table tblLog
(
  [Id] int primary key identity(1,1),
  [Date] DateTime,
  [ExceptionMessage] nvarchar(max)
)
Stored procedure to log the exception
create procedure spInsertLog
@ExceptionMessage nvarchar(max)
as
begin
insert into tblLog([Date], [ExceptionMessage])
values (Getdate(), @ExceptionMessage)
end
```

Database connection string in web.config. Change it accordingly, to connect to your sql server.

```
<connectionStrings>
  <add name="DBConnectionString"
    connectionString="data source=.; database=Sample; Integrated Security=SSPI"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Add a class file with name - Logger.cs.

```
public class Logger
{
  public static void Log(Exception exception)
  {
    StringBuilder sbExceptionMessage = new StringBuilder();

    do
    {
      sbExceptionMessage.Append("Exception Type" + Environment.NewLine);
      sbExceptionMessage.Append(exception.GetType().Name);
      sbExceptionMessage.Append(Environment.NewLine + Environment.NewLine);
      sbExceptionMessage.Append("Message" + Environment.NewLine);
      sbExceptionMessage.Append(exception.Message + Environment.NewLine
+ Environment.NewLine);
      sbExceptionMessage.Append("Stack Trace" + Environment.NewLine);
      sbExceptionMessage.Append(exception.StackTrace + Environment.NewLine
+ Environment.NewLine);

      exception = exception.InnerException;
    }
  }
}
```

```

        while (exception != null);

        LogToDB(sbExceptionMessage.ToString());
    }

    private static void LogToDB(string log)
    {
        // ConfigurationManager class is in System.Configuration namespace
        string connectionString =
        ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;
        // SqlConnection is in System.Data.SqlClient namespace
        using (SqlConnection con = new SqlConnection(connectionString))
        {
            SqlCommand cmd = new SqlCommand("spInsertLog", con);
            // CommandType is in System.Data namespace
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter parameter = new SqlParameter("@ExceptionMessage", log);
            cmd.Parameters.Add(parameter);

            con.Open();
            cmd.ExecuteNonQuery();
            con.Close();
        }
    }
}

```

Add a webform with name - Errors.aspx.

```

<div style="font-family: Arial">
    <table style="border:1px solid black">
        <tr>
            <td style="color:Red">
                <h2>Application Error</h2>
            </td>
        </tr>
        <tr>
            <td>
                <h3>
                    An unkown error has occured. We are aware of it and the IT team is currently working
                    on this issue. Sorry for the inconvenience caused.</h3>
                </td>
            </tr>
        <tr>
            <td>
                <h5>
                    If you need further assistance, please contact our helpdesk at
                    helpdesk@companyhelpdesk.com
                </h5>
            </td>
        </tr>
    </table>

```

</div>

Add a webform to the project. Drag and drop a gridview control. Copy and paste the following code in the code behind file, in the Page\_Load() event.

```
//try
//{
    // DataSet is in System.Data namespace
    DataSet ds = new DataSet();
    // This line will throw an exception, as Data.xml
    // file is not present in the project
    ds.ReadXml(Server.MapPath("~/Data.xml"));
    GridView1.DataSource = ds;
    GridView1.DataBind();
//}
//catch (Exception ex)
//{
    // Logger.Log(ex);
    // Server.Transfer("~/Errors.aspx");
//}
```

Global.asax code:

```
void Application_Error(object sender, EventArgs e)
{
    if (Server.GetLastError() != null)
    {
        // Log the exception
        Logger.Log(Server.GetLastError());
        // Clear the exception
        Server.ClearError();
        // Transfer the user to Errors.aspx page
        Server.Transfer("Errors.aspx");
    }
}
```

Run the application. The exception should be logged to the Database table - tblLog and the user will be redirected to Errors.aspx page.

## Customizing asp.net exception Logging

The ability to log exceptions must be configurable in web.config

Add a key to web.config file using appsettings element. This key determines, where the exceptions are logged.

```
<appSettings>
  <!--LogProvider = Database|Event Viewer|Both-->
  <add key="LogProvider" value="Both"/>
</appSettings>
```

The "Logger" class reads the "LogProvider" key from web.config and logs the exceptions accordingly.

```
public class Logger
{
    public static void Log(Exception exception)
    {
        StringBuilder sbExceptionMessage = new StringBuilder();

        do
        {
            sbExceptionMessage.Append("Exception Type" + Environment.NewLine);
            sbExceptionMessage.Append(exception.GetType().Name);
            sbExceptionMessage.Append(Environment.NewLine + Environment.NewLine);
            sbExceptionMessage.Append("Message" + Environment.NewLine);
            sbExceptionMessage.Append(exception.Message + Environment.NewLine
+ Environment.NewLine);
            sbExceptionMessage.Append("Stack Trace" + Environment.NewLine);
            sbExceptionMessage.Append(exception.StackTrace + Environment.NewLine
+ Environment.NewLine);

            exception = exception.InnerException;
        }
        while (exception != null);

        string logProvider = ConfigurationManager.AppSettings["LogProvider"];
        if (logProvider.ToLower() == "both")
        {
            LogToDB(sbExceptionMessage.ToString());
            LogToEventViewer(sbExceptionMessage.ToString());
        }
        else if (logProvider.ToLower() == "database")
        {
            LogToDB(sbExceptionMessage.ToString());
        }
        else if (logProvider.ToLower() == "eventviewer")
        {
            LogToEventViewer(sbExceptionMessage.ToString());
        }
    }
}
```



```

private static void LogToDB(string log)
{
    // ConfigurationManager class is in System.Configuration namespace
    string connectionString =
ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;
    // SqlConnection is in System.Data.SqlClient namespace
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        SqlCommand cmd = new SqlCommand("spInsertLog", con);
        // CommandType is in System.Data namespace
        cmd.CommandType = CommandType.StoredProcedure;

        SqlParameter parameter = new SqlParameter("@ExceptionMessage", log);
        cmd.Parameters.Add(parameter);

        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
    }
}

private static void LogToEventViewer(string log)
{
    if (EventLog.SourceExists("PragimTech.com"))
    {
        // Create an instance of the eventlog
        EventLog eventLog = new EventLog("PragimTech");
        // set the source for the eventlog
        eventLog.Source = "PragimTech.com";
        // Write the exception details to the event log as an error
        eventLog.WriteEntry(log, EventLogEntryType.Error);
    }
}
}

```

## Sending emails using asp.net

To compose the email message, use MailMessage class. To send the email use, SmtpClient class. Both of these classes are present in System.Net.Mail namespace.

```
public static void SendEmail(string emailbody)
{
    // Specify the from and to email address
    MailMessage mailMessage = new MailMessage("from_email@gmail.com",
    "To_Email@gmail.com");
    // Specify the email body
    mailMessage.Body = emailbody;
    // Specify the email Subject
    mailMessage.Subject = "Exception";

    // Specify the SMTP server name and port number
    SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587);
    // Specify your gmail address and password
    smtpClient.Credentials = new System.Net.NetworkCredential()
    {
        UserName = "from_email@gmail.com", Password = "your_password"
    };
    // Gmail works on SSL, so set this property to true
    smtpClient.EnableSsl = true;
    // Finally send the email message using Send() method
    smtpClient.Send(mailMessage);
}
```

SendEmail() method can then be called in our Logger class. Pass in the exception string as an input parameter.

```
SendEmail(sbExceptionMessage.ToString());
```

If you want the capability of sending emails to be configurable, add the following key in web.config.

```
<appSettings>
  <add key="SendEmail" value="true"/>
</appSettings>
```

Read "SendEmail" key from web.config. If SendEmail is set to true only then, send the email.

```
string sendEmail = ConfigurationManager.AppSettings["SendEmail"];
if (sendEmail.ToLower() == "true")
{
    SendEmail(sbExceptionMessage.ToString());
}
```

we discussed about sending emails using gmail smtp server and credentials. In reality organisations have their own SMTP server. If you want to send emails using your own SMTP server, use the respective smtp server address and credentials.

## Sending emails in asp.net using SMTP server settings from web.config

All the SMTP server settings were configured in code. we will discuss about specifying these settings in web.config file.

Web.config settings for SMTP server. All the attributes here are self explanatory.

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network" >
      <network host="smtp.gmail.com" enableSsl="true" port="587"
        userName="your_email@gmail.com" password="your_password"/>
    </smtp>
  </mailSettings>
</system.net>
```

Since the SMTP server settings are now configured in web.config. In code all you have to do is :

1. Create an instance of the MailMessage class. Specify the FROM & TO email address, subject and Body
2. Create an instance of SmtClient class, and send the email using Send() method. The SMTP settings will be automatically picked up from web.config file, when sending email.

SendEmail() method code

```
public static void SendEmail(string emailbody)
{
    // Specify the from and to email address
    MailMessage mailMessage = new MailMessage
        ("from_email@gmail.com", "to_email@gmail.com");
    // Specify the email body
    mailMessage.Body = emailbody;
    // Specify the email Subject
    mailMessage.Subject = "Exception";

    // No need to specify the SMTP settings as these
    // are already specified in web.config
    SmtClient smtpClient = new SmtClient();
    // Finall send the email message using Send() method
    smtpClient.Send(mailMessage);
}
```

## Tracing in asp.net

Tracing enables us to view diagnostic information about a request and is very useful when debugging application problems.

Tracing can be turned on or off :

1. At the application level or
2. At the page level

To enable tracing at the application level set "trace" element's "enabled" attribute to "true" in web.config. This enables tracing for all pages in the application.

```
<trace enabled="true"/>
```

To disable tracing for specific pages, set Trace="false" in the webform's "Page" directive

```
<%@ Page Language="C#" Trace="false" AutoEventWireup="true"  
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1" %>
```

If tracing is enabled at the application level, the trace messages are written to a file called trace.axd. Trace.axd file can only be accessed locally. To make the trace file available to remote users set localOnly="false". Tracing displays sensitive information, that could be useful to a hacker to hack into the server. So set this attribute to "false" only when it is required.

```
<trace enabled="true" localOnly="false"/>
```

To append trace messages at the end of the page set pageOutput="true".

```
<trace enabled="true" pageOutput="true" localOnly="false"/>
```

Use RequestLimit attribute to control the number of trace requests that are stored on the server. The default is 10. After this limit is reached, the sever will stop writing to the trace file.

```
<trace enabled="true" pageOutput="true" requestLimit="5" localOnly="false"/>
```

If you want to log trace messages, even after the requestLimit has reached set mostRecent attribute to true. When this attribute is set to true, the old entries in the trace log are discarded and the new entries get added.

```
<trace enabled="true" mostRecent="true" requestLimit="3"/>
```

## Writing custom asp.net tracing messages

To write custom asp.net trace messages `Trace.Write()` and `Trace.Warn()` methods can be used. The only difference, between these 2 methods is that, the messages written using `Trace.Warn()` are displayed in red colour, where as the messages written using `Trace.Write()` are displayed in black colour. In fact, What is the difference between `Trace.Write()` and `Trace.Warn()` is a very common asp.net interview question.

### Webform1.aspx HTML:

```
<div style="font-family: Arial">
  <table style="border: 1px solid black">
    <tr>
      <td>
        <b>First Number</b>
      </td>
      <td>
        <asp:TextBox ID="txtFirstNumber" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td>
        <b>Second Number</b>
      </td>
      <td>
        <asp:TextBox ID="txtSecondNumber" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:Button ID="btnDivide" runat="server" Text="Divide"
        OnClick="btnDivide_Click" />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:Label ID="lblMessage" runat="server" Font-Bold="true">
        </asp:Label>
      </td>
    </tr>
  </table>
</div>
```

### WebForm1.aspx.cs code behind:

```
protected void btnDivide_Click(object sender, EventArgs e)
{
    try
    {
        int FirstNumber = Convert.ToInt32(txtFirstNumber.Text);
```

```

int SecondNumber = Convert.ToInt32(txtSecondNumber.Text);

lblMessage.ForeColor = System.Drawing.Color.Navy;
int Result = FirstNumber / SecondNumber;
lblMessage.Text = Result.ToString();
}
catch (FormatException formatException)
{
    lblMessage.ForeColor = System.Drawing.Color.Red;
    lblMessage.Text = "Only numbers are allowed";
    // Check if tracing is enabled
    if (Trace.IsEnabled)
    {
        // Write the exception message to the trace log
        Trace.Write(formatException.Message);
    }
}
catch (OverflowException overflowException)
{
    lblMessage.ForeColor = System.Drawing.Color.Red;
    lblMessage.Text = "Numbers must be between " + Int32.MinValue.ToString() + " and " +
Int32.MaxValue.ToString();
    if (Trace.IsEnabled)
    {
        Trace.Warn(overflowException.Message);
    }
}
catch (DivideByZeroException divideByZeroException)
{
    lblMessage.ForeColor = System.Drawing.Color.Red;
    lblMessage.Text = "Denominator cannot be ZERO";
    if (Trace.IsEnabled)
    {
        Trace.Warn(divideByZeroException.Message);
    }
}
catch (Exception exception)
{
    lblMessage.ForeColor = System.Drawing.Color.Red;
    lblMessage.Text = "An unknown problem has occurred. Please try later";
    if (Trace.IsEnabled)
    {
        Trace.Warn(exception.Message);
    }
}
}

```

Trace.IsEnabled property can be used to check if tracing is enabled. In the following code Trace.Write() method is invoked only if tracing is enabled. If you don't check, Trace.IsEnabled property prior to writing out trace messages, we don't get an exception. But if you are going to do any sort of significant work to build the trace message, then you can avoid that work by checking the IsEnabled property first.

```
if (Trace.IsEnabled)
{
    Trace.Write("Debugging information");
}
```

With classic ASP, the only option for printing debugging information is Response.Write(). There are 2 problems with this:

1. Actual end users also, can see the debugging information that you have written using Response.Write(). But with tracing, if pageoutput attribute is set to false, then the trace messages are written to the trace.axd file. End users cannot see the trace information.
2. All the Response.Write() statements must be removed, before the application is deployed to production. But with tracing, all you have to do is disable tracing in web.config.

## Tracing in asp.net - A real time example

### ASP.NET Page is very slow. What will you do to make it fast?

This is a very common interview question. There are several reasons for the page being slow. We need to identify the cause.

We cannot debug an application on the production server, as we will usually, not have visual studio installed, and as the code is optimized for release builds.

#### Step 1:

First find out which is slow, is it the application or the database : If the page is executing SQL queries or stored procedures, run those on the database and check how long do they take to run. Alternatively, SQL profiler, can be used, to inspect the queries and the time they take. If the queries are taking most of the time, then you know you have to tune the queries for better performance. To tune the queries, there are several ways and I have listed some of them below.

- a) Check if there are indexes to help the query
- b) Select only the required columns, avoid Select \*.
- c) Check if there is a possibility to reduce the number of joins
- d) If possible use NO LOCK on your select statements
- e) Check if there are cursors and if you can replace them with joins

#### Step 2:

If the queries are running fast, then we know it is the application code that is causing the slowness. Isolate the page event that is causing the issue by turning tracing on. To turn tracing on, set Trace="true" in the page directive. Once you have tracing turned on you should see trace information at the bottom of the page or in trace.axd file. From the trace information, it should be clear to identify the piece of code that is taking the maximum time.

Stored Procedures used

```
Create proc spGetEmployees
as
begin
select Id, Name, Gender, DeptName
from tblEmployees
end
```

```
Create proc spGetEmployeesByGender
as
begin
select Gender, Count(Gender) as Total
from tblEmployees
Group by Gender
end
```

```
Create proc spGetEmployeesByDepartment
as
begin
select DeptName, Count(DeptName) as Total
```



```
from tblEmployees
Group by DeptName
end
```

### ASPX page HTML:

```
<div style="font-family: Arial">
  <b>All employees</b>
  <asp:GridView ID="gvAllEmployees" runat="server">
  </asp:GridView>
  <br /><br />
  <b>Total Employees by Department</b>
  <asp:GridView ID="gvEmployeesByDepartment" runat="server">
  </asp:GridView>
  <br /><br />
  <b>Total Employees by Gender</b>
  <asp:GridView ID="gvEmployeesByGender" runat="server">
  </asp:GridView>
</div>
```

### Code Behind:

```
protected void Page_Load(object sender, EventArgs e)
{
    Trace.Warn("GetAllEmployees() started");
    GetAllEmployees();
    Trace.Warn("GetAllEmployees() Complete");

    Trace.Warn("GetEmployeesByGender() started");
    GetEmployeesByGender();
    Trace.Warn("GetEmployeesByGender() Complete");

    Trace.Warn("GetEmployeesByDepartment() started");
    GetEmployeesByDepartment();
    Trace.Warn("GetEmployeesByDepartment() Complete");
}

private void GetAllEmployees()
{
    gvAllEmployees.DataSource = ExecuteStoredProcure("spGetEmployees");
    gvAllEmployees.DataBind();
}

private void GetEmployeesByGender()
{
    gvEmployeesByGender.DataSource = ExecuteStoredProcure("spGetEmployeesByGender");
    gvEmployeesByGender.DataBind();
}

private void GetEmployeesByDepartment()
{
    gvEmployeesByDepartment.DataSource =
```

```
ExecuteStoredProcudure("spGetEmployeesByDepartment");
    gvEmployeesByDepartment.DataBind();
}
```

```
private DataSet ExecuteStoredProcudure(string spname)
{
    string CS = ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;
    SqlConnection con = new SqlConnection(CS);
    SqlDataAdapter da = new SqlDataAdapter(spname, con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    DataSet DS = new DataSet();
    da.Fill(DS);
    if (spname == "spGetEmployeesByGender")
    {
        System.Threading.Thread.Sleep(7000);
    }
    return DS;
}
```

**Set Trace=true, in Page directive:**

```
<%@ Page Language="C#" Trace="true" AutoEventWireup="true"
CodeBehind="WebForm3.aspx.cs" Inherits="AdoDemo.WebForm3" %>
```

Many a times, the issue is not reproducible on the development environment. The issue happens only on the production server. In this case tracing is an invaluable mechanism to get to the root cause of the issue.

## **Application pools in IIS**

### **What are application pools in IIS**

An Application Pool can contain one or more web applications. In IIS it is possible to create one or more application pools. Applications in different application pools, runs in its own worker process(w3wp.exe). Errors in one application pool will not affect the applications running in other application pools. For example, if an application pool is recycled, only the applications in that pool are affected(may lose state information if stored inside worker process), and applications in other application pools are unaffected. Deploying applications to different application pools enables us to achieve the degree of application isolation that we need, in terms of availability and security. For example, applications that require high security can be present in one application pool, and the other applications can be in a different application pool. Another example, hosting providers can place competing business applications in different application pools, so that they do not accidentally access the data belonging to their competitor.

### **Creating application pools in internet information services(IIS)**

1. Click on start
2. Type "RUN" and press "ENTER"
3. In the "RUN" window, type "INETMGR"
4. Click "OK"
5. In the IIS Manager window, expand the root node and right click on "Application Pools" and select "Add Application Pool"
6. Provide the "Name" for Application pool and click OK.

### **Application pool identities**

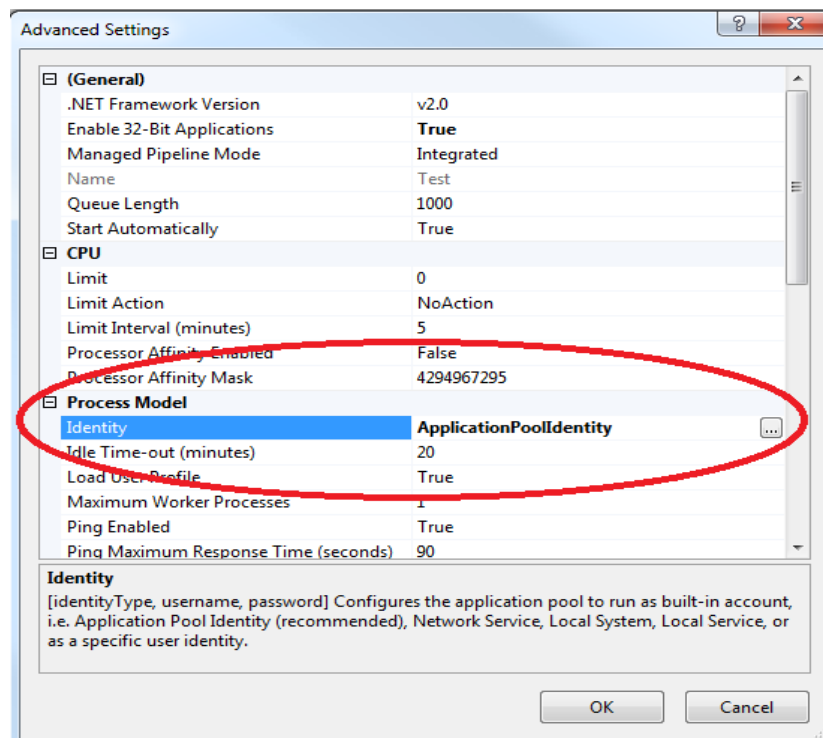
Asp.net applications execute inside asp.net worker process called w3wp.exe. The applications are executed by the worker process, using a windows identity. The windows identity that is used, is dependent on the application pool identity. The application pool identity can be any of the following built in accounts:

1. LocalService
2. LocalSystem
3. NetworkService
4. ApplicationPoolIdentity

In addition to these built-in accounts, we can also use a custom account, by specifying the username and password.

By default, when a new application pool is created, it uses ApplicationPoolIdentity. To change the application pool identity:

1. Right click on the application pool and select "Advanced Settings"
2. In the "Advanced Settings", click the ellipses button next to "Identity" under "Process Model" section
3. From the "Application Pool Identity" window, select one of the built-in accounts or enter the user and password, if you choose to use a custom account.
4. Finally click "OK"



**Local System :** Completely trusted account and has very high privileges and can also access network resources.

**Network Service :** Restricted or limited service account that is generally used to run, standard least-privileged services. This account has less privileges than Local System account. This account can access network resources.

**Local Service :** Restricted or limited service account that is very similar to Network Service and meant to run standard least-privileged services. This account cannot access network resources.

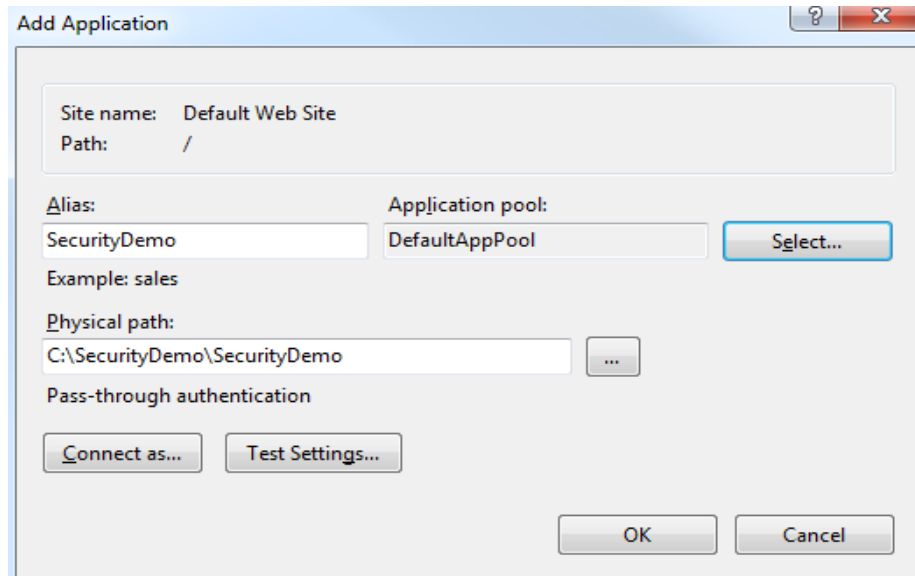
**ApplicationPoolIdentity :** When a new Application Pool is created, IIS creates a virtual account with the name of the new Application Pool and run the Application Pool's worker processes under this account. This is also a least privileged account.

Running an application using a low-privileged account is a good security practice, because, if there is a bug, that cannot be used by a malicious user to hack into your application or your system.

Associating an ASP.NET Web Application with an Application Pool:

1. Create a new asp.net web application project with name "SecurityDemo" in C:\
2. Open IIS (Type INETMGR in RUN window, and click OK)
3. Expand IIS root node
4. Expand "Sites"
5. Right click on "Default Web Site" and select "Add Application"
6. Enter the Alias Name for your application
7. Select the physical folder of the application by clicking on the ellipses button next "Physical Path" text box. If you are following along with me, then in the Physical Path text box you should have C:\SecurityDemo\SecurityDemo

8. To associate the application pool, click the "Select" button next to "Application pool" text box, and select the application pool from the drop down list.



The image shows a Windows-style dialog box titled "Add Application". It contains several input fields and buttons. At the top, there are fields for "Site name" (set to "Default Web Site") and "Path" (set to "/"). Below these, there are two main sections. The first section has an "Alias" field containing "SecurityDemo" and an "Application pool" field containing "DefaultAppPool". To the right of the "Application pool" field is a "Select..." button. Below the "Alias" field is an "Example" field containing "sales". The second section has a "Physical path" field containing "C:\SecurityDemo\SecurityDemo" and a browse button "...". Below the "Physical path" field is a "Pass-through authentication" checkbox. At the bottom left, there are two buttons: "Connect as..." and "Test Settings...". At the bottom right, there are two buttons: "OK" and "Cancel".

Site name: Default Web Site  
Path: /

Alias: SecurityDemo  
Application pool: DefaultAppPool  
Select...

Example: sales

Physical path: C:\SecurityDemo\SecurityDemo  
...

Pass-through authentication

Connect as... Test Settings...

OK Cancel

## Applications isolation using application pools in IIS

We will discuss about achieving isolation between applications, by associating with different application pools. Create an asp.net web application with name WebApplication1. Drag and drop a button control on webform1.aspx. Copy and paste the following code in webform1.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Session["Application1_Data"] = "Application 1 Data";
    }
    Response.Write("Identity used = " +
System.Security.Principal.WindowsIdentity.GetCurrent().Name + "<br/>");
}

protected void Button1_Click(object sender, EventArgs e)
{
    if (Session["Application1_Data"] != null)
    {
        Response.Write("Application1_Data = " + Session["Application1_Data"]);
    }
    else
    {
        Response.Write("Session Data not available");
    }
}

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Session["Application2_Data"] = "Application 2 Data";
    }
    Response.Write("Identity used = " +
System.Security.Principal.WindowsIdentity.GetCurrent().Name + "<br/>");
}

protected void Button1_Click(object sender, EventArgs e)
{
    if (Session["Application2_Data"] != null)
    {
        Response.Write("Application2_Data = " + Session["Application2_Data"]);
    }
    else
    {
        Response.Write("Session Data not available");
    }
}
```

Create an application pool in IIS with name "WebApplication1Pool":

1. Open IIS (Type INETMGR in RUN window, and press ENTER key)

2. Expand root node in IIS
3. Right click on "Application Pools" and select Add "Application Pool"
4. Enter "WebApplicationPool1" as the name and click OK.

Associate WebApplication1, with "WebApplication1Pool" we just created:

1. In IIS, right click on "Default Web Site" and Select "Add Application"
2. Set Alias="WebApplication1" and Select "WebApplication1Pool" as the application pool.
3. Set the physical path to the directory of WebApplication1.

Along the same lines, associate WebApplication2, with "WebApplication1Pool".

At this point, if you run WebApplication1, by using CTRL+F5, visual studio by default uses built-in asp.net development server. Configure visual studio to use local IIS:

1. Right click on the "WebApplication1" project in solution explorer in visual studio and select "Properties"
2. In the properties window click on "Web" tab.
3. Under "Servers" section, Select "Use Local IIS Web Server" radio button
4. Set project Url=http://localhost/WebApplication1 and save the changes.

Do the same thing for WebApplication2, but set project Url=http://localhost/WebApplication2 and save the changes.

Run both the applications. When you click the button on both the applications, Session data should be available.

Now let us recycle, application pools in IIS:

1. Open IIS
2. Select Application Pools
3. Right click on WebApplication1Pool, and select "Recycle"

Now click the button controls, on both the applications. Notice that both the applications have lost their session data.

Now create a new application pool in IIS, with name WebApplication2Pool and associate WebApplication2, with this new pool. Run both the applications again. Click the button controls. Session data should be available. Recycle WebApplication1Pool. Click the button controls again on both the applications. Notice, that, only WebApplication1 has lost the session data but not WebApplication2. WebApplication2 belong to WebApplication2Pool. We have not recycled WebApplication2Pool, and hence it retains its session data.

## Application pools in IIS Security

We will discuss about configuring different levels of security for different application pools, with an example.

In your C:\ drive, create a folder with name "Data". Open a notepad. Copy and paste the following XML into the notepad. Save the notepad as Application1Data.xml in C:\Data folder. Open, another notepad, copy and paste the same XML. Now save the notepad as Application2Data.xml in C:\Data folder. So, at this point, you should have Application1Data.xml and Application2Data.xml in C:\Data folder.

```
<?xml version="1.0" encoding="utf-8" ?>
<Countries>
  <Country>
    <Id>101</Id>
    <Name>India</Name>
    <Continent>Asia</Continent>
  </Country>
  <Country>
    <Id>102</Id>
    <Name>USA</Name>
    <Continent>North America</Continent>
  </Country>
  <Country>
    <Id>103</Id>
    <Name>UK</Name>
    <Continent>Europe</Continent>
  </Country>
  <Country>
    <Id>104</Id>
    <Name>France</Name>
    <Continent>Europe</Continent>
  </Country>
</Countries>
```

Create an asp.net web application with name WebApplication1. Drag and drop a GridView, FileUpload, Button and a Label control on to the webform. Set Text="Load Data" for the button control. Remove the Text property of the Label control. Double click the button control to generate the event handler. At this stage, the html of webform1.aspx should be as shown below.

```
<div>
  <asp:GridView ID="GridView1" runat="server">
  </asp:GridView>
  <br />
  <asp:FileUpload ID="FileUpload1" runat="server" />
  <asp:Button ID="Button1" runat="server"
  Text="Load Data" onclick="Button1_Click" />
  <br />
  <asp:Label ID="Label1" runat="server">
  </asp:Label>
</div>
```



### WebForm1.aspx.cs code:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("Identity used = " +
        System.Security.Principal.WindowsIdentity.GetCurrent().Name + "<br/>");
}

protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    {
        DataSet ds = new DataSet();
        ds.ReadXml(FileUpload1.PostedFile.FileName);
        GridView1.DataSource = ds;
        GridView1.DataBind();
        Label1.Text = "";
    }
    else
    {
        Label1.Text = "Please select a file first";
    }
}
```

Create an application pool with name Application1Pool, and associate WebApplication1, with this pool.

Create another asp.net web application with name WebApplication2. Copy and paste the HTML and code of WebForm1 from WebApplication1. Create an application pool with name Application2Pool, and associate WebApplication2, with this pool.

Run WebApplication1, and select Application1Data.xml from C:\Data folder, and click on "Load Data" button. The data should load fine. Now, select Application2Data.xml from C:\Data folder, and click on "Load Data" button. The data should load fine, from Application2Data.xml file as well. Test the same, with WebApplication2.

At this point, both WebApplication1 and WebApplication2, are able to read from Application1Data.xml and Application2Data.xml files. The idea is, we want to allow, WebApplication1 to be able to access only Application1Data.xml and not Application2Data.xml. Along the same lines, WebApplication2 should be able to access only Application2Data.xml and not Application1Data.xml.

The applications are deployed to different application pools. WebApplication1 is deployed to Application1Pool, and WebApplication2 to Application2Pool. So, WebApplication1 is executed using Application1Pool identity - IIS APPPOOL\Application1Pool, and WebApplication2 with Application2Pool identity - IIS APPPOOL\Application2Pool.

At this stage, all we have to do is, set the file permissions accordingly for the application pool identities.

Deny access to file Application1Data.xml for IIS APPPOOL\Application2Pool identity:

1. In C:\Data, right click on Application1Data.xml and select "Properties"
2. Click on the "Security" tab.
3. Click "Edit" button
4. Now click "Add"
5. Click on "Locations" button and select your "computer name" and click OK
6. In the "Enter the object names to select", text box, type IIS APPPOOL\Application2Pool and click "Check Names" button.
7. Click OK
8. In the permissions list, select "Full Control" under "Deny" and click OK.

Along the same lines, Deny access to file Application2Data.xml for IIS

APPPOOL\Application1Pool identity.

With these changes in place now, WebApplication1 should only be able to access Application1Data.xml and WebApplication2, only Application2Data.xml. Instead of showing the "Yellow screen of death", user friendly error message can be displayed in the label control by catching the security exception as shown below.

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    {
        try
        {
            DataSet ds = new DataSet();
            ds.ReadXml(FileUpload1.PostedFile.FileName);
            GridView1.DataSource = ds;
            GridView1.DataBind();
            Label1.Text = "";
        }
        catch (System.UnauthorizedAccessException)
        {
            Label1.Text = "You do not have access to this file";
        }
        catch (Exception)
        {
            Label1.Text = "An unexpected error has occurred, please contact administrator";
        }
    }
    else
    {
        Label1.Text = "Please select a file first";
    }
}
```

## Anonymous authentication in asp.net

Authentication is the process of identifying users. Authorization is the process of granting access to those users based on identity. Together, authentication and authorization secures our Web application.

**Authentication** - Who is the User?

**Authorization** - What rights the user has? What resources the user can access?

Most of the public web sites, does not ask the user to enter any user name and password. But still, we will be able to access the content of these web sites. ASP.NET Web applications provide anonymous access to resources on the server. Anonymous authentication allows users to access the public areas of the web site, without prompting the users for a user name or password.

Create an asp.net web application. Copy and paste the following code in the Page\_Load() event of WebForm1.aspx.cs

```
Response.Write("Application code executed using ");  
Response.Write(System.Security.Principal.WindowsIdentity.GetCurrent().Name + "<br/>");
```

```
Response.Write("Is User Authenticated: ");  
Response.Write(User.Identity.IsAuthenticated.ToString() + "<br/>");
```

```
Response.Write("Authentication Type, if Authenticated: ");  
Response.Write(User.Identity.AuthenticationType + "<br/>");
```

```
Response.Write("User Name, if Authenticated: ");  
Response.Write(User.Identity.Name + "<br/>");
```

Associate the web application, to the local IIS, instead of using the visual studio built-in asp.net development server. Use the DefaultAppPool as the application pool.

### In IIS 6.0

IUSR\_ComputerName is used for providing anonymous access.

### In IIS 7.0

IUSR account is used for providing anonymous access.

By default anonymous authentication is enabled in IIS. To verify this:

1. Open IIS
2. Expand the root node > Sites > Default Web Site
3. Select your web application
4. In the features window, double click "Authentication" icon
5. Notice that, anonymous authentication is enabled by default.

Run the application. Notice, that the application pool identity is used to execute the application code. we will discuss about asp.net impersonation with anonymous access.

To disable anonymous authentication, click "Disable" link under "actions" in the right hand side panel in IIS.

To change the account that is associated with anonymous access, click "Edit" link under actions in the right hand side panel in IIS. Notice, that the default account is IUSR. This can be changed to a custom windows account or Application pool identity.

## Anonymous authentication and asp.net impersonation

We discussed that IIS provides anonymous access to resources using IUSR account. Once the request is handed over to asp.net, the application code is executed using the application pool identity. In "C:\Data" folder, create an XML file with name Countries.xml.

```
<?xml version="1.0" encoding="utf-8" ?>
<Countries>
  <Country>
    <Id>101</Id>
    <Name>India</Name>
    <Continent>Asia</Continent>
  </Country>
  <Country>
    <Id>102</Id>
    <Name>UK</Name>
    <Continent>Europe</Continent>
  </Country>
  <Country>
    <Id>103</Id>
    <Name>US</Name>
    <Continent>North America</Continent>
  </Country>
  <Country>
    <Id>104</Id>
    <Name>France</Name>
    <Continent>Europe</Continent>
  </Country>
</Countries>
```

Create an asp.net web application. Drag and drop a gridview control and a button control on the webform. Copy and paste the following code in WebForm1.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("Application code executed using ");
    Response.Write(System.Security.Principal.WindowsIdentity.GetCurrent().Name + "<br/>");

    Response.Write("Is User Authenticated: ");
    Response.Write(User.Identity.IsAuthenticated.ToString() + "<br/>");

    Response.Write("Authentication Type, if Authenticated: ");
    Response.Write(User.Identity.AuthenticationType + "<br/>");

    Response.Write("User Name, if Authenticated: ");
    Response.Write(User.Identity.Name + "<br/>");
}

protected void Button1_Click(object sender, EventArgs e)
{
    DataSet ds = new DataSet();
    ds.ReadXml("C:\\Data\\Countries.xml");
}
```

```
GridView1.DataSource = ds;  
GridView1.DataBind();  
}
```

To enable impersonation, set impersonate="true" for the identity element in web.config.

```
<system.web>  
  <identity impersonate="true" />  
</system.web>
```

Impersonation can also be enabled or disabled from IIS.

1. Select the web application in IIS
2. Double click on "Authentication" icon
3. Select ASP.NET Impersonation
4. Click "Disable" or "Enable" link under actions in the right hand side panel in IIS.
5. This will automatically change the web.config file.

At this point, if you run the application, you may get an error stating

#### HTTP Error 500.24 - Internal Server Error

An ASP.NET setting has been detected that does not apply in Integrated managed pipeline mode.

To correct this, we need to set the "Managed pipeline mode" of the DefaultAppPool to "Classic".

Run the application, and notice that, the application code, is now executed, using 'NT AUTHORITY\IUSR' account, instead of 'IIS APPPOOL\DefaultAppPool'

So, when the application uses anonymous authentication and

1. If IMPERSONATION is disabled, then, the application pool identity is used to execute the application code
2. If IMPERSONATION is enabled, then, 'NT AUTHORITY\IUSR' account is used to execute the application code

#### When to use Application Pool Identity over IUSR

If there are 2 or more websites hosted on a machine, with IUSR as the anonymous account, then they can access each other's content. If we want to isolate, each applications content, the applications can be deployed to different application pools, and the NTFS file permissions can be set for the respective application pool identity. In fact, we have discussed about this in Part 84 - Application pools in IIS Security.

## Windows authentication in asp.net

we discussed about anonymous authentication. Anonymous authentication is fine for web sites that contain public information, that every one can see. However, if the web site contains private information or performs tasks such as booking tickets, placing orders etc, then the users need to be authenticated and authorised.

we will discuss about authenticating users, using Windows authentication. Security for an asp.net web application can be configured at 2 places. In IIS and in the application itself.

Windows authentication, identifies and authorizes users based on the server's user list. Access to resources on the server is then granted or denied based on the user account's privileges.

Windows authentication is best suited for Intranet Web applications.

The advantage of Windows authentication is that, the Web application can use the exact same security scheme that applies to your corporate network. User names, passwords, and permissions are the same for network resources and Web applications.

To enable windows authentication in IIS.

1. Open IIS (Type INETMGR in RUN window, and press enter)
2. Expand Root Server node > Sites > Default Web Site > WebApplication1
3. Double click "Authentication" icon, in the features window.
4. Notice that "Anonymous Authentication" is enabled by default.
5. Select "Windows Authentication" and click "Enable" link under "Actions" pane.

At this point, we have both anonymous and windows authentication enabled in IIS. We have not configured anything in the application yet. Run the application, and notice that, the user is still using anonymous authentication to access the webform.

So, if both, anonymous and windows authentication are enabled in IIS, and, if we don't have a deny entry for anonymous users, in the web.config file, then the resources on the web server are accessed using anonymous authentication.

Anonymous authentication can be disabled in IIS or in web.config file.

To disable anonymous authentication in web.config file, add the following entry

```
<authorization>  
  <deny users="?"/>  
</authorization>
```

Run the application now. Notice that the user is authenticated using the windows account, that is used to log into the computer. Also, notice that, the application code is executed using the application pool identity.

If you want to have the application code executed using the logged in user identity, then enable impersonation. Impersonation can be enabled thru IIS or by adding the following element to web.config file.

```
<identity impersonate="true"/>
```

If impersonation is enabled, the application executes using the permissions found in your user account. So, if the logged in user has access, to a specific network resource, only then will he be able to access that resource thru the application.



## Windows authentication and authorization in asp.net

? and \* have special meaning when used in the authorization element in web.config

? (Question Mark) - Indicates anonymous users

\* (Star) - Indicates all users

### Allowing or denying access to specific users:

When you run the application, with the following authorization list in web.config, only users "Venkat" and "Pragim" are allowed to access the application. If you are logged, into the computer, as any other user, the application prompts the user to provide user name and password. All the other users are denied access to the application.

```
<authorization>
  <allow users="Prasad-PC\Venkat, Prasad-PC\Pragim"/>
  <deny users="*/>
</authorization>
```

### Using windows roles to control access:

Windows operating system has several roles, like Administrators, Guests, Users etc. It is also possible to control access to resources using these roles in the web.config file. The following authorization list, only allows users belonging to Administrators role. All the other users are denied access.

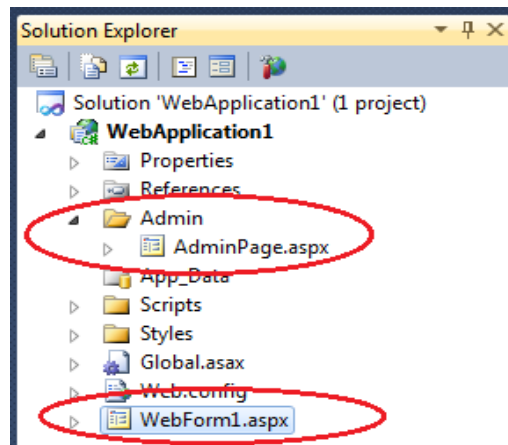
```
<authorization>
  <allow roles="Administrators"/>
  <deny users="*/>
</authorization>
```

### How to programmatically check if the user belongs to a specific role?

```
if (User.IsInRole("Administrators"))
{
    // Do Admin Stuff
}
else
{
    // Do Non-Admin stuff
}
```

## Windows authentication and folder level authorization

We will discuss about folder level authorization, with an example. Consider the project structure, shown in the solution explorer below.



Only administrators should be able to access the pages in "Admin" folder. The rest of the pages can be accessed by anyone. To achieve this, add another web.config file to the "Admin" folder and include the following authorization element.

```
<authorization>
  <allow roles="Administrators" />
  <deny users="*" />
</authorization>
```

Application root level web.config file. This allows access to all authenticated users.

```
<authorization>
  <deny users="?" />
</authorization>
```

### A very common asp.net interview question:

Is it possible to have more than one web.config file? If yes, when and why would you use more than one web.config file.

This is one of the classic examples, where we need more than one web.config files.

If you want to execute the application code, using the logged in Administrator account, then enable impersonation, in the web.config file of the Admin folder. With this setting in place, all the pages in the Admin folder are executed using the logged in user account, whereas the pages outside of the folder are executed using the identity of the application pool.

```
<system.web>
  <authorization>
    <allow roles="Administrators" />
    <deny users="*" />
  </authorization>
  <identity impersonate="true" />
```

</system.web>

It is also possible to impersonate, with a specific user name and password. With this setting, whenever any user belonging to the "Administrators" group requests a page from the Admin folder, the code will be executed using "Venkat" account.

<system.web>

<authorization>

<allow roles="Administrators" />

<deny users="\*" />

</authorization>

<identity impersonate="true" userName="Venkat" password="test"/>

</system.web>

## Forms authentication using user names list in web.config

Anonymous authentication is fine for web sites that contain public information that every one can see.

Windows authentication is used for intranet web applications, where the users are part of a windows domain-based network.

### When to use Forms Authentication?

Forms authentication is used for internet web applications. The advantage of Forms authentication is that users do not have to be member of a domain-based network to have access to your application. Many internet web sites like Gmail.com, Amazon.com, facebook.com etc uses forms authentication. To access these applications we do not have to be member of their domain-based network.

### How to enable Forms Authentication?

Create an asp.net web application project. Add a webform with name Welcome.aspx, and Login.aspx. Add a new folder with name "Registration", to the project. Add Register.aspx web form to the "Registration" folder.

#### Welcome.aspx HTML:

```
<h1>Welcome Page</h1>
```

#### Login.aspx HTML:

```
<div style="font-family:Arial">
<table style="border: 1px solid black">
  <tr>
    <td colspan="2">
      <b>Login</b>
    </td>
  </tr>
  <tr>
    <td>
      User Name
    </td>
    <td>
      <asp:TextBox ID="txtUserName" runat="server">
      </asp:TextBox>
    </td>
  </tr>
  <tr>
    <td>
      Password
    </td>
    <td>
      <asp:TextBox ID="txtPassword" TextMode="Password" runat="server">
      </asp:TextBox>
    </td>
  </tr>
</table>
```

```

</tr>
<td>

</td>
<td>
    <asp:Button ID="btnLogin" runat="server" Text="Login" />
</td>
</tr>
</table>
<br />
<a href="Registration/Register.aspx">Click here to register</a>
if you do not have a user name and password.
</div>

```

### Register.aspx HTML:

```
<h1>Registration Page</h1>
```

If you run the application now, we will be able to navigate to any page, just by changing the name of the page in the address bar. We are not logged in, but we are still able to access all the pages in the application.

Let us enable forms authentication now. To enable forms authentication, set authentication element's mode attribute to forms in web.config file of the application.

```

<authentication mode="Forms">
  <forms loginUrl="Login.aspx" timeout="30"
    defaultUrl="Welcome.aspx" protection="All">
    <credentials passwordFormat="Clear">
      <user name="venkat" password="venkat"/>
      <user name="pragim" password="pragim"/>
      <user name="prasad" password="prasad"/>
    </credentials>
  </forms>
</authentication>

<authorization>
  <deny users="?" />
</authorization>

```

The description of the attributes:

**loginUrl** - The URL of the login Page

**timeout** - Specifies the number of minutes the authentication cookie persists on the clients's computer. The default is 30 minutes.

**defaultUrl** - The url the user will be redirected after authentication

**Protection** - Specifies the protection for authentication cookie stored on the clients's computer. The default is All, which performs encryption and data validation. Other possible settings are Encryption, Validation, and None.

Double click the login button on the Login.aspx page. Copy and paste the following code in the button click event handler.

```
// Authenticate againsts the list stored in web.config
if (FormsAuthentication.Authenticate(txtUserName.Text, txtPassword.Text))
{
    // Create the authentication cookie and redirect the user to welcome page
    FormsAuthentication.RedirectFromLoginPage(txtUserName.Text,
chkBoxRememberMe.Checked);
}
else
{
    lblMessage.Text = "Invalid UserName and/or password";
}
```

Run the application. Try to navigate to Welcome.aspx or Registration/Register.aspx pages, you will be redirected to Login page. After you login, you will be able to access these pages.

There are 2 problems with this application at the moment.

1. It is not a good practise to store user names and passwords in web.config file. If you want to create the user names and passwords dynamically, you need to change the web.config file. If you change the web.config file at run time, the application restarts and all the session data will be lost, if stored inside the worker process.
2. At the moment, users are not able to access Register.aspx page, if they are not logged in. If a user does not have user name and password, he should be able to register himself using Register.aspx page.

## Forms authentication in asp.net and user registration

One of the problem is that, we are not able to navigate to Registration/Register.aspx page if we are not logged in.

To solve this issue, add another web.config file to the "Registration" folder, and specify the authorization element to allow all users.

```
<authorization>
  <allow users="*" />
</authorization>
```

At this point, without logging into the application, users should be able to navigate to Registration/Register.aspx page.

Copy and paste the following HTML in Register.aspx page.

```
<div style="font-family:Arial">
<table style="border: 1px solid black">
  <tr>
    <td colspan="2">
      <b>User Registration</b>
    </td>
  </tr>
  <tr>
    <td>
      User Name
    </td>
    <td>
      <asp:TextBox ID="txtUserName" runat="server">
      </asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidatorusername"
runat="server" ErrorMessage="User Name required" Text="*"
ControlToValidate="txtUserName" ForeColor="Red">
      </asp:RequiredFieldValidator>
    </td>
  </tr>
  <tr>
    <td>
      Password
    </td>
    <td>
      <asp:TextBox ID="txtPassword" TextMode="Password" runat="server">
      </asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidatorPassword"
runat="server" ErrorMessage="Password required" Text="*"
ControlToValidate="txtPassword" ForeColor="Red">
      </asp:RequiredFieldValidator>
    </td>
  </tr>
  <tr>
    <td>
```

```

Confirm Password
</td>
<td>
    <asp:TextBox ID="txtConfirmPassword" TextMode="Password" runat="server">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidatorConfirmPassword"
    runat="server" ErrorMessage="Confirm Password required" Text="*"
    ControlToValidate="txtConfirmPassword" ForeColor="Red"
    Display="Dynamic"></asp:RequiredFieldValidator>
    <asp:CompareValidator ID="CompareValidatorPassword" runat="server"
    ErrorMessage="Password and Confirm Password must match"
    ControlToValidate="txtConfirmPassword" ForeColor="Red"
    ControlToCompare="txtPassword" Display="Dynamic"
    Type="String" Operator="Equal" Text="*">
    </asp:CompareValidator>
</td>
</tr>
<tr>
<td>
    Email
</td>
<td>
    <asp:TextBox ID="txtEmail" runat="server">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidatorEmail"
    runat="server" ErrorMessage="Email required" Text="*"
    ControlToValidate="txtEmail" ForeColor="Red"
    Display="Dynamic"></asp:RequiredFieldValidator>
    <asp:RegularExpressionValidator ID="RegularExpressionValidatorEmail"
    runat="server" ErrorMessage="Invalid Email" ControlToValidate="txtEmail"
    ForeColor="Red" Display="Dynamic" Text="*"
    ValidationExpression="\w+([-+.']\w+)*@\w+([-.']\w+)*\.\w+([-.']\w+)*">
    </asp:RegularExpressionValidator>
</td>
</tr>
<tr>
<td>
</td>
<td>
    <asp:Button ID="btnRegister" runat="server" Text="Register"
    onclick="btnRegister_Click"/>
</td>
</tr>
<tr>
<td colspan="2">
    <asp:Label ID="lblMessage" runat="server" ForeColor="Red">
    </asp:Label>
</td>
</tr>
<tr>
<td colspan="2">

```



```

        <asp:ValidationSummary ID="ValidationSummary1" ForeColor="Red" runat="server" />
    </td>
</tr>
</table>
</div>

```

Copy and Paste the following code in the "Register" button click event.

```

// If the Page has no validation errors
if (Page.IsValid)
{
    // Read the connection string from web.config.
    // ConfigurationManager class is in System.Configuration namespace
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    // SqlConnection is in System.Data.SqlClient namespace
    using (SqlConnection con = new SqlConnection(CS))
    {
        SqlCommand cmd = new SqlCommand("spRegisterUser", con);
        cmd.CommandType = CommandType.StoredProcedure;

        SqlParameter username = new SqlParameter("@UserName", txtUserName.Text);
        // FormsAuthentication class is in System.Web.Security namespace
        string encryptedPassword = FormsAuthentication.
            HashPasswordForStoringInConfigFile(txtPassword.Text, "SHA1");
        SqlParameter password = new SqlParameter("@Password", encryptedPassword);
        SqlParameter email = new SqlParameter("@Email", txtEmail.Text);

        cmd.Parameters.Add(username);
        cmd.Parameters.Add(password);
        cmd.Parameters.Add(email);

        con.Open();
        int ReturnCode = (int)cmd.ExecuteScalar();
        if (ReturnCode == -1)
        {
            lblMessage.Text = "User Name already in use, please choose another user name";
        }
        else
        {
            Response.Redirect("~/Login.aspx");
        }
    }
}

```

Run the application. Fill in the required details, and click "Register" button. The user should be added to the database.

## Forms authentication against users in database table

Authenticating users against a list stored in web.config file is very easy. FormsAuthentication class exposes a static method Authenticate(), which does all the hardwork of authenticating users.

If we want to authenticate users against a list stored in a database table, we will have to write the stored procedure and a method in the application to authenticate users.

First let us create a stored procedure, that accepts username and password as input parameters and authenticate users.

```
Create Procedure spAuthenticateUser
@UserName nvarchar(100)
@Password nvarchar(100)
as
Begin
Declare @Count int

Select @Count = COUNT(Username) from tblUsers
where [UserName] = @UserName and [Password] = @Password

if(@Count = 1)
Begin
Select 1 as ReturnCode
End
Else
Begin
Select -1 as ReturnCode
End
End
```

Copy and paste the following private method in Login.aspx.cs page. This method invokes stored procedure 'spAuthenticateUser'.

```
private bool AuthenticateUser(string username, string password)
{
    // ConfigurationManager class is in System.Configuration namespace
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    // SqlConnection is in System.Data.SqlClient namespace
    using (SqlConnection con = new SqlConnection(CS))
    {
        SqlCommand cmd = new SqlCommand("spAuthenticateUser", con);
        cmd.CommandType = CommandType.StoredProcedure;

        // FormsAuthentication is in System.Web.Security
        string EncryptedPassword =
FormsAuthentication.HashPasswordForStoringInConfigFile(password, "SHA1");
        // SqlParameter is in System.Data namespace
        SqlParameter paramUsername = new SqlParameter("@UserName", username);
        SqlParameter paramPassword = new SqlParameter("@Password", EncryptedPassword);

        cmd.Parameters.Add(paramUsername);
```

```
cmd.Parameters.Add(paramPassword);

con.Open();
int ReturnCode = (int)cmd.ExecuteScalar();
return ReturnCode == 1;
}
}
```

Invoke AuthenticateUser() method, in the login button click event handler

```
if (AuthenticateUser(txtUserName.Text, txtPassword.Text))
{
    FormsAuthentication.RedirectFromLoginPage(txtUserName.Text,
chkBoxRememberMe.Checked);
}
else
{
    lblMessage.Text = "Invalid User Name and/or Password";
}
```

## Forms authentication and locking user accounts

We will discuss about locking or disabling user accounts, after repeated invalid attempts to login.

For example, if a user enters wrong username and password, he will be given 3 more chances, to enter the correct password. After the 3 chances are elapsed, the account will be locked. After the account is locked, the user will not be able to log in, even, if he provides a correct user name and password.

Most of the banking applications does this for security reasons.

Drop the table, tblUsers, Recreate tblUsers table using the script below.

Create table tblUsers

```
(  
  [Id] int identity primary key,  
  [UserName] nvarchar(100),  
  [Password] nvarchar(200),  
  [Email] nvarchar(100),  
  [RetryAttempts] int,  
  [IsLocked] bit,  
  [LockedDateTime] datetime  
)
```

Since, we have changed the structure of the table. The stored procedure 'spRegisterUser' , will break. The corrected stored procedure is show below.

```
Alter proc spRegisterUser  
@UserName nvarchar(100),  
@Password nvarchar 200),  
@Email nvarchar 200)
```

as

Begin

Declare @Count int

Declare @ReturnCode int

Select @Count = COUNT(UserName)

from tblUsers where UserName = @UserName

If @Count > 0

Begin

Set @ReturnCode = -1

End

Else

Begin

Set @ReturnCode = 1

--Change: Column list specified while inserting

Insert into tblUsers([UserName], [Password], [Email])

values (@UserName, @Password, @Email)

End

Select @ReturnCode as ReturnValue

End

Stored procedure - 'spAuthenticateUser', needs to be changed as shown below, to support the Account locking functionality.

```
Alter proc spAuthenticateUser
@UserName nvarchar(100),
@Password nvarchar(200)
as
Begin
  Declare @AccountLocked bit
  Declare @Count int
  Declare @RetryCount int

  Select @AccountLocked = IsLocked
  from tblUsers where UserName = @UserName

  --If the account is already locked
  if(@AccountLocked = 1)
  Begin
    Select 1 as AccountLocked, 0 as Authenticated, 0 as RetryAttempts
  End
  Else
  Begin
    -- Check if the username and password match
    Select @Count = COUNT(UserName) from tblUsers
    where [UserName] = @UserName and [Password] = @Password

    -- If match found
    if(@Count = 1)
    Begin
      -- Reset RetryAttempts
      Update tblUsers set RetryAttempts = 0
      where UserName = @UserName

      Select 0 as AccountLocked, 1 as Authenticated, 0 as RetryAttempts
    End
    Else
    Begin
      -- If a match is not found
      Select @RetryCount = IsNULL(RetryAttempts, 0)
      from tblUsers
      where UserName = @UserName

      Set @RetryCount = @RetryCount + 1

      if(@RetryCount <= 3)
      Begin
        -- If re-try attempts are not completed
        Update tblUsers set RetryAttempts = @RetryCount
        where UserName = @UserName

        Select 0 as AccountLocked, 0 as Authenticated, @RetryCount as RetryAttempts
      End
    End
  End
End
```

```

Else
Begin
-- If re-try attempts are completed
Update tblUsers set RetryAttempts = @RetryCount,
IsLocked = 1, LockedDateTime = GETDATE()
where UserName = @UserName

Select 1 as AccountLocked, 0 as Authenticated, 0 as RetryAttempts
End
End
End
End

```

Copy and Paste the following version of AuthenticateUser() method in Login.aspx.cs page.

```

private void AuthenticateUser(string username, string password)
{
// ConfigurationManager class is in System.Configuration namespace
string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
// SqlConnection is in System.Data.SqlClient namespace
using (SqlConnection con = new SqlConnection(CS))
{
SqlCommand cmd = new SqlCommand("spAuthenticateUser", con);
cmd.CommandType = CommandType.StoredProcedure;

//Formsauthentication is in system.web.security
string encryptedpassword =
FormsAuthentication.HashPasswordForStoringInConfigFile(password, "SHA1");

//sqlparameter is in System.Data namespace
SqlParameter paramUsername = new SqlParameter("@UserName", username);
SqlParameter paramPassword = new SqlParameter("@Password", encryptedpassword);

cmd.Parameters.Add(paramUsername);
cmd.Parameters.Add(paramPassword);

con.Open();
SqlDataReader rdr = cmd.ExecuteReader();
while (rdr.Read())
{
int RetryAttempts = Convert.ToInt32(rdr["RetryAttempts"]);
if (Convert.ToBoolean(rdr["AccountLocked"]))
{
lblMessage.Text = "Account locked. Please contact administrator";
}
else if (RetryAttempts > 0)
{
int AttemptsLeft = (4 - RetryAttempts);
lblMessage.Text = "Invalid user name and/or password. " +
AttemptsLeft.ToString() + "attempt(s) left";
}
else if (Convert.ToBoolean(rdr["Authenticated"]))

```

```
        {  
            FormsAuthentication.RedirectFromLoginPage(txtUserName.Text,  
chkBoxRememberMe.Checked);  
        }  
    }  
}
```

Invoke AuthenticateUser() method in the click event handler of the login button control.

```
AuthenticateUser(txtUserName.Text, txtPassword.Text);
```

## Unlocking the locked user accounts

we will discuss about unlocking the locked user accounts. There are several ways to unlock the user accounts.

**Approach 1:** The end user calls the technical help desk. The authorised person can issue a simple update query to remove the lock.

```
Update tblUsers  
set RetryAttempts = null, IsLocked = 0, LockedDateTime = null  
where username='CallersUserName'
```

However, running UPDATE queries manually against a production database is not recommended, as it is error prone and we may un-intentionally modify other rows that we do not intend to update.

**Approach 2:** Another approach would be to provide a web page that lists all the locked user accounts. From this page, the helpdesk agent, can unlock the account by clicking a button. This is not as dangerous as running a manual update query, but still a manual process and may be inefficient. If you know how to write basic ADO.NET code, this approach should not be very difficult to achieve.

**Approach 3:** Another approach would be, to create a SQL Server job. This job checks tblUsers table for locked accounts periodically and then unlocks them. The frequency at which the job should run is configurable.

First let us write the update query to unlock the user accounts. For example, The organization's policy is that, the user account can only be unlocked after 24 hours, since the account is locked. The update query to satisfy the organization's policy is shown below. DateDiff function is used in the update query.

```
Update tblUsers  
set RetryAttempts = null, IsLocked = 0, LockedDateTime = null  
where IsLocked = 1  
and datediff(HOUR,LockedDateTime,GETDATE()) > 24
```

Let us now, schedule this update query to run every 30 minutes, every day. This can be very easily done using sql server agent jobs. we will discuss about creating and scheduling sql server agent jobs, for sql server 2008.

1. Open sql server management studio
2. In the object explorer, check if "SQL Server Agent" is running.
3. If "SQL Server Agent" is not running, right click and select "Start".
4. Click on the "+" sign, next to "SQL Server Agent" to expand.
5. Right click on "Jobs" folder and select "New Job".
6. In the "New Job" dialog box, provide a meaningful name. Let us call it, "Unlock user accounts job".
7. Fill in Owner, Category and Description fields accordingly. Make sure the Enabled checkbox is selected.
8. Select "Steps" tab, and click "New" button
9. In the "New Job Step" dialog box, give a meaningful step name. Let us call it "Execute Update Query"
10. Select Transact-SQL Script as "Type"



11. Select the respective Database.
12. In the "Command" text box, copy and paste the UPDATE query, and click OK
13. In the "New Job" dialog box, select "Schedules" and click "New" button
14. In the "New Job Schedule" dialog box, give a meaningful name to the schedule. Let us call it "Run Every 30 Minutes Daily"
15. Choose "Recurring" as "Schedule type"
16. Under "Frequency", set "Occurs" = "Daily" and "Recur every" = "1" Days.
17. Under "Daily Frequency", set "Occurs every" = "30" Minutes.
18. Finally fill in the schedule start and end dates, under "Duration"
19. Click OK, twice and you are done.

This job, will run every 30 minutes daily, and unlocks the accounts that has been locked for more than 24 hours.

## Implementing password reset link in asp.net

### Step 1:

The first step is to design a page, that allows the user to enter their user name, for requesting, the reset of the password. Add a webform, with name "ResetPassword.aspx" to the "Registration" folder. The web.config file in this folder, allows anonymous access to all the pages without having the need to login.

### Step 2:

Copy and paste the following HTML on "ResetPassword.aspx" page.

```
<div style="font-family:Arial">
  <table style="border: 1px solid black; width:300px">
    <tr>
      <td colspan="2">
        <b>Reset my password</b>
      </td>
    </tr>
    <tr>
      <td>
        User Name
      </td>
      <td>
        <asp:TextBox ID="txtUserName" Width="150px" runat="server">
        </asp:TextBox>
      </td>
    </tr>
    <tr>
      <td>
      </td>
      <td>
        <asp:Button ID="btnResetPassword" runat="server"
        Width="150px" Text="Reset Password" onclick="btnResetPassword_Click" />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <asp:Label ID="lblMessage" runat="server"></asp:Label>
      </td>
    </tr>
  </table>
</div>
```

### Step 3:

Create a table "tblResetPasswordRequests" in sql server. This table is going to store a unique GUID (Globally Unique Identifier) along with the user id, each time a user requests a password recovery. This GUID will then be passed as part of the querystring in the link to the password reset page. This link will then be emailed to the email address that is associated with the user id. When a user clicks on the link the page will look up the GUID in "tblResetPasswordRequests" table and get the user id

from there allowing the user to change their password. I didn't use, UserId, as the querystring parameter, because it maybe open to abuse.

```
Create table tblResetPasswordRequests
(
  Id UniqueIdentifier Primary key,
  UserId int Foreign key references tblUsers(Id),
  ResetRequestDateTime DateTime
)
```

#### Step 4:

Create a stored procedure to check if the username exists, and to insert a row into "tblResetPasswordRequests" table.

```
Create proc spResetPassword
@UserName nvarchar(100)
as
Begin
  Declare @UserId int
  Declare @Email nvarchar(100)

  Select @UserId = Id, @Email = Email
  from tblUsers
  where UserName = @UserName

  if(@UserId IS NOT NULL)
  Begin
    --If username exists
    Declare @GUID UniqueIdentifier
    Set @GUID = NEWID()

    Insert into tblResetPasswordRequests
    (Id, UserId, ResetRequestDateTime)
    Values(@GUID, @UserId, GETDATE())

    Select 1 as ReturnCode, @GUID as UniqueId, @Email as Email
  End
  Else
  Begin
    --If username does not exist
    SELECT 0 as ReturnCode, NULL as UniqueId, NULL as Email
  End
End
```

#### Step 5:

Invoke the stored procedure and email the link, to the email address that is registered against the username. Copy and paste the following code in ResetPassword.aspx.cs page.

```
protected void btnResetPassword_Click(object sender, EventArgs e)
{
```

```

string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
using (SqlConnection con = new SqlConnection(CS))
{
    SqlCommand cmd = new SqlCommand("spResetPassword", con);
    cmd.CommandType = CommandType.StoredProcedure;

    SqlParameter paramUsername = new SqlParameter("@UserName", txtUserName.Text);

    cmd.Parameters.Add(paramUsername);

    con.Open();
    SqlDataReader rdr = cmd.ExecuteReader();
    while (rdr.Read())
    {
        if (Convert.ToBoolean(rdr["ReturnCode"]))
        {
            SendPasswordResetEmail(rdr["Email"].ToString(), txtUserName.Text,
rdr["UniqueId"].ToString());
            lblMessage.Text = "An email with instructions to reset your password is sent to your
registered email";
        }
        else
        {
            lblMessage.ForeColor = System.Drawing.Color.Red;
            lblMessage.Text = "Username not found!";
        }
    }
}

private void SendPasswordResetEmail(string ToEmail, string UserName, string UniqueId)
{
    // MailMessage class is present in System.Net.Mail namespace
    MailMessage mailMessage = new MailMessage("YourEmail@gmail.com", ToEmail);

    // StringBuilder class is present in System.Text namespace
    StringBuilder sbEmailBody = new StringBuilder();
    sbEmailBody.Append("Dear " + UserName + "<br/><br/>");
    sbEmailBody.Append("Please click on the following link to reset your password");
    sbEmailBody.Append("<br/>");
    sbEmailBody.Append("http://localhost/WebApplication1/Registration/ChangePassword.aspx?uid="
+ UniqueId);
    sbEmailBody.Append("<br/><br/>");
    sbEmailBody.Append("<b>Pragim Technologies</b>");

    mailMessage.IsBodyHtml = true;

    mailMessage.Body = sbEmailBody.ToString();
    mailMessage.Subject = "Reset Your Password";
    SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587);

```

```
smtpClient.Credentials = new System.Net.NetworkCredential()
{
    UserName = "YourEmail@gmail.com",
    Password = "YourPassword"
};

smtpClient.EnableSsl = true;
smtpClient.Send(mailMessage);
}
```

### **Step 6:**

Add a webform with name, "ChangePassword.aspx", to "Registration" folder. Copy and paste the following HTML in the aspx page.

```
<h1>Change Password Page</h1>
```

## Implementing change password page in asp.net

We discussed about, generating and emailing the password reset link. The password reset link looks as shown below.

<http://localhost/WebApplication1/Registration/ChangePassword.aspx?uid=c19b3a4a-7fd2-47dc-9c2a-be541daed8fa>

Notice that, ChangePassword.aspx page has a query string "uid". This GUID(Globally unique identifier), is used to look up UserID, for whom the password needs to be changed. After updating the password, delete the row from "tblResetPasswordRequests", so the link becomes invalid after the user has changed his/her password. Since, user id's are integers, they may be open for abuse as it is very easy to use random integers as query string values, to change other users password.

tblUsers						
Id	UserName	Password	Email	RetryAttempts	IsLocked	LockedDateTime
1	test	A94A8FE5CCB1	kudvenk	0	0	NULL

tblResetPasswordRequests		
Id	UserId	ResetRequestDateTime
C19B3A4A-7FD2-47DC-9C2A-BE541DAED8FA	1	25/12/2012 17:03:45

Stored Procedure to check, if the password reset link, is a valid link.

```
Create Proc spIsPasswordResetLink Valid
@GUID uniqueidentifier
as
Begin
Declare @UserId int

If(Exists(Select UserId from tblResetPasswordRequests where Id = @ GUID))
Begin
Select 1 as IsValidPasswordResetLink
End
Else
Begin
Select 0 as IsValidPasswordResetLink
End
End
```

Stored Procedure to change password

```
Create Proc spChangePassword
@GUID uniqueidentifier,
@Password nvarchar(100)
as
Begin
Declare @UserId int
```

```

Select @UserId = UserId
from tblResetPasswordRequests
where Id= @ GUID

if(@UserId is null)
Begin
-- If UserId does not exist
Select 0 as IsPasswordChanged
End
Else
Begin
-- If UserId exists, Update with new password
Update tblUsers set
[Password] = @Password
where Id = @ UserId

-- Delete the password reset request row
Delete from tblResetPasswordRequests
where Id = @ GUID

Select 1 as IsPasswordChanged
End
End

```

ChangePassword.aspx.cs page code

```

<div style="font-family: Arial">
<table style="border: 1px solid black">
  <tr>
    <td colspan="2">
      <b>Change Password</b>
    </td>
  </tr>
  <tr>
    <td>
      New Password
    </td>
    <td>
      <asp:TextBox ID="txtNewPassword" TextMode="Password"
runat="server"></asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidatorNewPassword"
runat="server" ErrorMessage="New Password required"
Text="*" ControlToValidate="txtNewPassword" ForeColor="Red">
      </asp:RequiredFieldValidator>
    </td>
  </tr>
  <tr>
    <td>
      Confirm New Password
    </td>
    <td>

```

```

        :<asp:TextBox ID="txtConfirmNewPassword" TextMode="Password" runat="server">
        </asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidatorConfirmNewPassword"
            runat="server" ErrorMessage="Confirm New Password required" Text="*"
            ControlToValidate="txtConfirmNewPassword"
            ForeColor="Red" Display="Dynamic"></asp:RequiredFieldValidator>
        <asp:CompareValidator ID="CompareValidatorPassword" runat="server"
            ErrorMessage="New Password and Confirm New Password must match"
            ControlToValidate="txtConfirmNewPassword" ForeColor="Red"
            ControlToCompare="txtNewPassword"
            Display="Dynamic" Type="String" Operator="Equal" Text="*">
        </asp:CompareValidator>
    </td>
</tr>
<tr>
    <td>

    </td>
    <td>
        &nbsp;<asp:Button ID="btnSave" runat="server"
            Text="Save" onclick="btnSave_Click" Width="70px" />
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:Label ID="lblMessage" runat="server">
        </asp:Label>
    </td>
</tr>
<tr>
    <td colspan="2">
        <asp:ValidationSummary ID="ValidationSummary1"
            ForeColor="Red" runat="server" />
    </td>
</tr>
</table>
</div>

```

ChangePassword.aspx.cs page code

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (!IsPasswordResetLinkValid())
        {
            lblMessage.ForeColor = System.Drawing.Color.Red;
            lblMessage.Text = "Password Reset link has expired or is invalid";
        }
    }
}

```



```

protected void btnSave_Click(object sender, EventArgs e)
{
    if (ChangeUserPassword())
    {
        lblMessage.Text = "Password Changed Successfully!";
    }
    else
    {
        lblMessage.ForeColor = System.Drawing.Color.Red;
        lblMessage.Text = "Password Reset link has expired or is invalid";
    }
}

private bool ExecuteSP(string SPName, List<SqlParameter> SPParameters)
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(CS))
    {
        SqlCommand cmd = new SqlCommand(SPName, con);
        cmd.CommandType = CommandType.StoredProcedure;

        foreach (SqlParameter parameter in SPParameters)
        {
            cmd.Parameters.Add(parameter);
        }

        con.Open();
        return Convert.ToBoolean(cmd.ExecuteScalar());
    }
}

private bool IsPasswordResetLinkValid()
{
    List<SqlParameter> paramList = new List<SqlParameter>()
    {
        new SqlParameter()
        {
            ParameterName = "@GUID",
            Value = Request.QueryString["uid"]
        }
    };

    return ExecuteSP("spIsPasswordResetLinkValid", paramList);
}

private bool ChangeUserPassword()
{
    List<SqlParameter> paramList = new List<SqlParameter>()
    {
        new SqlParameter()
        {
            ParameterName = "@GUID",

```

```
        Value = Request.QueryString["uid"]
    },
    new SqlParameter()
    {
        ParameterName = "@Password",
        Value = FormsAuthentication.HashPasswordForStoringInConfigFile(txtNewPassword.Text,
"SHA1")
    }
};

return ExecuteSP("spChangePassword", paramList);
}
```

## Changing password by providing current password

We will discuss about, Changing password by providing current password. In real time, users can change their password any time, by providing their current password.

Stored procedure to change password, using their current password

```
Create Proc spChangePasswordUsingCurrentPassword
```

```
@UserName nvarchar(100),
```

```
@CurrentPassword nvarchar(100),
```

```
@NewPassword nvarchar(100)
```

```
as
```

```
Begin
```

```
if(Exists(Select Id from tblUsers
```

```
    where UserName = @UserName
```

```
    and [Password] = @CurrentPassword))
```

```
Begin
```

```
Update tblUsers
```

```
Set [Password] = @NewPassword
```

```
where UserName = @UserName
```

```
Select 1 as IsPasswordChanged
```

```
End
```

```
Else
```

```
Begin
```

```
Select 0 as IsPasswordChanged
```

```
End
```

```
End
```

ChangePassword.aspx HTML

```
<div style="font-family: Arial">
```

```
<table style="border: 1px solid black">
```

```
<tr>
```

```
<td colspan="2">
```

```
<b>Change Password</b>
```

```
</td>
```

```
</tr>
```

```
<tr id="trCurrentPassword" runat="server">
```

```
<td>
```

```
Current Password
```

```
</td>
```

```
<td>
```

```
<asp:TextBox ID="txtCurrentPassword" TextMode="Password"
```

```
runat="server"></asp:TextBox>
```

```
<asp:RequiredFieldValidator ID="RequiredFieldValidatorCurrentPassword"
```

```
runat="server" ErrorMessage="Current Password required"
```

```
Text="*" ControlToValidate="txtCurrentPassword" ForeColor="Red">
```

```
</asp:RequiredFieldValidator>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td>
```

```
New Password
```

```

</td>
<td>
    <:asp:TextBox ID="txtNewPassword" TextMode="Password"
    runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidatorNewPassword"
    runat="server" ErrorMessage="New Password required"
    Text="*" ControlToValidate="txtNewPassword" ForeColor="Red">
    </asp:RequiredFieldValidator>
</td>
</tr>
<tr>
<td>
    Confirm New Password
</td>
<td>
    <:asp:TextBox ID="txtConfirmNewPassword" TextMode="Password" runat="server">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidatorConfirmNewPassword"
    runat="server" ErrorMessage="Confirm New Password required" Text="*"
    ControlToValidate="txtConfirmNewPassword"
    ForeColor="Red" Display="Dynamic"></asp:RequiredFieldValidator>
    <asp:CompareValidator ID="CompareValidatorPassword" runat="server"
    ErrorMessage="New Password and Confirm New Password must match"
    ControlToValidate="txtConfirmNewPassword" ForeColor="Red"
    ControlToCompare="txtNewPassword"
    Display="Dynamic" Type="String" Operator="Equal" Text="*">
    </asp:CompareValidator>
</td>
</tr>
<tr>
<td>
</td>
<td>
    &nbsp;<asp:Button ID="btnSave" runat="server"
    Text="Save" onclick="btnSave_Click" Width="70px" />
</td>
</tr>
<tr>
<td colspan="2">
    <asp:Label ID="lblMessage" runat="server">
    </asp:Label>
</td>
</tr>
<tr>
<td colspan="2">
    <asp:ValidationSummary ID="ValidationSummary1"
    ForeColor="Red" runat="server" />
</td>
</tr>
</table>
</div>

```

ChangePassword.aspx.cs code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.QueryString["uid"] == null && User.Identity.Name == "")
    {
        Response.Redirect("~/Login.aspx");
    }
    if (!IsPostBack)
    {
        if (Request.QueryString["uid"] != null)
        {
            if (!IsPasswordResetLinkValid())
            {
                lblMessage.ForeColor = System.Drawing.Color.Red;
                lblMessage.Text = "Password Reset link has expired or is invalid";
            }
            trCurrentPassword.Visible = false;
        }
        else if (User.Identity.Name != "")
        {
            trCurrentPassword.Visible = true;
        }
    }
}

protected void btnSave_Click(object sender, EventArgs e)
{
    if ((Request.QueryString["uid"] != null && ChangeUserPassword()) ||
        (User.Identity.Name != "" && ChangeUserPasswordUsingCurrentPassword()))
    {
        lblMessage.Text = "Password Changed Successfully!";
    }
    else
    {
        lblMessage.ForeColor = System.Drawing.Color.Red;
        if (trCurrentPassword.Visible)
        {
            lblMessage.Text = "Invalid Current Password!";
        }
        else
        {
            lblMessage.Text = "Password Reset link has expired or is invalid";
        }
    }
}

private bool ExecuteSP(string SPName, List<SqlParameter> SPParameters)
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(CS))
```

```

{
    SqlCommand cmd = new SqlCommand(SPName, con);
    cmd.CommandType = CommandType.StoredProcedure;

    foreach (SqlParameter parameter in SPParameters)
    {
        cmd.Parameters.Add(parameter);
    }

    con.Open();
    return Convert.ToBoolean(cmd.ExecuteScalar());
}
}

private bool IsPasswordResetLinkValid()
{
    List<SqlParameter> paramList = new List<SqlParameter>()
    {
        new SqlParameter()
        {
            ParameterName = "@GUID",
            Value = Request.QueryString["uid"]
        }
    };

    return ExecuteSP("spIsPasswordResetLinkValid", paramList);
}

private bool ChangeUserPassword()
{
    List<SqlParameter> paramList = new List<SqlParameter>()
    {
        new SqlParameter()
        {
            ParameterName = "@GUID",
            Value = Request.QueryString["uid"]
        },
        new SqlParameter()
        {
            ParameterName = "@Password",
            Value = FormsAuthentication.HashPasswordForStoringInConfigFile(txtNewPassword.Text,
"SHA1")
        }
    };

    return ExecuteSP("spChangePassword", paramList);
}

private bool ChangeUserPasswordUsingCurrentPassword()
{
    List<SqlParameter> paramList = new List<SqlParameter>()
    {

```

```

new SqlParameter()
{
    ParameterName = "@UserName",
    Value = User.Identity.Name
},
new SqlParameter()
{
    ParameterName = "@CurrentPassword",
    Value =
FormsAuthentication.HashPasswordForStoringInConfigFile(txtCurrentPassword.Text, "SHA1")
},
new SqlParameter()
{
    ParameterName = "@NewPassword",
    Value = FormsAuthentication.HashPasswordForStoringInConfigFile(txtNewPassword.Text,
"SHA1")
}
};

return ExecuteSP("spChangePasswordUsingCurrentPassword", paramList);
}

```

## Unlocking the locked user accounts using a web page

If a user repeatedly enters the wrong password. The accounts are locked to prevent hackers from guessing passwords and making dictionary attacks.

we will discuss about unlocking the locked user accounts, using a web page that lists all the locked user accounts. From this page, the help desk agent, can unlock the account by clicking a button. This is not as dangerous as running a manual update query, but still a manual process and may be inefficient.

Stored procedure to get the information about, all the locked user accounts.

```
Create proc spGetAllLocakedUserAccounts
as
Begin
Select UserName, Email, LockedDateTime,
DATEDIFF(hour, LockedDateTime, GETDATE()) as HoursElapsed
from tblUsers
where IsLocked = 1
End
```

Add a webform, with name "AccessDenied.aspx".

```
<div style="font-family:Arial;">
  <h1 style="color:Red">Access Denied</h1>
</div>
```

Add a webform, with name "LockedAccounts.aspx". Copy and paste the following HTML on this page.

```
<div style="font-family:Arial">
  <asp:GridView ID="gvLockedAccounts" runat="server" AutoGenerateColumns="False">
    <Columns>
      <asp:BoundField DataField="UserName" HeaderText="User Name" />
      <asp:BoundField DataField="Email" HeaderText="Email" />
      <asp:BoundField DataField="LockedDateTime"
        HeaderText="Locked Date & Time" />
      <asp:BoundField DataField="HoursElapsed" HeaderText="Hours Elapsed" >
        <ItemStyle HorizontalAlign="Center" />
      </asp:BoundField>
      <asp:TemplateField HeaderText="Enable">
        <ItemTemplate>
          <asp:Button ID="btnEnable" runat="server" Text="Enable"
            Enabled='<%#Convert.ToInt32(Eval("HoursElapsed")) > 24%>' />
        </ItemTemplate>
      </asp:TemplateField>
    </Columns>
  </asp:GridView>
</div>
```

"LockedAccounts.aspx.cs" code



```

protected void Page_Load(object sender, EventArgs e)
{
    if (User.Identity.Name.ToLower() == "test")
    {
        if (!IsPostBack)
        {
            GetData();
        }
    }
    else
    {
        Response.Redirect("~/AccessDenied.aspx");
    }
}

private void GetData()
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(CS))
    {
        SqlCommand cmd = new SqlCommand("spGetAllLockedUserAccounts", con);
        cmd.CommandType = CommandType.StoredProcedure;

        con.Open();
        gvLockedAccounts.DataSource = cmd.ExecuteReader();
        gvLockedAccounts.DataBind();
    }
}

```

## Implementing Enable button to unlock user accounts

To implement, the "Enable" button, make the following changes to the gridview control.

First Change: Specify the CommandArgument attribute of the Button control in the Template column.

```
<asp:TemplateField HeaderText="Enable">
    <ItemTemplate>
        <asp:Button ID="btnEnable" runat="server" CommandArgument='<%# Eval("UserName")
%>'
            Text="Enable" Enabled='<%#Convert.ToInt32(Eval("HoursElapsed")) > 24%>' />
    </ItemTemplate>
</asp:TemplateField>
```

Second Change: Generate the "RowCommand" event handler for the GridView control.

1. Right Click on the GridView Control and Select properties.
2. In the "Properties Window", click on events icon.
3. In the events windows, double click on the text box next to "Row Command" event.

With these 2 changes the HTML of the "LockedAccounts.aspx" should look as shown below.

```
<div style="font-family: Arial">
<asp:GridView ID="gvLockedAccounts" runat="server" AutoGenerateColumns="False"
    OnRowCommand="gvLockedAccounts_RowCommand">
    <Columns>
        <asp:BoundField DataField="UserName" HeaderText="User Name" />
        <asp:BoundField DataField="Email" HeaderText="Email" />
        <asp:BoundField DataField="LockedDateTime" HeaderText="Locked Date & Time" />
        <asp:BoundField DataField="HoursElapsed" HeaderText="Hours Elapsed">
            <ItemStyle HorizontalAlign="Center" />
        </asp:BoundField>
        <asp:TemplateField HeaderText="Enable">
            <ItemTemplate>
                <asp:Button ID="btnEnable" CommandArgument='<%# Eval("UserName") %>'
runat="server"
                    Text="Enable" Enabled='<%#Convert.ToInt32(Eval("HoursElapsed")) > 24%>' />
            </ItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
</div>
```

Copy and paste the following private method in "LockedAccounts.aspx.cs" page.

```
private void EnableUserAccount(string UserName)
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(CS))
    {
        SqlCommand cmd = new SqlCommand("spEnableUserAccount", con);
```

```

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter paramUserName = new SqlParameter()
{
    ParameterName = "@UserName",
    Value = UserName
};

cmd.Parameters.Add(paramUserName);

con.Open();
cmd.ExecuteNonQuery();
}
}

```

Invoke EnableUserAccount() method, in RowCommand() event handler as shown below.

```

protected void gvLockedAccounts_RowCommand(object sender, GridViewCommandEventArgs e)
{
    EnableUserAccount(e.CommandArgument.ToString());
    GetData();
}

```

## **Secure Socket Layer in asp.net**

We will discuss about:

1. The advantage of using HTTPS over HTTP protocol
2. How to identify, if the web application i am accessing, use HTTPS protocol
3. How to configure HTTPS instead of HTTP for asp.net web applications
4. What is SSL or Secure Socket Layer and how is it different from HTTPS
5. Who issues server certificates and can't I generate test certificates
6. What about performance when using HTTPS over HTTP

### **Advantages of using HTTPS**

HTTP stands for Hyper Text Transfer Protocol. HTTPS, stands for Hyper Text Transfer Protocol Secure. As the name suggests, HTTPS is more secure than HTTP. When the web server and the client communicate, using HTTP, protocol, the messages that are exchanged over the internet are not encrypted. Any one can secretly listen and see the messages that are exchanged between the client and the web server. That's why, any sensitive information like passwords, financial transactions should never be done over HTTP protocol. Most of the banking applications use HTTPS protocol. Messages exchanged between the client and web server, using the HTTPS protocol are encrypted and are very secure. HTTP use port 80 and HTTPS use port 443.

### **How to identify, if the web application i am accessing, use HTTPS protocol**

There are 2 ways:

1. Browser displays a LOCK symbol either in the address or status bar. Click on the lock icon, for more information like, the certificate issuing authority, encryption key length etc.
2. In the address bar look for HTTPS instead of HTTP

### **How to configure HTTPS instead of HTTP for asp.net web applications**

IIS is the web server for asp.net web applications. so the configuration to use HTTPS, is usually done in IIS. The encryption and decryption of messages exchanged between the client and the server is done by server certificates. These server certificates needs to be installed on the IIS server.

### **What is Secure Socket Layer and how is it different from HTTPS**

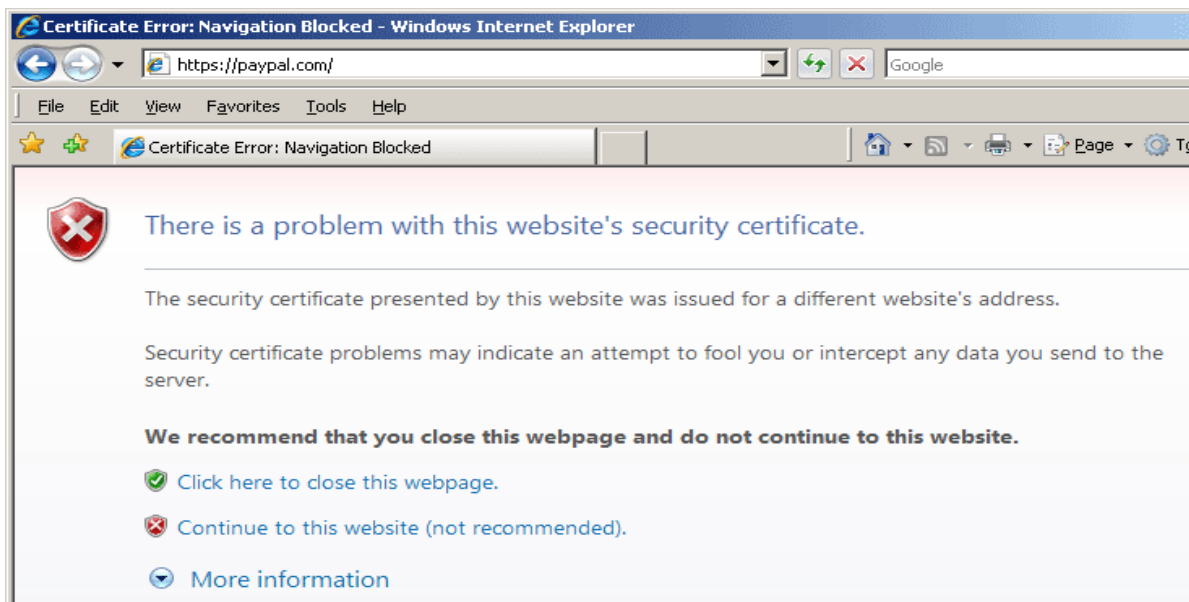
HTTPS is HTTP (HyperText Transfer Protocol) plus SSL (Secure Socket Layer). SSL standing for Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a web server and a browser, so that the data sent over the Internet can't be read by others. When a user requests a secure Web page, the server generates an encryption key for the user's session and then encrypts the page's data before sending a response. On the client side, the browser uses that same encryption key to decrypt the requested Web page and to encrypt new requests sent from that page. SSL uses server certificates for encryption and decryption. An SSL certificate contains a public key and certificate issuer. Not only can clients use the certificate to communicate with a server, clients can verify that the certificate was cryptographically signed by an official Certificate Authority. For example, if your browser trusts the VeriSign Certificate Authority, and VeriSign signs my SSL certificate, your browser will inherently trust my SSL certificate.

## Who issues server certificates and can't I generate test certificates

Server certificates are issued by an entity called certificate authority. There are several trusted certificate authorities like:

1. verisign
2. Thawte
3. Geotrust
4. Comodo
5. GoDaddy

The certificate authority acts as a clearing house to verify the server's identity over the Internet. When a browser requests a page over https, the browser also, requests the server certificate and checks it against a list of trusted sites provided by the certificate authority. If the server certificate does not match one of the sites already authorized by the user, or if the server certificate does not match the Web address for which it was registered, or if there are any other problems with the server certificate, a warning message is displayed. The warning message from internet explorer is shown below.



Besides providing encryption and decryption for secure data transmission, certificate authority also provides assurance to users that a website is authentic.

It is also possible to generate our own server certificates, using a tool called makecert.exe. This tool comes with visual studio and can be used from visual studio command prompt. The certificates that are generated using this tool, can only be used for testing purposes and not for production use.

## What about performance when using HTTPS over HTTP

Extra processing time is required for HTTPS, for key negotiation. Key negotiation is also termed as SSL handshake. The handshake allows the server to authenticate itself to the client.

## Implementing SSL in asp.net web application

We will discuss about:

1. Understand the term self-signed certificates
2. Creating self-signed certificates
3. Configuring an asp.net web application to use SSL, that is use HTTPS instead of HTTP
4. Importing and exporting certificates

### What are self signed certificates

A self-signed certificate is an identity certificate that is signed by its own creator. Certificates are signed by Certificate Authority. In general self signed certificates are fine for testing purposes and not for production use.

### Creating self-signed certificates

There are several ways to create self signed test certificates. Let us explore 2 of the easier options available. The easiest and simpler approach is to use IIS to create these certificates. In IIS 7.5

1. Click on the "Server Name"
2. Double click "Server Certificates" feature
3. Click on "Create Self Signed Certificate" link, under "Actions"
4. Specify a friendly name for the certificate and click OK. The friendly name is not part of the certificate itself, but is used by the server administrator to easily distinguish the certificate.

The generated test certificate, is also automatically installed into the certificate store.

**MakeCert.exe** tool can be used as another way to generate, test certificates. The following link from microsoft explains, various options that can be used with this tool. This is a command line tool and must be run from visual studio command prompt.

<http://msdn.microsoft.com/en-us/library/bfskty3.aspx>

**Makecert** -r -pe -n "CN=YourComputerName" -b 01/01/2000 -e 01/01/2100 -ss my -sr localMachine -sky exchange -sp "Microsoft RSA SChannel Cryptographic Provider" -sy 12

Note: Replace YourComputerName, with the name of your computer.

### Associating an asp.net web application with a specific certificate

Add HTTPS site binding, if it is not already present:

1. Open IIS
2. Expand the "Server Name"
3. Expand "Sites"
4. Select "Default Web Site"
5. Click "Binding" under "Edit Site" in "Actions" pane.
6. In the "Site Bindings" window, Click "Add"
7. Select Type = "https" and the SSL Certificate and click "OK"
8. Click "Close" on "Site Bindings" window

At this point, you will be able to access your application using both HTTP and HTTPS protocol.

When the site is accessed over HTTPS, you may receive a browser warning about the authenticity of the website.

If you want to dis-allow, access over HTTP protocol there are 2 ways:

**First Way:** Remove HTTP binding at the IIS Server level. This option will prevent all the web applications, running on that server to use only HTTPS binding.

**Second Way:** Let both the bindings be available at the server level and configure SSL settings at an application or web site level.

1. Select your web application in IIS
2. Double click "SSL Settings" from the features window
3. Make sure "Require SSL" check box is checked.
4. Click "Apply" under "Actions" pane

Now, if you try to access the application using HTTP instead of HTTPS, you will get an error

**HTTP Error 403.4 - Forbidden**

**The page you are trying to access is secured with Secure Sockets Layer (SSL)**

Use Import and Export feature of IIS to import and export certificates

## Redirect http to https in IIS

To redirect users from HTTP to HTTPS automatically, there are several ways:

### Step 1:

Please download and install "URL ReWrite" module from the following link.

<http://www.iis.net/downloads/microsoft/url-rewrite>

### Step 2:

Uncheck "Require SSL" option from "SSL Settings" for the web application in IIS.

### Step 3:

Copy and paste the following in the root web.config file of your application.

```
<system.webServer>
  <httpRedirect enabled="false" destination="" httpResponseStatus="Found" />
  <rewrite>
    <rules>
      <rule name="HTTP to HTTPS Redirection" stopProcessing="true">
        <match url="(.*)" />
        <conditions>
          <add input="{HTTPS}" pattern="off" />
        </conditions>
        <action type="Redirect" url="https://{HTTP_HOST}{REQUEST_URI}"
redirectType="Found" />
      </rule>
    </rules>
  </rewrite>
</system.webServer>
```

Now try to navigate to the application using HTTP, you will automatically be redirected to HTTPS.

These rules can also be created in IIS directly using the "URL Rewrite" module



## Redirect http to https in IIS using custom errors

We will discuss about redirecting users from HTTP to HTTPS, using "IIS Error Pages". We discussed about redirecting users using "URL ReWrite" module.

Custom error pages can be set at the server level or at a specific application level in IIS. In this demo, we will discuss about setting custom error pages at the server level. There are 3 simple steps.

### Step 1:

Make sure "Require SSL" option from "SSL Settings" is checked for your web application in IIS. Now, browse the web site, using HTTP, and you will receive the following error. Pay attention to HTTP error code - 403.4, which we will be using later.

**HTTP Error 403.4 - Forbidden**

The page you are trying to access is secured with Secure Sockets Layer (SSL).

### Step 2:

Copy and paste the following HTML in a notepad and save it as "RedirectToHttps.htm" in "C:\inetpub".

```
<html>
<head>
  <title>
    Redirecting to HTTPS
  </title>
</head>
<script language="JavaScript">
function redirectHttpToHttps()
{
  var httpURL= window.location.hostname + window.location.pathname;
  var httpsURL= "https://" + httpURL;
  window.location = httpsURL;
}
redirectHttpToHttps();
</script>
<body>
</body>
</html>
```

### Step 3:

1. In IIS, select the "Server Name" and double click "Error Pages" to open the feature.
2. Click on "Add" link under "Actions"
3. Set Status Code = 403.4, File Path = C:\Inetpub\RedirectToHttps.htm and click "OK"
4. Now click "Edit Feature Settings" link under "Actions"
5. Select "Custom Error Pages" and Path = C:\inetpub\RedirectToHttps.htm

Now, access the application using HTTP. You will be automatically redirected to HTTPS.

## User controls in asp.net

Web user controls combine one or more server or HTML controls on a Web user control page, which can, in turn, be used on a Web form as a single control. For example, to capture dates from the end user on a webform, we need a TextBox, ImageButton and, a Calendar control. A web form to capture date of birth is shown below in the image.



To select the date:

1. User clicks on the calendar image.
2. The Calendar control becomes visible.
3. User selects a date from the calendar.
4. Textbox control is automatically populated with the selected date and the calendar becomes invisible.

To achieve this functionality, considerable amount of code needs to be written in the webform.

If, I am capturing dates, on multiple web forms, rather than repeating the same HTML mark up and code, on each and every web form, we can encapsulate everything into a single web user control, which in turn, can be used on multiple web forms. This way we are reusing the same code, which saves a lot of time in terms of development and testing. So in short, user controls, increase re-usability of code, implement encapsulation and reduce development and maintenance time.

Designing and implementing web user controls is very similar to web forms. Web forms, have the extension of .aspx, where as web user controls have the extension of .ascx. Webforms begin with @Page directive and can have <html>, <head>, and <body> elements, where as a web user controls begin with @ Control directive and cannot have html, head, and body elements. Just, like webforms, user controls also have code behind files.

We will create a custom calendar user control, that can be reused on multiple webforms. To create a user control:

1. Right click on the web application project in solution explorer
2. Select Add >> New Item
3. From the "Add New Item" dialog box, select "Web User Control"
4. Set Name = CalendarUserControl
5. Click on "Add"

Notice that, CalendarUserControl.ascx page is created. Copy and paste the following HTML.

```
<asp:TextBox ID="txtDate" runat="server" Width="115px"></asp:TextBox>
<asp:ImageButton ID="ImgBtn" runat="server"
    ImageUrl="~/Images/Calendar.png" onclick="ImgBtn_Click" />
<asp:Calendar ID="Calendar1" runat="server"
    onselectionchanged="Calendar1_SelectionChanged">
</asp:Calendar>
```

CalendarUserControl.ascx.cs code

```
public partial class CalendarUserControl : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            Calendar1.Visible = false;
        }
    }

    protected void ImgBtn_Click(object sender, ImageClickEventArgs e)
    {
        if (Calendar1.Visible)
        {
            Calendar1.Visible = false;
        }
        else
        {
            Calendar1.Visible = true;
        }
    }

    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        txtDate.Text = Calendar1.SelectedDate.ToShortDateString();
        Calendar1.Visible = false;
    }
}
```

We are done creating the calendar user control.

## Adding and using user controls on a webform

We will discuss about:

1. Adding and using user controls on a webform
  2. Adding properties to the user control
- Adding and using user controls on a webform

Adding user controls to a web page is very straight forward. Simply drag the user control from solution explorer and drop it on the web page. Make sure, the "Design" view of the webform is selected before dragging and dropping the user control on the webform. This will automatically:

1. Add a "Register" directive for the user control and
2. The control declaration

"Register" directive for the CalendarUserControl

```
<% @ Register src="CalendarUserControl.ascx" tagname="CalendarUserControl" tagprefix="uc1" %>
```

Control declaration for the CalendarUserControl

```
<uc1:CalendarUserControl ID="CalendarUserControl1" runat="server" />
```

Notice, the "tagprefix" and "tagname" in the "Register" directive. These are used in the control declaration. For asp.net controls, the "tagprefix" is "asp". Tagprefix, can be changed, if you wish to do so.

If you intend to add the user control on multiple web forms, rather than including the "Register" directive on each and every web form, every time, the control can be registered once in web.config file and can be used on any number of web forms, without the "Register" directive.

```
<system.web>
  <pages>
    <controls>
      <add src="~/CalendarUserControl.ascx" tagName="CalendarUserControl" tagPrefix="uc1" />
    </controls>
  </pages>
</system.web>
```

At this point, you get the following error, if both, the user control and the webform are in the same directory. This limitation is by design due to an internal design consideration for performance. The page '/WebForm2.aspx' cannot use the user control '/CalendarUserControl.ascx', because it is registered in web.config and lives in the same directory as the page.

To solve this error move the user control to a different folder, and update the "src" attribute of the "Register" directive in web.config file accordingly.

### Adding properties to the user control:

A user control can also have it's own properties and methods. At the moment, CalendarUserControl does not expose any property that returns the selected date.

For example, drag and drop a button control on the same webform. when I click this button, we want

to print the selected date. To do this let's add the following SelectedDate property for the CalendarUserControl.

```
public string SelectedDate
{
    get
    {
        return txtDate.Text;
    }
    set
    {
        txtDate.Text = value;
    }
}
```

On the webform, in the button click event, I should now be able to retrieve, the selected date using "SelectedDate" property of the "CalendarUserControl" as shown below.

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write(CalendarUserControl1.SelectedDate);
}
```

You can also set this property declaratively in the HTML at design time as shown below. When this webform, loads, it shows the date, that we have set.

```
<uc1:CalendarUserControl SelectedDate="01/01/2013" ID="CalendarUserControl1" runat="server" />
```

But one limitation, here with the user control, is that the design time value is not shown in the control at design time. This is by design, and there are 2 ways to solve this issue:

1. Create a custom control instead of user control.
2. Compile the user control into a DLL.

## Raising custom events from user controls

we will discuss about :

1. Adding events to UserControl
2. Events and delegates

Most people feel "events and delegates" are complex and difficult to understand. Events and delegates are not that complex to understand, if the basics are right.

Very important points to keep in mind, when understanding "Events and Delegates":

1. Delegates are function pointers, and their syntax is very similar to that of a function.
2. Events are variables of type delegates with an event keyword.

If these points are not clear at the moment, don't worry, they will be much clear as we progress. At the moment, the CalendarUserControl does not have any custom events. Let us say, we want to raise CalendarVisibilityChanged event every time the visibility of the calendar changes. The visibility of the calendar is toggled by clicking on the image button.

The following are the steps to raise CalendarVisibilityChanged event from the CalendarUserControl:

**Step 1:** Create CalendarVisibilityChangedEventArgs class that will contain the event data.

```
public class CalendarVisibilityChangedEventArgs : EventArgs
{
    private bool _isCalendarVisible;

    // Constructor to initialize event data
    public CalendarVisibilityChangedEventArgs(bool isCalendarVisible)
    {
        this._isCalendarVisible = isCalendarVisible;
    }

    // Returns true if the calendar is visible otherwise false
    public bool IsCalendarVisible
    {
        get
        {
            return this._isCalendarVisible;
        }
    }
}
```

**Step 2:** Create CalendarVisibilityChangedEventHandler delegate. "sender" is the reference variable that points to the instance of the CalendarUserControl, that raises this event.

"CalendarVisibilityChangedEventArgs" object will contain "CalendarVisibilityChanged" event data.

```
public delegate void CalendarVisibilityChangedEventHandler(object sender,
CalendarVisibilityChangedEventArgs e);
```

**Step 3:** Create CalendarVisibilityChanged event. Remember that, an event is a variable of type delegate. In the line below, we are just creating a variable "CalendarVisibilityChanged" of type "CalendarVisibilityChangedEventHandler" with delegate keyword in front of it.

```
public event CalendarVisibilityChangedEventHandler CalendarVisibilityChanged;
```

**Step 4:** Create a protected virtual method to raise the event. Since this method is protected and virtual, all classes deriving from the CalendarUserControl class can override this method, if they wish to do so. This method enables the derived classes to do some additional work before the event can be raised. Just before raising the event, we are checking if CalendarVisibilityChanged is null. If you are not sure about this, please don't worry.

```
protected virtual void OnCalendarVisibilityChanged(CalendarVisibilityChangedEventArgs e)
{
    if (CalendarVisibilityChanged != null)
    {
        CalendarVisibilityChanged(this, e);
    }
}
```

For example, if we have a class "DerivedCalendarUserControl" that derives from CalendarUserControl class. "DerivedCalendarUserControl" can override the virtual "OnCalendarVisibilityChanged()" method as shown below. "CalendarVisibilityChanged" will only be raised when "base.OnCalendarVisibilityChanged(e);" is invoked. So, using a "protected virtual" method to raise events is a very useful technique.

```
public class DerivedCalendarUserControl : CalendarUserControl
{
    // Other methods, properties etc..

    protected override void OnCalendarVisibilityChanged(CalendarVisibilityChangedEventArgs e)
    {
        // Do some additional work before raising the event
        base.OnCalendarVisibilityChanged(e);
    }
}
```

**Step 5:** Finally raise the event, whenever the visibility of the Calendar is changed in the CalendarUserControl. The calendar visibility is changed, whenever the user clicks on the image button and when the date in the calendar is selected. So, raise "CalendarVisibilityChanged" event from ImgBtn\_Click() and Calendar1\_SelectionChanged(). Before raising the event, create an instance of "CalendarVisibilityChangedEventArgs" and pass event data, that is "true" or "false" to the constructor of this class.

```
protected void ImgBtn_Click(object sender, ImageClickEventArgs e)
{
    if (Calendar1.Visible)
    {
        Calendar1.Visible = false;
        CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventData =
            new CalendarVisibilityChangedEventArgs(false);
        OnCalendarVisibilityChanged(calendarVisibilityChangedEventData);
    }
}
```

```

    }
    else
    {

        Calendar1.Visible = true;
        CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventArgs =
            new CalendarVisibilityChangedEventArgs(true);
        OnCalendarVisibilityChanged(calendarVisibilityChangedEventArgs);
    }
}

```

```

protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    txtDate.Text = Calendar1.SelectedDate.ToShortDateString();
    Calendar1.Visible = false;
    CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventArgs =
        new CalendarVisibilityChangedEventArgs(false);
    OnCalendarVisibilityChanged(calendarVisibilityChangedEventArgs);
}

```

Here is the complete CalendarUserControl code

```

public partial class CalendarUserControl : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            Calendar1.Visible = false;
        }
    }

    protected void ImgBtn_Click(object sender, ImageClickEventArgs e)
    {
        if (Calendar1.Visible)
        {
            Calendar1.Visible = false;
            CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventArgs =
                new CalendarVisibilityChangedEventArgs(false);
            OnCalendarVisibilityChanged(calendarVisibilityChangedEventArgs);
        }
        else
        {
            Calendar1.Visible = true;
            CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventArgs =
                new CalendarVisibilityChangedEventArgs(true);
            OnCalendarVisibilityChanged(calendarVisibilityChangedEventArgs);
        }
    }

    protected void Calendar1_SelectionChanged(object sender, EventArgs e)

```



```

{
    txtDate.Text = Calendar1.SelectedDate.ToShortDateString();
    Calendar1.Visible = false;
    CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventData =
        new CalendarVisibilityChangedEventArgs(false);
    OnCalendarVisibilityChanged(calendarVisibilityChangedEventData);
}

public string SelectedDate
{
    get
    {
        return txtDate.Text;
    }
    set
    {
        txtDate.Text = value;
    }
}

public event CalendarVisibilityChangedEventHandler CalendarVisibilityChanged;

protected virtual void OnCalendarVisibilityChanged(CalendarVisibilityChangedEventArgs e)
{
    if (CalendarVisibilityChanged != null)
    {
        CalendarVisibilityChanged(this, e);
    }
}

public class CalendarVisibilityChangedEventArgs : EventArgs
{
    private bool _isCalendarVisible;

    // Constructor to initialize event data
    public CalendarVisibilityChangedEventArgs(bool isCalendarVisible)
    {
        this._isCalendarVisible = isCalendarVisible;
    }

    // Returns true if the calendar is visible otherwise false
    public bool IsCalendarVisible
    {
        get
        {
            return this._isCalendarVisible;
        }
    }
}

```

```
public delegate void CalendarVisibilityChangedEventHandler(object sender,  
CalendarVisibilityChangedEventArgs e);
```

## Consuming user control custom events

We will discuss about:

1. Consuming custom events of the user control
2. Understanding the importance of, checking if the event is null, before raising the event.

```
protected virtual void OnCalendarVisibilityChanged(CalendarVisibilityChangedEventArgs e)
{
    // NULL check
    if (CalendarVisibilityChanged != null)
    {
        CalendarVisibilityChanged(this, e);
    }
}
```

### Consuming custom event "CalendarVisibilityChanged"

To consume the event, there are 2 simple steps:

#### Step 1:

Create an event handler method as shown below. The method signature must match the signature of the "CalendarVisibilityChangedEventHandler" delegate. Notice that, in the event handler method, we are retrieving event data using "IsCalendarVisible" property.

```
protected void CalendarUserControl1_CalendarVisibilityChanged(object sender,
CalendarVisibilityChangedEventArgs e)
{
    Response.Write("Calendar Visible = " + e.IsCalendarVisible.ToString());
}
```

**Step 2:** Register event handler method "CalendarUserControl1\_CalendarVisibilityChanged()" with "CalendarVisibilityChanged" events of the "CalendarUserControl" using "+=" as shown below. Do this, in the Page\_Load() event of "WebForm1". To unregister we can use "-=".

```
protected void Page_Load(object sender, EventArgs e)
{
    CalendarUserControl1.CalendarVisibilityChanged +=
    new
    CalendarVisibilityChangedEventHandler(CalendarUserControl1_CalendarVisibilityChanged);
}
```

That's it. Run the project and click on the calendar image to toggle the display, the custom event will be raised and handled. You should see a message "Calendar Visible = true" or "Calendar Visible = false" depending on the visibility of the calendar control.

Understanding the importance of, checking if the event is null, before raising the event  
Now comment the line that registers event handler method in the Page\_Load() event. Run the application and click on the image button. Nothing happens and also we don't get any run time errors.

Now comment the line that checks for null in "OnCalendarVisibilityChanged()" method as shown below.

```
protected virtual void OnCalendarVisibilityChanged(CalendarVisibilityChangedEventArgs e)
{
    // NULL check
    //if (CalendarVisibilityChanged != null)
    //{
        CalendarVisibilityChanged(this, e);
    //}
}
```

Run the application and click on the image button. You should get a "NullReferenceException". The exception is due to CalendarVisibilityChanged() being null. So, if there are no subscribers for the event, that is, if there are no event handler methods registered with CalendarVisibilityChanged event, and if we try to raise the event, we get the exception. To avoid this it is always better to check for null, before raising the event.

## Events and delegates in c#

We will discuss about raising DateSelected custom event, whenever, the user selects a "Date" from the "CalendarUserControl".

The following are the steps to raise this event:

**Step 1:** Create "DateSelectedEventArgs" class that will contain the event data.

```
public class DateSelectedEventArgs : EventArgs
{
    private DateTime _selectedDate;

    public DateSelectedEventArgs(DateTime selectedDate)
    {
        this._selectedDate = selectedDate;
    }

    public DateTime SelectedDate
    {
        get
        {
            return this._selectedDate;
        }
    }
}
```

**Step 2:** Create DateSelectedEventHandler delegate.

```
public delegate void DateSelectedEventHandler(object sender, DateSelectedEventArgs e);
```

**Step 3:** Create DateSelected event.

```
public event DateSelectedEventHandler DateSelected;
```

**Step 4:** Create a protected virtual method "OnDateSelection" to raise the event.

```
protected virtual void OnDateSelection(DateSelectedEventArgs e)
{
    if (DateSelected != null)
    {
        DateSelected(this, e);
    }
}
```

**Step 5:** Finally raise the event, whenever a date is selected from the CalendarUserControl.

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    txtDate.Text = Calendar1.SelectedDate.ToShortDateString();
    DateSelectedEventArgs dateSelectedEventData = new
    DateSelectedEventArgs(Calendar1.SelectedDate);
```

```
OnDateSelection(dateSelectedEventData);
```

```
// Rest of the code  
}
```

Finally, here is the complete CalendarUserControl code, with both the custom events:

1. "DateSelected" event
2. "CalendarVisibilityChanged" event

```
public partial class CalendarUserControl : System.Web.UI.UserControl  
{  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        if (!IsPostBack)  
        {  
            Calendar1.Visible = false;  
        }  
    }  
  
    protected void ImgBtn_Click(object sender, ImageClickEventArgs e)  
    {  
        if (Calendar1.Visible)  
        {  
            Calendar1.Visible = false;  
            CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventData =  
                new CalendarVisibilityChangedEventArgs(false);  
            OnCalendarVisibilityChanged(calendarVisibilityChangedEventData);  
        }  
        else  
        {  
            Calendar1.Visible = true;  
            CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventData = new  
CalendarVisibilityChangedEventArgs(true);  
            OnCalendarVisibilityChanged(calendarVisibilityChangedEventData);  
        }  
    }  
  
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)  
    {  
        txtDate.Text = Calendar1.SelectedDate.ToShortDateString();  
        DateSelectedEventArgs dateSelectedEventData = new  
DateSelectedEventArgs(Calendar1.SelectedDate);  
        OnDateSelection(dateSelectedEventData);  
  
        Calendar1.Visible = false;  
        CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventData = new  
CalendarVisibilityChangedEventArgs(false);  
        OnCalendarVisibilityChanged(calendarVisibilityChangedEventData);  
    }  
}
```

```

public string SelectedDate
{
    get
    {
        return txtDate.Text;
    }
    set
    {
        txtDate.Text = value;
    }
}

public event DateSelectedEventHandler DateSelected;

public event CalendarVisibilityChangedEventHandler CalendarVisibilityChanged;

protected virtual void OnCalendarVisibilityChanged(CalendarVisibilityChangedEventArgs e)
{
    if (CalendarVisibilityChanged != null)
    {
        CalendarVisibilityChanged(this, e);
    }
}

protected virtual void OnDateSelection(DateSelectedEventArgs e)
{
    if (DateSelected != null)
    {
        DateSelected(this, e);
    }
}
}

public class DateSelectedEventArgs : EventArgs
{
    private DateTime _selectedDate;

    public DateSelectedEventArgs(DateTime selectedDate)
    {
        this._selectedDate = selectedDate;
    }

    public DateTime SelectedDate
    {
        get
        {
            return this._selectedDate;
        }
    }
}

public class CalendarVisibilityChangedEventArgs : EventArgs

```

```

{
    private bool _isCalendarVisible;

    public CalendarVisibilityChangedEventArgs(bool isCalendarVisible)
    {
        this._isCalendarVisible = isCalendarVisible;
    }

    public bool IsCalendarVisible
    {
        get
        {
            return this._isCalendarVisible;
        }
    }
}

public delegate void CalendarVisibilityChangedEventHandler(object sender,
CalendarVisibilityChangedEventArgs e);

public delegate void DateSelectedEventHandler(object sender, DateSelectedEventArgs e);

```

To consume the event, create an event handler method as shown below:

```

protected void CalendarUserControl1_DateSelected(object sender, DateSelectedEventArgs e)
{
    Response.Write("Selected Date = " + e.SelectedDate.ToShortDateString());
}

```

Register event handler method "CalendarUserControl1\_DateSelected()" with "DateSelected" event of the "CalendarUserControl" using "+=" as shown below. Do this, in the Page\_load() event of "WebForm1". To unregister we can use "-=".

```

protected void Page_Load(object sender, EventArgs e)
{
    CalendarUserControl1.DateSelected +=
        new DateSelectedEventHandler(CalendarUserControl1_DateSelected);
}

```

Here is the complete code for WebForm1.aspx.cs, that consumes both the custom events.

```

public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        CalendarUserControl1.CalendarVisibilityChanged +=
            new
CalendarVisibilityChangedEventHandler(CalendarUserControl1_CalendarVisibilityChanged);

        CalendarUserControl1.DateSelected +=
            new DateSelectedEventHandler(CalendarUserControl1_DateSelected);
    }
}

```



```
protected void CalendarUserControl1_CalendarVisibilityChanged(object sender,
CalendarVisibilityChangedEventArgs e)
{
    Response.Write("Calendar Visible = " + e.IsCalendarVisible.ToString());
}

protected void CalendarUserControl1_DateSelected(object sender, DateSelectedEventArgs e)
{
    Response.Write("Selected Date = " + e.SelectedDate.ToShortDateString());
}

protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write(CalendarUserControl1.SelectedDate);
}
}
```

Hopefully, by now we should have a good understanding of events & delegates, and raising and consuming custom events. If there are any questions or feedback, please feel free to leave a comment.

## Loading user controls dynamically

We will discuss about loading user controls dynamically. Normally, we drag and drop user controls on the webform, at design time. However, there could be several reasons in real time, for loading user controls dynamically. For example, depending on logged in user preferences like age, gender, location etc, we may want to load different user controls. Along the same line, based on Admin and Non-Admin users, different user controls should be loaded.

In most of the internet articles, I read that, if we want, the dynamically created controls, to maintain viewstate across postback, the controls should be loaded in Page\_Init() event of the webform. However, I used the Page\_Load() event, but the controls are still able to retain their state across postback.

Let us add the CalendarUserControl to the webform. Drag and drop, Placeholder control, where we want the controls to be on the page. If we don't use the PlaceholderControl, and if we try to add controls using the following code, we may get a runtime HttpException.

Control 'CU1\_txtDate' of type 'TextBox' must be placed inside a form tag with runat=server  
`this.Page.Controls.Add(LoadControl("~/UserControls/CalendarUserControl.ascx"));`

WebForm1.aspx HTML

```
<form id="form1" runat="server">
<div>
  <asp:Placeholder ID="Placeholder1" runat="server">
  </asp:Placeholder>
  <br />
  <asp:Button ID="Button1" runat="server" Text="Button"
  onclick="Button1_Click" />
</div>
</form>
```

Webform1.aspx.cs Code

```
protected void Page_Load(object sender, EventArgs e)
{
    CalendarUserControl calendarUserControl =
        (CalendarUserControl)LoadControl("~/UserControls/CalendarUserControl.ascx");
    calendarUserControl.ID = "CU1";
    calendarUserControl.DateSelected +=
        new DateSelectedEventHandler(calendarUserControl_DateSelected);
    calendarUserControl.CalendarVisibilityChanged +=
        new
    CalendarVisibilityChangedEventHandler(calendarUserControl_CalendarVisibilityChanged);
    Placeholder1.Controls.Add(calendarUserControl);
}

protected void calendarUserControl_CalendarVisibilityChanged(object sender,
CalendarVisibilityChangedEventArgs e)
{
    Response.Write("Calende Visible = " + e.IsCalendarVisible.ToString());
}
```

```
protected void calendarUserControl1_DateSelected(object sender, DateSelectedEventArgs e)
{
    Response.Write(e.SelectedDate.ToShortDateString());
}
```

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write(((CalendarUserControl)Placeholder1.FindControl("CU1")).SelectedDate);
}
```

## Loading controls dynamically in ASP.NET

We will discuss about:

1. Adding controls dynamically
2. Tips and tricks to maintain the state of dynamically added controls across postback

To maintain the state of controls across postback

1. Add the controls in the page load event, and set their visibility to false.
2. Based on the DropDownList selection, show/hide the dynamically added controls

ASPX Page HTML

```
<div>
  <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True">
    <asp:ListItem Text="Please Select" Value="-1"></asp:ListItem>
    <asp:ListItem Text="Select City" Value="DDL"></asp:ListItem>
    <asp:ListItem Text="Enter Post Code" Value="TB"></asp:ListItem>
  </asp:DropDownList>
  <br />
  <asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
  <br />
  <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
</div>
```

Code Behind Code

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
    TextBox TB = new TextBox();
    TB.ID = "TB1";
    PlaceHolder1.Controls.Add(TB);
    TB.Visible = false;

    DropDownList DDL = new DropDownList();
    DDL.ID = "DDL1";
    DDL.Items.Add("New York");
    DDL.Items.Add("New Jersey");
    DDL.Items.Add("Washington DC");
    PlaceHolder1.Controls.Add(DDL);
    DDL.Visible = false;

    if (DropDownList1.SelectedValue == "TB")
    {
        TB.Visible = true;
    }
    else if (DropDownList1.SelectedValue == "DDL")
    {
        DDL.Visible = true;
    }
}
```

```
protected void Button1_Click(object sender, EventArgs e)
```

```
{
  if (DropDownList1.SelectedValue == "TB")
  {
    Response.Write(((TextBox)Placeholder1.FindControl("TB1")).Text);
  }
  else if (DropDownList1.SelectedValue == "DDL")
  {
    Response.Write(((DropDownList)Placeholder1.FindControl("DDL1")).SelectedItem.Text);
  }
}
```

## Navigating to a specific month and an year in an asp.net calendar control

There are several reasons or situations in real time, where we need to navigate to a specific month and an year in an asp.net calendar control. For example, if the date of birth of a person is in the year 1960, we may have to click on the calendar control several times to navigate to that year.

Let's see, how to skip months and years, so that the dates that are too far in the past or future can be easily selected. To achieve this

**Step 1:** Create an XML file with name Years.XML. Copy and paste the following XML.

```
<?xml version="1.0" encoding="utf-8" ?>
<Years>
  <Year>
    <Number>2000</Number>
    <Value>2000</Value>
  </Year>
  <Year>
    <Number>2001</Number>
    <Value>2001</Value>
  </Year>
  <Year>
    <Number>2002</Number>
    <Value>2002</Value>
  </Year>
  <Year>
    <Number>2003</Number>
    <Value>2003</Value>
  </Year>
  <Year>
    <Number>2004</Number>
    <Value>2004</Value>
  </Year>
  <Year>
    <Number>2005</Number>
    <Value>2005</Value>
  </Year>
  <Year>
    <Number>2006</Number>
    <Value>2006</Value>
  </Year>
  <Year>
    <Number>2007</Number>
    <Value>2007</Value>
  </Year>
  <Year>
    <Number>2008</Number>
    <Value>2008</Value>
  </Year>
  <Year>
    <Number>2009</Number>
    <Value>2009</Value>
  </Year>
```

```

</Year>
<Year>
  <Number>2010</Number>
  <Value>2010</Value>
</Year>
<Year>
  <Number>2011</Number>
  <Value>2011</Value>
</Year>
<Year>
  <Number>2012</Number>
  <Value>2012</Value>
</Year>
<Year>
  <Number>2013</Number>
  <Value>2013</Value>
</Year>
</Years>

```

**Step 2:** Create an XML file with name Months.XML. Copy and paste the following XML.

```

<?xml version="1.0" encoding="utf-8" ?>
<Months>
  <Month>
    <Number>1</Number>
    <Name>January</Name>
  </Month>
  <Month>
    <Number>2</Number>
    <Name>February</Name>
  </Month>
  <Month>
    <Number>3</Number>
    <Name>March</Name>
  </Month>
  <Month>
    <Number>4</Number>
    <Name>April</Name>
  </Month>
  <Month>
    <Number>5</Number>
    <Name>May</Name>
  </Month>
  <Month>
    <Number>6</Number>
    <Name>June</Name>
  </Month>
  <Month>
    <Number>7</Number>
    <Name>July</Name>
  </Month>
  <Month>

```

```

    <Number>8</Number>
    <Name>August</Name>
</Month>
<Month>
    <Number>9</Number>
    <Name>September</Name>
</Month>
<Month>
    <Number>10</Number>
    <Name>October</Name>
</Month>
<Month>
    <Number>11</Number>
    <Name>November</Name>
</Month>
<Month>
    <Number>12</Number>
    <Name>December</Name>
</Month>
</Months>

```

**Step 3:** Copy and paste the following HTML in any webform

```

<div style="font-family:Arial">
    Year :
    <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
        onselectedindexchanged="DropDownList1_SelectedIndexChanged">
    </asp:DropDownList> &nbsp;
    Month:
    <asp:DropDownList ID="DropDownList2" runat="server" AutoPostBack="True"
        onselectedindexchanged="DropDownList2_SelectedIndexChanged">
    </asp:DropDownList>
    <br />
    <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
        Text="Get Selected Date" />
</div>

```

**Step 4:** Copy and paste the following code in the code-behind file

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        LoadYears();
        LoadMonths();
    }
}

private void LoadMonths()
{
    DataSet dsMonths = new DataSet();

```



```

dsMonths.ReadXml(Server.MapPath("~/Data/Months.xml"));

DropDownList2.DataTextField = "Name";
DropDownList2.DataValueField = "Number";

DropDownList2.DataSource = dsMonths;
DropDownList2.DataBind();
}

private void LoadYears()
{
    DataSet dsYears = new DataSet();
    dsYears.ReadXml(Server.MapPath("~/Data/Years.xml"));

    DropDownList1.DataTextField = "Number";
    DropDownList1.DataValueField = "Number";

    DropDownList1.DataSource = dsYears;
    DropDownList1.DataBind();
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    int year = Convert.ToInt16(DropDownList1.SelectedValue);
    int month = Convert.ToInt16(DropDownList2.SelectedValue);
    Calendar1.VisibleDate = new DateTime(year, month, 1);
    Calendar1.SelectedDate = new DateTime(year, month, 1);
}

protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
{
    int year = Convert.ToInt16(DropDownList1.SelectedValue);
    int month = Convert.ToInt16(DropDownList2.SelectedValue);
    Calendar1.VisibleDate = new DateTime(year, month, 1);
    Calendar1.SelectedDate = new DateTime(year, month, 1);
}

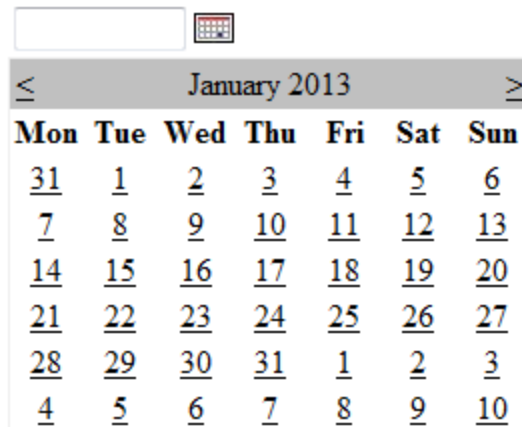
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write(Calendar1.SelectedDate.ToShortDateString());
}

```

At this point, run the project and you should be able to skip to any month and year in an asp.net calendar control. This code can be encapsulated in an user control, and reused anywhere in the entire project.

## Asp.net custom server controls

We have discussed about achieving the same, by building asp.net CalendarUserControl.



The custom calendar control should be capable of doing the following:

1. When the user clicks on the calendar image button, the calendar should be visible.
2. Once the user selects a date, from the calendar control, the textbox should be populated with the selected date, and the calendar should become invisible.

1. To get started open visual studio
2. File > New Project
3. In the "New Project" dialog box, select "Web" from "Installed Templates".
4. Select "ASP.NET Server Control"
5. Type "CustomControls" in the "Name" text box
6. Choose C:\ as the location and click "OK"

At this point "CustomControls" project should be created. In the solution explorer rename "ServerControl1.cs" file to "CustomCalendar.cs". Click "Yes" on the message you get.

Copy and paste the following code in CustomCalendar.cs file. The code is well commented, to describe the purpose of various attributes and methods, that we used.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CustomControls
{
    // Specifies the default tag to generate for our CustomCalendar control
    // when it is dragged from visual studio toolbox on to the webform
    [ToolboxData("<{0} :CustomCalendar runat=server></{0} :CustomCalendar>")]
    // All composite custom controls inherit from the base CompositeControl class
    // that contains all the common functionality needed by all composite controls.
```

```

public class CustomCalendar : CompositeControl
{
    // Child controls required by the CustomCalendar control
    TextBox textBox;
    ImageButton imageButton;
    Calendar calendar;

    // All child controls are required to be created by overriding
    // CreateChildControls method inherited from the base Control class
    // CompositeControl inherits from WebControl and WebControl class
    // inherits from Control class
    protected override void CreateChildControls()
    {
        Controls.Clear();

        textBox = new TextBox();
        textBox.ID = "dateTextBox";
        textBox.Width = Unit.Pixel(80);

        imageButton = new ImageButton();
        imageButton.ID = "calendarImageButton";

        calendar = new Calendar();
        calendar.ID = "calendarControl";

        // Add Child controls to CustomCalendar control
        this.Controls.Add(textBox);
        this.Controls.Add(imageButton);
        this.Controls.Add(calendar);
    }

    // Render the child controls using HtmlTextWriter object
    protected override void Render(HtmlTextWriter writer)
    {
        textBox.RenderControl(writer);
        imageButton.RenderControl(writer);
        calendar.RenderControl(writer);
    }
}

```

At this point we are done, building the composite custom control. However, there will be several problems with this control.

## **Adding composite custom controls to visual studio tool box**

We will discuss about:

1. Adding composite custom controls to visual studio toolbox
2. Using composite custom controls in an asp.net web application

### **Adding composite custom controls to visual studio toolbox:**

1. Build the "CustomControls" project and Building the project should produce "CustomControls.dll" assembly which contains our "CustomCalendar" control. Since, we have created the project in C:\ drive, the assembly should be present at the following path.  
C:\CustomControls\CustomControls\bin\Debug
2. Open visual studio. Create a new asp.net web application project.
3. In visual studio toolbox, right click anywhere, and select "Add Tab"
4. Name the tab "Custom Controls".
5. Right click under "Custom Controls" and select "Choose Items"
6. In the "Choose Toolbox Items" dialog box, click "Browse" button.
7. Navigate to "C:\CustomControls\CustomControls\bin\Debug" and select "CustomControls.dll" and click "Open"
8. Now click "OK" on the "Choose Toolbox Items" dialog box.
9. Notice that "CustomCalendar" control is added to the toolbox under "Custom Controls" tab.

### **Using composite custom controls in an asp.net web application:**

Flip "WebForm1.aspx" to "Design" mode. Drag and Drop "CustomCalendar" control onto "WebForm1.aspx". This should automatically add the control to the webform and the control declaration looks as as shown below, in the source mode of webform1.aspx.

```
<cc1:CustomCalendar ID="CustomCalendar1" runat="server" />
```

Notice that a "Register" directive is also automatically, added under "Page" directive.

```
<%@ Register assembly="CustomControls" namespace="CustomControls" tagprefix="cc1" %>
```

At this point, run the project and notice that the "CustomCalendar" control has several issues. For example:

1. There is no image associated with the Image button
2. The calendar is always visible. We want the Calendar to be invisible at first. The calendar should be shown, when the user clicks the image button.
3. Also, when select a date with in calendar, nothing happens. We want the text box, to be populated with the selected date.

## Adding properties to composite custom controls

At the moment, the calendar composite control does not have an image associated with the image button. Let us create "ImageButtonImageUrl" property to associate an image.

Copy and paste the following code in CustomCalendar.cs file.

```
[Category("Appearance")]
[Description("Sets the image icon for the calendar control")]
public string ImageButtonImageUrl
{
    get
    {
        // This method checks if the child controls are created, if not,
        // it triggers a call to CreateChildControls() method.
        EnsureChildControls();
        return imageUrl != null ? imageUrl : string.Empty;
    }
    set
    {
        EnsureChildControls();
        imageUrl = value;
    }
}
```

Rebuild CustomControls project.

Flip to the asp.net web application project, where the CustomCalendar is being tested. Please remove the CustomCalendar control from visual studio tool box, and add it again. Drag and drop CustomCalendar control on to the webform. Right click and select properties. Notice that "ImageButtonImageUrl" property is now displayed in the properties window. If you change the properties display mode to "Categorized", then notice that "ImageButtonImageUrl" property is displayed under "Appearance" category. Also, notice that, when the property is selected, the property description is displayed at the bottom of the properties window.

Create a folder with Name=Images in the asp.net web application project. Download Calendar.jpg image from this website. To download the image, simply right click on the image, and select "Save Image as" and save it to the specified location on your computer. Copy the image into the "Images" folder in your asp.net web application project.



Now, in the properties window, set ImageButtonImageUrl="Images/Calendar.jpg". Notice that, at design time, the image is not shown on the image button. Run the project, and notice that, at run time, the image button shows the image as expected.

To correct the design time problem, override RecreateChildControls() method. This method is called by visual studio designer, to recreate child controls at design time. Copy and paste the following code CustomCalendar.cs file.

```
protected override void RecreateChildControls()
{
    EnsureChildControls();
}
```

At this point, re-test the custom calendar. When you set `ImageButtonImageUrl="Images/Calendar.jpg"`, notice that the property change is immediately picked up by the control at design time.

At the moment, another cosmetic issue with this control is that, the "Image Button" and the "TextBox" are not properly alligned. To correct this problem, change the `Render()` method in `CustomCalendar.cs` file, as shown below. This method adds, cellpadding attribute, and puts the "textbox" and "calendar" in a table.

```
protected override void Render(HtmlTextWriter writer)
{
    AddAttributesToRender(writer);
    writer.AddAttribute(HtmlTextWriterAttribute.Cellpadding, "1");

    writer.RenderBeginTag(HtmlTextWriterTag.Table);

    writer.RenderBeginTag(HtmlTextWriterTag.Tr);

    writer.RenderBeginTag(HtmlTextWriterTag.Td);
    textBox.RenderControl(writer);
    writer.RenderEndTag();

    writer.RenderBeginTag(HtmlTextWriterTag.Td);
    imageButton.RenderControl(writer);
    writer.RenderEndTag();

    writer.RenderEndTag();
    writer.RenderEndTag();

    calendar.RenderControl(writer);
}
```

Here is the complete code for `CustomCalendar`:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CustomControls
{
    [ToolboxData("<{0}.CustomCalendar runat=server></{0}.CustomCalendar>")]
```

```

public class CustomCalendar : CompositeControl
{
    TextBox textBox;
    ImageButton imageButton;
    Calendar calendar;

    [Category("Appearance")]
    [Description("Sets the image icon for the calendar control")]
    public string ImageButtonImageUrl
    {
        get
        {
            EnsureChildControls();
            return imageButton.ImageUrl != null ? imageButton.ImageUrl : string.Empty;
        }
        set
        {
            EnsureChildControls();
            imageButton.ImageUrl = value;
        }
    }

    protected override void RecreateChildControls()
    {
        EnsureChildControls();
    }

    protected override void CreateChildControls()
    {
        Controls.Clear();

        textBox = new TextBox();
        textBox.ID = "dateTextBox";
        textBox.Width = Unit.Pixel(80);

        imageButton = new ImageButton();
        imageButton.ID = "calendarImageButton";

        calendar = new Calendar();
        calendar.ID = "calendarControl";

        this.Controls.Add(textBox);
        this.Controls.Add(imageButton);
        this.Controls.Add(calendar);
    }

    protected override void Render(HtmlTextWriter writer)
    {
        AddAttributesToRender(writer);
        writer.AddAttribute(HtmlTextWriterAttribute.Cellpadding, "1");

        writer.RenderBeginTag(HtmlTextWriterTag.Table);
    }
}

```

```
writer.RenderBeginTag(HtmlTextWriterTag.Tr);

writer.RenderBeginTag(HtmlTextWriterTag.Td);
textBox.RenderControl(writer);
writer.RenderEndTag();

writer.RenderBeginTag(HtmlTextWriterTag.Td);
imageButton.RenderControl(writer);
writer.RenderEndTag();

writer.RenderEndTag();
writer.RenderEndTag();

calendar.RenderControl(writer);
}
}
}
```



## Solving the problems of asp.net composite custom calendar control

We will discuss about:

1. Hiding the calendar control, when the CustomCalendar control is first loaded
2. Displaying and hiding the calendar, when the image button is clicked
3. Populating the textbox control, with a date that is selected from the calendar
4. Retrieving the selected date from the CustomCalendar control on a webform.

### Hiding the calendar control, when the CustomCalendar control is first loaded:

To hide the calendar set `calendar.Visible = false` in `CreateChildControls()` method in `CustomCalendar.cs` file

### Displaying and hiding the calendar, when the image button is clicked:

To toggle the visibility of the calendar object, first register an event handler method, with the click event of the image button in `CreateChildControls()` method as shown below.

```
imageButton.Click += new ImageClickEventHandler(imageButton_Click);
```

Provide the implementation for `imageButton_Click()` event handler method as shown below:

```
void imageButton_Click(object sender, ImageClickEventArgs e)
{
    // If the calendar is visible, make it invisible
    if (calendar.Visible)
    {
        calendar.Visible = false;
    }
    // If the calendar is invisible, make it visible
    else
    {
        calendar.Visible = true;
        // if the user has not already selected a date, then set
        // set the VisibleDate of the calendar to today's date
        if (string.IsNullOrEmpty(textBox.Text))
        {
            calendar.VisibleDate = DateTime.Today;
        }
        // if the user has already selected a date, then set
        // set the VisibleDate of the calendar to the selected date
        // by retrieving it from the textbox
        else
        {
            DateTime output = DateTime.Today;
            bool isDateTimeConversionSuccessful = DateTime.TryParse(textBox.Text, out output);
            calendar.VisibleDate = output;
        }
    }
}
```

Populating the textbox control, with a date that is selected from the calendar:

First register an event handler method, with the "SelectionChanged" event of the calendar in CreateChildControls() method as shown below.

```
calendar.SelectionChanged += new EventHandler(calendar_SelectionChanged);
```

Provide the implementation for calendar\_SelectionChanged() event handler method as shown below.

```
void calendar_SelectionChanged(object sender, EventArgs e)
{
    // Populate the text box with the selected date
    textBox.Text = calendar.SelectedDate.ToShortDateString();
    // Make the calendar invisible
    calendar.Visible = false;
}
```

### Retrieving the selected date from the CustomCalendar control on a webform:

To retrieve the selected date from the CustomCalendar control, add SelectedDate property to the CustomCalendar class as shown below.

```
[Category("Appearance")]
[Description("Gets or sets the selected date of custom calendar control")]
public DateTime SelectedDate
{
    get
    {
        EnsureChildControls();
        return string.IsNullOrEmpty(textBox.Text) ? DateTime.MinValue :
Convert.ToDateTime(textBox.Text);
    }

    set
    {
        if (value != null)
        {
            EnsureChildControls();
            textBox.Text = value.ToShortDateString();
        }
        else
        {
            EnsureChildControls();
            textBox.Text = "";
        }
    }
}
```

Here is the complete code for CustomCalendar control so far:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CustomControls
{
    [ToolboxData("<{0}:CustomCalendar runat=server></{0}:CustomCalendar>")]
    public class CustomCalendar : CompositeControl
    {
        TextBox textBox;
        ImageButton imageButton;
        Calendar calendar;

        protected override void CreateChildControls()
        {
            Controls.Clear();

            textBox = new TextBox();
            textBox.ID = "dateTextBox";
            textBox.Width = Unit.Pixel(80);

            imageButton = new ImageButton();
            imageButton.ID = "calendarImageButton";
            imageButton.Click += new ImageClickEventHandler(imageButton_Click);

            calendar = new Calendar();
            calendar.ID = "calendarControl";
            calendar.SelectionChanged += new EventHandler(calendar_SelectionChanged);
            calendar.Visible = false;

            this.Controls.Add(textBox);
            this.Controls.Add(imageButton);
            this.Controls.Add(calendar);
        }

        void calendar_SelectionChanged(object sender, EventArgs e)
        {
            textBox.Text = calendar.SelectedDate.ToShortDateString();
            calendar.Visible = false;
        }

        void imageButton_Click(object sender, ImageClickEventArgs e)
        {
            if (calendar.Visible)
            {

```

```

        calendar.Visible = false;
    }
    else
    {
        calendar.Visible = true;
        if (string.IsNullOrEmpty(textBox.Text))
        {
            calendar.VisibleDate = DateTime.Today;
        }
        else
        {
            DateTime output = DateTime.Today;
            bool isDateTimeConversionSuccessful = DateTime.TryParse(textBox.Text, out output);
            calendar.VisibleDate = output;
        }
    }
}

```

```

[Category("Appearance")]
[Description("Sets the image icon for the calendar control")]
public string ImageButtonImageUrl
{
    get
    {
        EnsureChildControls();
        return imageUrl != null ? imageUrl : string.Empty;
    }
    set
    {
        EnsureChildControls();
        imageUrl = value;
    }
}

```

```

[Category("Appearance")]
[Description("Gets or sets the selected date of custom calendar control")]
public DateTime SelectedDate
{
    get
    {
        EnsureChildControls();
        return string.IsNullOrEmpty(textBox.Text) ? DateTime.MinValue :
Convert.ToDateTime(textBox.Text);
    }

    set
    {
        if (value != null)
        {
            EnsureChildControls();
            textBox.Text = value.ToShortDateString();
        }
    }
}

```

```

        else
        {
            EnsureChildControls();
            textBox.Text = "";
        }
    }
}

protected override void RecreateChildControls()
{
    EnsureChildControls();
}

protected override void Render(HtmlTextWriter writer)
{
    AddAttributesToRender(writer);
    writer.AddAttribute(HtmlTextWriterAttribute.Cellpadding, "1");

    writer.RenderBeginTag(HtmlTextWriterTag.Table);

    writer.RenderBeginTag(HtmlTextWriterTag.Tr);

    writer.RenderBeginTag(HtmlTextWriterTag.Td);
    textBox.RenderControl(writer);
    writer.RenderEndTag();

    writer.RenderBeginTag(HtmlTextWriterTag.Td);
    imageButton.RenderControl(writer);
    writer.RenderEndTag();

    writer.RenderEndTag();
    writer.RenderEndTag();

    calendar.RenderControl(writer);
}
}
}

```

Please build CustomControls project, and add a reference to CustomCalendar in asp.net web application for testing. Drag and drop the CustomCalendar control and a button onto the webform.

Double click the button control to generate the click event handler.

```

protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write(CustomCalendar1.SelectedDate.ToShortDateString());
}

```

Run the web application and select a date from the calendar control. Notice that the textbox is populated with the selected date. Now click the button. The selected date should be printed on the web form.

## Adding custom events to asp.net composite custom control

There are 5 simple steps to add "DateSelected" custom event to composite custom calendar control:

**Step 1:** Create "DateSelectedEventArgs" that will contain event data

```
public class DateSelectedEventArgs : EventArgs
{
    private DateTime _selectedDate;

    public DateSelectedEventArgs(DateTime selectedDate)
    {
        this._selectedDate = selectedDate;
    }

    public DateTime SelectedDate
    {
        get
        {
            return this._selectedDate;
        }
    }
}
```

**Step 2:** Create "DateSelectedEventHandler" delegate

```
public delegate void DateSelectedEventHandler(object sender, DateSelectedEventArgs e);
```

**Step 3:** Create "DateSelected" event.

```
public event DateSelectedEventHandler DateSelected;
```

**Step 4:** Create a protected virtual method to raise the event.

```
protected virtual void OnDateSelection(DateSelectedEventArgs e)
{
    if (DateSelected != null)
    {
        DateSelected(this, e);
    }
}
```

**Step 5:** Finally raise the event, when the date selection in the calendar changes.

```
DateSelectedEventArgs dateSelectedEventData = new
DateSelectedEventArgs(calendar.SelectedDate);
OnDateSelection(dateSelectedEventData);
```

Here is the complete code of the composite custom calendar control:

```
using System;
using System.Collections.Generic;
```

```

using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace CustomControls
{

    [ToolboxData("<{0}:CustomCalendar runat=server></{0}:CustomCalendar>")]
    public class CustomCalendar : CompositeControl
    {
        TextBox textBox;
        ImageButton imageButton;
        Calendar calendar;

        protected override void CreateChildControls()
        {
            Controls.Clear();

            textBox = new TextBox();
            textBox.ID = "dateTextBox";
            textBox.Width = Unit.Pixel(80);

            imageButton = new ImageButton();
            imageButton.ID = "calendarImageButton";
            imageButton.Click += new ImageClickEventHandler(imageButton_Click);

            calendar = new Calendar();
            calendar.ID = "calendarControl";
            calendar.SelectionChanged += new EventHandler(calendar_SelectionChanged);
            calendar.Visible = false;

            this.Controls.Add(textBox);
            this.Controls.Add(imageButton);
            this.Controls.Add(calendar);
        }

        void calendar_SelectionChanged(object sender, EventArgs e)
        {
            textBox.Text = calendar.SelectedDate.ToShortDateString();

            DateSelectedEventArgs dateSelectedEventData = new
DateSelectedEventArgs(calendar.SelectedDate);
            OnDateSelection(dateSelectedEventData);

            calendar.Visible = false;
        }

        void imageButton_Click(object sender, ImageClickEventArgs e)
        {

```

```

if (calendar.Visible)
{
    calendar.Visible = false;
}
else
{
    calendar.Visible = true;
    if (string.IsNullOrEmpty(textBox.Text))
    {
        calendar.VisibleDate = DateTime.Today;
    }
    else
    {
        DateTime output = DateTime.Today;
        bool isDateTimeConversionSuccessful = DateTime.TryParse(textBox.Text, out output);
        calendar.VisibleDate = output;
    }
}
}

```

```

[Category("Appearance")]
[Description("Sets the image icon for the calendar control")]
public string ImageButtonImageUrl
{
    get
    {
        EnsureChildControls();
        return imageUrl != null ? imageUrl : string.Empty;
    }
    set
    {
        EnsureChildControls();
        imageUrl = value;
    }
}

```

```

[Category("Appearance")]
[Description("Gets or sets the selected date of custom calendar control")]
public DateTime SelectedDate
{
    get
    {
        EnsureChildControls();
        return string.IsNullOrEmpty(textBox.Text) ? DateTime.MinValue :
Convert.ToDateTime(textBox.Text);
    }

    set
    {
        if (value != null)
        {
            EnsureChildControls();

```



```

        textBox.Text = value.ToShortDateString();
    }
    else
    {
        EnsureChildControls();
        textBox.Text = "";
    }
}
}

```

```

protected override void RecreateChildControls()
{
    EnsureChildControls();
}

```

```

protected override void Render(HtmlTextWriter writer)
{
    AddAttributesToRender(writer);
    writer.AddAttribute(HtmlTextWriterAttribute.Cellpadding, "1");

    writer.RenderBeginTag(HtmlTextWriterTag.Table);

    writer.RenderBeginTag(HtmlTextWriterTag.Tr);

    writer.RenderBeginTag(HtmlTextWriterTag.Td);
    textBox.RenderControl(writer);
    writer.RenderEndTag();

    writer.RenderBeginTag(HtmlTextWriterTag.Td);
    imageButton.RenderControl(writer);
    writer.RenderEndTag();

    writer.RenderEndTag();
    writer.RenderEndTag();

    calendar.RenderControl(writer);
}

```

```

public event DateSelectedEventHandler DateSelected;

```

```

protected virtual void OnDateSelection(DateSelectedEventArgs e)
{
    if (DateSelected != null)
    {
        DateSelected(this, e);
    }
}
}

```

```

public class DateSelectedEventArgs : EventArgs
{
    private DateTime _selectedDate;
}

```

```

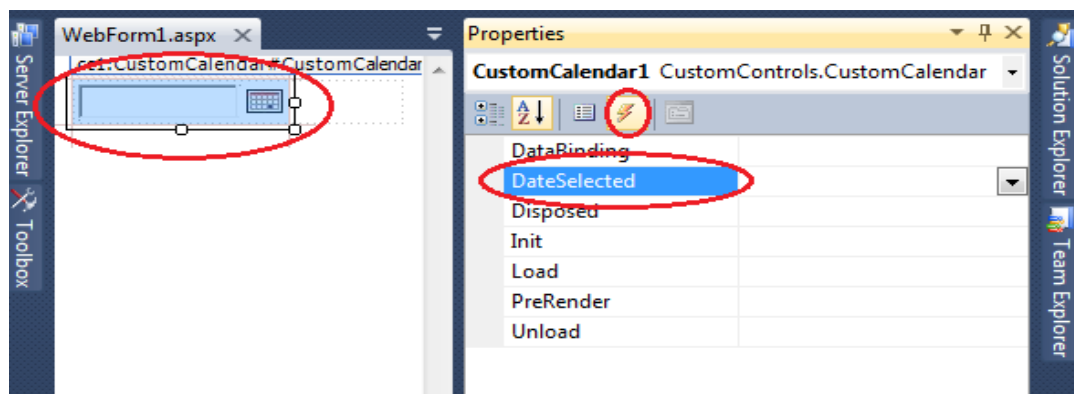
public DateSelectedEventArgs(DateTime selectedDate)
{
    this._selectedDate = selectedDate;
}

public DateTime SelectedDate
{
    get
    {
        return this._selectedDate;
    }
}

public delegate void DateSelectedEventHandler(object sender, DateSelectedEventArgs e);
}

```

Build the CustomControls project. In an asp.net web application add a reference to the CustomCalendar control. Drag and drop the CustomCalendar control on a webform. Right click on the control and select properties. In the properties window, click on the events icon. Notice that, the event "DateSelected" is displayed.



Double click on the "DateSelected" event. This should automatically generate an event handler method. Implement the event handler method as shown below.

```

protected void CustomCalendar1_DateSelected(object sender,
CustomControls.DateSelectedEventArgs e)
{
    Response.Write(e.SelectedDate.ToShortDateString());
}

```

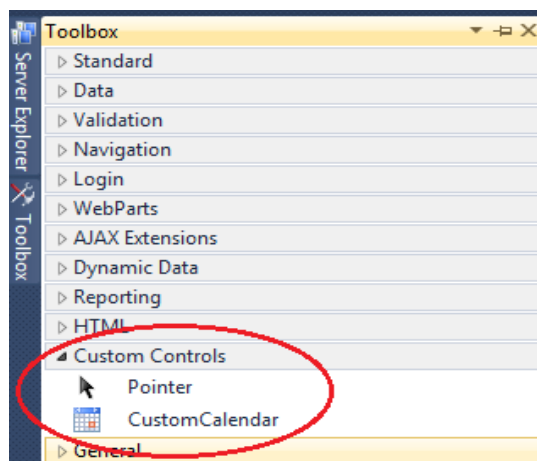
## Assigning an image to the composite custom control in visual studio tool box

To associate an image to the composite custom calendar control use ToolboxBitmap attribute. This attribute is present in System.Drawing namespace. There are 2 ways we can use this attribute:

Using ToolboxBitmap attribute to associate, the image of an existing control: In this example, we are assigning the image of the asp.net calendar control to the composite custom calendar control. Please notice the usage of ToolboxBitmap attribute with "typeof" keyword.

```
[System.Drawing.ToolboxBitmap(typeof(Calendar))]  
[ToolboxData("<{0}:CustomCalendar runat=server></{0}:CustomCalendar>")]  
public class CustomCalendar : CompositeControl  
{  
    TextBox textBox;  
    ImageButton imageButton;  
    Calendar calendar;  
  
    // Rest of the code.....  
}
```

Rebuild the CustomControls project. Re-add the CustomCalendar control to visual studio tool box and notice that the asp.net calendar control icon is also used for the CustomCalendar control.



Using ToolboxBitmap attribute to associate a custom image: Create an image measuring 16X16 pixels in dimensions. I have used Microsoft paint to come up with the custom image that you can see below.



Save this image to "C:\Images" folder or in a folder of your choice. I have given the name of "CustomCalendarIcon.bmp" to the image.

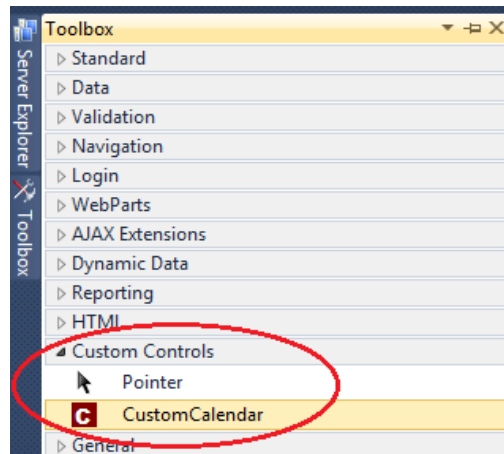
Finally use the ToolboxBitmap attribute and specify the path to the image.

```
[System.Drawing.ToolboxBitmap(@"C:\Images\CustomCalendarIcon.bmp")]  
[ToolboxData("<{0}:CustomCalendar runat=server></{0}:CustomCalendar>")]  
public class CustomCalendar : CompositeControl  
{
```

```
TextBox textBox;  
ImageButton imageButton;  
Calendar calendar;
```

```
// Rest of the code.....  
}
```

Rebuild the CustomControls project. Re-add the CustomCalendar control to visual studio tool box and notice that the custom image "CustomCalendarIcon.bmp" is used for the CustomCalendar control.



## **Difference between user controls and custom controls**

Custom controls are compiled into their own assembly(.dll) where as user controls are not. User controls are compiled into the web application project's assembly that contain them.

Custom controls can be added to toolbox where as user controls cannot be added.

User controls are easier to create as they are similar to creating web pages. Custom controls are relatively complex, as there is no designer, and every thing has to be done in code.

A separate copy of user control is required in each application you want to use; where as a single copy of custom control can be used in multiple projects.

Please Note: When you drag and drop, a custom control, from the tool box onto a web form designer, the following 2 things can happen:

1. If the custom controls assembly is not installed in GAC, then custom controls assembly is copied into the "BIN" folder of the application. So, in this case, if you need to use the custom control in multiple applications, then a copy of the custom control's assembly, will be made into the "BIN" folder of every application.
2. If the custom controls assembly is installed in GAC, then custom controls assembly is directly referenced from the GAC. So, in this case, a single copy of custom control can be used in multiple projects.

## Caching in asp.net

Caching improves the performance and scalability of an application. Caching is the technique of storing frequently used data/pages in memory. Let us practically understand caching, with an example.

### Create tblproducts table in sql server

```
Create Table tblProducts
(
    [Id] int identity primary key,
    [Name] nvarchar(50),
    [Description] nvarchar(250)
)
```

### Populate tblProducts with sample data

```
Insert into tblProducts values ('Laptops', 'Dell Laptops')
Insert into tblProducts values ('iPhone', 'iPhone 4S')
Insert into tblProducts values ('LCD TV', 'Samsung LCD TV')
Insert into tblProducts values ('Desktop', 'HP Desktop Computer')
```

Create "spGetProducts" stored procedure. In this procedure we are using **WAITFOR DELAY**, to block the execution of the stored procedure for 5 seconds. In real time, we may have large tables, where the query processing can take some time before the data is returned. Table "tblProducts" is a very small table, with only 4 rows. So the stored procedure "spGetProducts" would execute in a fraction of second. Just to simulate artificial query processing time of 5 seconds, we are using **WAITFOR DELAY**.

```
Create Procedure spGetProducts
As
Begin
    Waitfor Delay '00:00:05'

    Select Id, Name, Description
    from tblProducts
End
```

Now, let us invoke the stored procedure in an asp.net web application, and display the "Products" data in a gridview control. Drag and drop a "gridview" control onto the web form. Copy and paste the following code in the code-behind file Page\_Load() event.

```
string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
SqlConnection con = new SqlConnection(CS);
SqlDataAdapter da = new SqlDataAdapter("spGetProducts", con);
da.SelectCommand.CommandType = CommandType.StoredProcedure;
DataSet DS = new DataSet();
da.Fill(DS);
GridView1.DataSource = DS;
GridView1.DataBind();
```

Also make sure you have the following "using" declarations

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

At this point, if you run the application, the page takes about 5 seconds to load. This is because, when you request the webform, the web server has to process the web form events, execute the stored procedure, create objects, generate HTML and send that HTML to the client browser.

Now let us cache the webform. To cache a webform, use the @OutputCache page directive. The @OutputCache directive has 2 mandatory attributes:

**Duration** - Specifies the time in seconds for which the webform should be cached

**VaryByParam** - Cache multiple responses of a single webform. For now set the value to "None".

Webform with the following "OutputCache" directive is cached for 30 seconds.

```
<%@ OutputCache Duration="30" VaryByParam="None" %>
```

When any user requests this Web form for the first time, the web server will process the web form events, execute the stored procedure, create objects, generate HTML and send that HTML to the client browser, and retains a copy of the response in memory for the next 30 seconds. Any subsequent requests during that time receive the cached response.

After the cache duration has expired, the next request for the Web form, has to process the web form events, execute the stored procedure, create objects, generate HTML, which is then cached for another 30 seconds. So this web form is processed by the server, once every 30 second, at the most.

## Caching multiple responses for a single webform

First create a stored procedure to get products from "tblProducts" table by product name. This procedure returns "ALL Products", if "All" is supplied as the product name, else, only the product whose name matches with the parameter "@ProductName" is returned.

Create Procedure spGetProductByName

@ProductName nvarchar(50)

as

Begin

if(@ProductName = 'All')

Begin

Select Id, Name, Description

from tblProducts

End

Else

Begin

Select Id, Name, Description

from tblProducts

where Name = @ProductName

End

End

### WebForm1.aspx HTML:

```
<div style="font-family:Arial">
```

```
    Select Product:
```

```
    <asp:DropDownList ID="DropDownList1" AutoPostBack="true" runat="server"
        onselectedindexchanged="DropDownList1_SelectedIndexChanged">
```

```
        <asp:ListItem Text="All" Value="All"></asp:ListItem>
```

```
        <asp:ListItem Text="Laptops" Value="Laptops"></asp:ListItem>
```

```
        <asp:ListItem Text="iPhone" Value="iPhone"></asp:ListItem>
```

```
        <asp:ListItem Text="LCD TV" Value="LCD TV"></asp:ListItem>
```

```
        <asp:ListItem Text="Desktop" Value="Desktop"></asp:ListItem>
```

```
    </asp:DropDownList>
```

```
    <br />
```

```
    <br />
```

```
    <asp:GridView ID="GridView1" runat="server">
```

```
    </asp:GridView>
```

```
    <br />
```

```
    <br />
```

```
    Server Time:
```

```
    <asp:Label ID="Label1" runat="server"></asp:Label>
```

```
    <br />
```

```
    <br />
```

```
    Client Time:
```

```
    <script type="text/javascript">
```

```
        document.write(Date());
```

```
    </script>
```

```
</div>
```



### WebForm1.aspx.cs Code:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        GetProductByName("All");
    }
    Label1.Text = DateTime.Now.ToString();
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    GetProductByName(DropDownList1.SelectedValue);
}

private void GetProductByName(string ProductName)
{
    string CS = ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;
    SqlConnection con = new SqlConnection(CS);
    SqlDataAdapter da = new SqlDataAdapter("spGetProductByName", con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;

    SqlParameter paramProductName = new SqlParameter();
    paramProductName.ParameterName = "@ProductName";
    paramProductName.Value = ProductName;
    da.SelectCommand.Parameters.Add(paramProductName);

    DataSet DS = new DataSet();
    da.Fill(DS);
    GridView1.DataSource = DS;
    GridView1.DataBind();
}
```

Include "@OutputCache" attribute with the following settings. Notice that, since "VaryByParam" is set to "None", only one response of "WebForm1" is cached.

```
<%@ OutputCache Duration="60" VaryByParam="None"%>
```

Now change the "@OutputCache" attribute as shown below.

```
<%@ OutputCache Duration="60" VaryByParam="DropDownList1"%>
```

Since "VaryByParam" is set to "DropDownList1", up to 5 different responses will be cached for this Web form. One for each possible selection from DropDownList1 control.

## Controlling asp.net caching in code

"Cache" property of the "Response" object, can be used to control caching in code.

"Response.Cache" property returns the "HttpCachePolicy" object, which can be used in code just like the "OutputCache" directive is used in webform's HTML.

Copy and paste the following HTML on the webform.

```
<div style="font-family: Arial">
  Server Time :
  <asp:Label ID="Label1" runat="server" Font-Bold="true" ></asp:Label>
  <br /><br />
  Client Time:
  <b>
    <script type="text/javascript">
      document.write(Date());
    </script>
  </b>
</div>
```

Copy and paste the following code in the code-behind file

1. SetExpires() method, is used to set the duration for which we want to cache the webform. This is similar to the "Duration" attribute of the "OutputCache" directive.
2. Response.Cache.VaryByParams["None"] = true. This is similar to setting VaryByParam="None" for the "OutputCache" directive.
3. To control the location where items are cached, we can use "Location" attribute of the "OutputCache" directive, and SetCacheability() method in code.

```
protected void Page_Load(object sender, EventArgs e)
{
  Label1.Text = "This page is cached by the server @ " + DateTime.Now.ToString();
  Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));
  Response.Cache.VaryByParams["None"] = true;
  Response.Cache.SetCacheability(HttpCacheability.Server);
}
```

Equivalent OutputCache settings

```
<%@ OutputCache Duration="60" VaryByParam="None" Location="Server" %>
```

## Fragment caching in asp.net

Caching parts of webform is called as partial caching or fragment caching. In a web application development, there might be scenarios where most parts of the web page changes, but only a specific section of the page is static. Let's say that specific section also takes a long time to load. This is an ideal scenario, where fragment caching can be used.

We will be using tblProducts table to understand fragment caching. Please use the script below to create and populate table tblProducts

Create Table tblProducts

```
(  
    Id int primary key,  
    Name nvarchar(50),  
    ProductDescription nvarchar(100)  
)
```

Insert into tblProducts Values (1, 'Laptop', 'Dell Latitude Laptops')

Insert into tblProducts Values (2, 'iPhone', 'Apple iPhone 4S')

Insert into tblProducts Values (3, 'Desktop', 'Dell high performance desktops')

Insert into tblProducts Values (4, 'Server', 'HP Servers')

Script to create stored procedure that gets all the products. To introduce artificial query processing time, we are using "waitfor delay".

Create Procedure spGetProducts

as

Begin

Waitfor Delay '00:00:05'

Select \* from tblProducts

End

Steps to fragment cache a webform:

1. Encapsulate that specific sections of the page that does not constantly change into a user control.
2. Use the OutputCache directive on the user control to specify the cache settings.
3. Drag and drop the user control on the webform.

**Step 1:** Encapsulate specific sections of the page that does not constantly change into a user control  
Create an asp.net web application project. Add a user control to the project, with name "UCProductsControl.ascx". Copy and paste the following HTML in the ascx page of the user control.

```
<table style="border: 1px solid black">  
    <tr>  
        <td style="background-color: Gray; font-size:12pt">  
            Products User Control  
        </td>  
    </tr>  
    <tr>  
        <td>  
            <asp:GridView ID="GridView1" runat="server">
```

```

        </asp:GridView>
    </td>
</tr>
<tr>
    <td>
        <b>User Control Server Time:
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </b>
    </td>
</tr>
<tr>
    <td>
        <b>User Control Client Time:
        <script type="text/javascript">
            document.write(Date());
        </script>
        </b>
    </td>
</tr>
</table>

```

Copy and paste the following code in the code-behind file of the usercontrol

```

protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString();

    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    SqlConnection con = new SqlConnection(CS);
    SqlDataAdapter da = new SqlDataAdapter("spGetProducts", con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    DataSet DS = new DataSet();
    da.Fill(DS);

    GridView1.DataSource = DS;
    GridView1.DataBind();
}

```

Please make sure to include the following using declarations as well

```

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

```

**Step 2:** Use the OutputCache directive on the user control to specify the cache settings

```
<%@ OutputCache Duration="60" VaryByParam="None" %>
```

**Step 3:** Drag and drop the user control on the webform

Copy and paste the following HTML on the webform.

```

<div style="font-family: Arial">
  <table style="border: 1px solid black">
    <tr>
      <td>
        Page content that changes
      </td>
    </tr>
    <tr>
      <td>
        <b>Page Server Time:
          <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </b>
      </td>
    </tr>
    <tr>
      <td>
        <b>Page Client Time:
          <script type="text/javascript">
            document.write(Date());
          </script>
        </b>
      </td>
    </tr>
    <tr>
      <td>
        <br /><br />
        <uc1:UCProductsControl ID="UCProductsControl1" runat="server" />
      </td>
    </tr>
  </table>
</div>

```

Copy and paste the following code in the code-behind file of the webform.

```

protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString();
}

```

Notice that, when you run the web application, the user control is cached. The server time of the user control is not changed on refreshing the page but the other times on the user control and page changes. This proves that, the user control is cached, but not the rest of the webform. Fragment caching is that easy.

### **"Shared" attribute of the "OutputCache" directive:**

"Shared" attribute can be used with "OutputCache" directive, to cache a single response from a user control for use on multiple Web forms. By default, ASP.NET caches a separate response for each Web form that uses a cached user control. Let us understand this with an example.

Add another webform with name WebForm2.aspx to the asp.net web application. Copy and paste the

relevant HTML and code from WebForm1.aspx onto WebForm2.aspx.

Run the application and navigate to WebForm1.aspx. It takes around 5 seconds to load. Now navigate to WebForm2.aspx, and notice that WebForm2.aspx also takes 5 minutes. Since by default, ASP.NET caches a separate response for each Web form that uses a cached user control, both WebForm1.aspx and WebForm2.aspx, are taking 5 seconds.

Now, let's set the Shared="true" for "OutputCache" directive on "UCProductsControl" user control. This should cache a single response from the user control for use on WebForm1.aspx and WebForm2.aspx.

```
<%@ OutputCache Duration="60" VaryByParam="None" Shared="true" %>
```

Run the application and navigate to WebForm1.aspx. It takes around 5 seconds to load. Now navigate to WebForm2.aspx, and notice that WebForm2.aspx loads instantly. Also notice that, the server time of the user control is same on both the webforms. This proves that both WebForm1.aspx and WebForm2.aspx are using the single cached response of the user control.

## **Web form caching based on GET and POST requests**

**First Question: Why was the GET request for the webform not cached for "All" products selection in the dropdownlist?**

**Answer:** Actually, the GET request for "All" products selection in the dropdownlist is cached. To retrieve the cached response of the GET request, click any where in the URL address bar in the browser, and press ENTER key. This sends another GET request to the server, which should to return the cached response of the GET request.

**Second Question: Why am I not getting the cached response of the GET request when I change the selection in the dropdownlist to "All"?**

**Answer:** This is because, when the selection in the dropdownlist is changed to "All", a POSTBACK request is sent to the server. Since, for the POSTBACK request there is already a cached version of the webform, we get that version.

## Caching multiple versions of user control using VaryByControl

We will be using "tblProducts" table for this demo. If you need the script to create and populate this table,

Stored procedure to get products by "Name"

```
Create Procedure spGetProductByName
@ProductName nvarchar(50)
as
Begin
if(@ProductName = 'All')
Begin
Select Id, Name, Description
from tblProducts
End
Else
Begin
Select Id, Name, Description
from tblProducts
where Name = @ProductName
End
End
```

Add a user control to the project, with name = "UCProductsControl.ascx". Copy and paste the following HTML in the usercontrol.

```
<table style="border: 1px solid black">
<tr>
<td style="background-color: Gray; font-size: 12pt">
Products User Control
</td>
</tr>
<tr>
<td>
Select Product:
<asp:DropDownList ID="DropDownList1" AutoPostBack="true" runat="server"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
<asp:ListItem Text="All" Value="All"></asp:ListItem>
<asp:ListItem Text="Laptops" Value="Laptops"></asp:ListItem>
<asp:ListItem Text="iPhone" Value="iPhone"></asp:ListItem>
<asp:ListItem Text="LCD TV" Value="LCD TV"></asp:ListItem>
<asp:ListItem Text="Desktop" Value="Desktop"></asp:ListItem>
</asp:DropDownList>
</td>
</tr>
<tr>
<td>
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
</td>
</tr>
</table>
```



```

<tr>
  <td>
    <b>User Control Server Time:
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </b>
  </td>
</tr>
<tr>
  <td>
    <b>User Control Client Time:
      <script type="text/javascript">
        document.write(Date());
      </script>
    </b>
  </td>
</tr>
</table>

```

Copy and paste the following code in UCProductsControl.ascx.cs

```

public partial class UCProductsControl : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            GetProductByName("DropDownList1");
        }
        Label1.Text = DateTime.Now.ToString();
    }

    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {
        GetProductByName(DropDownList1.SelectedValue);
    }

    private void GetProductByName(string ProductName)
    {
        string CS =
ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;
        SqlConnection con = new SqlConnection(CS);
        SqlDataAdapter da = new SqlDataAdapter("spGetProductByName", con);
        da.SelectCommand.CommandType = CommandType.StoredProcedure;

        SqlParameter paramProductName = new SqlParameter();
        paramProductName.ParameterName = "@ProductName";
        paramProductName.Value = ProductName;
        da.SelectCommand.Parameters.Add(paramProductName);

        DataSet DS = new DataSet();
        da.Fill(DS);
        GridView1.DataSource = DS;
    }
}

```

```

        GridView1.DataBind();
    }
}

```

Caching multiple responses of a user control declaratively, using "VaryByControl" attribute of the "OutputCache" directive

To cache multiple response of the user control, include "OutputCache" in the aspx of the UCProductsControl.ascx. VaryByControl="DropDownList1", indicates that a separate response must be cached for each varying value in DropDownList1.

```

<%@ OutputCache Duration="60" VaryByControl="DropDownList1" %>

```

Caching multiple responses of a user control programatically, using "VaryByControls" property of the PartialCachingAttribute

We can also achieve the same thing, by specifying "PartialCachingAttribute" on the UCProductsControl class as shown below.

```

[PartialCaching(60, VaryByControls = "DropDownList1")]
public partial class UCProductsControl : System.Web.UI.UserControl
{
    //...Rest of the UCProductsControl code
}

```

Please run the application and test. Notice that, as the product selections change in the dropdownlist, for each different selection a response from the user control is cached for 60 seconds. The difference between "User Control Server Time" and "User Control Client Time" proves this. Since, we don't have "Caching" set on the WebForm1.aspx, "Page Server Time" and "Page Client Time" stays the same always.

## Caching multiple versions of user control using VaryByParam

**When should we use VaryByParam over VaryByControl and vice versa?**

**OR**

**What is the difference between VaryByParam and VaryByControl?**

If you want to cache multiple responses of a user control, based on a query string or a form "POST" parameter, then use VaryByParam. On the other hand, if you want to cache multiple responses of a user control, based on a control value then use "VaryByControl".

First let us modify "UCProductsControl.ascx" user control, to load products into the gridview control, using a query string parameter. This means we can remove the "DropDownList1" control from "UCProductsControl.ascx" file. The HTML is shown below.

```
<table style="border: 1px solid black">
  <tr>
    <td style="background-color: Gray; font-size: 12pt">
      Products User Control
    </td>
  </tr>
  <tr>
    <td>
      <asp:GridView ID="GridView1" runat="server">
      </asp:GridView>
    </td>
  </tr>
  <tr>
    <td>
      <b>User Control Server Time:
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
      </b>
    </td>
  </tr>
  <tr>
    <td>
      <b>User Control Client Time:
      <script type="text/javascript">
        document.write(Date());
      </script>
      </b>
    </td>
  </tr>
</table>
```

Make the following changes to the code-behind file of the user control:

1. In Page\_Load() event, remove "!IsPostBack" condition
2. Since, we want to load products using query string parameter, remove the reference to "DropDownList1" and use Request.QueryString["ProductName"]

```
protected void Page_Load(object sender, EventArgs e)
{
    GetProductByName(Request.QueryString["ProductName"]);
    Label1.Text = DateTime.Now.ToString();
}
```

DropDownList1\_SelectedIndexChanged() method can be completely removed. The private method GetProductByName() will not change in any way.

Caching multiple responses of a user control declaratively, using "VaryByParam" attribute of the "OutputCache" directive

To cache multiple responses of the user control, include "OutputCache" directive in the aspx of the UCProductsControl.ascx. Set VaryByParam="ProductName", which indicates that a separate response must be cached for each varying value of "ProductName" query string.

```
<%@ OutputCache Duration="60" VaryByParam="ProductName" %>
```

Caching multiple responses of a user control programmatically, using "VaryByParams" property of the PartialCachingAttribute

We can also achieve the same thing, by specifying "PartialCachingAttribute" on the UCProductsControl class as shown below.

```
[PartialCaching(60, VaryByParams = "ProductName")]
public partial class UCProductsControl : System.Web.UI.UserControl
{
    //...Rest of the UCProductsControl code
}
```

Please run the application and test. Notice that, as the "ProductName" query string value changes, for each different value, a response from the user control is cached for 60 seconds. The difference between "User Control Server Time" and "User Control Client Time" proves this. Since, we don't have "Caching" set on the WebForm1.aspx, "Page Server Time" and "Page Client Time" stays the same always.

## Caching application data in asp.net

we will discuss about caching application data. It is possible to store application data in the web server memory, using the CACHE object, so that the data can be retrieved faster. For example, let us say, we have a stored procedure that takes 5 seconds to execute and return data. We can cache the data returned by this stored procedure with in an asp.net web application using the CACHE object, so that, next time when we try to access the data, we can get it from the cache, rather than reprocessing the stored procedure again.

The following stored procedure takes 5 seconds to execute and return data. We are using WAITFOR DELAY, to introduce artificial query processing time of 5 seconds.

```
CREATE Procedure spGetProducts
as
Begin
    Waitfor Delay '00:00:05'
    Select * from tblProducts
End
```

Create an asp.net web application, copy and paste the following HTML in WebForm1.aspx.

```
<div style="font-family:Arial">
    <asp:Button ID="btnGetProducts" runat="server" Text="Get Products Data"
        onclick="btnGetProducts_Click" />
    <br /><br />
    <asp:GridView ID="gvProducts" runat="server">
    </asp:GridView>
    <br />
    <asp:Label ID="lblMessage" Font-Bold="true" runat="server"></asp:Label>
</div>
```

Copy and paste the following code in WebForm1.aspx.cs. The code is well documented and is self explanatory.

```
protected void btnGetProducts_Click(object sender, EventArgs e)
{
    DateTime dtStartDateTime = DateTime.Now;
    System.Text.StringBuilder sbMessage = new System.Text.StringBuilder();
    // Check if the data is already cached
    if (Cache["ProductsData"] != null)
    {
        // If data is cached, retrieve data from Cache using the key "ProductsData"
        DataSet ds = (DataSet)Cache["ProductsData"];
        // Set the dataset as the datasource
        gvProducts.DataSource = ds;
        gvProducts.DataBind();
        // Retrieve the total rows count
        sbMessage.Append(ds.Tables[0].Rows.Count.ToString() + " rows retrieved from cache.");
    }
    // If the data is not cached
    else
    {

```

```

// Get the data from the database
DataSet ds = GetProductsData();
// Cache the dataset using the key "ProductsData"
Cache["ProductsData"] = ds;
// Set the dataset as the datasource
gvProducts.DataSource = ds;
gvProducts.DataBind();
sbMessage.Append(ds.Tables[0].Rows.Count.ToString() + " rows retrieved from database.");
}
DateTime dtEndTime = DateTime.Now;
sbMessage.Append((dtEndTime - dtStartTime).Seconds.ToString() + " Seconds Load
Time");
lblMessage.Text = sbMessage.ToString();
}

private DataSet GetProductsData()
{
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    SqlConnection con = new SqlConnection(CS);
    SqlDataAdapter da = new SqlDataAdapter("spGetProducts", con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;

    DataSet dsProducts = new DataSet();
    da.Fill(dsProducts);

    return dsProducts;
}

```

## Different ways to cache application data in asp.net

We will discuss about:

1. Different ways to cache application data
2. Removing an item from cache
3. Similarities and difference between cache and application state

Apart from caching data, using assignment, there are 2 other ways to cache application data :

1. Using "Cache" object's Insert() method
2. Using "Cache" object's Add() method

### Caching application data, using "Cache" object's Insert() method:

Cache object's, Insert() method has got 5 overloaded versions. Depending on application requirement, we can choose the overloaded version that best suits our needs. The simplest overloaded version, takes 2 parameters:

1. The Cache Key and
2. The data that we want to cache

we cached products dataset, using assignment as shown below

```
Cache["ProductsData"] = ds;
```

The same thing can be achieved, using cache object's "Insert()" method, as shown below.

```
Cache.Insert("ProductsData", ds);
```

Caching application data, using "Cache" object's Add() method:

Adding application data, to the cache object, using "Add()" method is similar to "Insert()" method. As "Insert()" method has got several overloaded versions, some of the parameters are optional. Where as, "Add()" method does not have any overloads, and hence, all parameters must be specified. `Cache.Add("ProductsData", ds, null, System.Web.Caching.Cache.NoAbsoluteExpiration, System.Web.Caching.Cache.NoSlidingExpiration, System.Web.Caching.CacheItemPriority.Default, null);`

Be cautious when dealing with cache keys: Assigning a value to a cache key, that is already being used will silently overwrite the existing value, without any warnings. For example The value stored in "MyKey" is "Value1 "

```
Cache["MyKey"] = "Value 1 ";
```

The following line will silently overwrite "MyKey" value to "Value 2"

```
Cache["MyKey"] = "Value 2 ";
```

To Remove an item from cache explicitly, use Remove() method. The following line would remove the cache item with key "MyKey"

```
Cache.Remove("MyKey");
```

Please Note: An item may be removed automatically, from cache, when any of the following conditions are true:

1. The cached item has expired.
2. The cache is full.
3. There is a cache dependency, and the item that the cache object is dependent on has changed.

In a way, Cache object is similar to Application state. The objects that we store in application state variables are available anywhere within a Web application. Objects stored in cache are also available anywhere within a Web application. But the difference is that, items stored in cache can expire, where as items in application state will never expire.



## Caching in asp.net - AbsoluteExpiration, SlidingExpiration, and CacheItemPriority

When you cache an item using the Insert() or Add() method, you can also specify, how long you want the item to be cached. There are 2 ways to do this.

**AbsoluteExpiration:** A DateTime object that specifies when the data should be removed from the cache. When you specify "AbsoluteExpiration", the cache item will expire at that time, irrespective of whether the cached item is accessed or not.

For example, the following line will cache dataset, ds, for 10 seconds, irrespective of whether the dataset, is accessed or not within those 10 seconds, after it is cached. Since we are using absolute expiration, we specified Cache.NoSlidingExpiration for "SlidingExpiration" parameter of the Insert() method.

```
Cache.Insert("ProductsData", ds, null, DateTime.Now.AddSeconds(10),  
System.Web.Caching.Cache.NoSlidingExpiration);
```

**SlidingExpiration:** A TimeSpan object that identifies how long the data should remain in the cache after the data was last accessed.

For example, the following line will cache dataset, ds, for 10 seconds. If the dataset is accessed from the cache within the specified 10 seconds, then, from that point, the dataset will remain in cache for the next 10 seconds. Since we are using sliding expiration, we specified Cache.NoAbsoluteExpiration for "AbsoluteExpiration" parameter of the Insert() method.

```
Cache.Insert("ProductsData", ds, null, System.Web.Caching.Cache.NoAbsoluteExpiration,  
TimeSpan.FromSeconds(10));
```

**What happens if you specify both, AbsoluteExpiration and SlidingExpiration when caching application data?**

You get a run time exception stating 'absoluteExpiration must be DateTime.MaxValue or slidingExpiration must be TimeSpan.Zero'

**CacheItemPriority:** Sliding expiration and absolute expiration can be used to control, how long the item is cached, but please note, if the web server is running low on memory, and if it requires memory, it may remove cached items that may not have expired. However, the order in which the items are removed is determined by the cached item's priority. Cache item's priority can be specified using CacheItemPriority enum.

**CacheItemPriority enum values:**

```
CacheItemPriority.Low  
CacheItemPriority.BelowNormal  
CacheItemPriority.Normal  
CacheItemPriority.Default  
CacheItemPriority.AboveNormal  
CacheItemPriority.High  
CacheItemPriority.NotRemovable
```

For example, let us say we have 3 items with the following priorities and they are cached.  
Item 1 with CacheItemPriority.NotRemovable

Item 2 with CacheItemPriority.High  
Item 3 with CacheItemPriority.AboveNormal

Now, if the server is running low on memory, it may remove the items from cache, even if they have not expired. Since, Item 3 has the least priority, it will be removed first, followed by Item 2 and then finally Item 1.

## Cache dependency on files in ASP.NET

we will discuss about cache dependency on files in asp.net. Let us understand this with an example. Create an asp.net web application. Add a folder with name = "Data" to your web application. Right click on the "Data" folder and add an xml file with name = "Countries.xml".

Copy and paste the following xml data into Countries.xml file.

```
<?xml version="1.0" encoding="utf-8" ?>
<Countries>
  <Country>
    <Id>101</Id>
    <Name>India</Name>
    <Continent>Asia</Continent>
  </Country>
  <Country>
    <Id>102</Id>
    <Name>China</Name>
    <Continent>Asia</Continent>
  </Country>
  <Country>
    <Id>103</Id>
    <Name>France</Name>
    <Continent>Europe</Continent>
  </Country>
  <Country>
    <Id>104</Id>
    <Name>United Kingdom</Name>
    <Continent>Europe</Continent>
  </Country>
  <Country>
    <Id>105</Id>
    <Name>United State of America</Name>
    <Continent>North America</Continent>
  </Country>
</Countries>
```

### WebForm1.aspx HTML:

```
<div style="font-family: Arial">
  <asp:Button ID="btnGetCountries" runat="server" Text="Get Countries"
    OnClick="btnGetCountries_Click" />
  <br />
  <br />
  <asp:GridView ID="gvCountries" runat="server">
  </asp:GridView>
  <br />
  <asp:Label ID="lblMessage" Font-Bold="true" runat="server"></asp:Label>
</div>
```

The following code reads xml data from "Countries.xml" file into dataset which is then cached. Notice, that we are using "cache" object's "Insert()" method to cache the DataSet. Since we are

passing "null" as the argument for "CacheDependency" parameter of the "Insert()" method, when the data in "Countries.xml" file changes, the data in the cache is unaffected.

```
protected void btnGetCountries_Click(object sender, EventArgs e)
{
    // Check if the data is already cached
    if (Cache["CountriesData"] != null)
    {
        // If data is cached, retrieve data from Cache
        DataSet ds = (DataSet)Cache["CountriesData"];
        // Set the dataset as the datasource
        gvCountries.DataSource = ds;
        gvCountries.DataBind();
        // Retrieve the total rows count
        lblMessage.Text = ds.Tables[0].Rows.Count.ToString() + " rows retrieved from cache.";
    }
    // If the data is not cached
    else
    {
        // Get data from xml file
        DataSet ds = new DataSet();
        ds.ReadXml(Server.MapPath("~/Data/Countries.xml"));

        //Cache Countries and set dependency on file
        Cache.Insert("CountriesData", ds, null, DateTime.Now.AddSeconds(20),
        System.Web.Caching.Cache.NoSlidingExpiration);

        // Set the dataset as the datasource
        gvCountries.DataSource = ds;
        gvCountries.DataBind();
        lblMessage.Text = ds.Tables[0].Rows.Count.ToString() + " rows retrieved from the file.";
    }
}
```

When the data in Countries.xml file changes, we want the data in the cache to be removed automatically. For this to happen we need to establish a dependency on the xml file as shown below.

```
Cache.Insert("CountriesData", ds, new
CacheDependency(Server.MapPath("~/Data/Countries.xml")), DateTime.Now.AddSeconds(20),
System.Web.Caching.Cache.NoSlidingExpiration);
```

Now run the application. After the dataset is cached, change the xml file and click "Get Countries" button. Notice that the data is now retrieved from file directly, as the dataset is removed from cache.

## Reloading or refreshing cache automatically, when cached data is removed

We will discuss about:

1. When and how to use "CacheItemRemovedCallback" delegate
2. Reloading or refreshing cache automatically, when cached data has expired

Create an asp.net web application. Copy and paste the following HTML on WebForm1.aspx.

```
<div style="font-family: Arial">
    <asp:Button ID="btnLoadCountriesAndCache" runat="server" Text="Load Countries & Cache"
        OnClick="btnLoadCountriesAndCache_Click" />
    &nbsp;
    <asp:Button ID="btnGetCountriesFromCache" runat="server" Text="Get Countries from Cache"
        OnClick="btnGetCountriesFromCache_Click" />
    <br />
    <br />
    <asp:GridView ID="gvCountries" runat="server">
    </asp:GridView>
    <br />
    <asp:Button ID="btnRemoveCachedItem" runat="server" Text="Remove Cached Item"
        OnClick="btnRemoveCachedItem_Click" />
    &nbsp;
    <asp:Button ID="btnGetCacheStatus" runat="server" Text="Get Cache Status"
        OnClick="btnGetCacheStatus_Click" />
    <br />
    <br />
    <asp:Label ID="lblMessage" Font-Bold="true" runat="server"></asp:Label>
</div>
```

Copy and paste the following code in WebForm1.aspx.cs

```
protected void btnLoadCountriesAndCache_Click(object sender, EventArgs e)
{
    // Load countries data from XML file into dataset
    DataSet ds = new DataSet();
    ds.ReadXml(Server.MapPath("~/Data/Countries.xml"));

    // Create an instance of CacheItemRemovedCallback delegate. Notice that this delegate
    // points to CacheItemRemovedCallbackMethod. When cache item is removed
    // for any reason
    // the delegate gets invoked, which in turn will invoke the method it is pointing to.
    CacheItemRemovedCallback onCacheItemRemoved =
        new CacheItemRemovedCallback(CacheItemRemovedCallbackMethod);

    // Cache countries dataset. Please note that we are passing the delegate instance as an
    // argument for CacheItemRemovedCallback delegate parameter of the insert() method.
    Cache.Insert("CountriesData", ds, new
    CacheDependency(Server.MapPath("~/Data/Countries.xml")), DateTime.Now.AddSeconds(60),
        System.Web.Caching.Cache.NoSlidingExpiration, CacheItemPriority.Default,
    onCacheItemRemoved);
```

```

// Set the dataset as the datasource for the gridview
gvCountries.DataSource = ds;
gvCountries.DataBind();
lblMessage.Text = ds.Tables[0].Rows.Count.ToString() + " rows retrieved from XML file.";
}

```

```

protected void btnGetCountriesFromCache_Click(object sender, EventArgs e)
{
    // Check if countries dataset exists in cache
    if (Cache["CountriesData"] != null)
    {
        // If countries dataset is in cache, retrieve it
        DataSet ds = (DataSet)Cache["CountriesData"];
        // Set the dataset as the datasource
        gvCountries.DataSource = ds;
        gvCountries.DataBind();
        // Retrieve the total rows count
        lblMessage.Text = ds.Tables[0].Rows.Count.ToString() + " rows retrieved from cache.";
    }
    else
    {
        lblMessage.Text = "Cache item with key CountriesData is not present in cache";
    }
}

```

```

protected void btnRemoveCachedItem_Click(object sender, EventArgs e)
{
    // Remove cached item explicitly
    Cache.Remove("CountriesData");
}

```

```

// This method gets invoked automatically, whenever the cached item is removed from cache
public void CacheItemRemovedCallbackMethod(string key, object value,
CacheItemRemovedReason reason)
{
    // Retrieve the key and reason for removal
    string dataToStore = "Cache item with key = \"" + key + "\" is no longer present. Reason = " +
reason.ToString();
    // Cache the message
    Cache["CacheStatus"] = dataToStore;

    // ADO.NET code to store the message in database
    // string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    // SqlConnection con = new SqlConnection(cs);
    // SqlCommand cmd = new SqlCommand("insert into tblAuditLog values('" + dataToStore + "')",
con);
    // con.Open();
    // cmd.ExecuteNonQuery();
    // con.Close();

    // Reload data into cache
    // DataSet ds = new DataSet();
}

```

```

// ds.ReadXml(Server.MapPath("~/Data/Countries.xml"));

// CacheItemRemovedCallback onCacheItemRemoved = new
CacheItemRemovedCallback(CacheItemRemovedCallbackMethod);
// Cache.Insert("CountriesData", ds, new
CacheDependency(Server.MapPath("~/Data/Countries.xml")), DateTime.Now.AddSeconds(60),
// System.Web.Caching.Cache.NoSlidingExpiration, CacheItemPriority.Default,
onCacheItemRemoved);
}

protected void btnGetCacheStatus_Click(object sender, EventArgs e)
{
    if (Cache["CountriesData"] != null)
    {
        lblMessage.Text = "Cache item with key \"CountriesData\" is still present in cache";
    }
    else
    {
        if (Cache["CacheStatus"] != null)
        {
            lblMessage.Text = Cache["CacheStatus"].ToString();
        }
    }
}

```

### **Explanantion of code in btnLoadCountriesAndCache\_Click() event handler:**

When you click "Load Countries & Cache" button, the xml data is read from "Countries.xml" into a dataset, which is then cached. Notice that, before we cache data using the "Cache" objects "Insert()" method, we are creating an instance of CacheItemRemovedCallback delegate. To the constructor of this delegate, we are passing, the name of the function that we want to have executed automatically, when the cache item is removed from cache. An item may be removed, from cache, when any of the following conditions are true:

1. The cached item has expired.
2. The cache is full.
3. There is a cache dependency, and the item that the cache object is dependent on has changed.
4. The cache item may also be explicitly removed using Cache object's Remove() method.

The delegate object is then passed as an argument for CacheItemRemovedCallback delegate parameter of the insert() method. So, when the item, with cache key="CountriesData" is removed the delegate gets invoked, which in turn will automatically invoke, the function it is pointing to, in our case "CacheItemRemovedCallbackMethod()".

### **Explanantion of code in CacheItemRemovedCallbackMethod() function:**

This method gets invoked automatically, whenever the cached item is removed from cache. So, this is the opportunity for us to decide what we want to do when the cached item is removed from cache. For example, in this method the following 2 lines get the key of the item that is removed from the cache and the removed reason, and stores the message in another cache object, with key="CacheStatus".

```
string dataToStore = "Cache item with key = \" + key + "\" is no longer present. Reason = " +  
reason.ToString();  
Cache["CacheStatus"] = dataToStore;
```

Alternatively we can also store information about the removed cache object, in a database table. The ado.net code that does this is commented.

Finally we can also reload data into cache, from the XML file. The code to reload data into cache, is also commented.

### **Explanantion of code in btnGetCountriesFromCache\_Click() event handler:**

The code in "Get Countries from Cache" button click event handler is straight forward. We check if countries data is present in cache, and if it is, we retrieve the data from cache and display in gridview control.

Code in btnRemoveCachedItem\_Click() and btnGetCacheStatus\_Click() is self explanatory.



## Cache dependency on sql server database table

We will discuss about removing data from cache, when the data in the table from which it has come, has changed. Let us understand cache dependency on sql server database table, with an example. First create "tblProducts" table and populate with sample data using the script below:

```
CREATE TABLE [tblProducts]
(
  [Id] [int] PRIMARY KEY IDENTITY,
  [Name] [nvarchar](50) NULL,
  [Description] [nvarchar](250) NULL
)
```

```
Insert into tblProducts values ('Laptops', 'Dell Laptops')
Insert into tblProducts values ('iPhone', 'iPhone 4S')
Insert into tblProducts values ('LCD TV', 'Samsung LCD TV')
Insert into tblProducts values ('Desktop', 'HP Desktop Computer')
```

Create an asp.net web application project. Copy and paste the following HTML on WebForm1.aspx

```
<div style="font-family: Arial">
  <asp:Button ID="btnGetData" runat="server" Text="Get Data"
  OnClick="btnGetData_Click" />
  <br />
  <br />
  <asp:GridView ID="gvProducts" runat="server">
  </asp:GridView>
  <br />
  <asp:Label ID="lblStatus" runat="server" Font-Bold="true">
  </asp:Label>
</div>
```

Copy and paste the following code in WebForm1.aspx.cs

```
protected void btnGetData_Click(object sender, EventArgs e)
{
  // Check if the DataSet is present in cache
  if (Cache["ProductsData"] != null)
  {
    // If data available in cache, retrieve and bind it to gridview control
    gvProducts.DataSource = Cache["ProductsData"];
    gvProducts.DataBind();

    lblStatus.Text = "Data retrieved from cache @ " + DateTime.Now.ToString();
  }
  else
  {
    // Read connection string from web.config file
    string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

    // Enable change notifications on the database,
    // so that when the data changes the cached item will be removed
  }
}
```

```

System.Web.Caching.SqlCacheDependencyAdmin.EnableNotifications(CS);
// Enable change notifications on the database table,
// so that when the data changes the cached item will be removed
System.Web.Caching.SqlCacheDependencyAdmin.EnableTableForNotifications(CS,
"tblProducts");

SqlConnection con = new SqlConnection(CS);
SqlDataAdapter da = new SqlDataAdapter("select * from tblProducts", con);
DataSet ds = new DataSet();
da.Fill(ds);

// Build SqlCacheDependency object using the database and table names
SqlCacheDependency sqlDependency = new SqlCacheDependency("Sample", "tblProducts");

// Pass SqlCacheDependency object, when caching data
Cache.Insert("ProductsData", ds, sqlDependency);

gvProducts.DataSource = ds;
gvProducts.DataBind();
lblStatus.Text = "Data retrieved from database @ " + DateTime.Now.ToString();
}
}

```

In the above code, notice that, to enable SqlCacheDependency on the database and table, we are using "EnableNotifications()" and "EnableTableForNotifications()" methods, as shown below.

```

System.Web.Caching.SqlCacheDependencyAdmin.EnableNotifications(CS)
System.Web.Caching.SqlCacheDependencyAdmin.EnableTableForNotifications(CS, "tblProducts");

```

Alternatively, to enable SqlCacheDependency on the database and table, we can use a command line tool, aspnet\_regsql.exe. Open visual studio command prompt and execute the following 2 commands

**To enable SqlCacheDependency on the "Sample" database:**

```
aspnet_regsql -ed -E -d Sample
```

**To enable SqlCacheDependency on "tblProducts" table in "Sample" database:**

```
aspnet_regsql -et -E -d Sample -t tblProducts
```

If you need to understand the purpose of -et, -E, -d, -t, use the help, by typing the following command

```
aspnet_regsql /?
```

Finally, in web.config specify database connection string and SqlCacheDependency.

```

<connectionStrings>
  <add name="DBCS" connectionString="data source=.; database=Sample; integrated
security=SSPI"/>
</connectionStrings>
<system.web>

```

```
<キャッシング>  
  <sqlCacheDependency pollTime="2000" enabled="true">  
    <databases>  
      <add name="Sample" connectionString="DBCS"/>  
    </databases>  
  </sqlCacheDependency>  
</キャッシング>  
</system.web>
```

Notice that, we have set pollTime="2000". pollTime attribute specifies the frequency, at which, asp.net is going to check the database for changes. This time is in milli-seconds. Since, we have specified 2000 milli-seconds, asp.net is going to check the database for changes every 2 seconds. The default is 500 milli-seconds.

Run the application and when we click "Get Products" button for the first time, data is loaded from database. Click again, the data should be loaded from cache. Now, execute the following UPDATE query.

```
Update tblProducts set Name='Laptops' where Id = 1
```

Now, click the button again, and notice that the data is loaded from the database, as existing data is removed from cache.

## Reload data into cache automatically when data in the table changes

To load data automatically into cache, when cached data is removed, we need to define a callback method. This callback method will be invoked when the respective item is removed from cache. So, the code to reload and re-cache data can be written in this method. The callback method signature should match, with the signature of "CacheItemRemovedCallback" delegate. The call back method is defined below.

```
public void CacheItemRemovedCallbackMethod(string key, object value,
CacheItemRemovedReason reason)
{
    string CS = ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;

    SqlConnection con = new SqlConnection(CS);
    SqlDataAdapter da = new SqlDataAdapter("spGetProducts", con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    DataSet ds = new DataSet();
    da.Fill(ds);

    CacheItemRemovedCallback onCacheItemRemoved = new
CacheItemRemovedCallback(CacheItemRemovedCallbackMethod);

    SqlCacheDependency sqlDependency = new SqlCacheDependency("Sample", "tblProducts");
    Cache.Insert("ProductsData", ds, sqlDependency, DateTime.Now.AddHours(24),
Cache.NoSlidingExpiration,
    CacheItemPriority.Default, onCacheItemRemoved);
}
```

Now, create an instance of "CacheItemRemovedCallback" delegate, and to the constructor pass the name of the callback method, that should be executed automatically when, the cached item is removed.

```
CacheItemRemovedCallback onCacheItemRemoved = new
CacheItemRemovedCallback(CacheItemRemovedCallbackMethod);
```

To cache data, we used cache object's, Insert() method that takes 3 parameters, as shown below.

```
Cache.Insert("ProductsData", ds, sqlDependency);
```

Instead, let's use the overloaded version that takes 7 parameters and pass "onCacheItemRemoved" as an argument for "CacheItemRemovedCallback" parameter.

```
Cache.Insert("ProductsData", ds, sqlDependency, DateTime.Now.AddHours(24),
Cache.NoSlidingExpiration,
CacheItemPriority.Default, onCacheItemRemoved);
```

That's it. We are done. Now, please run the application. When "Get Data" button is clicked for the first time, the data is loaded from database. Once the data is loaded into cache, we always get it from cache when we click "Get Data". Now, execute an update statement on the database table. Notice that, when we click "Get Data" button now, we still get data from cache, but notice that, the updated data is loaded.

## What is AutoEventWireup in asp.net

Let us understand the use of AutoEventWireup property with an example. Create an asp.net web application project.

A webform in asp.net raises several events in it's life cycle. The following are few of those events:

1. Page Load
2. Page Load Complete
3. Page PreRender
4. Page PreRenderComplete

Copy and paste the following code in WebForm1.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("Page Load <br/>");
}

protected void Page_LoadComplete(object sender, EventArgs e)
{
    Response.Write("Page LoadComplete <br/>");
}

protected void Page_PreRender(object sender, EventArgs e)
{
    Response.Write("Page PreRender <br/>");
}

protected void Page_PreRenderComplete(object sender, EventArgs e)
{
    Response.Write("Page PreRenderComplete <br/>");
}
```

Now in WebForm1.aspx, set **AutoEventWireup=true**

Run the application, and notice that the above event handler methods are executed as expected. We did not explicitly associate event handler methods to the events, but still the event handler methods are hooked up to their respective events. This is because we have AutoEventWireup attribute set to true.

Now, set **AutoEventWireup=false**

Run the application, and notice that none of the above event handler methods are executed. This is because, when "AutoEventWireup" property is set to false, the event handler methods does not get automatically associated with their respective events. We have to explicitly associate them by overriding OnInit() method as shown below. Now, copy and paste the following code in webform1.aspx.cs

```
protected override void OnInit(EventArgs e)
{
    this.Load += new EventHandler(Page_Load);
}
```

```

this.LoadComplete += new EventHandler(Page_LoadComplete);
this.PreRender += new EventHandler(Page_PreRender);
this.PreRenderComplete += new EventHandler(Page_PreRenderComplete);
}

```

Run the application, and notice that the above event handler methods are executed as expected.

Now, set `AutoEventWireup=true`

Run the application, and notice that every event handler method is executed twice. This is because,

1. Setting `AutoEventWireup=true`, registered the event handler method's once, and
2. Overriding `OnInit()` method, has registered the same event handler method again

### Important points to remember:

1. When `AutoEventWireup` is set to true and if you want the event handlers to be wired up with their events automatically, the event handler names should follow the standard naming convention - `Page_EventName`.
2. `AutoEventWireup` can be set in the page directive or in `web.config` file.
3. To set `autoEventWireup` in `web.config`, use `pages` element as shown below.

```

<configuration>
  <system.web>
    <pages autoEventWireup="true" />
  </system.web>
</configuration>

```

4. If `autoEventWireup`, is set at both webform and `web.config` level, webform setting will take precedence over `web.config` setting.

So, `AutoEventWireup` is a boolean property which, when set to true, the page event handler methods are automatically wired with their respective events. If this property is set to false, then the event handler methods need to be explicitly associated with their respective events.

## Add image slideshow to your website using asp.net ajax and c#

We will discuss adding image slideshow to a website or web application.:

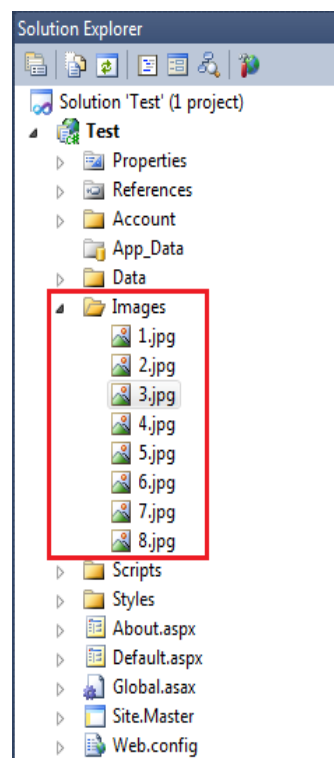
**Step 1:** Create an asp.net web application project.

**Step 2:** In the solution explorer, right click on the project name, and add "Images" folder.

**Step 3:** For this demo, we will use the sample pictures that are shipped with Microsoft operating system. Copy the images that are present at the following path, and paste them into the images folder.

C:\Users\Public\Pictures\Sample Pictures

Rename the images to use numbers. Since on my machine there are 8 images, I have named them 1.jpg, 2.jpg to 8.jpg. At this point, your solution explorer should look as show below.



**Step 4:** Drag and drop "ScriptManager" control onto the webform. This control can be found under "AJAX Extensions" in toolbox. ScriptManager control is required on any asp.net page, where you want to take advantage of asp.net ajax framework. We will discuss more about ScriptManager control in our upcoming asp.net ajax tutorial.

**Step 5:** Drag and drop "UpdatePanel" control. This control, allow us to perform partial page postbacks as opposed to a full page postback. The responsiveness of a page can be significantly increased using partial page postback, as only the data that is relevant to that UpdatePanel is sent to the server, and only the corresponding data is returned. Another benefit of partial page postbacks is that, they avoid screen flickers that are very common with full page postbacks.

All the content of the updatepanel, must be placed inside ContentTemplate element, so include

<ContentTemplate> tag directly inside updatepanel. Drag and drop, the timer and image controls onto the webform, so that they are placed inside the <ContentTemplate> tag.

1. The Timer control raises a tick event. This event is raised when the specified timer interval has elapsed and the timer is enabled.
2. Timer interval is specified in milli-seconds. For example, If you want the tick event to be raised every one second, then set "Interval" property of timer control to "1000" milliseconds.
3. We will use the tick event of the timer control to change the image dynamically every one second. So, flip the webform to design mode, if it's not already in design mode. Double click on the timer control. This should generate an event handler for tick event.
4. Set the Image control height and width to 100px.
5. Finally copy and paste the following code in the code-behind file.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        SetImageUrl();
    }
}

private void SetImageUrl()
{
    // Create an instance of Random class
    Random _rand = new Random();
    // Generate a random number between 1 and 8
    int i = _rand.Next(1, 8);
    // Set ImageUrl using the generated random number
    Image1.ImageUrl = "~/Images/" + i.ToString() + ".jpg";
}

// This event is raised every one second as we have set
// the interval to 1000 milliseconds
protected void Timer1_Tick(object sender, EventArgs e)
{
    SetImageUrl();
}
```

At the moment, the problem with this code is that, it displays a random image every one second. Let's say our requirement is such that, we want to display images in order from 1.jpg, 2.jpg to 8.jpg. Also, below the image, display the number of the image that is being displayed.



## Display images in sequence in an image slideshow

To achieve this,

**Step 1:** Include "Label1" control in the aspx page. This label control displays the image number that is being displayed.

```
<br /><asp:Label ID="Label1" Font-Bold="true" runat="server"></asp:Label>
```

**Step 2:** Change the code in SetImageUrl() function as shown below

```
private void SetImageUrl()
{
    if (ViewState["ImageDisplayed"] == null)
    {
        Image1.ImageUrl = "~/Images/1.jpg";
        ViewState["ImageDisplayed"] = 1;
        Label1.Text = "Displaying Image - 1";
    }
    else
    {
        int i = (int)ViewState["ImageDisplayed"];
        if (i == 8)
        {
            Image1.ImageUrl = "~/Images/1.jpg";
            ViewState["ImageDisplayed"] = 1;
            Label1.Text = "Displaying Image - 1";
        }
        else
        {
            i = i + 1;
            Image1.ImageUrl = "~/Images/" + i.ToString() + ".jpg";
            ViewState["ImageDisplayed"] = i;
            Label1.Text = "Displaying Image - " + i.ToString();
        }
    }
}
```

At the moment, there is no mechanism in place to start or stop the slideshow.

## Provide capability to start and stop image slideshow



1. Provide a button control as shown in the image
2. If the Sildeshow has not already started, the text on the button will be "Start Slideshow"
3. Once the button is clicked, the "slideshow" starts, and then, the text on the button will be changed to "Stop Slideshow". The images should be displayed in sequence from 1 to 8. The images should be changed dynamically in sequence until "Stop Slideshow" button is clicked.
4. Once "Stop SlideShow" button is clicked, the slideshow should stop. If the user clicks "start slideshow", then the slide show sould resume from where it was left.

To achieve this:

1. Drag and drop a button control on the webform.
2. Set Text= "Stop Slideshow".
3. Generate click event handler from Button1. Copy and paste the following code.

```
protected void Button1_Click(object sender, EventArgs e)
{
    // If timer is enabled, disable timer and change
    // the text on the button control accordingly
    if (Timer1.Enabled)
    {
        Timer1.Enabled = false;
        Button1.Text = "Start Slideshow";
    }
    // If timer is disabled, enable timer and change
    // the text on the button control accordingly
    else
    {
        Timer1.Enabled = true;
        Button1.Text = "Stop Slideshow";
    }
}
```

At the moment, there are 2 problems with this code. If we want to add a new image to the slide show,

1. We will have to modify the application code
2. The new image has to be named in a specific way. Since we already have 8 images, the next image has to be named 9.jpg.

## Add images to slideshow using xml file

There are 2 problems with the image slideshow:

1. We will have to modify the application code
2. The new image has to be named in a specific way. Since we already have 8 images, the next image has to be named 9.jpg.

There are two ways to fix the above 2 issues.

1. Using an XML file
2. Using a database table

**Step 1:** At the moment the images in "Images" folder have the following names

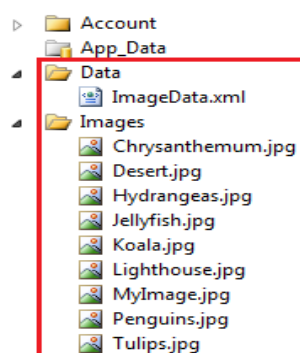
- 1.jpg
- 2.jpg
- 3.jpg
- etc...

Delete all these 8 images. Now copy the images with their original names from "C:\Users\Public\Pictures\Sample Pictures".

**Step 2:** Right click on the project name in solution explorer, and add "Data" folder. Add "ImageData.xml" file. Copy and paste the following XML

```
<?xml version="1.0" encoding="utf-8" ?>
<Images>
  <image name="Chrysanthemum.jpg" order="1"></image>
  <image name="Desert.jpg" order="2"></image>
  <image name="Hydrangeas.jpg" order="3"></image>
  <image name="Jellyfish.jpg" order="4"></image>
  <image name="Koala.jpg" order="5"></image>
  <image name="Lighthouse.jpg" order="6"></image>
  <image name="Penguins.jpg" order="7"></image>
  <image name="Tulips.jpg" order="8"></image>
</Images>
```

At this point your solution explorer, should be as shown below.



### Default.aspx code:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="ImageSlideShow._Default" %>
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Timer ID="Timer1" runat="server" Interval="1000" OnTick="Timer1_Tick">
            </asp:Timer>
            <asp:Image ID="Image1" Height="200px" Width="200px" runat="server" />
            <br />
            <br />
            Name: <asp:Label ID="lblImageName" runat="server"></asp:Label>
            <br />
            Order: <asp:Label ID="lblImageOrder" runat="server"></asp:Label>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Stop Slideshow"
                onclick="Button1_Click" />
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

### Default.aspx.cs code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;

namespace ImageSlideShow
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                LoadImageData();
            }
        }

        private void LoadImageData()
        {

```

```

DataSet ds = new DataSet();
ds.ReadXml(Server.MapPath("~/Data/ImageData.xml"));
ViewState["ImageData"] = ds;

ViewState["ImageDisplayed"] = 1;
DataRow imageDataRow = ds.Tables["image"].Select().FirstOrDefault(x =>
x["order"].ToString() == "1");
Image1.ImageUrl = "~/Images/" + imageDataRow["name"].ToString();
lblImageName.Text = imageDataRow["name"].ToString();
lblImageOrder.Text = imageDataRow["order"].ToString();
}

protected void Timer1_Tick(object sender, EventArgs e)
{
    int i = (int)ViewState["ImageDisplayed"];
    i = i + 1;
    ViewState["ImageDisplayed"] = i;

    DataRow imageDataRow =
((DataSet)ViewState["ImageData"]).Tables["image"].Select().FirstOrDefault(x =>
x["order"].ToString() == i.ToString());
    if (imageDataRow != null)
    {
        Image1.ImageUrl = "~/Images/" + imageDataRow["name"].ToString();
        lblImageName.Text = imageDataRow["name"].ToString();
        lblImageOrder.Text = imageDataRow["order"].ToString();
    }
    else
    {
        LoadImageData();
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    if (Timer1.Enabled)
    {
        Timer1.Enabled = false;
        Button1.Text = "Start Slideshow";
    }
    else
    {
        Timer1.Enabled = true;
        Button1.Text = "Stop Slideshow";
    }
}
}
}

```

To add a new image to the slideshow, there are 2 simple steps:

1. Add your new image to the images folder
2. Update "ImageData.xml" file

## Add images to slideshow using database table

To store image data, create table tblImages

Create table tblImages

```
(  
[ID] int identity primary key,  
[Name] nvarchar(50),  
[Order] int  
)
```

SQL script to insert image data

```
Insert into tblImages values('Chrysanthemum.jpg',1)  
Insert into tblImages values('Desert.jpg',2)  
Insert into tblImages values('Hydrangeas.jpg',3)  
Insert into tblImages values('Jellyfish.jpg',4)  
Insert into tblImages values('Koala.jpg',5)  
Insert into tblImages values('Lighthouse.jpg',6)  
Insert into tblImages values('Penguins.jpg',7)  
Insert into tblImages values('Tulips.jpg',8)  
Insert into tblImages values('MyImage.jpg',9)
```

Stored procedure to retrieve image data

Create procedure spGetImageData

as

Begin

Select [Name], [Order] from tblImages

End

After the table is created, create a connection string in web.config.

```
<connectionStrings>  
  <add name="DBCS"  
        connectionString="data source=.;Integrated Security=SSPI;database=Sample"  
        providerName="System.Data.SqlClient" />  
</connectionStrings>
```

We now have to write ADO.NET code to retrieve image data from the database table. The rest of the logic remains unchanged. Here's the complete code for your reference.

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
using System.Data;  
using System.Data.SqlClient;  
using System.Configuration;
```



```

namespace ImageSlideShow
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                SetImageUrl();
            }
        }

        protected void Timer1_Tick(object sender, EventArgs e)
        {
            int i = (int)ViewState["ImageDisplayed"];
            i = i + 1;
            ViewState["ImageDisplayed"] = i;

            DataRow imageDataRow =
            ((DataSet)ViewState["ImageData"]).Tables["image"].Select().FirstOrDefault(x =>
            x["order"].ToString() == i.ToString());
            if (imageDataRow != null)
            {
                Image1.ImageUrl = "~/Images/" + imageDataRow["name"].ToString();
                lblImageName.Text = imageDataRow["name"].ToString();
                lblImageOrder.Text = imageDataRow["order"].ToString();
            }
            else
            {
                SetImageUrl();
            }
        }

        private void SetImageUrl()
        {
            DataSet ds = new DataSet();
            string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
            SqlConnection con = new SqlConnection(CS);
            SqlDataAdapter da = new SqlDataAdapter("spGetImageData", con);
            da.Fill(ds, "image");
            ViewState["ImageData"] = ds;
            ViewState["ImageDisplayed"] = 1;

            DataRow imageDataRow = ds.Tables["image"].Select().FirstOrDefault(x =>
            x["order"].ToString() == "1");
            Image1.ImageUrl = "~/Images/" + imageDataRow["name"].ToString();
            lblImageName.Text = imageDataRow["name"].ToString();
            lblImageOrder.Text = imageDataRow["order"].ToString();
        }

        protected void Button1_Click(object sender, EventArgs e)
        {

```

```
if (Timer1.Enabled)
{
    Timer1.Enabled = false;
    Button1.Text = "Start Slideshow";
}
else
{
    Timer1.Enabled = true;
    Button1.Text = "Stop Slideshow";
}
}
}
```

To add a new image to the slideshow:

1. Copy the image to the images folder
2. Insert the new image name and it's order into tblImages table.

## How to upload and download files using asp.net and c#

We will discuss:

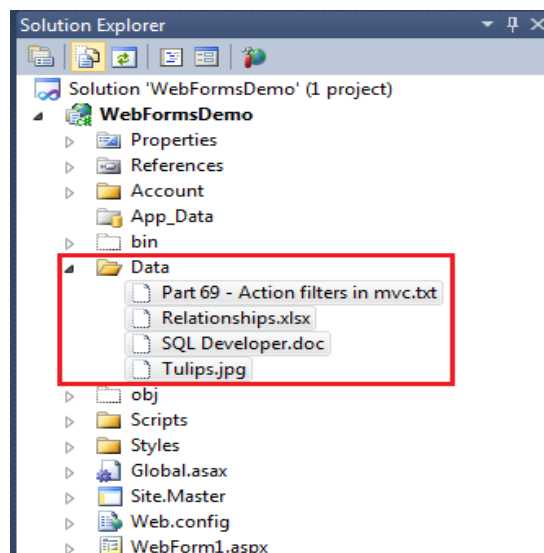
1. Uploading files
2. Displaying the list of files that are already uploaded
3. Downloading files

The user interface to upload and download files, should be as shown below.

No file chosen

File	Size in Bytes	File Type
<a href="#">Part 69 - Action filters in mvc.txt</a>	813	Text Document
<a href="#">Relationships.xlsx</a>	229705	Microsoft Excel Document
<a href="#">SQL Developer.doc</a>	28160	Microsoft Word Document
<a href="#">Tulips.jpg</a>	620888	Image

When the files are uploaded, they should be uploaded to a folder on the web server. In our case, we will be uploading to "Data" folder.



### WebForm1.aspx code:

```
<div style="font-family:Arial">
<asp:FileUpload ID="FileUpload1" runat="server" />
<asp:Button ID="Button1" runat="server" Text="Upload"
    OnClick="Button1_Click" />
<br />
<br />
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    OnRowCommand="GridView1_RowCommand" BackColor="White"
    BorderColor="#CC9966" BorderStyle="None"
    BorderWidth="1px" CellPadding="4">
    <Columns>
```

```

<asp:TemplateField HeaderText="File" ShowHeader="False">
  <ItemTemplate>
    <asp:LinkButton ID="LinkButton1" runat="server"
      CausesValidation="False"
      CommandArgument='<%# Eval("File") %>'
      CommandName="Download" Text='<%# Eval("File") %>'>
    </asp:LinkButton>
  </ItemTemplate>
</asp:TemplateField>
<asp:BoundField DataField="Size" HeaderText="Size in Bytes" />
<asp:BoundField DataField="Type" HeaderText="File Type" />
</Columns>
<FooterStyle BackColor="#FFFFCC" ForeColor="#330099" />
<HeaderStyle BackColor="#990000" Font-Bold="True"
  ForeColor="#FFFFCC" />
<PagerStyle BackColor="#FFFFCC" ForeColor="#330099"
  HorizontalAlign="Center" />
<RowStyle BackColor="White" ForeColor="#330099" />
<SelectedRowStyle BackColor="#FFCC66" Font-Bold="True"
  ForeColor="#663399" />
<SortedAscendingCellStyle BackColor="#FEECEB" />
<SortedAscendingHeaderStyle BackColor="#AF0101" />
<SortedDescendingCellStyle BackColor="#F6F0C0" />
<SortedDescendingHeaderStyle BackColor="#7E0000" />
</asp:GridView>
</div>

```

### WebForm1.aspx.cs code:

```

protected void Button1_Click(object sender, EventArgs e)
{
  if (FileUpload1.HasFile)
  {
    string fileName = FileUpload1.FileName;
    FileUpload1.PostedFile
      .SaveAs(Server.MapPath("~/Data/") + fileName);
  }

  DataTable dt = new DataTable();
  dt.Columns.Add("File");
  dt.Columns.Add("Size");
  dt.Columns.Add("Type");

  foreach (string strfile in Directory.GetFiles(Server.MapPath("~/Data")))
  {
    FileInfo fi = new FileInfo(strfile);
    dt.Rows.Add(fi.Name, fi.Length.ToString(),
      GetFileTypeByExtension(fi.Extension));
  }

  GridView1.DataSource = dt;
  GridView1.DataBind();
}

```

```

}

private string GetFileTypeByExtension(string fileExtension)
{
    switch (fileExtension.ToLower())
    {
        case ".docx":
        case ".doc":
            return "Microsoft Word Document";
        case ".xlsx":
        case ".xls":
            return "Microsoft Excel Document";
        case ".txt":
            return "Text Document";
        case ".jpg":
        case ".png":
            return "Image";
        default:
            return "Unknown";
    }
}

protected void GridView1_RowCommand(object sender,
    GridViewCommandEventArgs e)
{
    Response.Clear();
    Response.ContentType = "application/octet-stream";
    Response.AppendHeader("Content-Disposition", "filename="
        + e.CommandArgument);
    Response.TransmitFile(Server.MapPath("~/Data/")
        + e.CommandArgument);
    Response.End();
}

```

Please make sure to include the following using declarations in the code behind file.

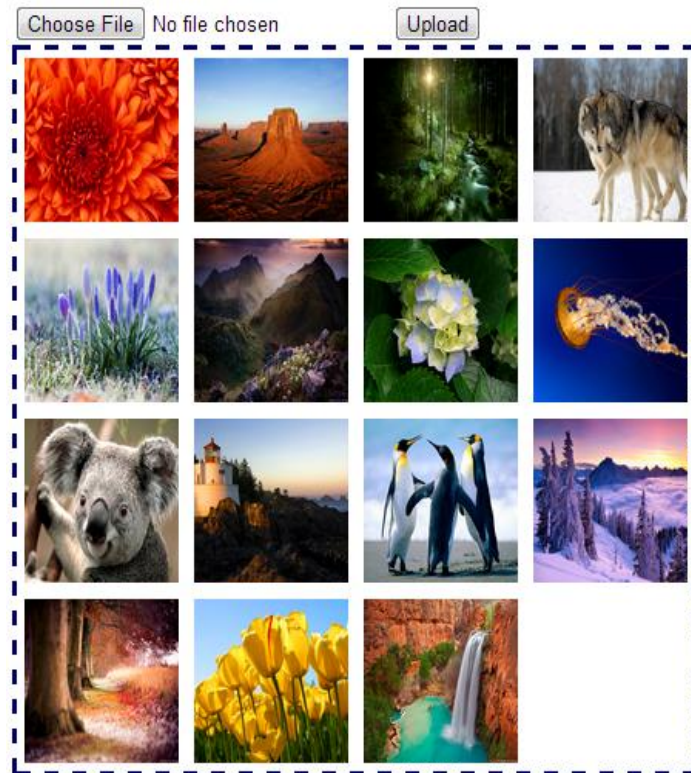
```

using System.IO;
using System.Data;

```

## Creating an image gallery using asp.net and c#

We will discuss creating an image gallery using asp.net and c#. The images in the gallery should be displayed as click-able thumbnails. Upon clicking the thumbnail, the user should be redirected to a page, where he can see the original image. Upon uploading an image, the image should be added to the gallery immediately. The output should be as shown below.



Add "Data" folder to the project. This folder stores the uploaded images.

### WebForm1.aspx

```
<asp:FileUpload ID="FileUpload1" runat="server" />
<asp:Button ID="Button1" runat="server" Text="Upload" OnClick="Button1_Click" />
<asp:Panel ID="Panel1" runat="server" Width="440px"
    BorderStyle="Dashed" BorderColor="#000066">
</asp:Panel>
```

### WebForm1.aspx.cs

```
public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        LoadImages();
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpload1.HasFile)
        {

```

```

        string fileName = FileUpload1.FileName;
        FileUpload1.PostedFile.SaveAs(Server.MapPath("~/Data/") + fileName);
    }
    Response.Redirect("~/WebForm1.aspx");
}

private void LoadImages()
{
    foreach (string strfile in Directory.GetFiles(Server.MapPath("~/Data/")))
    {
        ImageButton imageButton = new ImageButton();
        FileInfo fi = new FileInfo(strfile);
        imageButton.ImageUrl = "~/Data/" + fi.Name;
        imageButton.Height = Unit.Pixel(100);
        imageButton.Style.Add("padding", "5px");
        imageButton.Width = Unit.Pixel(100);
        imageButton.Click += new ImageClickEventHandler(imageButton_Click);
        Panel1.Controls.Add(imageButton);
    }
}

protected void imageButton_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("WebForm2.aspx?ImageURL=" +
        ((ImageButton)sender).ImageUrl);
}
}

```

### WebForm2.aspx

```

<asp:Button ID="Button2" Text="Back to Gallery" runat="server" onclick="Button1_Click" />
<br /><br />
<asp:Image ID="Image1" Width="800px" Height="550px" runat="server" />
<br /><br />
<asp:Button ID="Button1" Text="Back to Gallery" runat="server" onclick="Button1_Click" />

```

### WebForm2.aspx.cs

```

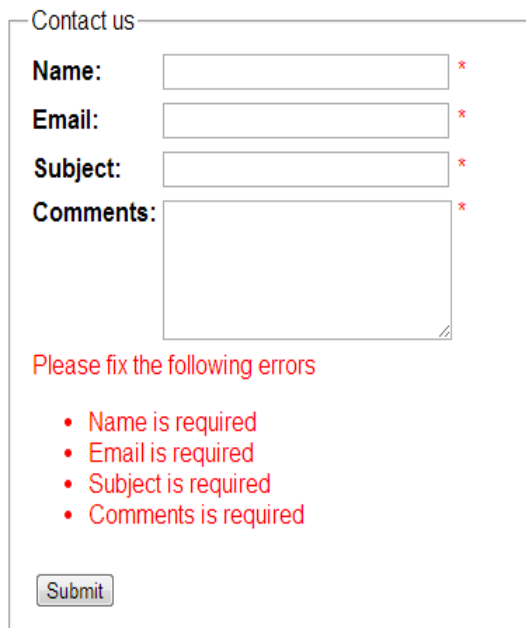
public partial class WebForm2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Image1.ImageUrl = Request.QueryString["ImageURL"];
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("WebForm1.aspx");
    }
}

```

## Contact us page using asp.net and c#

We will discuss designing contact us page using asp.net and c#. It's very common for web sites to contain a contact us page which allows the users of the site to contact either the administrator or the support group. A typical contact-us page is shown below.



### contactus.aspx:

```
<div style="font-family: Arial">
  <fieldset style="width: 350px">
    <legend title="Contact us">Contact us</legend>
    <table>
      <tr>
        <td>
          <b>Name:</b>
        </td>
        <td>
          <asp:TextBox
            ID="txtName"
            Width="200px"
            runat="server">
          </asp:TextBox>
        </td>
        <td>
          <asp:RequiredFieldValidator
            ForeColor="Red"
            ID="RequiredFieldValidator1"
            runat="server"
            ControlToValidate="txtName"
            ErrorMessage="Name is required"
            Text="*">
          </asp:RequiredFieldValidator>
        </td>
      </tr>
    </table>
```



```

<tr>
  <td>
    <b>Email:</b>
  </td>
  <td>
    <asp:TextBox
      ID="txtEmail"
      Width="200px"
      runat="server">
    </asp:TextBox>
  </td>
  <td>
    <asp:RequiredFieldValidator
      Display="Dynamic"
      ForeColor="Red"
      ID="RequiredFieldValidator2"
      runat="server"
      ControlToValidate="txtEmail"
      ErrorMessage="Email is required"
      Text="*">
    </asp:RequiredFieldValidator>
    <asp:RegularExpressionValidator
      Display="Dynamic"
      ForeColor="Red"
      ID="RegularExpressionValidator1"
      runat="server"
      ErrorMessage="Invalid Email"
      ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*"
      ControlToValidate="txtEmail"
      Text="*">
    </asp:RegularExpressionValidator>
  </td>
</tr>
<tr>
  <td>
    <b>Subject:</b>
  </td>
  <td>
    <asp:TextBox
      ID="txtSubject"
      Width="200px"
      runat="server">
    </asp:TextBox>
  </td>
  <td>
    <asp:RequiredFieldValidator
      ForeColor="Red"
      ID="RequiredFieldValidator3"
      runat="server"
      ControlToValidate="txtSubject"
      ErrorMessage="Subject is required"
      Text="*">

```

```

        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td style="vertical-align: top">
        <b>Comments:</b>
    </td>
    <td style="vertical-align: top">
        <asp:TextBox
            ID="txtComments"
            Width="200px"
            TextMode="MultiLine"
            Rows="5"
            runat="server">
        </asp:TextBox>
    </td>
    <td style="vertical-align: top">
        <asp:RequiredFieldValidator
            ForeColor="Red"
            ID="RequiredFieldValidator4"
            runat="server"
            ControlToValidate="txtComments"
            ErrorMessage="Comments is required"
            Text="*">
        </asp:RequiredFieldValidator>
    </td>
</tr>
<tr>
    <td colspan="3">
        <asp:ValidationSummary
            HeaderText="Please fix the following errors"
            ForeColor="Red"
            ID="ValidationSummary1"
            runat="server" />
    </td>
</tr>
<tr>
    <td colspan="3">
        <asp:Label
            ID="lblMessage"
            runat="server"
            Font-Bold="true">
        </asp:Label>
    </td>
</tr>
<tr>
    <td colspan="3">
        <asp:Button
            ID="Button1"
            runat="server"
            Text="Submit"
            OnClick="Button1_Click" />
    </td>
</tr>

```

```

        </td>
    </tr>
</table>
</fieldset>
</div>

```

### contactus.aspx.cs

```

public partial class contactus : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        // To Do. Implementation will be discussed in Part 142
    }
}

```

We discussed designing "contact us" page. Once the form is submitted, an email should be sent to the support group. The email should contain all the details that are entered in the contact us form.

Please make sure to include the following 2 namespaces.

```

using System;
using System.Net.Mail;

```

### contactus.aspx.cs:

```

protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        if (Page.IsValid)
        {
            MailMessage mailMessage = new MailMessage();
            mailMessage.From = new MailAddress("YourGmailID@gmail.com");
            mailMessage.To.Add("AdministratorID@YourCompany.com");
            mailMessage.Subject = txtSubject.Text;

            mailMessage.Body = "<b>Sender Name : </b>" + txtName.Text + "<br/>"

```

```
+ "<b>Sender Email : </b>" + txtEmail.Text + "<br/>"
+ "<b>Comments : </b>" + txtComments.Text;
mailMessage.IsBodyHtml = true;
```

```
SmtplibClient smtpClient = new SmtplibClient("smtp.gmail.com", 587);
smtpClient.EnableSsl = true;
smtpClient.Credentials = new
    System.Net.NetworkCredential("YourGmailID@gmail.com", "YourGmailPassowrd");
smtpClient.Send(mailMessage);
```

```
Label1.ForeColor = System.Drawing.Color.Blue;
Label1.Text = "Thank you for contacting us";
```

```
txtName.Enabled = false;
txtEmail.Enabled = false;
txtComments.Enabled = false;
txtSubject.Enabled = false;
Button1.Enabled = false;
```

```
}
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
    // Log the exception information to
```

```
    // database table or event viewer
```

```
Label1.ForeColor = System.Drawing.Color.Red;
```

```
Label1.Text = "There is an unkown problem. Please try later";
```

```
}
```

```
}
```

## Difference between http get and http post methods

HTTP stands for Hypertext Transfer Protocol. Http is the most common protocol used for communication between a web server and a client. For example, to request home page of Pragim Technologies, you would type the following URL. Notice that the URL has http prefix.

The 2 most commonly used HTTP methods are GET and POST.

Let's understand GET and POST requests with an example.

1. Create an asp.net web application
2. Set the name of the project to WebFormsDemo
3. Add WebForm1.aspx. Copy and paste the following HTML

```
<div style="font-family:Arial">
<table>
  <tr>
    <td>First Name
    </td>
    <td>
      <asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>
    </td>
  </tr>
  <tr>
    <td>
      Last Name
    </td>
    <td>
      <asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>
    </td>
  </tr>
  <tr>
    <td>
      Email
    </td>
    <td>
      <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
    </td>
  </tr>
</table>
<asp:Button ID="Button1" runat="server" Text="Submit"
  onclick="Button1_Click" />
</div>
```

4. Copy and paste the following code in WebForm1.aspx.cs

```
public partial class WebForm1 : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
```

```

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("~/WebForm2.aspx?FirstName=" + txtFirstName.Text +
            "&LastName=" + txtLastName.Text + "&Email=" + txtEmail.Text);
    }
}

```

5. Add WebForm2.aspx. Copy and paste the following HTML

```

<div style="font-family:Arial">
    <table>
    <tr>
        <td>First Name
        </td>
        <td>
            <asp:Label ID="lblFirstName" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            Last Name
        </td>
        <td>
            <asp:Label ID="lblLastName" runat="server"></asp:Label>
        </td>
    </tr>
    <tr>
        <td>
            Email
        </td>
        <td>
            <asp:Label ID="lblEmail" runat="server"></asp:Label>
        </td>
    </tr>
    </table>
</div>

```

6. Copy and paste the following code in WebForm2.aspx.cs

```

public partial class WebForm2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblFirstName.Text = Request.QueryString["FirstName"];
        lblLastName.Text = Request.QueryString["LastName"];
        lblEmail.Text = Request.QueryString["Email"];
    }
}

```

Download Fiddler tool from <http://fiddler2.com/get-fiddler> and run it.

Build the solution. Open a browser window. In the Address bar type the following address and press and enter key.

<http://localhost/WebFormsDemo/WebForm1.aspx>

So here we are issuing a GET request for WebForm1.aspx. In the fiddler tool, you can clearly notice that a GET request is issued for WebForm1.aspx

At this point, you will have the user interface to enter FirstName, LastName and Email. Enter the details and when you click "Submit" button, the data will be posted to the server. So, here we are issuing a POST request to submit the data on WebForm1.aspx. In the fiddler tool, you can clearly notice that a POST request is issued to submit WebForm1 data.

In the Button1\_Click() event handler, we have Response.Redirect() call, which issues a GET request to WebForm2.aspx

**GET Request** - is generally used to get data from the web server. A GET request is generally issued,

1. When you click on a hyperlink
2. When Response.Redirect() statement is executed
3. When you type URL in the address bar and hit enter

**POST Request** - is generally used to submit data to the server. A POST request is generally issued,

1. When you click on a submit button
2. When AUTOPOST back is set true and when a selection in the DropDownList is changed

### **Difference between GET and POST method**

1. GET method appends data to the URL, where as with the POST method data can either be appended to the URL or in the message body.
2. As GET request rely on querystrings to send data to the server, there is a length restriction, where as POST requests have no restrictions on data length.
3. While it is possible to change the state of data in database using GET request, they should only be used to retrieve data.

In asp.net webforms, IsPostBack page property is used to check if the request is a GET or a POST request. If IsPostBack property returns true, then the request is POST, else the request is GET.

## How to check if the request method is a GET or a POST in MVC

In asp.net webforms, IsPostBack page property is used to check if the request is a GET or a POST request. If IsPostBack property returns true, then the request is POST, else the request is GET.

In asp.net mvc, use Request object's HttpMethod property to check if the request is a GET or a POST request. Let's discuss using Request.HttpMethod with an example.

1. Create a new asp.net mvc application and name MVCDemo.
2. Add HomeController using "Empty MVC controller" scaffolding template
3. Copy and paste the following code

```
public class HomeController : Controller
{
    // Action method that responds to the GET request
    [HttpGet]
    public ActionResult Index()
    {
        ViewBag.IsPostBack = IsPostBack();
        return View();
    }

    // Action method that responds to the POST request
    [HttpPost]
    [ActionName("Index")]
    public ActionResult Index_Post()
    {
        ViewBag.IsPostBack = IsPostBack();
        return View();
    }

    // This method checks if a request is a GET or a POST request
    private bool IsPostBack()
    {
        return Request.HttpMethod == "POST";
    }
}
```

4. Right click on the Index() action method in HomeController and select "Add View" from the context menu. Set

View name = Index

View engine = Razor

Create a strongly typed view = unchecked

Create a partial view = unchecked

Use a layout or master page = unchecked

Click Add.

5. Copy and paste the following code in Index.cshtml view.

```
<h3 style="font-family:Arial">
    IsPosback = @ViewBag.IsPostBack
```



</h3>

```
@using (@Html.BeginForm())  
{  
    <input type="submit" value="Submit" />  
}
```

Build the project and navigate to <http://localhost/MVCDemo/Home/Index>. Notice that when you first visit the Page, a GET request is issued and hence `IsPostBack = False`. Now click Submit button and notice that `IsPostBack = True` is displayed, as a POST request is issued upon clicking the Submit button.

## Implementing autocomplete textbox in asp.net web forms

We will discuss implementing auto-complete functionality in an asp.net web forms application using jQuery Autocomplete Widget and a web service.

Name:

Name	Gender	TotalMarks
Mark Hastings		
Mary Ward		900
Pam Nicholas	Female	760
John Stenson	Male	980
Ram Gerald	Male	990
Ron Simpson	Male	440
Able Wicht	Male	320
Steve Thompson	Male	983
James Bynes	Male	720
Mary Ward	Female	870
Nick Niron	Male	680

**Step 1:** We will be using tblStudents table in this demo. Please find the sql script below, to create and populate this table with some data.

Create Table tblStudents

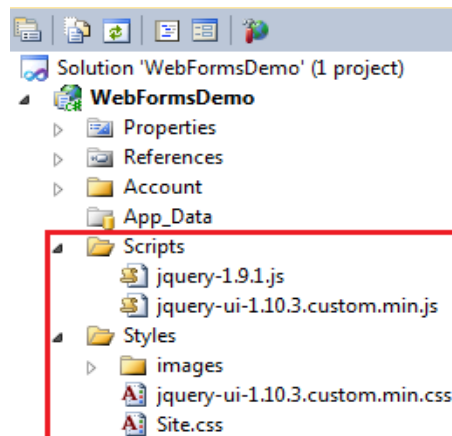
```
(  
  ID int identity primary key,  
  Name nvarchar(50),  
  Gender nvarchar(20),  
  TotalMarks int  
)
```

```
Insert into tblStudents values('Mark Hastings','Male',900)  
Insert into tblStudents values('Pam Nicholas','Female',760)  
Insert into tblStudents values('John Stenson','Male',980)  
Insert into tblStudents values('Ram Gerald','Male',990)  
Insert into tblStudents values('Ron Simpson','Male',440)  
Insert into tblStudents values('Able Wicht','Male',320)  
Insert into tblStudents values('Steve Thompson','Male',983)  
Insert into tblStudents values('James Bynes','Male',720)  
Insert into tblStudents values('Mary Ward','Female',870)  
Insert into tblStudents values('Nick Niron','Male',680)
```

**Step 2:** Download autocomplete widget from <http://jqueryui.com/download>. The following folders and files will be downloaded.

Name	Date modified	Type	Size
css	19/09/201...	File folder	
development-bundle	19/09/201...	File folder	
js	19/09/201...	File folder	
index.html	19/09/201...	Chrome H...	25 KB

**Step 3:** Open "js" folder copy "jquery-1.9.1.js" and "jquery-ui-1.10.3.custom.min.js" files and paste them in the "Scripts" folder of your asp.net project. Now open "css" folder. This folder will be present in "ui-lightness" folder. Copy "images" folder and "jquery-ui-1.10.3.custom.min.css" file and paste them in "Styles" folder in your asp.net project. If you are following along, at this point your solution explorer should look as shown below.



**Step 4:** Right click on the project name in "Solution Explorer" and add a Web Service with name = StudentService.asmx. Copy and paste the following code. This web service is responsible for retrieving matching student names to implement auto-complete when the user types the name of the student in the textbox.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace WebFormsDemo
{
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    [System.Web.Script.Services.ScriptService]
    public class StudentService : System.Web.Services.WebService
    {
        [WebMethod]
        public List<string> GetStudentNames(string searchTerm)
        {
            List<string> studentNames = new List<string>();
            string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
            using (SqlConnection con = new SqlConnection(cs))
            {
                SqlCommand cmd = new SqlCommand("spGetStudentNames", con);
```

```

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter parameter = new SqlParameter("@searchTerm", searchTerm);
cmd.Parameters.Add(parameter);
con.Open();
SqlDataReader rdr = cmd.ExecuteReader();
while (rdr.Read())
{
    studentNames.Add(rdr["Name"].ToString());
}

return studentNames;
}
}
}

```

**Step 5:** Copy and paste the following code in WebForm1.aspx

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <script src="Scripts/jquery-1.9.1.js" type="text/javascript"></script>
    <script src="Scripts/jquery-ui-1.10.3.custom.min.js" type="text/javascript"></script>
    <link href="Styles/jquery-ui-1.10.3.custom.min.css" rel="stylesheet" type="text/css" />
    <script type="text/javascript" language="javascript">
        $(function () {
            $('#<%= txtStudentName.ClientID %>').autocomplete({
                source: function (request, response) {
                    $.ajax({
                        url: "StudentService.asmx/GetStudentNames",
                        data: "{ 'searchTerm': '" + request.term + "' }",
                        type: "POST",
                        dataType: "json",
                        contentType: "application/json; charset=utf-8",
                        success: function (data) {
                            response(data.d);
                        },
                        error: function (result) {
                            alert('There is a problem processing your request');
                        }
                    });
                },
                minLength: 0
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
        <div style="font-family:Arial">
            <asp:TextBox ID="txtStudentName" runat="server">
            </asp:TextBox>

```

```

<asp:Button ID="Button1" runat="server" Text="Button"
    onclick="Button1_Click" />
<br />
<asp:GridView ID="gvStudents" runat="server">
</asp:GridView>
</div>
</form>
</body>
</html>

```

**Step 6:** Copy and paste the following code in WebForm1.aspx.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace WebFormsDemo
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                GetStudents(null);
            }
        }

        private void GetStudents(string studentName)
        {
            string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
            using (SqlConnection con = new SqlConnection(cs))
            {
                SqlCommand cmd = new SqlCommand("spGetStudents", con);
                cmd.CommandType = CommandType.StoredProcedure;

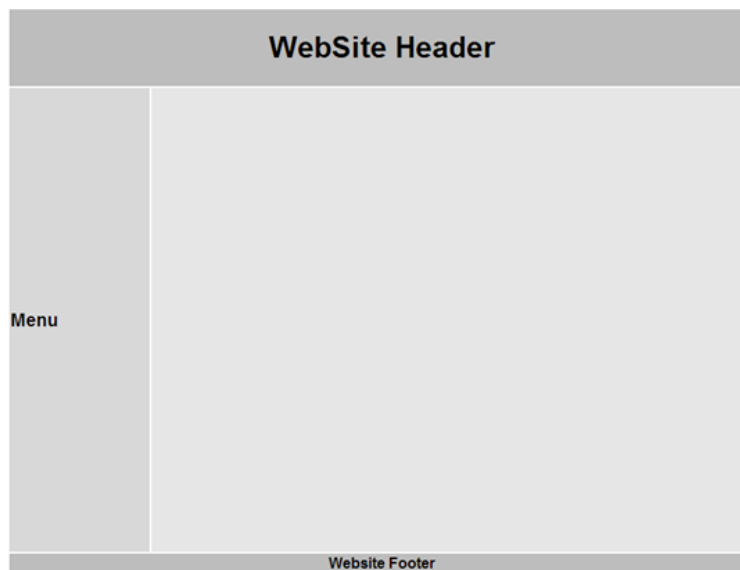
                if (!string.IsNullOrEmpty(studentName))
                {
                    SqlParameter parameter = new SqlParameter("@EmployeeName", studentName);
                    cmd.Parameters.Add(parameter);
                }
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();
                gvStudents.DataSource = rdr;
                gvStudents.DataBind();
            }
        }
    }
}

```

```
    }  
  
    protected void Button1_Click(object sender, EventArgs e)  
    {  
        GetStudents(txtStudentName.Text);  
    }  
}
```

## Why use master pages in asp.net

It is common for a web site to have a consistent look and behaviour across all the pages in the web application. For example, we want all the pages in our application to have the layout as shown below.



To have a layout (similar to the one above) across all our pages, without the use of master pages. There are 2 options:

**Option 1:** Copy and paste the following HTML across all pages in the application

```
<table style="font-family:Arial">
  <tr>
    <td colspan="2" style="width:800px; height:80px; background-color:#BDBDBD;
      text-align:center">
      <h1>
        WebSite Header
      </h1>
    </td>
  </tr>
  <tr>
    <td style="height:500px; background-color:#D8D8D8; width:150px">
      <h3>Menu</h3>
    </td>
    <td style="height:500px; background-color:#E6E6E6; width:650px">
      <h2>This section changes on a page by page basis</h2>
    </td>
  </tr>
  <tr>
    <td colspan="2" style="background-color:#BDBDBD; text-align:center">
      <b>Website Footer</b>
    </td>
  </tr>
</table>
```

The following are the problems with this approach:

1. We have lot of duplicated HTML on every page
2. If we have to change something in the common layout, we will have to make the change in all the pages, which is time consuming and error prone.

**Option 2:** Implement the layout using the following user controls.

1. Header user control
2. Menu user control
3. Footer user control

The following are the problems with using user controls:

1. We will end up with complicated HTML and CSS if we have to design the layout similar to the one in the above image
2. All the user controls have to be manually added on each and every page of the site

So, the best approach to have a consistent look and feel across all pages in a web application is to use master pages.



## Master pages in asp.net

Master pages in asp.net allow you to create a consistent look and behaviour for all the pages in an asp.net web application.

### To create a master page

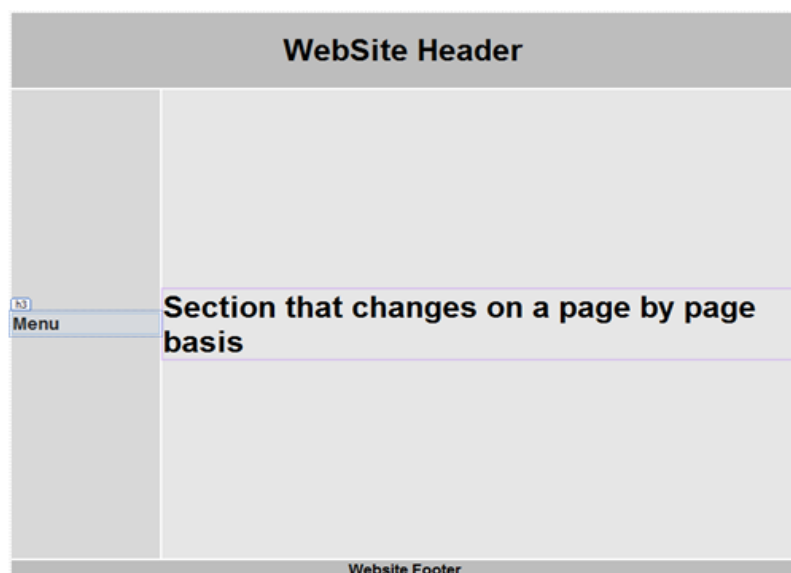
1. Right click on the "Project Name" in "Solution Explorer", and select "Add - New Item"
2. From the "Add New Item" dialog box, select "Master Page"
3. Give your master page a name and click OK.

Master pages have .master extension. Notice that there are 2 ContentPlaceHolder controls on the master page.

1. One in the "head" section and
2. Second one in the "body" section

ContentPlaceHolder control defines a region on the master page, where the content pages can plugin page specific content.

Now, let's design the master page so that it looks as shown below.



### Designing Master Page:

Remove the "ContentPlaceHolder" that is present in the "Head" section of the master page. Also, delete everything that is present in the "Body" section of the master page. Copy and paste the following HTML between the opening and closing <body> tag.

```
<form id="form1 " runat="server">
<table style="font-family: Arial">
  <tr>
    <td colspan="2" style="width: 800px; height: 80px; text-align: center;
      background-color: #BDBDBD;">
      <h1>
        WebSite Header
      </h1>
```

```

        </td>
    </tr>
    <tr>
        <td style="height: 500px; background-color: #D8D8D8; width: 150px">
            <h3>
                Menu</h3>
            </td>
        <td style="height: 500px; background-color: #E6E6E6; width: 650px">
            <asp:ContentPlaceHolder ID="MainContent" runat="server">
                <h1>Section that changes on a page by page basis</h1>
            </asp:ContentPlaceHolder>
        </td>
    </tr>
    <tr>
        <td colspan="2" style="background-color: #BDBDBD; text-align: center">
            <b>Website Footer</b>
        </td>
    </tr>
</table>
</form>

```

Notice that, we only have one ContentPlaceHolder now. Please pay attention to the ID of the ContentPlaceHolder control. In a bit, we will see, how content pages use this ID to plugin their page specific content.

Now, let's add a content page, that is going to use the master page for it's layout.

1. Right click on the "Master Page" in solution explorer and select "Add Content Page"

At this point, you should have a webform, with the following HTML.

```

<% @ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebFormsDemo.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
</asp:Content>

```

### Points to remember:

1. html, head and body sections are not present on the content page, as they are already present on the master page, that this content page is subscribed to.
2. Content pages are associated with master pages using "MasterPageFile" attribute of the "Page" directive
3. Content control on a content page is linked to "ContentPlaceHolder" on the master page using "ContentPlaceHolderID" attribute of the "Content" control. Controls like textbox, button, gridview etc.. and any text that you want in the content page must be placed inside the content control. If you try to place anything outside of the content control, you would get an error - Only Content controls are allowed directly in a content page that contains Content controls.
4. When you flip the content page to design mode, notice that, the region outside of the "ContentPlaceHolder" is greyed out and disabled. This is because on a content page, you should only

be able to edit "Content" that is specific to that page.

So, an asp.net master page allows us to define a common layout for all the pages in the application. Later, if we have to change something in the common layout, we only have to do it at one place, that is in the master page.

## Passing data from content page to master page in asp.net

Let us understand how to pass data from a content page to a master page, with an example.

In the example below,

1. We have a textbox in the master page. The ID of the textbox is txtBoxOnMasterPage
2. On the Content page, we have another textbox and a Button. The ID of this textbox is TextBox1 and the ID of the button is Button1
3. When you type some text in the textbox on the content page and when you hit Set Text button, we want to display the text entered in the textbox on the content page in the textbox that is present in the master page.

Here are the steps:

**Step 1:** Place a TextBox control with ID="txtBoxOnMasterPage" on the master page. Here is the HTML

Content Page Data:<br />  
<asp:TextBox ID="txtBoxOnMasterPage" runat="server"></asp:TextBox>

**Step 2:** In the code-behind file of the master page, include a public property that returns the textbox control. Content pages will use this property to set the text of the textbox on the master page.

```
// The property returns a TextBox
public TextBox TextBoxOnMasterPage
{
    get
    {
        // Return the textbox on the master page
        return this.txtBoxOnMasterPage;
    }
}
```

**Step 3:** Include a TextBox and a Button control on the content page. Here is the HTML.

<b>Type the text in the textbox that you want to display in Master Page TextBox and click Set Text Button</b>  
<asp:TextBox ID="TextBox1" runat="server">  
</asp:TextBox>  
<asp:Button ID="Button1" runat="server" Text="Set Text" onclick="Button1\_Click" />

**Step 4:** Copy and paste the following Button1\_Click() event handler method in the content page

```
protected void Button1_Click(object sender, EventArgs e)
{
    // Retrieve the master page associated with this content page using
    // Master property and typecast to the actual master page and then
    // reference the property
```

```
((Site)Master).TextBoxOnMasterPage.Text = TextBox1.Text;  
}
```

If you want Master property to return a strongly typed reference of the associated master page, include the following MasterType directive on the content page.

```
<%@ MasterType VirtualPath=~/.Site.Master" %>
```

Once you have included the above MasterType directive on the content page, you will now have a strongly typed reference of the master page and can access it's properties without having to typecast.

```
protected void Button1_Click(object sender, EventArgs e)  
{  
    Master.TextBoxOnMasterPage.Text = TextBox1.Text;  
}
```

To retrieve a master page associated with a content page, we can use either:

1. Master property
2. Page.Master property

Page.Master property always returns an object of type MasterPage and we need to explicitly typecast it to the actual master page, if we need to access it's properties and methods.

Where as Master property returns a strongly typed reference of the actual master page if the content page has MasterType directive specified. Otherwise Master property works in the same way as Page.Master property.

## Passing data from master page to content page in asp.net

Let us understand how to pass data from a master page to a content page, with an example.

In the example below,

1. We have a search textbox and a search button in the master page.
2. On the Content page, we have a GridView control that displays students from a database table.
3. When we type some text in the search textbox on the master page and when we hit "Search" button, we want to display all the students whose name contains the search term entered in the search textbox.

Here are the steps:

**Step 1:** Include a panel, textbox and a button control on the master page.

```
<asp:Panel ID="panelSearch" runat="server">
    <b>Search : </b>
    <asp:TextBox ID="txtSerach" runat="server"></asp:TextBox>
    <asp:Button ID="btnSearch" runat="server" Text="Search"/>
</asp:Panel>
```

**Step 2:** Include the following 3 properties in the code-behind file of the master page. We will use these properties in the content page.

```
public Panel SearchPanel
{
    get
    {
        return panelSearch;
    }
}

public string SearchTerm
{
    get
    {
        return txtSerach.Text;
    }
}

public Button SearchButton
{
    get
    {
        return btnSearch;
    }
}
```

**Step 3:** We will be using table tblStudents. Here is the SQL Script to create and populate this table with sample data.

```
Create Table tblStudents
(
    ID int Identity Primary Key,
    Name nvarchar(50),
    Gender nvarchar(20),
    TotalMarks int,
)
```

```
Insert into tblStudents values ('Mark Hastings','Male',900)
Insert into tblStudents values ('Pam Nicholas','Female',760)
Insert into tblStudents values ('John Stenson','Male',980)
Insert into tblStudents values ('Ram Gerald','Male',990)
Insert into tblStudents values ('Ron Simpson','Male',440)
Insert into tblStudents values ('Able Wicht','Male',320)
Insert into tblStudents values ('Steve Thompson','Male',983)
Insert into tblStudents values ('James Bynes','Male',720)
Insert into tblStudents values ('Mary Ward','Female',870)
Insert into tblStudents values ('Nick Niron','Male',680)
```

**Step 4:** Search stored procedure below, returns all students whose name contains @SearchTerm parameter.

```
Create Proc spSearch
@SearchTerm nvarchar(50)
as
Begin
    Select * from tblStudents where Name like '%' + @SearchTerm + '%'
End
```

**Step 5:** Add a content page, and include a GridView control to display the students.

```
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
```

Include the following MasterType directive on the content page

```
<% @ MasterType VirtualPath="~/Site.Master" %>
```

**Step 6:** Copy and paste the following code in the code-behind file of the content page.

```
protected void Page_Init(object sender, EventArgs e)
{
    Master.SearchButton.Click += new EventHandler(SearchButton_Click);
}

protected void Page_Load(object sender, EventArgs e)
{
    GetData(null);
}

protected void SearchButton_Click(object sender, EventArgs e)
```

```

{
    GetData(Master.SearchTerm);
}

private void GetData(string searchTerm)
{
    string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(cs))
    {
        SqlCommand cmd = new SqlCommand("spSearch", con);
        cmd.CommandType = CommandType.StoredProcedure;

        SqlParameter searchParameter = new SqlParameter("@SearchTerm", searchTerm ??
string.Empty);
        cmd.Parameters.Add(searchParameter);
        con.Open();
        GridView1.DataSource = cmd.ExecuteReader();
        GridView1.DataBind();
    }
}

```

Note: Please include the following USING declarations

```

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

```

At this point, run the application and enter a search term in the search textbox on the master page, and click Search button. Notice that the matching students are displayed in the grid view control on the content page.

Let us look at another example of implementing search functionality on another content page. This time we will use table tblEmployee. Here is the sql script to create and populate this table.

Create Table tblEmployee

```

(
    Id int Identity Primary Key,
    Name nvarchar(50),
    Email nvarchar(50),
    Age int,
    Gender nvarchar(50),
    HireDate date,
)

```

```

Insert into tblEmployee values
('Sara Nan','Sara.Nan@test.com',35,'Female','04/04/1999')
Insert into tblEmployee values
('James Histo','James.Histo@test.com',33,'Male','12/07/2008')
Insert into tblEmployee values
('Mary Jane','Mary.Jane@test.com',28,'Female','11/11/2005')
Insert into tblEmployee values
('Paul Sensit','Paul.Sensit@test.com',29,'Male','10/10/2007')

```



```
Insert into tblEmployee values
('Todd Grove','todd.grove@test.com',31,'Male','11/11/2008')
```

Search stored procedure below, returns all employees whose name contains @SearchTerm parameter.

```
Create Proc spSearchEmployees
@SearchTerm nvarchar(50)
as
Begin
Select * from tblEmployee where Name like '%' + @SearchTerm + '%'
End
```

Add a content page, and include a GridView control to display the employees.

```
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
```

Include the following MasterType directive on the content page

```
<% @ MasterType VirtualPath="~/Site.Master" %>
```

Copy and paste the following code in the code-behind file of the content page.

```
protected void Page_Init(object sender, EventArgs e)
{
    Master.SearchButton.Click += new EventHandler(SearchButton_Click);
}

protected void Page_Load(object sender, EventArgs e)
{
    GetData(null);
}

protected void SearchButton_Click(object sender, EventArgs e)
{
    GetData(Master.SearchTerm);
}

private void GetData(string searchTerm)
{
    string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    using (SqlConnection con = new SqlConnection(cs))
    {
        SqlCommand cmd = new SqlCommand("spSearchEmployees", con);
        cmd.CommandType = CommandType.StoredProcedure;

        SqlParameter searchParameter = new SqlParameter("@SearchTerm", searchTerm
?? string.Empty);
        cmd.Parameters.Add(searchParameter);
        con.Open();
        GridView1.DataSource = cmd.ExecuteReader();
    }
}
```

```
        GridView1.DataBind();  
    }  
}
```

Note: Please include the following USING declarations

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

At this point, run the application and enter a search term in the search textbox on the master page, and click Search button. Notice that the matching employees are displayed in the gridview control on the content page.

Let's say on a specific content page, we don't want this search functionality. If that's the case, we can very easily hide the search interface on the master page. Here are the steps.

**Step 1:** Add a new content page

**Step 2:** Include the following MasterType directive on the content page

```
<%@ MasterType VirtualPath="~/Site.Master" %>
```

**Step 3:** Copy and paste the following code in the code-behind file of the content page

```
protected void Page_Load(object sender, EventArgs e)  
{  
    Master.SearchPanel.Visible = false;  
}
```

## Default content in contentplaceholder of a master page

A ContentPlaceHolder in a master page can include default content.

Let's understand this with an example. The following ContentPlaceHolder in the master page has some default content in an <h1> tag.

```
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
    <h1>Content pages can override this content if they wish to</h1>
</asp:ContentPlaceHolder>
```

Now right click on the master page and select "Add Content Page". A new content page should be added and would see the following HTML on the content page

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site2.Master"
AutoEventWireup="true" CodeBehind="WebForm5.aspx.cs"
Inherits="WebFormsDemo.WebForm5" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
</asp:Content>
```

Notice that we have a content control in the content page. Normally we would place the asp.net controls and may be some text inside the content control. However, if we want the content page to default to the content in the ContentPlaceHolder of the master page.

1. Flip the content page to the design mode
2. Right click on the "ContentPlaceHolder" control and select "Default to Master's content"
3. You would get a confirmation dialog box  
If you default to the Master Page content, everything in this region will be removed from the page.  
Do you want to continue?
4. Click Yes

Notice that the default content from the master page "ContentPlaceHolder" is displayed. Now, flip the content page to Source mode, and notice that, the content control is removed. So, if you want a content page to default to master page default content, all you need to do is to remove the content control from the content page.

## Assigning a master page dynamically in asp.net

We will discuss, how to select a master page dynamically at runtime. Let us understand this with an example.

This is what we want to achieve.

1. We have 2 master pages in our application.
2. Include a dropdownlist on both the master pages. The dropdownlist should display both the master pages.
3. As we select a master page from the dropdownlist, we want the selected master page to be dynamically set as the master page.

WebSite Header

Search :

ID	Name	Gender	TotalMarks
1	Mark Hastings	Male	900
2	Pam Nicholas	Female	760
3	John Stenson	Male	980
4	Ram Gerald	Male	990
5	Ron Simpson	Male	440
6	Able Wicht	Male	320
7	Steve Thompson	Male	983
8	James Bynes	Male	720
9	Mary Ward	Female	870
10	Nick Niron	Male	680

Select Master Page  
Master Page 1 ▼

WebSite Header

Search :

ID	Name	Gender	TotalMarks
1	Mark Hastings	Male	900
2	Pam Nicholas	Female	760
3	John Stenson	Male	980
4	Ram Gerald	Male	990
5	Ron Simpson	Male	440
6	Able Wicht	Male	320
7	Steve Thompson	Male	983
8	James Bynes	Male	720
9	Mary Ward	Female	870
10	Nick Niron	Male	680

Select Master Page  
Master Page 2 ▼

Here are the steps:

1. Create a new asp.net webforms application and name it WebFormsDemo.
2. Add a class file to the project and name it BaseMaterPage.cs. Copy and paste the following code. Notice that the BaseMaterPage class inherits from System.Web.UI.MasterPage. This BaseMaterPage class will be used as the base class for the 2 master pages that we will be creating later.

```

using System.Web.UI.WebControls;
namespace WebFormsDemo
{
    public abstract class BaseMaterPage : System.Web.UI.MasterPage
    {
        public abstract Panel SearchPanel { get; }

        public abstract string SearchTerm { get; }

        public abstract Button SearchButton { get; }
    }
}

```

3. Add a master page and name it Site.Master. Copy and paste the following html.

```

<table style="font-family: Arial">
<tr>
    <td colspan="2" style="width: 800px; height: 80px; text-align:
        center; background-color: #BDBDBD;">
        <h1>WebSite Header</h1>
        <asp:Panel ID="panelSearch" runat="server">
            <b>Search : </b>
            <asp:TextBox ID="txtSerach" runat="server"></asp:TextBox>
            <asp:Button ID="btnSearch" runat="server" Text="Search" />
        </asp:Panel>
    </td>
</tr>
<tr>
    <td style="height: 500px; background-color: #D8D8D8; width: 150px">
        <b>Select Master Page</b>
        <asp:DropDownList ID="DropDownList1" runat="server"
            AutoPostBack="true"
            OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
            <asp:ListItem Text="Master Page 1" Value="~/Site.Master">
            </asp:ListItem>
            <asp:ListItem Text="Master Page 2" Value="~/Site2.Master">
            </asp:ListItem>
        </asp:DropDownList>
    </td>
    <td style="height: 500px; background-color: #E6E6E6; width: 650px">
        <asp:ContentPlaceHolder ID="MainContent" runat="server">
            <h1>
                Section that changes on a page by page basis</h1>
        </asp:ContentPlaceHolder>
    </td>
</tr>
<tr>
    <td colspan="2" style="background-color: #BDBDBD; text-align: center">
        <b>Website Footer</b>
    </td>
</tr>

```

</table>

4. Copy and paste the following code in the code behind file. Notice that the Site master page class inherits from the [BaseMaterPage](#) class that we created in Step 1.

```
public partial class Site : BaseMaterPage
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    public override Panel SearchPanel
    {
        get
        {
            return panelSearch;
        }
    }

    public override string SearchTerm
    {
        get
        {
            return txtSerach.Text;
        }
    }

    public override Button SearchButton
    {
        get
        {
            return btnSearch;
        }
    }

    protected void DropDownList1_SelectedIndexChanged
        (object sender, EventArgs e)
    {
        Session["SELECTED_MASTERPAGE"] = DropDownList1.SelectedValue;
        Response.Redirect(Request.Url.AbsoluteUri);
    }

    protected void Page_PreRender(object sender, EventArgs e)
    {
        if (Session["SELECTED_MASTERPAGE"] != null)
        {
            DropDownList1.SelectedValue =
                Session["SELECTED_MASTERPAGE"].ToString();
        }
    }
}
```

5. Add another master page and name it Site2.Master. Copy and paste the following html.

```
<table style="font-family: Arial; color:White">
<tr>
  <td colspan="2" style="width: 800px; height: 80px;
    text-align: center; background-color:Red;">
    <h1>WebSite Header</h1>
    <asp:Panel ID="panelSearch" runat="server">
      <b>Search : </b>
      <asp:TextBox ID="txtSerach" runat="server"></asp:TextBox>
      <asp:Button ID="btnSearch" runat="server" Text="Search"/>
    </asp:Panel>
  </td>
</tr>
<tr>
  <td style="height: 500px; background-color:Green; width: 650px">
    <asp:ContentPlaceHolder ID="MainContent" runat="server">
      <h1>Section that changes on a page by page basis</h1>
    </asp:ContentPlaceHolder>
  </td>
  <td style="height: 500px; background-color:Blue; width: 150px">
    <b>Select Master Page</b>
    <asp:DropDownList ID="DropDownList1" runat="server"
      AutoPostBack="true"
      OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
      <asp:ListItem Text="Master Page 1" Value="~/Site.Master">
      </asp:ListItem>
      <asp:ListItem Text="Master Page 2" Value="~/Site2.Master">
      </asp:ListItem>
    </asp:DropDownList>
  </td>
</tr>
<tr>
  <td colspan="2" style="background-color:Red; text-align: center">
    <b>Website Footer</b>
  </td>
</tr>
</table>
```

6. Copy and paste the following code in the code behind file. Notice that the Site2 master page class inherits from the BaseMaterPage class that we created in Step 1.

```
public partial class Site2 : BaseMaterPage
{
  protected void Page_Load(object sender, EventArgs e)
  {
  }

  public override Panel SearchPanel
  {
    get
```

```

    {
        return panelSearch;
    }
}

public override string SearchTerm
{
    get
    {
        return txtSerach.Text;
    }
}

public override Button SearchButton
{
    get
    {
        return btnSearch;
    }
}

protected void DropDownList1_SelectedIndexChanged
(object sender, EventArgs e)
{
    Session["SELECTED_MASTERPAGE"] = DropDownList1.SelectedValue;
    Response.Redirect(Request.Url.AbsoluteUri);
}

protected void Page_PreRender(object sender, EventArgs e)
{
    if (Session["SELECTED_MASTERPAGE"] != null)
    {
        DropDownList1.SelectedValue =
            Session["SELECTED_MASTERPAGE"].ToString();
    }
}
}

```

7. Add a class file and name it BasePage.cs. Copy and paste the following code. Notice that the BasePage class inherits from System.Web.UI.Page. This BasePage class will be used as the base class for all the content pages in our application.

```

using System;
namespace WebFormsDemo
{
    public class BasePage : System.Web.UI.Page
    {
        protected override void OnPreInit(EventArgs e)
        {
            if (Session["SELECTED_MASTERPAGE"] != null)
            {
                this.MasterPageFile = Session["SELECTED_MASTERPAGE"].ToString();
            }
        }
    }
}

```



```

    }
}
}
}

```

8. Right click on Site.Master page and select Add Content Page. Copy and paste the following html. Notice that we have set MasterType to BaseMaterPage class.

```

<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs" Inherits="WebFormsDemo.WebForm1" %>
<%@ MasterType TypeName="WebFormsDemo.BaseMaterPage" %>
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
    <asp:GridView ID="GridView1" runat="server">
    </asp:GridView>
</asp:Content>

```

9. In the code-behind file of the content page, copy and paste the following code.

```

public partial class WebForm1 : BasePage
{
    protected void Page_Init(object sender, EventArgs e)
    {
        Master.SearchButton.Click += new
            EventHandler(SearchButton_Click);
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            GetData(null);
        }
    }

    protected void SearchButton_Click(object sender, EventArgs e)
    {
        GetData(Master.SearchTerm);
    }

    private void GetData(string searchTerm)
    {
        string cs = ConfigurationManager.ConnectionStrings["DBCS"]
            .ConnectionString;
        using (SqlConnection con = new SqlConnection(cs))
        {
            SqlCommand cmd = new SqlCommand("spSearch", con);
            cmd.CommandType = CommandType.StoredProcedure;

            SqlParameter searchParameter = new SqlParameter
                ("@SearchTerm", searchTerm ?? string.Empty);
            cmd.Parameters.Add(searchParameter);
            con.Open();

```

```
        GridView1.DataSource = cmd.ExecuteReader();
        GridView1.DataBind();
    }
}
```

Note: Please include the following USING declarations

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

## Master page content page user control life cycle in asp.net

We will understand the order of events execution, when we have a content page with a master page and a user control.

1. Create a new asp.net webforms application and name it WebFormsDemo.
2. Add a UserControl with name="TestUserControl.ascx". Copy and paste the following html. This is a very simple user control, with just an h1 tag with some static text.

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="TestUserControl.ascx.cs"
Inherits="WebFormsDemo.TestUserControl" %>
<h1>This is a Test User Control</h1>
```

3. Add a MasterPage and name it Site.Master. Copy and paste the following code in the code-behind file.

```
public partial class Site : System.Web.UI.MasterPage
{
    protected void Page_Init(object sender, EventArgs e)
    {
        Response.Write("Master Page Init Event<br/>");
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Master Page Load Event<br/>");
    }

    protected void Page_PreRender(object sender, EventArgs e)
    {
        Response.Write("Master Page PreRender Event<br/>");
    }
}
```

4. Right click on the MasterPage, and select Add Content Page. This should add "WebForm1.aspx" with a content control in it. Flip WebForm1.aspx to design mode. Drag and drop a textbox control and TestUserControl onto the content page. Associate event handler methods for OnInit, OnLoad and OnPreRender events for the textbox and the usercontrol. At this point the HTML should look as shown below.

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebFormsDemo.WebForm1" %>
<%@ Register src="TestUserControl.ascx"
    tagname="TestUserControl" tagprefix="uc1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
    runat="server">
    <asp:TextBox ID="TextBox1" runat="server"
        OnInit="TextBox1_Init" OnLoad="TextBox1_Load"
        OnPreRender="TextBox1_PreRender">
    </asp:TextBox>
```

```

<br />
<uc1:TestUserControl ID="TestUserControl1" runat="server"
OnInit="TestUC1_Init" OnLoad="TestUC1_Load"
OnPreRender="TestUC1_PreRender"/>
</asp:Content>

```

5. Finally copy and paste the following code in the code-behind file of WebForm1.

```

public partial class WebForm1 : System.Web.UI.Page
{
    protected void Page_Init(object sender, EventArgs e)
    {
        Response.Write("Content Page Init Event<br/>");
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Write("Content Page Load Event<br/>");
    }
    protected void Page_PreRender(object sender, EventArgs e)
    {
        Response.Write("Content Page PreRender Event<br/>");
    }
    protected void TextBox1_Init(object sender, EventArgs e)
    {
        Response.Write("TextBox Init Event<br/>");
    }
    protected void TextBox1_Load(object sender, EventArgs e)
    {
        Response.Write("TextBox Load Event<br/>");
    }
    protected void TextBox1_PreRender(object sender, EventArgs e)
    {
        Response.Write("TextBox PreRender Event<br/>");
    }
    protected void TestUC1_Init(object sender, EventArgs e)
    {
        Response.Write("UserControl Init Event<br/>");
    }
    protected void TestUC1_Load(object sender, EventArgs e)
    {
        Response.Write("UserControl Load Event<br/>");
    }
    protected void TestUC1_PreRender(object sender, EventArgs e)
    {
        Response.Write("UserControl PreRender Event<br/>");
    }
}

```

**Run the application and notice that, the events are executed in the following order.**

TextBox Init Event  
UserControl Init Event

Master Page Init Event  
Content Page Init Event  
Content Page Load Event  
Master Page Load Event  
TextBox Load Event  
UserControl Load Event  
Content Page PreRender Event  
Master Page PreRender Event  
TextBox PreRender Event  
UserControl PreRender Event

So, in general the initialization events are raised from the innermost control to the outermost one, and all other events are raised from the outermost control to the innermost one.

Please note that the master page is merged into the content page and treated as a control in the content page. Master page is merged into the content page during the initialization stage of page processing.

**Question 1:** We have a master page, content page and a user control, with the following events

Master init & master load  
Content init & Content load  
User control init & User control load

When we run the page containing above things, in what sequence the above events are fired.

**Answer:** Here is the order

User control init  
Master init  
Content Init  
Content load  
Master load  
UserControl Load

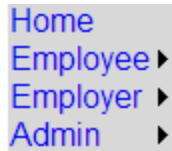
## Menu control in asp.net

As the name speaks for itself, Menu control in asp.net is used to display a menu in an asp.net web application.

The content for the Menu control can be specified directly in the control or the menu control can be bound to a data source.

A Menu control is a collection of MenuItem objects.

To get a menu as shown in the image below,



we would configure the menu control as shown below.

```
<asp:Menu ID="Menu1" runat="server">
  <Items>
    <asp:MenuItem NavigateUrl="~/Home.aspx" Text="Home" Value="Home">
    </asp:MenuItem>
    <asp:MenuItem NavigateUrl="~/Employee.aspx" Text="Employee" Value="Employee">
      <asp:MenuItem NavigateUrl="~/UploadResume.aspx" Text="Upload Resume"
Value="Upload Resume">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/EditResume.aspx" Text="Edit Resume" Value="Edit
Resume">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/ViewResume.aspx" Text="View Resume" Value="View
Resume">
      </asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem NavigateUrl="~/Employer.aspx" Text="Employer" Value="Employer">
      <asp:MenuItem NavigateUrl="~/UploadJob.aspx" Text="Upload Job" Value="Upload Job">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/EditJob.aspx" Text="Edit Job" Value="Edit Job">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/ViewJob.aspx" Text="View Job" Value="View Job">
      </asp:MenuItem>
    </asp:MenuItem>
    <asp:MenuItem NavigateUrl="~/Admin.aspx" Text="Admin" Value="Admin">
      <asp:MenuItem NavigateUrl="~/AddUser.aspx" Text="Add User" Value="Add User">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/EditUser.aspx" Text="Edit User" Value="Edit User">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/ViewUser.aspx" Text="View User" Value="View User">
      </asp:MenuItem>
    </asp:MenuItem>
  </Items>
```

</asp:Menu>

By default, the menu control is displayed using vertical orientation. If you want to change its orientation to Horizontal, use Orientation property.

Home Employee ▶ Employer ▶ Admin ▶  
Add User  
Edit User  
View User

<asp:Menu ID="Menu1" runat="server" Orientation="Horizontal">

By default, StaticDisplayLevels Property is set to 1. This means only the first level is statically displayed. If you want 2 levels to be displayed statically, you would set this property to 2.

Home  
Employee  
Upload Resume  
Edit Resume  
View Resume  
Employer  
Upload Job  
Edit Job  
View Job  
Admin  
Add User  
Edit User  
View User

<asp:Menu ID="Menu1" runat="server" StaticDisplayLevels="2">

To control the amount of time it takes for the dynamically displayed portion of a menu to disappear, use DisappearAfter property. This is in milliseconds. If you want the menu to disappear after 2 seconds, you would set it to 2000 milliseconds as shown below.

<asp:Menu ID="Menu1" runat="server" DisappearAfter="2000">

## Using styles with asp.net menu control

To configure styles for the statically displayed portion of the menu, use:

**StaticMenuStyle** - Applicable for the entire statically displayed portion of the menu

**StaticMenuItemStyle** - Applicable for the individual menu items in the statically displayed portion of the menu

**StaticSelectedStyle** - Applicable for the selected menu item in the statically displayed portion of the menu

**StaticHoverStyle** - Applicable for the statically displayed menu item over which mouse is hovered over

To configure styles for the dynamically displayed portion of the menu, use:

**DynamicMenuStyle** - Applicable for the entire dynamically displayed portion of the menu

**DynamicMenuItemStyle** - Applicable for the individual menu items in the dynamically displayed portion of the menu

**DynamicSelectedStyle** - Applicable for the selected menu item in the dynamically displayed portion of the menu

**DynamicHoverStyle** - Applicable for the dynamically displayed menu item over which mouse is hovered over

To configure styles for each level in the menu control, use:

**LevelMenuItemStyles** - If the menu control has 2 levels, you will have the html as shown below. MenuLevel1 css class will be applied for menu items at level 1. Along the same lines MenuLevel2 css class will be applied for menu items at level 2.

```
<LevelMenuItemStyles>
  <asp:MenuItemStyle CssClass="MenuLevel1" />
  <asp:MenuItemStyle CssClass="MenuLevel2" />
</LevelMenuItemStyles>
```

**LevelSelectedStyles** - If the menu control has 2 levels, you will have the html as shown below. Yellow color will be applied for the menu item selected at level 1. Maroon color will be applied for menu item selected at level 2.

```
<LevelSelectedStyles>
  <asp:MenuItemStyle ForeColor="Yellow" />
  <asp:MenuItemStyle ForeColor="Maroon" />
</LevelSelectedStyles>
```

If you are following along with the example in the demo, for LevelSelectedStyles, StaticSelectedStyle and DynamicSelectedStyle to work, you need the following code in Page\_Load of the master page.

```
protected void Page_Load(object sender, EventArgs e)
{
    foreach (MenuItem item in Menu1.Items)
    {
        Check(item);
    }
}
```



```

}

private void Check(MenuItem item)
{
    if (item.NavigateUrl.Equals(Request.AppRelativeCurrentExecutionFilePath,
StringComparison.InvariantCultureIgnoreCase))
    {
        item.Selected = true;
    }
    else if (item.ChildItems.Count > 0)
    {
        foreach (MenuItem menuItem in item.ChildItems)
        {
            Check(menuItem);
        }
    }
}
}

```

HTML of the menu control

```

<asp:Menu ID="Menu1" runat="server">
    <LevelMenuItemStyles>
        <asp:MenuItemStyle CssClass="MenuLevel1" />
        <asp:MenuItemStyle CssClass="MenuLevel2" />
    </LevelMenuItemStyles>
    <LevelSelectedStyles>
        <asp:MenuItemStyle ForeColor="Yellow" />
        <asp:MenuItemStyle ForeColor="Maroon" />
    </LevelSelectedStyles>
    <!--<StaticSelectedStyle ForeColor="Green" />
    <DynamicSelectedStyle ForeColor="Green" />--%>
    <Items>
        <asp:MenuItem NavigateUrl="/Home.aspx" Text="Home" Value="Home">
        </asp:MenuItem>
        <asp:MenuItem NavigateUrl="/Employee.aspx" Text="Employee" Value="Employee">
            <asp:MenuItem NavigateUrl="/UploadResume.aspx" Text="Upload Resume"
Value="Upload Resume">
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="/EditResume.aspx" Text="Edit Resume" Value="Edit
Resume">
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="/ViewResume.aspx" Text="View Resume" Value="View
Resume">
            </asp:MenuItem>
        </asp:MenuItem>
        <asp:MenuItem NavigateUrl="/Employer.aspx" Text="Employer" Value="Employer">
            <asp:MenuItem NavigateUrl="/UploadJob.aspx" Text="Upload Job" Value="Upload Job">
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="/EditJob.aspx" Text="Edit Job" Value="Edit Job">
            </asp:MenuItem>
            <asp:MenuItem NavigateUrl="/ViewJob.aspx" Text="View Job" Value="View Job">
            </asp:MenuItem>

```

```
</asp:MenuItem>
<asp:MenuItem NavigateUrl="~/Admin.aspx" Text="Admin" Value="Admin">
  <asp:MenuItem NavigateUrl="~/AddUser.aspx" Text="Add User" Value="Add User">
    </asp:MenuItem>
    <asp:MenuItem NavigateUrl="~/EditUser.aspx" Text="Edit User" Value="Edit User">
      </asp:MenuItem>
      <asp:MenuItem NavigateUrl="~/ViewUser.aspx" Text="View User" Value="View User">
        </asp:MenuItem>
      </asp:MenuItem>
    </Items>
  </asp:Menu>
```

## Binding asp.net menu control to an xml file using xmldatasource control

we will discuss binding asp.net menu control to an xml file using xmldatasource control.

1. Add an XML file and name it MenuData.xml. Copy and paste the following xml.

```
<?xml version="1.0" encoding="utf-8" ?>
<Items>
  <MenuItem NavigateUrl=~ /Home.aspx" Text="Home"/>
  <MenuItem NavigateUrl=~ /Employee.aspx" Text="Employee">
    <MenuItem NavigateUrl=~ /UploadResume.aspx" Text="Upload Resume"/>
    <MenuItem NavigateUrl=~ /EditResume.aspx" Text="Edit Resume"/>
    <MenuItem NavigateUrl=~ /ViewResume.aspx" Text="View Resume"/>
  </MenuItem>
  <MenuItem NavigateUrl=~ /Employer.aspx" Text="Employer">
    <MenuItem NavigateUrl=~ /UploadJob.aspx" Text="Upload Job"/>
    <MenuItem NavigateUrl=~ /EditJob.aspx" Text="Edit Job"/>
    <MenuItem NavigateUrl=~ /ViewJob.aspx" Text="View Job"/>
  </MenuItem>
  <MenuItem NavigateUrl=~ /Admin.aspx" Text="Admin">
    <MenuItem NavigateUrl=~ /AddUser.aspx" Text="Add User"/>
    <MenuItem NavigateUrl=~ /EditUser.aspx" Text="Edit User"/>
    <MenuItem NavigateUrl=~ /ViewUser.aspx" Text="View User"/>
  </MenuItem>
</Items>
```

2. Drag and drop an XmlDataSource control on the webform. Set XPath and DataFile attributes as shown below. Notice that DataFile attribute points to the XML file that we added in Step 1.

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
  XPath="/Items/MenuItem" DataFile=~ /MenuData.xml">
</asp:XmlDataSource>
```

3. Drag and drop a menu control and set DataSourceID attribute to the xmldatasource control we created in Step 2. Also, set DataBindings as shown below.

```
<asp:Menu ID="Menu1" runat="server" DataSourceID="XmlDataSource1"
  OnPreRender="Menu1_PreRender">
  <DataBindings>
    <asp:MenuItemBinding DataMember="MenuItem"
      NavigateUrlField="NavigateUrl" TextField="Text" />
  </DataBindings>
</asp:Menu>
```

4. To set the styles for the selected menu item, copy and paste the following code in the code-behind file.

```
private void Check(MenuItem item)
{
  if (item.NavigateUrl.Equals(Request.AppRelativeCurrentExecutionFilePath,
    StringComparison.InvariantCultureIgnoreCase))
  {

```

```
        item.Selected = true;
    }
    else if (item.ChildItems.Count > 0)
    {
        foreach (MenuItem menuItem in item.ChildItems)
        {
            Check(menuItem);
        }
    }
}

protected void Menu1_PreRender(object sender, EventArgs e)
{
    foreach (MenuItem item in Menu1.Items)
    {
        Check(item);
    }
}
```

## Binding asp.net menu control to database table

we will discuss binding asp.net menu control to database table.

1. Create the required database tables using the script below.

Create Table tblMenuItemsLevel1

```
(  
  ID int identity primary key,  
  MenuText nvarchar(50),  
  NavigateURL nvarchar(50)  
)
```

```
Insert into tblMenuItemsLevel1 values('Home','~/Home.aspx')  
Insert into tblMenuItemsLevel1 values('Employee','~/Employee.aspx')  
Insert into tblMenuItemsLevel1 values('Employer','~/Employer.aspx')  
Insert into tblMenuItemsLevel1 values('Admin','~/Admin.aspx')
```

Create Table tblMenuItemsLevel2

```
(  
  ID int identity primary key,  
  MenuText nvarchar(50),  
  NavigateURL nvarchar(50),  
  ParentId int foreign key references tblMenuItemsLevel1 (ID)  
)
```

```
Insert into tblMenuItemsLevel2 values ('Upload Resume','~/UploadResume.aspx',2)  
Insert into tblMenuItemsLevel2 values ('Edit Resume','~/EditResume.aspx',2)  
Insert into tblMenuItemsLevel2 values ('View Resume','~/ViewResume.aspx',2)
```

```
Insert into tblMenuItemsLevel2 values ('Upload Job','~/UploadJob.aspx',3)  
Insert into tblMenuItemsLevel2 values ('Edit Job','~/EditJob.aspx',3)  
Insert into tblMenuItemsLevel2 values ('View Job','~/ViewJob.aspx',3)
```

```
Insert into tblMenuItemsLevel2 values ('Add User','~/AddUser.aspx',4)  
Insert into tblMenuItemsLevel2 values ('Edit User','~/EditUser.aspx',4)  
Insert into tblMenuItemsLevel2 values ('View User','~/ViewUser.aspx',4)
```

2. Create a stored procedure that returns data from both the tables.

```
Create Proc spGetMenuData  
as  
Begin  
  Select * from tblMenuItemsLevel1  
  Select * from tblMenuItemsLevel2  
End
```

3. Drag and drop a menu control on the webform

```
<asp:Menu ID="Menu1" runat="server">  
</asp:Menu>
```

4. Copy and paste the following ado.net code in the code-behind file.

```
protected void Page_Load(object sender, EventArgs e)
{
    GetMenuItems();
}

private void GetMenuItems()
{
    string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    SqlConnection con = new SqlConnection(cs);
    SqlDataAdapter da = new SqlDataAdapter("spGetMenuData", con);
    da.SelectCommand.CommandType = CommandType.StoredProcedure;
    DataSet ds = new DataSet();
    da.Fill(ds);

    ds.Relations.Add("ChildRows", ds.Tables[0].Columns["ID"], ds.Tables[1].Columns["ParentId"]);

    foreach (DataRow level1DataRow in ds.Tables[0].Rows)
    {
        MenuItem item = new MenuItem();
        item.Text = level1DataRow["MenuText"].ToString();
        item.NavigateUrl = level1DataRow["NavigateURL"].ToString();

        DataRow[] level2DataRows = level1DataRow.GetChildRows("ChildRows");
        foreach (DataRow level2DataRow in level2DataRows)
        {
            MenuItem childItem = new MenuItem();
            childItem.Text = level2DataRow["MenuText"].ToString();
            childItem.NavigateUrl = level2DataRow["NavigateURL"].ToString();
            item.ChildItems.Add(childItem);
        }
        Menu1.Items.Add(item);
    }
}
```

Note: Please include the following using declarations.

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

If you want the menu control to apply selected menu item styles.

1. Include OnPreRender attribute in the aspx page
2. Copy and paste the following code in the code-behind file.

```
private void Check(MenuItem item)
{
    if (item.NavigateUrl.Equals(Request.AppRelativeCurrentExecutionFilePath,
        StringComparison.InvariantCultureIgnoreCase))
```

```
{
    item.Selected = true;
}
else if (item.ChildItems.Count > 0)
{
    foreach (MenuItem menuItem in item.ChildItems)
    {
        Check(menuItem);
    }
}
}

protected void Menu1_PreRender(object sender, EventArgs e)
{
    foreach (MenuItem item in Menu1.Items)
    {
        Check(item);
    }
}
```

## SiteMapPath control in asp.net

SiteMapPath control displays navigation path. This navigation path is often called as breadcrumb. Using SiteMapPath control is straight forward.

Here are the steps:

1. Drag and drop SiteMapPath control onto the master page.

```
<asp:SiteMapPath ID="SiteMapPath1" runat="server">
</asp:SiteMapPath>
```

2. Right click on the project name in solution explorer and add a SiteMap file. Copy and paste the following content.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode title="DummyRoot">
    <siteMapNode url="~/Home.aspx" title="Home"/>
    <siteMapNode url="~/Employee.aspx" title="Employee">
      <siteMapNode url="~/UploadResume.aspx" title="Upload Resume"/>
      <siteMapNode url="~/EditResume.aspx" title="Edit Resume"/>
      <siteMapNode url="~/ViewResume.aspx" title="View Resume"/>
    </siteMapNode>
    <siteMapNode url="~/Employer.aspx" title="Employer">
      <siteMapNode url="~/UploadJob.aspx" title="Upload Job"/>
      <siteMapNode url="~/EditJob.aspx" title="Edit Job"/>
      <siteMapNode url="~/ViewJob.aspx" title="View Job"/>
    </siteMapNode>
    <siteMapNode url="~/Admin.aspx" title="Admin">
      <siteMapNode url="~/AddUser.aspx" title="Add User"/>
      <siteMapNode url="~/EditUser.aspx" title="Edit User"/>
      <siteMapNode url="~/ViewUser.aspx" title="View User"/>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

That's it we are done. Run the application and notice that SiteMapPath control displays the navigation path. The SiteMapPath control automatically reads data from Web.sitemap file.

At the moment, the only problem is that it is displaying DummyRoot node as well. Here are the steps to hide the Root node in SiteMapPath control in asp.net.

1. Specify OnItemCreated attribute for the SiteMapPath control on the master page.
2. Copy and paste the following code in the code-behind file of the master page

```
protected void SiteMapPath1_ItemCreated(object sender, SiteMapNodeItemEventArgs e)
{
    if (e.Item.ItemType == SiteMapNodeItemType.Root ||
        (e.Item.ItemType == SiteMapNodeItemType.PathSeparator && e.Item.ItemIndex == 1))
    {
        e.Item.Visible = false;
    }
}
```



}  
}

## Binding asp.net menu control to web.sitemap file using sitemapdatasource control

We discussed binding menu control to an xml file using xmldatasource control.

It is also possible to bind menu control to web.sitemap file.

Here are the steps to achieve this:

1. Add web.sitemap file. Copy and paste the following XML

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode title="DummyRoot">
    <siteMapNode url="~/Home.aspx" title="Home"/>
    <siteMapNode url="~/Employee.aspx" title="Employee">
      <siteMapNode url="~/UploadResume.aspx" title="Upload Resume"/>
      <siteMapNode url="~/EditResume.aspx" title="Edit Resume"/>
      <siteMapNode url="~/ViewResume.aspx" title="View Resume"/>
    </siteMapNode>
    <siteMapNode url="~/Employer.aspx" title="Employer">
      <siteMapNode url="~/UploadJob.aspx" title="Upload Job"/>
      <siteMapNode url="~/EditJob.aspx" title="Edit Job"/>
      <siteMapNode url="~/ViewJob.aspx" title="View Job"/>
    </siteMapNode>
    <siteMapNode url="~/Admin.aspx" title="Admin">
      <siteMapNode url="~/AddUser.aspx" title="Add User"/>
      <siteMapNode url="~/EditUser.aspx" title="Edit User"/>
      <siteMapNode url="~/ViewUser.aspx" title="View User"/>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

2. Drag and drop "SiteMapDataSource" control onto the webform. By default, a SiteMapDataSource control reads data automatically from web.sitemap file.

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
```

3. Finally drag and drop Menu control onto the webform and set DataSourceID attribute, to the ID of the SiteMapDataSource control created in Step 2.

```
<asp:Menu ID="Menu1" DataSourceID="SiteMapDataSource1" runat="server">
</asp:Menu>
```

That's it. Run the application and notice that, the Menu control is bound to web.sitemap file. One problem here is that, the Root node is also displayed.

### Hiding Root node in the Menu control.

To hide the Root node in the Menu control, set ShowStartingNode attribute of the SiteMapDataSource control to false.

```
<asp:SiteMapDataSource ShowStartingNode="false" ID="SiteMapDataSource1" runat="server" />
```

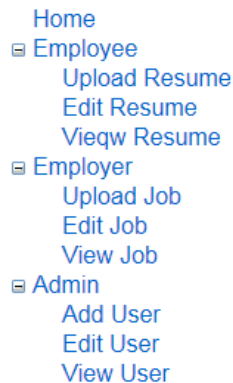
## TreeView control in asp.net

TreeView Web control is useful to display hierarchical data in a tree structure. The content for the TreeView control can be specified directly in the control or the TreeView control can be bound to :

1. XML file
2. web.sitemap file
3. Database table

A TreeView is a collection of TreeNode objects.

To get a TreeView control as shown in the image below,



We would configure the TreeView control as shown below. Notice that we have set Target attribute of TreeNode to \_blank. This ensures that the target page is opened in a new window.

```
<asp:TreeView ID="TreeView1" runat="server">
  <Nodes>
    <asp:TreeNode Text="Home" NavigateUrl="~/Home.aspx" Target="_blank" />
    <asp:TreeNode Text="Employee" NavigateUrl="~/Employee.aspx" Target="_blank">
      <asp:TreeNode Text="Upload Resume" NavigateUrl="~/UploadResume.aspx"
Target="_blank"/>
      <asp:TreeNode Text="Edit Resume" NavigateUrl="~/EditResume.aspx" Target="_blank"/>
      <asp:TreeNode Text="View Resume" NavigateUrl="~/ViewResume.aspx"
Target="_blank"/>
    </asp:TreeNode>
    <asp:TreeNode Text="Employer" NavigateUrl="~/Employer.aspx" Target="_blank">
      <asp:TreeNode Text="Upload Job" NavigateUrl="~/UploadJob.aspx" Target="_blank"/>
      <asp:TreeNode Text="Edit Job" NavigateUrl="~/EditJob.aspx" Target="_blank"/>
      <asp:TreeNode Text="View Job" NavigateUrl="~/ViewJob.aspx" Target="_blank"/>
    </asp:TreeNode>
    <asp:TreeNode Text="Admin" NavigateUrl="~/Admin.aspx" Target="_blank">
      <asp:TreeNode Text="Add User" NavigateUrl="~/AddUser.aspx" Target="_blank"/>
      <asp:TreeNode Text="Edit User" NavigateUrl="~/EditUser.aspx" Target="_blank"/>
      <asp:TreeNode Text="View User" NavigateUrl="~/ViewUser.aspx" Target="_blank"/>
    </asp:TreeNode>
  </Nodes>
</asp:TreeView>
```

To configure the look and feel of the TreeView control, the following styles can be used.

1. HoverNodeStyle
2. LeafNodeStyle
3. RootNodeStyle
4. SelectedNodeStyle etc.

## Binding asp.net treeview control to an xml file using xmldatasource control

we will discuss binding asp.net TreeView control to an xml file using xmldatasource control.

1. Add an XML file and name it TreeViewData.xml. Copy and paste the following xml.

```
<?xml version="1.0" encoding="utf-8" ?>
<Items>
  <TreeViewItem NavigateUrl="~/Home.aspx" Text="Home"/>
  <TreeViewItem NavigateUrl="~/Employee.aspx" Text="Employee">
    <TreeViewItem NavigateUrl="~/UploadResume.aspx" Text="Upload Resume"/>
    <TreeViewItem NavigateUrl="~/EditResume.aspx" Text="Edit Resume"/>
    <TreeViewItem NavigateUrl="~/ViewResume.aspx" Text="View Resume"/>
  </TreeViewItem>
  <TreeViewItem NavigateUrl="~/Employer.aspx" Text="Employer">
    <TreeViewItem NavigateUrl="~/UploadJob.aspx" Text="Upload Job"/>
    <TreeViewItem NavigateUrl="~/EditJob.aspx" Text="Edit Job"/>
    <TreeViewItem NavigateUrl="~/ViewJob.aspx" Text="View Job"/>
  </TreeViewItem>
  <TreeViewItem NavigateUrl="~/Admin.aspx" Text="Admin">
    <TreeViewItem NavigateUrl="~/AddUser.aspx" Text="Add User"/>
    <TreeViewItem NavigateUrl="~/EditUser.aspx" Text="Edit User"/>
    <TreeViewItem NavigateUrl="~/ViewUser.aspx" Text="View User"/>
  </TreeViewItem>
</Items>
```

2. Drag and drop an XmlDataSource control on the webform. Set XPath and DataFile attributes as shown below. Notice that DataFile attribute points to the XML file that we added in Step 1.

```
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
  XPath="/Items/TreeViewItem" DataFile="~/TreeViewData.xml">
</asp:XmlDataSource>
```

3. Drag and drop a TreeView control and set DataSourceID attribute to the xmldatasource control we created in Step 2. Also, set DataBindings as shown below.

```
<asp:TreeView ID="TreeView1" DataSourceID="XmlDataSource1" runat="server">
  <DataBindings>
    <asp:TreeNodeBinding DataMember="TreeViewItem" TextField="Text"
      NavigateUrlField="NavigateUrl" Target="_blank" />
  </DataBindings>
</asp:TreeView>
```

## Binding treeview control to web.sitemap file

We discussed binding menu control to web.sitemap file using sitemapdatasource control. Binding treeview control to web.sitemap file is similar.

Here are the steps to achieve this:

1. Add web.sitemap file. Copy and paste the following XML

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode title="DummyRoot">
    <siteMapNode url="~/Home.aspx" title="Home"/>
    <siteMapNode url="~/Employee.aspx" title="Employee">
      <siteMapNode url="~/UploadResume.aspx" title="Upload Resume"/>
      <siteMapNode url="~/EditResume.aspx" title="Edit Resume"/>
      <siteMapNode url="~/ViewResume.aspx" title="View Resume"/>
    </siteMapNode>
    <siteMapNode url="~/Employer.aspx" title="Employer">
      <siteMapNode url="~/UploadJob.aspx" title="Upload Job"/>
      <siteMapNode url="~/EditJob.aspx" title="Edit Job"/>
      <siteMapNode url="~/ViewJob.aspx" title="View Job"/>
    </siteMapNode>
    <siteMapNode url="~/Admin.aspx" title="Admin">
      <siteMapNode url="~/AddUser.aspx" title="Add User"/>
      <siteMapNode url="~/EditUser.aspx" title="Edit User"/>
      <siteMapNode url="~/ViewUser.aspx" title="View User"/>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

2. Drag and drop "SiteMapDataSource" control onto the webform. By default, a SiteMapDataSource control reads data automatically from web.sitemap file.

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
```

3. Finally drag and drop TreeView control onto the webform and set DataSourceID attribute, to the ID of the SiteMapDataSource control created in Step 2.

```
<asp:TreeView ID="Treeview1" DataSourceID="SiteMapDataSource1" runat="server">
</asp:TreeView>
```

That's it. Run the application and notice that, the TreeView control is bound to web.sitemap file. One problem here is that, the Root node is also displayed.

### Hiding Root node in the TreeView control.

To hide the Root node in the TreeView control, set ShowStartingNode attribute of the SiteMapDataSource control to false.

```
<asp:SiteMapDataSource ShowStartingNode="false" ID="SiteMapDataSource1" runat="server" />
```

To open the target page in a new window set Target attribute of TreeView control to \_blank as shown below.

```
<asp:TreeView ID="Treeview1" Target="_blank" DataSourceID="SiteMapDataSource1"
runat="server">
</asp:TreeView>
```

## Binding asp.net treeview control to database table

We will discuss binding asp.net treeview control to database table.

1. Create the required database table using the script below.

Create table tblTreeViewItems

```
(
  ID int identity primary key,
  TreeViewText nvarchar(50),
  NavigateURL nvarchar(50),
  ParentId int references tblTreeViewItems(ID)
)
-- Insert top level TreeView items
Insert into tblTreeViewItems values ('Home', '~/Home.aspx', NULL)
Insert into tblTreeViewItems values ('Employee', '~/Employee.aspx', NULL)
Insert into tblTreeViewItems values ('Employer', '~/Employer.aspx', NULL)
Insert into tblTreeViewItems values ('Admin', '~/Admin.aspx', NULL)
-- Insert Employee child items
Insert into tblTreeViewItems values ('Upload Resume', '~/UploadResume.aspx', 2)
Insert into tblTreeViewItems values ('Edit Resume', '~/EditResume.aspx', 2)
Insert into tblTreeViewItems values ('View Resume', '~/ViewResume.aspx', 2)
-- Insert Employer child items
Insert into tblTreeViewItems values ('Upload Job', '~/UploadJob.aspx', 3)
Insert into tblTreeViewItems values ('Edit Job', '~/EditJob.aspx', 3)
Insert into tblTreeViewItems values ('View Job', '~/ViewJob.aspx', 3)
-- Insert Admin child items
Insert into tblTreeViewItems values ('Add User', '~/AddUser.aspx', 4)
Insert into tblTreeViewItems values ('Edit User', '~/EditUser.aspx', 4)
Insert into tblTreeViewItems values ('View User', '~/ViewUser.aspx', 4)
```

2. Create a stored procedure that returns data from tblTreeViewItems table.

```
Create proc spGetTreeViewItems
as
Begin
  Select ID, TreeViewText, NavigateURL, ParentId
  from tblTreeViewItems
End
```

3. Drag and drop a treeview control on the webform

```
<asp:TreeView ID="Tree view1" runat="server">
</asp:TreeView>
```

4. Copy and paste the following ado.net code in the code-behind file.

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!IsPostBack)
  {
    GetTreeViewItems();
  }
}
```



```

}

private void GetTreeViewItems()
{
    string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
    SqlConnection con = new SqlConnection(cs);
    SqlDataAdapter da = new SqlDataAdapter("spGetTreeViewItems", con);
    DataSet ds = new DataSet();
    da.Fill(ds);

    ds.Relations.Add("ChildRows", ds.Tables[0].Columns["ID"],
        ds.Tables[0].Columns["ParentId"]);

    foreach (DataRow level1DataRow in ds.Tables[0].Rows)
    {
        if (string.IsNullOrEmpty(level1DataRow["ParentId"].ToString()))
        {
            TreeNode treeNode = new TreeNode();
            treeNode.Text = level1DataRow["TreeViewText"].ToString();
            treeNode.NavigateUrl = level1DataRow["NavigateURL"].ToString();

            DataRow[] level2DataRows = level1DataRow.GetChildRows("ChildRows");
            foreach (DataRow level2DataRow in level2DataRows)
            {
                TreeNode childTreeNode = new TreeNode();
                childTreeNode.Text = level2DataRow["TreeViewText"].ToString();
                childTreeNode.NavigateUrl = level2DataRow["NavigateURL"].ToString();
                treeNode.ChildNodes.Add(childTreeNode);
            }
            Treeview1.Nodes.Add(treeNode);
        }
    }
}

```

Note: Please include the following using declarations.

```

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

```

The above code works as expected only with 2 levels of TreeNode objects. If you include a third level, those TreeNodes will not be displayed in the TreeView control. For example, if you execute the following insert sql script. These rows will not be displayed in the TreeView control.

```

Insert into tblTreeViewItems values ('AAA', '~/AAA.aspx', 5)
Insert into tblTreeViewItems values ('BBB', '~/BBB.aspx', 5)

```

## Dynamically adding treenodes to treeview control in asp.net

We want to bind this data to a TreeView control. If you need the SQL Script to create and populate the table with sample data

ID	TreeViewText	NavigateURL	ParentId
1	Home	~/Home.aspx	NULL
2	Employee	~/Employee.aspx	NULL
3	Employer	~/Employer.aspx	NULL
4	Admin	~/Admin.aspx	NULL
5	Upload Resume	~/UploadResume.aspx	2
6	Edit Resume	~/EditResume.aspx	2
7	View Resume	~/ViewResume.aspx	2
8	Upload Job	~/UploadJob.aspx	3
9	Edit Job	~/EditJob.aspx	3
10	View Job	~/ViewJob.aspx	3
11	Add User	~/AddUser.aspx	4
12	Edit User	~/EditUser.aspx	4
13	View User	~/ViewUser.aspx	4
16	AAA	~/AAA.aspx	5
17	BBB	~/BBB.aspx	5
18	AA1	AA1.aspx	16
19	AA2	AA2.aspx	16
20	BB1	BB1.aspx	17
21	BB2	BB2.aspx	17

This is how the treeview control should look.

- Home
- [-] Employee
  - [-] Upload Resume
    - [-] AAA
      - AA1
      - AA2
    - [-] BBB
      - BB1
      - BB2
  - Edit Resume
  - View Resume
- [-] Employer
  - Upload Job
  - Edit Job
  - View Job
- [-] Admin
  - Add User
  - Edit User
  - View User

## Drag and drop a treeview control

```
<asp:TreeView ID="Treeview1" Target="_blank" runat="server">
</asp:TreeView>
```

Copy and paste the following code in the code-behind file.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace WebFormsDemo
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                GetTreeViewItems();
            }
        }

        private void GetTreeViewItems()
        {
            string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
            SqlConnection con = new SqlConnection(cs);
            SqlDataAdapter da = new SqlDataAdapter("spGetTreeViewItems", con);
            DataSet ds = new DataSet();
            da.Fill(ds);

            ds.Relations.Add("ChildRows", ds.Tables[0].Columns["ID"],
                ds.Tables[0].Columns["ParentId"]);

            foreach (DataRow level1DataRow in ds.Tables[0].Rows)
            {
                if (string.IsNullOrEmpty(level1DataRow["ParentId"].ToString()))
                {
                    TreeNode parentTreeNode = new TreeNode();
                    parentTreeNode.Text = level1DataRow["TreeViewText"].ToString();
                    parentTreeNode.NavigateUrl = level1DataRow["NavigateURL"].ToString();
                    GetChildRows(level1DataRow, parentTreeNode);
                    Treeview1.Nodes.Add(parentTreeNode);
                }
            }
        }
    }
}
```

```
}

private void GetChildRows(DataRow dataRow, TreeNode treeNode)
{
    DataRow[] childRows = dataRow.GetChildRows("ChildRows");
    foreach (DataRow row in childRows)
    {
        TreeNode childTreeNode = new TreeNode();
        childTreeNode.Text = row["TreeViewText"].ToString();
        childTreeNode.NavigateUrl = row["NavigateURL"].ToString();
        treeNode.ChildNodes.Add(childTreeNode);

        if (row.GetChildRows("ChildRows").Length > 0)
        {
            GetChildRows(row, childTreeNode);
        }
    }
}
}
```

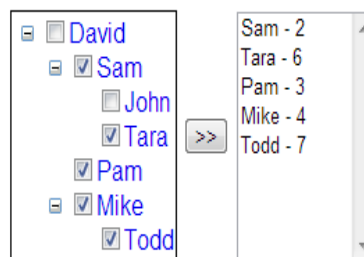
## Displaying organization employee chart using treeview control in asp.net

We will discuss, displaying organization employee chart using TreeView control.

We will use the following table tblEmployee

ID	Name	ManagerId
1	David	NULL
2	Sam	1
3	Pam	1
4	Mike	1
5	John	2
6	Tara	2
7	Todd	4

We want to display above data in a TreeView control as shown below. A check box should be displayed next to every TreeNode. On clicking the button, the selected employees must be added to the listbox.



SQL script to create and populate table tblEmployee

Create table tblEmployee

```
(  
    ID int identity primary key,  
    Name nvarchar(50),  
    ManagerId int foreign key references tblEmployee(ID)  
)
```

```
Insert into tblEmployee values('David', NULL)  
Insert into tblEmployee values('Sam', 1)  
Insert into tblEmployee values('Pam', 1)  
Insert into tblEmployee values('Mike', 1)  
Insert into tblEmployee values('John', 2)  
Insert into tblEmployee values('Tara', 2)  
Insert into tblEmployee values('Todd', 4)
```

Stored procedure to retrieve data from table tblEmployee

Create Proc spGetEmployees

```
as
Begin
Select ID, Name, ManagerId from tblEmployee
End
```

## ASPX HTML

```
<div style="font-family:Arial">
<table>
<tr>
<td style="border:1px solid black">
<asp:TreeView ID="TreeView1" ShowCheckBoxes="All"
runat="server">
</asp:TreeView>
</td>
<td>
<asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text=">>" />
</td>
<td>
<asp:ListBox ID="ListBox1" runat="server" Height="145px"
Width="100px">
</asp:ListBox>
</td>
</tr>
</table>
</div>
```

## ASPX.CS Code:

```
using System;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace WebFormsDemo
{
public partial class WebForm1 : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
if (!IsPostBack)
{
GetTreeViewItems();
}
}

private void GetTreeViewItems()
{
string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
SqlConnection con = new SqlConnection(cs);
```

```

SqlDataAdapter da = new SqlDataAdapter("spGetEmployees", con);
DataSet ds = new DataSet();
da.Fill(ds);

ds.Relations.Add("ChildRows", ds.Tables[0].Columns["ID"],
    ds.Tables[0].Columns["ManagerId"]);

foreach (DataRow level1DataRow in ds.Tables[0].Rows)
{
    if (string.IsNullOrEmpty(level1DataRow["ManagerId"].ToString()))
    {
        TreeNode parentTreeNode = new TreeNode();
        parentTreeNode.Text = level1DataRow["Name"].ToString();
        parentTreeNode.Value = level1DataRow["ID"].ToString();
        GetChildRows(level1DataRow, parentTreeNode);
        Tree View1.Nodes.Add(parentTreeNode);
    }
}

private void GetChildRows(DataRow dataRow, TreeNode treeNode)
{
    DataRow[] childRows = dataRow.GetChildRows("ChildRows");
    foreach (DataRow row in childRows)
    {
        TreeNode childTreeNode = new TreeNode();
        childTreeNode.Text = row["Name"].ToString();
        childTreeNode.Value = row["ID"].ToString();
        treeNode.ChildNodes.Add(childTreeNode);

        if (row.GetChildRows("ChildRows").Length > 0)
        {
            GetChildRows(row, childTreeNode);
        }
    }
}

protected void Button1_Click(object sender, EventArgs e)
{
    ListBox1.Items.Clear();
    GetSelectedTreeNodes(Tree View1.Nodes[0]);
}

private void GetSelectedTreeNodes(TreeNode parentTreeNode)
{
    if (parentTreeNode.Checked)
    {
        ListBox1.Items.Add(parentTreeNode.Text + " - " + parentTreeNode.Value);
    }
    if (parentTreeNode.ChildNodes.Count > 0)
    {
        foreach (TreeNode childTreeNode in parentTreeNode.ChildNodes)

```

```
        {  
            GetSelectedTreeNodes(childTreeNode);  
        }  
    }  
}
```