**Index**

Contents

1. [What is ASP.NET - Part 1 3](#_Toc384387060)
2. [Creating ASP.NET website - Part 2 5](#_Toc384387061)
3. [What is viewstate in asp.net - Part 3 9](#_Toc384387062)
4. [Events in the life cycle of a web application - Part 4 13](#_Toc384387063)

[Types of State management: 16](#_Toc384387064)

[Control State 18](#_Toc384387065)

[Hidden Fields: 18](#_Toc384387066)

1. [Difference between ViewState, Session State and Application State in asp.net - Part 5 20](#_Toc384387067)
2. [Cookies: 24](#_Toc384387068)

[Types of Cookies: 24](#_Toc384387069)

[Use of Cookies: 25](#_Toc384387070)

[Implementation: 26](#_Toc384387071)

[Limitation of cookies 26](#_Toc384387072)

[Cookie Munging: 27](#_Toc384387073)

1. [ASP.NET Page Life Cycle Events - Part 6 27](#_Toc384387074)

[ASP.NET Page Life Cycle Overview 29](#_Toc384387075)

[Data Binding Events for Data-Bound Controls 33](#_Toc384387076)

[Login Control Events 34](#_Toc384387077)

1. [.NET Server control events - Part 7 35](#_Toc384387078)
2. [IsPostBack in asp.net - Part 8 38](#_Toc384387079)

[Internet Information Services and ASP.NET - Part 9 41](#_Toc384387080)

1. [C#.Net How To: Publish a Web Service in IIS using Visual Studio 2010 44](#_Toc384387081)

[Steps to publish a web service in c#.net 44](#_Toc384387082)

1. [ASP.NET TextBox Control - Part 10 48](#_Toc384387083)
2. [ASP.NET Radio Button Control - Part 11 50](#_Toc384387084)
3. [ASP.NET CheckBox Control - Part 12 51](#_Toc384387085)
4. [ASP.NET Hyperlink control - Part 13 54](#_Toc384387086)
5. [ASP.NET Button, LinkButton and ImageButton Controls - Part 14 55](#_Toc384387087)
6. [Command Event of an asp.net button control - Part 15 58](#_Toc384387088)
7. [Dropdownlist in asp.net - Part 16 61](#_Toc384387089)
8. [Data bind asp.net dropdownlist with data from the database - Part 17 63](#_Toc384387090)
9. [Binding an asp.net dropdownlist with an XML file - Part 18 65](#_Toc384387091)
10. [Mapping virtual path to physical path using Server.MapPath method - Part 19 67](#_Toc384387092)
11. [Mapping virtual path to physical path using Server.MapPath method Example - Part 20 70](#_Toc384387093)
12. [21. Retrieving selected item text value and index of dropdownlist 71](#_Toc384387094)
13. [Cascading dropdown in asp.net - Part 22 72](#_Toc384387095)
14. [Asp.net checkboxlist control - Part 23 78](#_Toc384387096)
15. [Fileupload control in asp.net - Part 30 80](#_Toc384387097)

[Adrotator control in asp.net - Part 31 82](#_Toc384387098)

1. [Different page navigation techniques in asp.net - Part 51 84](#_Toc384387099)
2. [Response.Redirect in asp.net - Part 52 85](#_Toc384387100)
3. [Server.Transfer in asp.net - Part 53 86](#_Toc384387101)
4. [Server.execute in asp.net - Part 54 90](#_Toc384387102)
5. [Cross page posting in asp.net - Part 55 94](#_Toc384387103)
6. [Cross page postback strongly typed reference - Part 56 97](#_Toc384387104)
7. [Application pools in IIS - Part 82 99](#_Toc384387105)
8. [Applications isolation using application pools in IIS - Part 83 103](#_Toc384387106)
9. [Application pools in IIS Security - Part 84 105](#_Toc384387107)
10. [Anonymous authentication in asp.net - Part 85 109](#_Toc384387108)
11. [Anonymous authentication and asp.net impersonation - Part 86 111](#_Toc384387109)
12. [Windows authentication in asp.net - Part 87 113](#_Toc384387110)
13. [Windows authentication and authorization in asp.net - Part 88 115](#_Toc384387111)
14. [Windows authentication and folder level authorization - Part 89 116](#_Toc384387112)
15. [Forms authentication using user names list in web.config - Part 90 118](#_Toc384387113)
16. [Forms authentication in asp.net and user registration - Part 91 121](#_Toc384387114)
17. [Forms authentication against users in database table - Part 92 126](#_Toc384387115)
18. [Forms authentication and locking user accounts - Part 93 128](#_Toc384387116)
19. [Unlocking the locked user accounts - Part 94 132](#_Toc384387117)
20. [Implementing password reset link in asp.net - Part 95 134](#_Toc384387118)
21. [Implementing change password page in asp.net - Part 96 138](#_Toc384387119)
22. [Changing password by providing current password - Part 97 144](#_Toc384387120)
23. [Unlocking the locked user accounts using a web page - Part 98 150](#_Toc384387121)
24. [Implementing Enable button to unlock user accounts - Part 99 152](#_Toc384387122)
25. [Secure Socket Layer in asp.net - Part 100 154](#_Toc384387123)
26. [Implementing SSL in asp.net web application - Part 101 156](#_Toc384387124)
27. [Redirect http to https in IIS - Part 102 158](#_Toc384387125)
28. [Redirect http to https in IIS using custom errors - Part 103 159](#_Toc384387126)
29. [User controls in asp.net - Part 104 161](#_Toc384387127)
30. [Adding and using user controls on a webform - Part 105 163](#_Toc384387128)
31. [Raising custom events from user controls - Part 106 165](#_Toc384387129)
32. [Consuming user control custom events - Part 107 170](#_Toc384387130)
33. [Add image slideshow to your website using asp.net ajax and c# - Part 134 172](#_Toc384387131)
34. [Display images in sequence in an image slideshow - Part 135 176](#_Toc384387132)
35. [Provide capability to start and stop image slideshow - Part 136 178](#_Toc384387133)
36. [Add images to slideshow using xml file - Part 137 180](#_Toc384387134)
37. [Add images to slideshow using database table - Part 138 184](#_Toc384387135)

# What is ASP.NET - Part 1

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, whereas 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

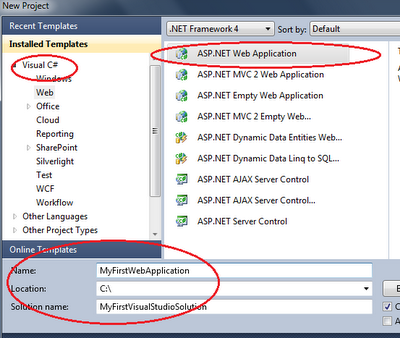
2. On the client all you need is a browser that can understand HTML

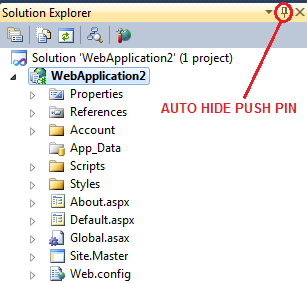
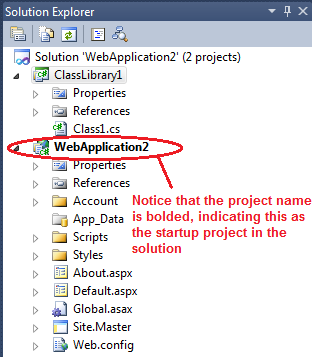
3. On the server side, the Web application runs under Microsoft Internet Information Services (IIS)

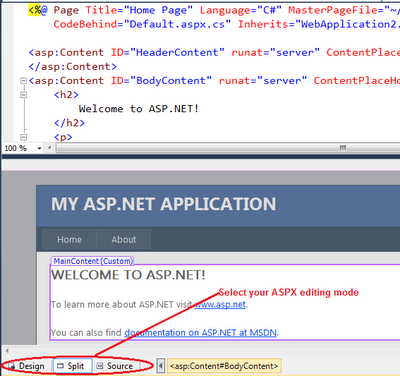
When the client enters the URL of the web application in the browser, and submits the request. The web server which hosts the web application receives the request. The request is then processed by the application. The application generates, the HTML and hands it over to the IIS (web server). Finally, 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 - Part 2

**1.** We will learn about using visual studio  
**2.** Creating your first ASP.NET web application  
**3.** Learn about different windows in visual studio  
  
**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 - Part 3

**In this video session, we will learn about**  
**1.** Stateless nature of HTTP protocol  
**2.** How a webform is processed  
**3.** What is ViewState  
**4.** The role of ViewState in ASP.NET  
**5.** Primary difference between ASP.NET Server controls and HTML controls  
  
**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 a **PostBack** 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 a **PostBack** 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. We will discuss about, all the events in the life cycle of a webform and the order in which they occur in a later session.  
  
**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 - Part 4

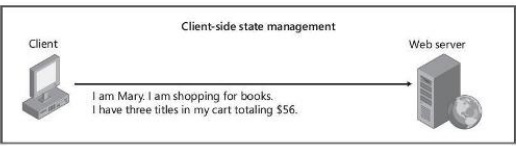
**In a web application, events can occur at 3 levels**  
**1.** At the Application Level(Example: Application Start)  
**2.** At the Page Level(Example: Page Load)  
**3.** At the Control Level (Example: Button Click)  
  
**In this video, we will learn about Application Level events**. Before understanding Application level events, lets talk about Session State and Application State variables. [In Part 3](http://csharp-video-tutorials.blogspot.com/2012/10/what-is-viewstate-in-aspnet-part-3.html) of this video series we have learnt about ViewState. ViewState variables are used to preserve data across page post back. By default, ViewState of one webform is not available in another webform.   
  
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  
  
**We will discuss about Query Strings and Cookies in a later video.**   
  
**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 conatins 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.  
}

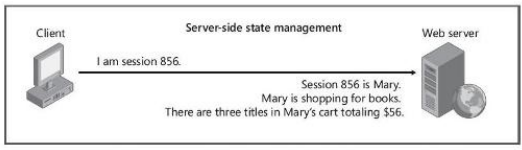
**In general, Application events** are used to initialize data that needs to be available to all the current sessions of the application. Where as **Session events** are used to initialize data that needs to be available only for a given individual session, but not between multiple sessions.  
  
Now,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

Types of State management:

There are mainly two types of state management that ASP.NET provides:

1. Client side state management
2. Server side state management





|  |  |
| --- | --- |
| **Client Side** | **Server-Side** |
| The major benefit of having this kind of state management is that we relieve the server from the burden of keeping the state related information, it saves a lot of server memory. The downside of client side state management is that it takes more bandwidth as considerable amount of data is traveling back and forth. | Server side state management, in contrast to client side, keeps all the information in user memory. |
| But there is one more problem which is bigger than the bandwidth usage problem. The client side state management makes the information travel back and forth and hence this information can be intercepted by anyone in between. So there is no way we can store the sensitive information like passwords, creditcard number and payable amount on client side, we need server side state management for such things. | The downside of this is more memory usage on server and the benefit is that users' confidential and sensitive information is secure. |
| Better scalability  Support for multiple browser | Better security  Reduced bandwidth |

#### Client side state management techniques

* View State
* Control State
* Hidden fields
* Cookies
* Query Strings

#### Server side state management techniques

* Application State
* Session State

Control State

We now know what a viewstate is and we also know that we can disable viewstate for controls on the page. But imagine if we are developing a custom control and we internally are using viewstate to store some information but the user of the control can disable the viewstate for our control. To avoid this problem, we can have viewstate like behavior which cannot be disabled by control users and it is called ControlState. Control states lies inside custom controls and work the same as viewstate works.

To use control state in a custom control, we have to override the OnInit method and call the RegisterRequiresControlState method during initialization. Then we have to override the SaveControlState and LoadControlState methods.

Hidden Fields:

Hidden field are the controls provided by the ASP.NET and they let use store some information in them. The only constraint on hidden filed is that it will keep the information when HTTP post is being done, i.e., button clicks. It will not work with HTTP get. Let us do the same exercise of keeping track of postbacks using HiddenFields now.   
(**Note**: ViewState also uses hidden field underneath.)

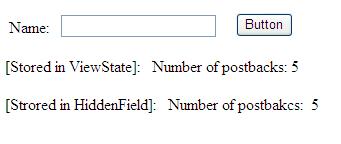
http://www.codeproject.com/images/minus.gifCollapse | [Copy Code](http://www.codeproject.com/Articles/331962/A-Beginner-s-Tutorial-on-ASP-NET-State-Management)

//Store in Hidden Field -----------------------------------------------------------

int newVal = Convert.ToInt32(HiddenField1.Value) + 1; //Hidden field default value was 0

HiddenField1.Value = newVal.ToString();

Label2.Text = HiddenField1.Value;



When we run the page and hit the button to do a postback, the web will show us the postbacks being done so far which is being stored in Hiddenfields (See code for details).

Query Strings:

Query strings are commonly used to store variables that identify specific pages, such as search terms or page numbers. A query string is information that is appended to the end of a page URL. They can be used to store/pass information from one page to another to even the same page. Let us work on storing the postback information in querystrings now:

//GetDataItem from querystring

if (Request.QueryString["number"] != null) //Lets retrieve, increase and store again

{

Label4.Text = Request.QueryString["number"];

}

//set in query string

int postbacks = 0;

if (Request.QueryString["number"] != null) //Lets retrieve, increase and store again

{

postbacks = Convert.ToInt32(Request.QueryString["number"]) + 1;

}

else //First postback, lets store the info

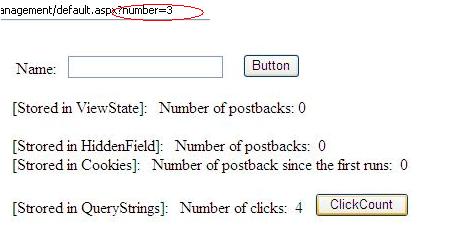
{

postbacks = 1;

}

Response.Redirect("default.aspx?number=" + postbacks);

One thing to notice here is that we can no way store the postback information in the query string we are dealing with same page. The reason is that the query string creates a new URL each time and it will be a fresh request each time we use query strings. SO we are now essentially tracking number of click here. The idea behind query string is to pass small information to OTHER pages that can be used to populate information on that page.



**NOTE**: The use of cookies and querystring here are just for the purpose of demonstration. In real scenarios, they should never be used to store information required for same page. The Querystrings should be used to store the information between multiple page visits. Cookies should be used to store information between multiple visits to our website from the same computer.

# Difference between ViewState, Session State and Application State in asp.net - Part 5

**Suggested videos before continuing with this session**  
[Part 3 - View State in ASP.NET](http://csharp-video-tutorials.blogspot.com/2012/10/what-is-viewstate-in-aspnet-part-3.html)  
[Part 4 - Events in the life cycle of a web application](http://csharp-video-tutorials.blogspot.com/2012/10/events-in-life-cycle-of-web-application.html)  
  
Let's understand the differences, with an example. Create a new asp.net web application.   
  
**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 with in 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 broswer 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.

# Cookies:

Cookies are the small text files that the Web server writes on the client machine when the client's browser accesses their web site. Cookies can be stored in plain text or can be stored in encrypted form.

Since web applications are stateless we need some way to manage the state of the current client request. The state management can be done at server side or client side. Cookies are actually used to identify the users and facilitate the state management.

Note: There are various other state management techniques. Cookies are client side state management techniques and there are other client side state management techniques available too.

## Types of Cookies:

Cookies can be classified into various types based on their lifetime behavior and the domain they are stored for. Major type of cookies is:

**1. Session Cookies**

**2. Persistent Cookies**

**3. Secure Cookies**

**4. Third Party Cookies**

**Session Cookies:** This cookie lives in memory of the client computer and its lifetime depends on the current browser session. If the user closes the browser these cookies are deleted from the client machine. If the user visits the website again after closing the browser these cookies will not be available.

**Persistent Cookies:** Persistent cookies are the cookies that are stored on the secondary storage of the client machine. These cookies do not depend on the browser session. If the user closes the browser and then access the website again then these cookies will still be available. The lifetime of these cookies are specified in cookies itself (as expiration time). The maximum age of such cookies could be 1 year.

**Secure Cookies:** These cookies have an attribute for security. These cookies can only be accessed by the HTTPS connections and not HTTP connections. The reason for having this type of cookie is that it lessen the chances of cookie stealing/eavesdropping (more on this later in the article)

**HttpOnly Cookies:** This mode will allow the cookie to be accessed using HTTP or HTTPS requests. Such cookies will not be accessible by any other methods(JavaScript APIs for instance)

**Third Party Cookies**: First party cookies are the cookies which set the domain of the cookie same as the domain or sub-domain of the website that is being browsed. Third Party Cookies on the other hand are the cookies with domain set to different domain then the website being browsed. These cookies are mainly used for tracking user browsing patterns and/or finding the Advertisement recommendations for the user.

## Use of Cookies:

**State Management (Session Management)**

Web Page Personalization: Web page personalization can also be achieved using cookies. User can set there personalization preferences and these preferences can be saved on server. Using cookies we can identify the same user and then load the personalized version for him.

**Tracking User:**

Cookies are also user to track the user browsing patterns. This is mainly done to identify whether the user is visiting the site for the first time or is he a returning user. Also This is being done to find the Ad recommendations for the user.

**Cookies Attribute:**

**Secure**: When this attribute is specified the cookie can only be accessed over HTTP. This reduces the chances of cookies getting stolen or eavesdropping of cookies.

**Domain and Path**: These two attributes are to identify the web site and the particular URL of that website for which this cookie is being set.

**HTTPOnly:** using this attribute the cookies are forced to be used over HTTP or HTTPS only. This reduced the chances of cross site scripting because the JavaScript APIs will not be able to access cookies.

**Expires:** This attribute specifies whether the cookie is persistent or non persistent. If we don't specify this attribute the cookie will be non persistent i.e. closing the browser will remove the cookie from the browser memory. If this attribute is specified then the cookie will be written on the client machine and will be valid till the time specified in this attribute is reached.

## Implementation:

|  |
| --- |
| HttpCookie cookiename = new HttpCookie("UName");  //Set the cookie value  cookie.Value = TextBox1.Text;  //make it a persistant cookie by setting the expiration time  cookie.Expires = DateTime.Now.AddDays(1);  //Push the cookie to the client computer.  Response.Cookies.Add(cookiename);  Label1.Text = TextBox1.Text;  if (Request.Cookies["UName"] != null)  {  Label1.Text = Request.Cookies["UName"].Value;  }  else  {  Label1.Text = "Guest";  } |

If I want to see the actual cookie file then I can find that in the "C:\Document and Settings\USERNAME\Cookies" folder. To view these files I have to set the folder options to view the hidden files and operating system files.

## Limitation of cookies

• Since cookies data travel over the network between client and server, there are chances that the cookie can be intercepted in between and someone can use it to mimic our session on the server.

• In case of persistent cookies we should never save the sensitive data in cookies as they can be read by anyone.

• Another limitation of using cookies is their size. The browsers often limit the size of the cookie file (4MB in most cases) which is why we should avoid storing large data in cookies.

## Cookie Munging:

• Now we have seen that the most crucial use of cookies in ASP.NET framework is in tracking sessions and implementing Forms authentication. Now what if the user has disabled cookies in his browser.

• If the user has disabled the cookies then the ASP.NET framework uses the URL to keep track of session and authentication data. the unique session ID is then put in the urls and used to track the user session. If the web page contain links i.e. hrefs then the same session ID will also be associated with all the href links. This process in ASP.NET terminology known as cookie munging.

# ASP.NET Page Life Cycle Events - Part 6

**Suggested videos before continuing with this session**  
[Part 3 - Understanding ViewState](http://csharp-video-tutorials.blogspot.com/2012/10/what-is-viewstate-in-aspnet-part-3.html)  
[Part 4 - Events in the life cycle of a web application](http://csharp-video-tutorials.blogspot.com/2012/10/events-in-life-cycle-of-web-application.html)  
[Part 5 - Difference between ViewState, SessionState and ApplicationState](http://csharp-video-tutorials.blogspot.com/2012/10/difference-between-viewstate-session.html)  
  
**In Part 4, of this video series, we have discussed that, 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)  
  
**In this video, we will discuss about events at the page level**. From the previous,parts of this video series, we have learnt that, 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 Page Life Cycle Overview

When an ASP.NET page runs, the page goes through a life cycle in which it performs a series of processing steps. These include initialization, instantiating controls, restoring and maintaining state, running event handler code, and rendering. It is important for you to understand the page life cycle so that you can write code at the appropriate life-cycle stage for the effect you intend.

If you develop custom controls, you must be familiar with the page life cycle in order to correctly initialize controls, populate control properties with view-state data, and run control behavior code. The life cycle of a control is based on the page life cycle, and the page raises many of the events that you need to handle in a custom control.

This topic contains the following sections:

* [General Page Life-cycle Stages](http://msdn.microsoft.com/en-us/library/ms178472.aspx#general_page_lifecycle_stages)
* [Life-cycle Events](http://msdn.microsoft.com/en-us/library/ms178472.aspx#lifecycle_events)
* [Additional Page Life Cycle Considerations](http://msdn.microsoft.com/en-us/library/ms178472.aspx#additional_page_life_cycle_considerations)
* [Catch-Up Events for Added Controls](http://msdn.microsoft.com/en-us/library/ms178472.aspx#catch_up_events_for_added_controls)
* [Data Binding Events for Data-Bound Controls](http://msdn.microsoft.com/en-us/library/ms178472.aspx#data_binding_events_for_databound_controls)
* [Login Control Events](http://msdn.microsoft.com/en-us/library/ms178472.aspx#login_control_events)

[General Page Life-Cycle Stages](javascript:void(0))(SILVER)

In general terms, the page goes through the stages outlined in the following table. In addition to the page life-cycle stages, there are application stages that occur before and after a request but are not specific to a page. For more information, see [Introduction to the ASP.NET Application Life Cycle](http://go.microsoft.com/fwlink/?LinkId=133108) and [ASP.NET Application Life Cycle Overview for IIS 7.0](http://msdn.microsoft.com/en-us/library/bb470252.aspx).

Some parts of the life cycle occur only when a page is processed as a postback. For postbacks, the page life cycle is the same during a partial-page postback (as when you use an [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) control) as it is during a full-page postback.

|  |  |
| --- | --- |
| **Stage** | **Description** |
| Page request | The page request occurs before the page life cycle begins. When the page is requested by a user, ASP.NET determines whether the page needs to be parsed and compiled (therefore beginning the life of a page), or whether a cached version of the page can be sent in response without running the page. |
| Start(S) | In the start stage, page properties such as [Request](http://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) and [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) are set. At this stage, the page also determines whether the request is a postback or a new request and sets the [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) property. The page also sets the [UICulture](http://msdn.microsoft.com/en-us/library/system.web.ui.page.uiculture.aspx) property. |
| Initialization(I) | During page initialization, controls on the page are available and each control's [UniqueID](http://msdn.microsoft.com/en-us/library/system.web.ui.control.uniqueid.aspx) property is set. A master page and themes are also applied to the page if applicable. If the current request is a postback, the postback data has not yet been loaded and control property values have not been restored to the values from view state. |
| Load(L) | During load, if the current request is a postback, control properties are loaded with information recovered from view state and control state. |
| Postback event handling(V-Validate, E-Event Handling) | If the request is a postback, control event handlers are called. After that, the [Validate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.validate.aspx) method of all validator controls is called, which sets the [IsValid](http://msdn.microsoft.com/en-us/library/system.web.ui.ivalidator.isvalid.aspx) property of individual validator controls and of the page. (There is an exception to this sequence: the handler for the event that caused validation is called after validation.) |
| Rendering(R) | Before rendering, view state is saved for the page and all controls. During the rendering stage, the page calls the [Render](http://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method for each control, providing a text writer that writes its output to the [OutputStream](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.outputstream.aspx) object of the page's [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) property. |
| Unload | The [Unload](http://msdn.microsoft.com/en-us/library/system.web.ui.control.unload.aspx) event is raised after the page has been fully rendered, sent to the client, and is ready to be discarded. At this point, page properties such as [Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) and [Request](http://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) are unloaded and cleanup is performed. |

Within each stage of the life cycle of a page, the page raises events that you can handle to run your own code. For control events, you bind the event handler to the event, either declaratively using attributes such as onclick, or in code.

Pages also support automatic event wire-up, meaning that ASP.NET looks for methods with particular names and automatically runs those methods when certain events are raised. If the AutoEventWireup attribute of the [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive is set to true, page events are automatically bound to methods that use the naming convention of Page\_event, such as Page\_Load and Page\_Init. For more information on automatic event wire-up, see [ASP.NET Web Server Control Event Model](http://msdn.microsoft.com/en-us/library/y3bwdsh3.aspx).

The following table lists the page life-cycle events that you will use most frequently. There are more events than those listed; however, they are not used for most page-processing scenarios. Instead, they are primarily used by server controls on the ASP.NET Web page to initialize and render themselves. If you want to write custom ASP.NET server controls, you need to understand more about these events. For information about creating custom controls, see [Developing Custom ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/zt27tfhy.aspx).

|  |  |
| --- | --- |
| **Page Event** | **Typical Use** |
| [PreInit](http://msdn.microsoft.com/en-us/library/system.web.ui.page.preinit.aspx) | Raised after the start stage is complete and before the initialization stage begins.  Use this event for the following:   * Check the [IsPostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) property to determine whether this is the first time the page is being processed. The [IsCallback](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) and [IsCrossPagePostBack](http://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) properties have also been set at this time. * Create or re-create dynamic controls. * Set a master page dynamically. * Set the [Theme](http://msdn.microsoft.com/en-us/library/system.web.ui.page.theme.aspx) property dynamically. * Read or set profile property values.   Note**Note**  If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event. |
| [Init](http://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) | Raised after all controls have been initialized and any skin settings have been applied. The [Init](http://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) event of individual controls occurs before the [Init](http://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) event of the page.  Use this event to read or initialize control properties. |
| [InitComplete](http://msdn.microsoft.com/en-us/library/system.web.ui.page.initcomplete.aspx) | Raised at the end of the page's initialization stage. Only one operation takes place between the [Init](http://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) and [InitComplete](http://msdn.microsoft.com/en-us/library/system.web.ui.page.initcomplete.aspx) events: tracking of view state changes is turned on. View state tracking enables controls to persist any values that are programmatically added to the [ViewState](http://msdn.microsoft.com/en-us/library/system.web.ui.control.viewstate.aspx) collection. Until view state tracking is turned on, any values added to view state are lost across postbacks. Controls typically turn on view state tracking immediately after they raise their [Init](http://msdn.microsoft.com/en-us/library/system.web.ui.control.init.aspx) event.  Use this event to make changes to view state that you want to make sure are persisted after the next postback. |
| [PreLoad](http://msdn.microsoft.com/en-us/library/system.web.ui.page.preload.aspx) | Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the [Request](http://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) instance. |
| [Load](http://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) | The [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls the [OnLoad](http://msdn.microsoft.com/en-us/library/system.web.ui.control.onload.aspx) method on the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object, and then recursively does the same for each child control until the page and all controls are loaded. The [Load](http://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event of individual controls occurs after the [Load](http://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event of the page.  Use the [OnLoad](http://msdn.microsoft.com/en-us/library/system.web.ui.control.onload.aspx) event method to set properties in controls and to establish database connections. |
| Control events | Use these events to handle specific control events, such as a [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) control's [Click](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.click.aspx) event or a [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) control's [TextChanged](http://msdn.microsoft.com/en-us/library/system.web.ui.mobilecontrols.textbox.textchanged.aspx) event.  Note**Note**  In a postback request, if the page contains validator controls, check the [IsValid](http://msdn.microsoft.com/en-us/library/system.web.ui.page.isvalid.aspx) property of the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) and of individual validation controls before performing any processing. |
| [LoadComplete](http://msdn.microsoft.com/en-us/library/system.web.ui.page.loadcomplete.aspx) | Raised at the end of the event-handling stage.  Use this event for tasks that require that all other controls on the page be loaded. |
| [PreRender](http://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) | Raised after the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object has created all controls that are required in order to render the page, including child controls of composite controls. (To do this, the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls [EnsureChildControls](http://msdn.microsoft.com/en-us/library/system.web.ui.control.ensurechildcontrols.aspx) for each control and for the page.)  The [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object raises the [PreRender](http://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event on the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object, and then recursively does the same for each child control. The [PreRender](http://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event of individual controls occurs after the [PreRender](http://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event of the page.  Use the event to make final changes to the contents of the page or its controls before the rendering stage begins. |
| [PreRenderComplete](http://msdn.microsoft.com/en-us/library/system.web.ui.page.prerendercomplete.aspx) | Raised after each data bound control whose [DataSourceID](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.databoundcontrol.datasourceid.aspx) property is set calls its [DataBind](http://msdn.microsoft.com/en-us/library/system.web.ui.control.databind.aspx) method. |
| [SaveStateComplete](http://msdn.microsoft.com/en-us/library/system.web.ui.page.savestatecomplete.aspx) | Raised after view state and control state have been saved for the page and for all controls. Any changes to the page or controls at this point affect rendering, but the changes will not be retrieved on the next postback. |
| [Render](http://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) | This is not an event; instead, at this stage of processing, the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls this method on each control. All ASP.NET Web server controls have a [Render](http://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method that writes out the control's markup to send to the browser.  If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the [Render](http://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method. For more information, see [Developing Custom ASP.NET Server Controls](http://msdn.microsoft.com/en-us/library/zt27tfhy.aspx).  A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code. |
| [Unload](http://msdn.microsoft.com/en-us/library/system.web.ui.control.unload.aspx) | Raised for each control and then for the page.  In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.  For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks.  Note**Note**  During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream. If you attempt to call a method such as the Response.Write method, the page will throw an exception. |

|  |  |
| --- | --- |
|  |  |

## Data Binding Events for Data-Bound Controls

To help you understand the relationship between the page life cycle and data binding events, the following table lists data-related events in data-bound controls such as the [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx), [DetailsView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.detailsview.aspx), and [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx) controls.

|  |  |
| --- | --- |
| **Control Event** | **Typical Use** |
| [DataBinding](http://msdn.microsoft.com/en-us/library/system.web.ui.control.databinding.aspx) | Raised after the control's [PreRender](http://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event, which occurs after the page's [PreRender](http://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) event. (This applies to controls whose [DataSourceID](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.databoundcontrol.datasourceid.aspx) property is set declaratively. Otherwise the event happens when you call the control's DataBind method.)  This event marks the beginning of the process that binds the control to the data. Use this event to manually open database connections, if required, and to set parameter values dynamically before a query is run |
| [RowCreated](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.rowcreated.aspx) ([GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) only) or [ItemCreated](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.itemcreated.aspx) ([DataList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.aspx), [DetailsView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.detailsview.aspx), [SiteMapPath](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sitemappath.aspx), [DataGrid](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datagrid.aspx), [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx), [Repeater](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeater.aspx), and [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) controls) | Raised after the control's [DataBinding](http://msdn.microsoft.com/en-us/library/system.web.ui.control.databinding.aspx) event.  Use this event to manipulate content that is not dependent on data binding. For example, at run time, you might programmatically add formatting to a header or footer row in a [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control. |
| [RowDataBound](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.rowdatabound.aspx) ([GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) only) or [ItemDataBound](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.itemdatabound.aspx) ([DataList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.aspx), [SiteMapPath](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sitemappath.aspx), [DataGrid](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datagrid.aspx), [Repeater](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeater.aspx), and [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) controls) | Raised after the control's [RowCreated](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.rowcreated.aspx) or [ItemCreated](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datalist.itemcreated.aspx) event.  When this event occurs, data is available in the row or item, so you can format data or set the [FilterExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.sqldatasource.filterexpression.aspx) property on child data source controls in order to display related data within the row or item. |
| [DataBound](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basedataboundcontrol.databound.aspx) | Raised at the end of data-binding operations in a data-bound control. In a [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control, data binding is complete for all rows and any child controls.  Use this event to format data-bound content or to initiate data binding in other controls that depend on values from the current control's content. (For more information, see [Catch-Up Events for Added Controls](http://msdn.microsoft.com/en-us/library/ms178472.aspx#catch_up_events_for_added_controls) earlier in this topic.) |

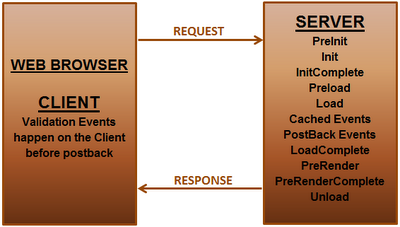
### [Login Control Events](javascript:void(0))

The [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control can use settings in the Web.config file to manage membership authentication automatically. However, if your application requires you to customize how the control works, or if you want to understand how [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control events relate to the page life cycle, you can use the events listed in the following table.

|  |  |
| --- | --- |
| **Control Event** | **Typical Use** |
| [LoggingIn](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.loggingin.aspx) | Raised during a postback, after the page's [LoadComplete](http://msdn.microsoft.com/en-us/library/system.web.ui.page.loadcomplete.aspx) event has occurred. This event marks the beginning of the login process.  Use this event for tasks that must occur prior to beginning the authentication process. |
| [Authenticate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.authenticate.aspx) | Raised after the [LoggingIn](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.loggingin.aspx) event.  Use this event to override or enhance the default authentication behavior of a [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx) control. |
| [LoggedIn](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.loggedin.aspx) | Raised after the user name and password have been authenticated.  Use this event to redirect to another page or to dynamically set the text in the control. This event does not occur if there is an error or if authentication fails. |
| [LoginError](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.loginerror.aspx) | Raised if authentication was not successful.  Use this event to set text in the control that explains the problem or to direct the user to a different page. |

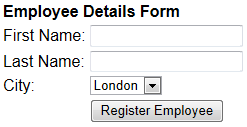
# .NET Server control events - Part 7

**Suggested Videos**  
[Part 4 - Events in the life cycle of a web application](http://csharp-video-tutorials.blogspot.com/2012/10/events-in-life-cycle-of-web-application.html)   
[Part 5 - Difference between ViewState, SessionState and ApplicationState](http://csharp-video-tutorials.blogspot.com/2012/10/difference-between-viewstate-session.html)  
[Part 6 - ASP.NET Page Life Cycle Events](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-page-life-cycle-events-part-6.html)  
  
In the previous parts of this video series, we have discussed that events can occur at application, page and control levels. In [Part 4](http://csharp-video-tutorials.blogspot.com/2012/10/events-in-life-cycle-of-web-application.html), we discussed application level events, and in [Part 6](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-page-life-cycle-events-part-6.html), we discussed about Page Level events.  
  
**In this session, we will discuss about control level 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.   
  
  
  
**In Part 6, of this video series**, 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.

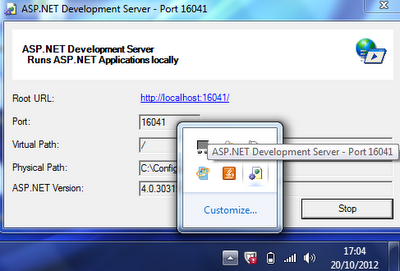
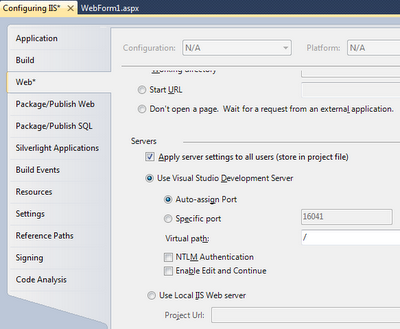


**To understand the order in which Page and Server control events execute**, add a Web form with a **TextBox, RequiredFieldValidator, and a Button control**. You can find **RequiredFieldValidator** under **Validation** tab, in the ToolBox. Double click the TextBox control on the WebForm, and the event 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 - Part 8

**Suggested Videos**  
[Part 3 - ViewState in ASP.NET](http://csharp-video-tutorials.blogspot.com/2012/10/what-is-viewstate-in-aspnet-part-3.html)   
[Part 6 - ASP.NET Page Life Cycle Events](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-page-life-cycle-events-part-6.html)  
[Part 7 - ASP.NET Server Control Events](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-server-control-events-part-7.html)  
  
**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.   
  
  
  
  
  
**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 selecttion 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();**  
**}**

### Internet Information Services and ASP.NET - Part 9

**In this videos we will learn about**  
**1.** What is a Web Server  
**2.** Do you need IIS to develop and test asp.net web applications  
**3.** How to check if IIS is installed?  
**4.** Installing IIS  
**5.** Configuring IIS server to run asp.net web applications  
  
**What is a web server?**  
In simple terms, a web server, is a 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 cick 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

# [C#.Net How To: Publish a Web Service in IIS using Visual Studio 2010](http://www.csharptutorial.in/2012/01/cnet-how-to-publish-web-service-in-iis.html)

Let’s learn to **publish Web Service** in this **C# tutorial**. I am assuming you have already created a **web service** to publish. If not then i would suggest to refer my previous article on [how to create a web service application](http://www.csharptutorial.in/2012/01/cnet-how-to-create-web-service-in-cnet.html). You can also refer another article that explains [how to **consume web service**](http://www.csharptutorial.in/2012/01/cnet-how-to-consume-web-service-in-cnet.html).  
  
**Publishing a web service** in **c# & .net** is very easy task. Follow below steps to **create web service** and **publish a web service in IIS**.

## Steps to publish a web service in c#.net

1. Open your Web Service Application project. Go to **Solution Explorer** and right click on your Web Service Project. Select “**Publish**” from the drop down menu.

|  |
| --- |
| [C# - Publishing web service through .Net IDE](http://3.bp.blogspot.com/-fIffRGyqKpk/UlPkgWyFtMI/AAAAAAAAAK8/lkDxrsh9cqw/s1600/csharptutorial.in_1_Publish_WebSerice.PNG) |
| Publishing web service through .Net IDE |

1. A “Publish Web” window will appear.

|  |
| --- |
| [C# - Publish Web Service Window](http://4.bp.blogspot.com/-toLWisS7YyM/UlPkf2ZVJAI/AAAAAAAAAK0/wc24JSSyxNI/s1600/csharptutorial.in_2_Web_Publish_Window.PNG) |
| C# - Publish Web Service Window |

1. Rename the Publish Profile name from “Profile1” to “***MyFirstWebServiceProfile***”. Select “File System” as Publish Method. And specify the target location to publish the Web Service. I am keeping target location under “***C:\inetpub\wwwroot\MyFirstPublishedWebService***”.

|  |
| --- |
| [C# - Select Profile for Publishing WebService](http://4.bp.blogspot.com/-yaYjTQINLHs/UlPkf0zEQSI/AAAAAAAAAK4/LMmR-O2ERtI/s1600/csharptutorial.in_3_Web_Publish_Profile.PNG) |
| C# - Select Profile for Publishing WebService |

1. Click on **Publish** button to publich the Web Service Application.
2. You will find following files and folder in your published folder.
   1. **bin** folder – this folder contains the .dll file.
   2. **Web.config** file - This is a configuration file.
   3. **Service1.asmx** file
3. Now open IIS (Internet Information Service).

|  |
| --- |
| [IIS - Internet Information Service](http://3.bp.blogspot.com/-7_6wGr-DO70/UlPkhPrho-I/AAAAAAAAALQ/YYmPZeIWlTw/s1600/csharptutorial.in_4_IIS.PNG) |
| IIS - Internet Information Service |

1. Here, you can see your published folder under “**Default Web Site**”. In the next step we would convert it to application format.
2. Right click on the published folder and select “**Convert to Application**”.

|  |
| --- |
| [IIS - Adding Web Service Application](http://1.bp.blogspot.com/-n7SPUyA7060/UlPkgm9-aHI/AAAAAAAAALI/G6IDNTrlhwo/s1600/csharptutorial.in_5_Add_Application_In_IIS.PNG) |
| IIS - Adding Web Service Application |

1. You can provide different Alias and application pool. The application pool must have a Read Access to the physical path.  
   Hitting on Ok button will publish your Web Service in IIS.
2. Now you will need to enable the Directory Browsing of our newly created Web Service Application. While selecting “**MyFirstPublishedWebService**” application double click on “**Directory Browsing**”.

|  |
| --- |
| [IIS - Directory Browsing for WebService](http://3.bp.blogspot.com/-Q9GHlQ2Fkm8/UlPkh97CLaI/AAAAAAAAALg/UCMbCUcwrzY/s1600/csharptutorial.in_6_IIS_Directory_Browsing.PNG) |
| IIS - Directory Browsing for WebService |

1. Click on **enable** to enable the Directory Browsing.

|  |
| --- |
| [C# - IIS - Enable Directory Browsing](http://2.bp.blogspot.com/-uPOrqJsCbxU/UlPkhTfRa5I/AAAAAAAAALY/ir0jlKRRW_U/s1600/csharptutorial.in_7_IIS_Enabling_Directory_Browsing_.PNG) |
| IIS - Enable Directory Browsing |

1. Now restart the IIS to take effect the changes.
2. Now browse *“http://localhost/MyFirstPublishedWebService/Service1.asmx”* in internet browser. And here is your first published web service interface.

In this article you have learn to publish a **web service in IIS using c#**. I hope this articles will meet your expectations. If you find this article helpful, then could you please share the article on your social network?

# ASP.NET TextBox Control - Part 10

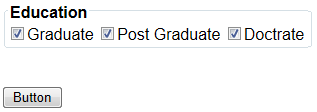
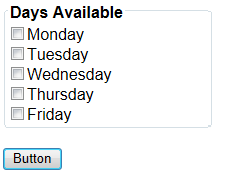
**The TextBox control is used to get the input from the user of the web application.** An asp.net textbox has several properties, that we need to be aware of as a developer.   
  
  
  
**Properties of a TextBox control**  
**1. TextMode Propertry** - SingleLine, MultiLine and Password.  
When you set 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 chatacters 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 - Part 11

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 - Part 12

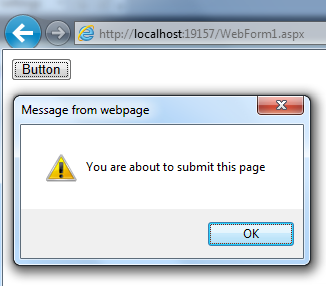
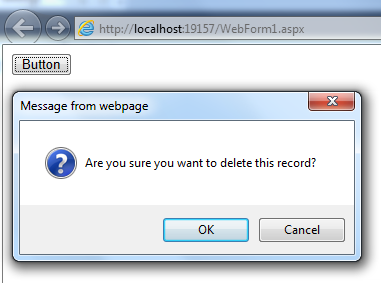
**Suggested Videos:**  
[Part 10 - TextBox Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-textbox-control-part-10.html)  
[Part 11 - RadioButton Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-radio-button-control-part-11.html)  
  
In this video we will learn about the properties, methods and events of an 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="DoctrateCheckBox" Text="Doctrate" 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 (DoctrateCheckBox.Checked)  
{  
sbUserChoices.Append(", " + DoctrateCheckBox.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 - Part 13

**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 - Part 14

**Suggested Videos:**  
[Part 11. RadioButton Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-radio-button-control-part-11.html)  
[Part 12. CheckBox Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-checkbox-control-part-12.html)  
[Part 13. HyperLink Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html)  
  
**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. We will talk about these properties in the next video session. These 3 button controls also support **CuasesValidation** and **ValidationGroup** properties. We will discuss about these properties, when we talk about validation controls in asp.net. We will discuss about **PostBackURL** 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 - Part 15

**Suggested Videos:**  
[Part 12 - CheckBox Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-checkbox-control-part-12.html)  
[Part 13 - HyperLink Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html)  
[Part 14 - Button, LinkButton and ImageButton Controls](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-button-linkbutton-and.html)  
  
**ASP.NET button control exposes 2 events - Click and Command events.** In [Part 14](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-button-linkbutton-and.html), we have discussed about the click event. In this session we will discuss about the Command event. 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.  
**Button1 Click event handled**   
**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. We will discuss about Repeater, GridView, and DataList in a later video session.  
  
**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 programatically 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 - Part 16

**Suggested Videos**  
[Part 13 - HyperLink Control](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html)  
[Part 14 - Button, LinkButton and ImageButton Controls](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-button-linkbutton-and.html)  
[Part 15 - Command Event of an asp.net button control](http://csharp-video-tutorials.blogspot.com/2012/10/command-event-of-aspnet-button-control.html)  
  
**In this video we will learn about**   
**1.** Adding items to the DropDownList control at design time using the HTML  
**2.** Adding items to the DropDownList control at runtime using the code  
  
**Drag and drop a DropDownList control onto the webform.**   
  
**To add items to the DropDownList at deign 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   
  
**In the next video session**, we will discuss about binding the dropdownlist to the data from a database and an XML file.

# Data bind asp.net dropdownlist with data from the database - Part 17

**Suggested Videos**  
[Part 14 - Button, LinkButton and ImageButton Controls](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-button-linkbutton-and.html)  
[Part 15 - Command Event of an asp.net button control](http://csharp-video-tutorials.blogspot.com/2012/10/command-event-of-aspnet-button-control.html)   
[Part 16 - DropDownList in asp.net](http://csharp-video-tutorials.blogspot.com/2012/10/dropdownlist-in-aspnet-part-16.html)   
  
**In this video**, we will learn about, 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 specifes 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 - Part 18

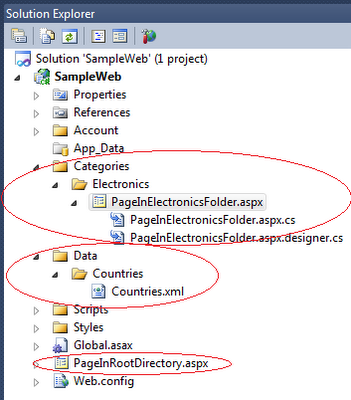
**Suggested Videos**  
[Part 15 - Command Event of an asp.net button control](http://csharp-video-tutorials.blogspot.com/2012/10/command-event-of-aspnet-button-control.html)  
[Part 16 - DropDownList in asp.net](http://csharp-video-tutorials.blogspot.com/2012/10/dropdownlist-in-aspnet-part-16.html)  
[Part 17 - Data bind asp.net dropdownlist with data from the database](http://csharp-video-tutorials.blogspot.com/2012/11/data-bind-aspnet-dropdownlist-with-data.html)   
  
In [**Part 17**](http://csharp-video-tutorials.blogspot.com/2012/11/data-bind-aspnet-dropdownlist-with-data.html) of this video series we have discussed about **binding a DropDownList to data that is retrieved from a database.** In this part, 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**. We will discuss about this method in a later video session.  
  
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 - Part 19

**Suggested Videos**  
[Part 16 - DropDownList in asp.net](http://csharp-video-tutorials.blogspot.com/2012/10/dropdownlist-in-aspnet-part-16.html)  
[Part 17 - Data bind asp.net dropdownlist with data from the database](http://csharp-video-tutorials.blogspot.com/2012/11/data-bind-aspnet-dropdownlist-with-data.html)  
[Part 18 - Binding an asp.net dropdownlist with an XML file](http://csharp-video-tutorials.blogspot.com/2012/11/binding-aspnet-dropdownlist-with-xml.html)   
  
**In this video we will discuss about 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>

**f 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 pysical 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 Example - Part 20

**Suggested Videos**  
[Part 17 - Data bind asp.net dropdownlist with data from the database](http://csharp-video-tutorials.blogspot.com/2012/11/data-bind-aspnet-dropdownlist-with-data.html)  
[Part 18 - Binding an asp.net dropdownlist with an XML file](http://csharp-video-tutorials.blogspot.com/2012/11/binding-aspnet-dropdownlist-with-xml.html)  
[Part 19 - Mapping virtual path to physical path using Server.MapPath method](http://csharp-video-tutorials.blogspot.com/2012/11/mapping-virtual-path-to-physical-path.html)   
  
**In Part 19 of asp.net video series, we discussed about Server.MapPath() method.** In this video, we will continue with a practical application of this method. Please watch Part 19 first, if haven't already done so.   
  
  
  
**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"));**

# [21. Retrieving selected item text value and index of dropdownlist](http://csharp-video-tutorials.blogspot.com/2012/11/retrieving-selected-item-text-value-and.html)

# Cascading dropdown in asp.net - Part 22

**Suggested Videos**  
[Part 16 - Dropdownlist in asp.net](http://csharp-video-tutorials.blogspot.com/2012/10/dropdownlist-in-aspnet-part-16.html)  
[Part 17 - Data bind dropdownlist with data from the database](http://csharp-video-tutorials.blogspot.com/2012/11/data-bind-aspnet-dropdownlist-with-data.html)  
[Part 21 - Retrieving selected item text, value and index of the dropdownlist](http://csharp-video-tutorials.blogspot.com/2012/11/retrieving-selected-item-text-value-and.html)   
  
**In this video we will discuss about cascading dropdownlists.** 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 dropsownlist 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, 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);  
}  
}

[http://img1.blogblog.com/img/icon18_wrench_allbkg.png](http://www.blogger.com/rearrange?blogID=6082652835152798567&widgetType=HTML&widgetId=HTML2&action=editWidget&sectionId=sidebar-right-1)

|  |  |
| --- | --- |
|  |  |

# Asp.net checkboxlist control - Part 23

**Suggested Videos**   
[Part 16 - Dropdownlist in asp.net](http://csharp-video-tutorials.blogspot.com/2012/10/dropdownlist-in-aspnet-part-16.html)  
[Part 17 - Data bind dropdownlist with data from the database](http://csharp-video-tutorials.blogspot.com/2012/11/data-bind-aspnet-dropdownlist-with-data.html)  
[Part 21 - Retrieving selected item text, value and index of the dropdownlist](http://csharp-video-tutorials.blogspot.com/2012/11/retrieving-selected-item-text-value-and.html)   
  
**In this video we will learn about asp.net checkboxlist control. Just like DropDownList**  
**1.** CheckBoxList is collection of ListItem objects.  
**2.** Items can be added to the CheckBoxList in the HTML source or in the code behind file  
**3.** CheckBoxList 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 CheckBoxList control can be used.  
  
**Create an asp.net web application. Copy and paste the following HTML**  
<asp:CheckBoxList ID="checkboxListEducation" 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>  
<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 checkboxListEducation.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 = " + checkboxListEducation.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:CheckBoxList ID="checkkboxListEducation" 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 CheckBoxList 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);**  
**}**

# Fileupload control in asp.net - Part 30

**Suggested Videos**  
[Part 19 - Mapping virtual path to physical path using Server.MapPath method](http://csharp-video-tutorials.blogspot.com/2012/11/mapping-virtual-path-to-physical-path.html)  
[Part 20 - Mapping virtual path to physical path using Server.MapPath method Example](http://csharp-video-tutorials.blogspot.com/2012/11/mapping-virtual-path-to-physical-path_6.html)   
  
**In this video 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 seleceted 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" />**  
**&nbsp;**  
**<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 - Part 31

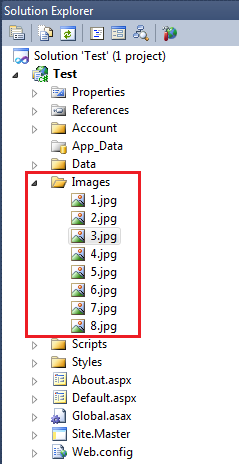
**Suggested Videos**  
[Part 28 - Bulleted list in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/bulleted-list-in-aspnet-part-28.html)  
[Part 29 - List controls in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/list-controls-in-aspnet-part-29.html)  
[Part 30 - Fileupload control in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/fileupload-control-in-aspnet-part-30.html)   
  
**In this video 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. In this video 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.**   
  
  
  
  
  
**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. More on this, when we discuss about master pages in a later video session.

# Different page navigation techniques in asp.net - Part 51

**Suggested Videos**  
[Part 48 - CustomValidator](http://csharp-video-tutorials.blogspot.com/2012/11/customvalidator-control-in-aspnet-part.html)  
[Part 49 - ValidationSummary](http://csharp-video-tutorials.blogspot.com/2012/11/validationsummary-control-in-aspnet.html)  
[Part 50 - ValidationGroups](http://csharp-video-tutorials.blogspot.com/2012/11/validationgroups-in-aspnet-part-50.html)   
  
**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. We have discussed in detail about the HyperLink control in Part 13 of ASP.NET video series.   
**2. Response.Redirect**   
**3. Server.Transfer**   
**4. Server.Execute**  
**5. Cross-Page postback**  
**6. Window.Open**   
  
We will discuss about the rest of the page navigation techniques in the subsequent videos.

|  |  |
| --- | --- |
| Response.Redirect in asp.net - Part 52 **Suggested Videos** [Part 49 - ValidationSummary](http://csharp-video-tutorials.blogspot.com/2012/11/validationsummary-control-in-aspnet.html) [Part 50 - ValidationGroups](http://csharp-video-tutorials.blogspot.com/2012/11/validationgroups-in-aspnet-part-50.html) [Part 51 - Different page navigation techniques](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html)   **The following are the different page navigation techniques in asp.net**  **1.** Hyperlink control - Discussed in [**Part 13**](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html) and [**Part 51**](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) of the [**ASP.NET video series**](http://www.youtube.com/playlist?list=PL6n9fhu94yhXQS_p1i-HLIftB9Y7Vnxlo&feature=view_all) **2.** Response.Redirect  **3.** Server.Transfer  **4.** Server.Execute **5.** Cross-Page postback **6.** Window.Open     **In this video, We will discuss about Response.Redirect.** 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.  Response.Redirect in asp.net  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 - Part 53 **Suggested Videos** [Part 50 - ValidationGroups](http://csharp-video-tutorials.blogspot.com/2012/11/validationgroups-in-aspnet-part-50.html) [Part 51 - Different page navigation techniques](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) [Part 52 - Response.Redirect in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html)   **The following are the different page navigation techniques in asp.net** **1.** Hyperlink control - Discussed in [Part 13](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html) and [Part 51](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) of the [ASP.NET video series](http://www.youtube.com/playlist?list=PL6n9fhu94yhXQS_p1i-HLIftB9Y7Vnxlo) **2.** Response.Redirect - Discussed in [Part 52](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html) of the [ASP.NET video series](http://www.youtube.com/playlist?list=PL6n9fhu94yhXQS_p1i-HLIftB9Y7Vnxlo) **3.** Server.Transfer  **4.** Server.Execute **5.** Cross-Page postback **6.** Window.Open   **In this video, We will discuss about**  **1.** Server.Transfer **2.** Difference between Server.Transfer and Response.Redirect     **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.  server.transfer vs response.redirect Server.execute in asp.net - Part 54 **Suggested Videos** [Part 51 - Different page navigation techniques](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) [Part 52 - Response.Redirect in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html) [Part 53 - Server.Transfer in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/servertransfer-in-aspnet-part-53.html)   **The following are the different page navigation techniques in asp.net** **1.** Hyperlink control - Discussed in [Part 13](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html) and [Part 51](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) of the ASP.NET video series **2.** Response.Redirect - Discussed in [Part 52](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html) **3.** Server.Transfer - Discussed in [Part 53](http://csharp-video-tutorials.blogspot.com/2012/11/servertransfer-in-aspnet-part-53.html) **4.** Server.Execute **5.** Cross-Page postback **6.** Window.Open     **In this video we will discuss about**  **1.** Server.Execute method **2.** Difference between Server.Transfer and Server.Execute  **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 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> <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 - Part 55 **Suggested Videos** [Part 52 - Response.Redirect in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html) [Part 53 - Server.Transfer in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/servertransfer-in-aspnet-part-53.html) [Part 54 - Server.execute in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/serverexecute-in-aspnet-part-54.html)   **The following are the different page navigation techniques in asp.net** **1.** Hyperlink control - Discussed in [Part 13](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html) and [Part 51](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) of the [ASP.NET video series](http://www.youtube.com/playlist?list=PL6n9fhu94yhXQS_p1i-HLIftB9Y7Vnxlo) **2.** Response.Redirect - Discussed in [Part 52](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html) **3.** Server.Transfer - Discussed in [Part 53](http://csharp-video-tutorials.blogspot.com/2012/11/servertransfer-in-aspnet-part-53.html) **4.** Server.Execute - Discussed in [Part 54](http://csharp-video-tutorials.blogspot.com/2012/11/serverexecute-in-aspnet-part-54.html) **5.** Cross-Page postback **6.** Window.Open     **In this video we will discuss about Cross page posting.** 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> <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 NullRefernceException. In the next video we will discuss about obtaining a strongly typed reference to the previous page, which can avoid NullRefernceExceptions Cross page postback strongly typed reference - Part 56 **Suggested Videos** [Part 53 - Server.Transfer in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/servertransfer-in-aspnet-part-53.html) [Part 54 - Server.execute in asp.net](http://csharp-video-tutorials.blogspot.com/2012/11/serverexecute-in-aspnet-part-54.html) [Part 55 - Cross page postback](http://csharp-video-tutorials.blogspot.com/2012/11/cross-page-posting-in-aspnet-part-55.html)   **The following are the different page navigation techniques in asp.net** **1.** Hyperlink control - Discussed in [Part 13](http://csharp-video-tutorials.blogspot.com/2012/10/aspnet-hyperlink-control-part-13.html) and [Part 51](http://csharp-video-tutorials.blogspot.com/2012/11/different-page-navigation-techniques-in.html) of the [ASP.NET video series](https://www.youtube.com/playlist?list=PL6n9fhu94yhXQS_p1i-HLIftB9Y7Vnxlo&feature=view_all) **2.** Response.Redirect - Discussed in [Part 52](http://csharp-video-tutorials.blogspot.com/2012/11/responseredirect-in-aspnet-part-52.html) **3.** Server.Transfer - Discussed in [Part 53](http://csharp-video-tutorials.blogspot.com/2012/11/servertransfer-in-aspnet-part-53.html) **4.** Server.Execute - Discussed in [Part 54](http://csharp-video-tutorials.blogspot.com/2012/11/serverexecute-in-aspnet-part-54.html) **5.** Cross-Page postback - Discussed in [Part 55](http://csharp-video-tutorials.blogspot.com/2012/11/cross-page-posting-in-aspnet-part-55.html) **6.** Window.Open     **In Part 55, we have discussed the basics of cross page posting.** In the previous session, 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.  **In this session, we will discuss about, obtaining a strongly types reference to the previous page**. There are 2 ways to obtain a strongly typed reference. We will explore both of these options in this part of the video. We will be using the same example used in Part 55.   **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 stronly typed reference //WebForm1 previousPage = this.PreviousPage; //PreviousPage also returns a stronly 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. Application pools in IIS - Part 82 **Suggested Videos** [Part 79 - Tracing in asp.net](http://csharp-video-tutorials.blogspot.com/2012/12/tracing-in-aspnet-part-79.html) [Part 80 - Writing custom asp.net tracing messages](http://csharp-video-tutorials.blogspot.com/2012/12/writing-custom-aspnet-tracing-messages.html) [Part 81 - Tracing in asp.net - A real time example](http://csharp-video-tutorials.blogspot.com/2012/12/tracing-in-aspnet-real-time-example.html)   **In this video we will discuss about** **1.** What are application pools in IIS **2.** Creating application pools in internet information services(IIS) **3.** Application pool identities **4.** Associating an ASP.NET Web Application with an Application Pool     **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 loose 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 idenity. The application pool identity can be any of the following built in aaccounts **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"  Changing application pool identity  **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 previlaged 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.  Associating an ASP.NET Web Application with an Application Pool Applications isolation using application pools in IIS - Part 83 **Suggested Videos** [Part 80 - Writing custom asp.net tracing messages](http://csharp-video-tutorials.blogspot.com/2012/12/writing-custom-aspnet-tracing-messages.html) [Part 81 - Tracing in asp.net - A real time example](http://csharp-video-tutorials.blogspot.com/2012/12/tracing-in-aspnet-real-time-example.html) [Part 82 - Application pools in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-part-82.html)   **In this video 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 it's session data.  **In the next video session, we will discuss about configuring different levels of security for different application pools.** Application pools in IIS Security - Part 84 **Suggested Videos** [Part 81 - Tracing in asp.net - A real time example](http://csharp-video-tutorials.blogspot.com/2012/12/tracing-in-aspnet-real-time-example.html) [Part 82 - Application pools in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-part-82.html) [Part 83 - Applications isolation using application pools](http://csharp-video-tutorials.blogspot.com/2012/12/applications-isolation-using.html)   **In this video, 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. We have discussed about creating application pools and associating web applications to an application pool in [**Part 82**](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-part-82.html) and [**Part 83**](http://csharp-video-tutorials.blogspot.com/2012/12/applications-isolation-using.html).  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 occured, please contact administrator"; } } else { Label1.Text = "Please select a file first"; } } Anonymous authentication in asp.net - Part 85 **Suggested Videos** [Part 82 - Application pools in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-part-82.html) [Part 83 - Applications isolation using application pools](http://csharp-video-tutorials.blogspot.com/2012/12/applications-isolation-using.html) [Part 84 - Application pools in IIS Security](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-security-part.html)   **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. For help on these topics, please check the following parts [Part 82 - Application pool in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-part-82.html) [Part 83 - Applications isolation using application pools in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/applications-isolation-using.html)  **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, dobule 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. In the next video session, 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 - Part 86 **Suggested Videos** [Part 83 - Applications isolation using application pools](http://csharp-video-tutorials.blogspot.com/2012/12/applications-isolation-using.html) [Part 84 - Application pools in IIS Security](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-security-part.html) [Part 85 - Anonymous authentication](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-in-aspnet-part.html)   **Please watch Part 85**, before watching this video. In Part 85, 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 this video, we will discuss the effects of turning impersonation on, with anonymous access.     **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 - Part 87 **Suggested Videos** [Part 84 - Application pools in IIS Security](http://csharp-video-tutorials.blogspot.com/2012/12/application-pools-in-iis-security-part.html) [Part 85 - Anonymous authentication](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-in-aspnet-part.html) [Part 86 - Anonymous authentication and asp.net impersonation](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-and-aspnet.html)   **In Parts 85 and 86 of this video series, 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.     **In this session, 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.  We will be using the same project that we worked with, in Part 86.   **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 - Part 88 **Suggested Videos** [Part 85 - Anonymous authentication](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-in-aspnet-part.html) [Part 86 - Anonymous authentication and asp.net impersonation](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-and-aspnet.html) [Part 87 - Windows authentication](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-in-aspnet-part-87.html)   **In Part 87, we have discussed the basics of windows authentication.** In this session, we will continue to discuss about windows authentication. Please watch Part 87, before proceeding.  **? 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 - Part 89 **Suggested Videos** [Part 86 - Anonymous authentication and asp.net impersonation](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-and-aspnet.html) [Part 87 - Windows authentication](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-in-aspnet-part-87.html) [Part 88 - Windows authentication and authorization](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-and.html)   **Please watch Parts 87 and 88**, before proceeding. In this video we will discuss about folder level authorization, with an example. Consider the project structure, shown in the solution explorer below. Folder level authorization    **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, where as 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 - Part 90 **Suggested Videos** [Part 87 - Windows authentication](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-in-aspnet-part-87.html) [Part 88 - Windows authentication and authorization](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-and.html) [Part 89 - Windows authentication and folder level authorization](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-and-folder-level.html)  **Anonymous authentication is fine for web sites that contain public information** that every one can see. We discussed about Anonymous authentication in [Part 85 - Anonymous authentication](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-in-aspnet-part.html) [Part 86 - Anonymous authentication and impersonation](http://csharp-video-tutorials.blogspot.com/2012/12/anonymous-authentication-and-aspnet.html)  **Windows authentication is used for intranet web applications**, where the users are part of a windows domain-based network. We discussed about Windows authentication in **Parts 87, 88 and 89**.     **In this video we will discuss about** **1.** When to use Forms Authentication **2.** How to enable Forms Authentication  **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> <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 againts 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. In a later video session, we will discuss about storing user names and passwords in a database table.  **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. In a later video session, we will discuss about this. Forms authentication in asp.net and user registration - Part 91 **Suggested Videos** [Part 88 - Windows authentication and authorization](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-and.html) [Part 89 - Windows authentication and folder level authorization](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-and-folder-level.html) [Part 90 - Forms authentication using user names list in web.config](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-using-user-names.html)     **In this code sample, we have used validation controls and ADO.NET**. If you have not watched the videos on validation controls and ADO.NET, I would strongly encourage you to do so, before continuing with this session.  **Please watch Part - 90**, before proceeding. In Part - 90, we have discussed the basics of Forms authentication. One of the problems, with the example in Part 90, 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>  CREATE 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  INSERT INTO tblUsers VALUES ( @UserName ,@Password ,@Email ) END  SELECT @ReturnCode AS ReturnValue END **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 calss 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. In the next video session, we will discuss about, authenticating with the credentials we stored in the database. Forms authentication against users in database table - Part 92 **Suggested Videos** [Part 89 - Windows authentication and folder level authorization](http://csharp-video-tutorials.blogspot.com/2012/12/windows-authentication-and-folder-level.html) [Part 90 - Forms authentication using user names list in web.config](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-using-user-names.html) [Part 91 - Forms authentication and user registration](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-in-aspnet-and-user.html)   **In Part 90, we have discussed about authenticating users against a list stored in web.config file**. In Part 91, we have discussed about, registering users, if they do not have a username and password to log in. In this session, we will disuss about authenticating users against a list stored in a database table.  **This is continuation to Part 91**. Please watch Part 91, before proceeding with this video. 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 - Part 93 **Suggested Videos** [Part 90 - Forms authentication using user names list in web.config](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-using-user-names.html) [Part 91 - Forms authentication and user registration](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-in-aspnet-and-user.html) [Part 92 - Forms authentication against users in database table](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-against-users-in.html)   **Please watch Parts 90, 91 and 92 before proceeding.** In this video 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, that we have created in Part 90**. 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'** that we created in Part 91, 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', that we created in Part 92**, 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  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);  **In the next video session, we will discuss about enabling the disabled user accounts.** Unlocking the locked user accounts - Part 94 **Suggested Videos** [Part 91 - Forms authentication and user registration](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-in-aspnet-and-user.html) [Part 92 - Forms authentication against users in database table](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-against-users-in.html) [Part 93 - Forms authentication and locking user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-and-locking-user.html)   **In Part 93, of this video series** we have discussed about locking user accounts, if a user repeatedly enters the wrong password. The accounts are locked to prevent hackers from guessing passwords and dictionary attacks. Please watch Part 93, before proceeding with this video.     **In this video, 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 in-efficient. If you know how to write basic ADO.NET code, this approach should not be very difficult to achieve. [If you are new to ADO.NET, Click here for a video series that I have recorded on ADO.NET](http://www.youtube.com/playlist?list=PL6n9fhu94yhX5dzHunAI2t4kE0kOuv4D7)  **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.   In this video, we will discuss about creating and scheduling the SQL Server Job to unlock user accounts.  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. [If you are new to DateTime functions in SQL Server, please check this video by clicking here.](http://www.youtube.com/watch?v=eYsizQVa_EU)  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. In this video, we will discuss about creating and scheduling sql server agent jobs, for sql server 2008. **1.** Open sql serevr 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 **"Recurs 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 - Part 95 **Suggested Videos** [Part 92 - Forms authentication against users in database table](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-against-users-in.html) [Part 93 - Forms authentication and locking user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-and-locking-user.html) [Part 94 - Unlocking the locked user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-part-94.html)     **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. We discussed about having multiple web.config files and allowing anonymous access to a set of pages in Part 91 of this video series. [Click here to watch Part 91, before proceeding.](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-in-aspnet-and-user.html)     **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 is 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. In the next video session we will implement ChangePassword page. <h1>Change Password Page</h1> Implementing change password page in asp.net - Part 96 **Suggested Videos** [Part 93 - Forms authentication and locking user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/forms-authentication-and-locking-user.html) [Part 94 - Unlocking the locked user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-part-94.html) [Part 95 - Implementing password reset link](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-password-reset-link-in.html)   **In this video we will discuss about**, implementing change password page in asp.net. When the user clicks on password reset link, the user lands on ChangePassword.aspx page. In Part 95, 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.     http://4.bp.blogspot.com/--GO7wDfdhJY/UNodEBe3LeI/AAAAAAAAA8U/aTFiMlSqGQk/s1600/tblUsers.png  http://2.bp.blogspot.com/-iqLvkttPLrk/UNodQBF1dZI/AAAAAAAAA8g/p7uJpIK6fD4/s1600/tblResetPasswordRequests.png  **Stored Procedure to check, if the password reset link, is a valid link.** Create Proc spIsPasswordResetLinkValid  @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); }  **In the next video**, we will discuss about changing password by providing the current password. Changing password by providing current password - Part 97 **Suggested Videos** [Part 94 - Unlocking the locked user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-part-94.html) [Part 95 - Implementing password reset link](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-password-reset-link-in.html) [Part 96 - Implementing change password page](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-change-password-page-in.html)   **In this video 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 - Part 98 **Suggested Videos** [Part 95 - Implementing password reset link](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-password-reset-link-in.html) [Part 96 - Implementing change password page](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-change-password-page-in.html) [Part 97 - Changing password by providing current password](http://csharp-video-tutorials.blogspot.com/2012/12/changing-password-by-providing-current.html)   **If a user repeatedly enters the wrong password.** The accounts are locked to prevent hackers from guessing passwords and making dictionary attacks. In Part 94, of this video series we have discussed about un-locking user accounts, using a SQL Server agent job. [Please watch Part 94, before proceeding with this video.](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-part-94.html)     **In this video, 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 in-efficient.  **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 &amp; 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("spGetAllLocakedUserAccounts", con); cmd.CommandType = CommandType.StoredProcedure;  con.Open(); gvLockedAccounts.DataSource = cmd.ExecuteReader(); gvLockedAccounts.DataBind(); } }  **In the next video session, we will discuss about implementing the "Enable" button.** Implementing Enable button to unlock user accounts - Part 99 **Suggested Videos** [Part 96 - Implementing change password page](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-change-password-page-in.html) [Part 97 - Changing password by providing current password](http://csharp-video-tutorials.blogspot.com/2012/12/changing-password-by-providing-current.html) [Part 98 - Unlocking the locked user accounts using a web page](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-using.html)   [**This is continuation to Part 98**.](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-using.html) 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 &amp; 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 - Part 100 **Suggested Videos** [Part 97 - Changing password by providing current password](http://csharp-video-tutorials.blogspot.com/2012/12/changing-password-by-providing-current.html) [Part 98 - Unlocking the locked user accounts using a web page](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-using.html) [Part 99 - Implementing Enable button to unlock user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-enable-button-to-unlock.html)   **In this video 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. We will discuss about IIS configuration in a later video session.   **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.  http://3.bp.blogspot.com/-bdoOvjQWXec/UNzObY5vxgI/AAAAAAAAA9M/woKVXlfyS7k/s1600/server%2Bcertificate%2Berror.gif  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. We will discuss about generating and installing server certificates in our next video session.  **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 - Part 101 **Suggested Videos** [Part 98 - Unlocking the locked user accounts using a web page](http://csharp-video-tutorials.blogspot.com/2012/12/unlocking-locked-user-accounts-using.html) [Part 99 - Implementing Enable button to unlock user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-enable-button-to-unlock.html) [Part 100 - Secure Socket Layer in asp.net](http://csharp-video-tutorials.blogspot.com/2012/12/secure-socket-layer-in-aspnet-part-100.html)   **In this video 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/bfsktky3.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. In a later video session we will discuss about resolving this.  **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 - Part 102 **Suggested Videos** [Part 99 - Implementing Enable button to unlock user accounts](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-enable-button-to-unlock.html) [Part 100 - Secure Socket Layer in asp.net](http://csharp-video-tutorials.blogspot.com/2012/12/secure-socket-layer-in-aspnet-part-100.html) [Part 101 - Implementing SSL in asp.net web application](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-ssl-in-aspnet-web.html)   **In this video we will discuss about redirecting users from HTTP to HTTPS**. In the previous session, we discussed about implementing SSL in asp.net. [Please watch Part 101](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-ssl-in-aspnet-web.html), before proceeding with this video.     **To redirect users from HTTP to HTTPS automatically**, there are several ways. In this video we will discuss about using "url rewrite" module. There are 3 simple steps  **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  In the next video we will discuss about using "IIS Error Pages" to redirect users from HTTP to HTTPS automatically. Redirect http to https in IIS using custom errors - Part 103 **Suggested Videos** [Part 100 - Secure Socket Layer in asp.net](http://csharp-video-tutorials.blogspot.com/2012/12/secure-socket-layer-in-aspnet-part-100.html) [Part 101 - Implementing SSL in asp.net web application](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-ssl-in-aspnet-web.html) [Part 102 - Redirect http to https in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/redirect-http-to-https-in-iis-part-102.html)   **In this video we will discuss about redirecting users from HTTP to HTTPS**, using "IIS Error Pages". In the previous session, 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 - Part 104 **Suggested Videos** [Part 101 - Implementing SSL in asp.net web application](http://csharp-video-tutorials.blogspot.com/2012/12/implementing-ssl-in-aspnet-web.html) [Part 102 - Redirect http to https in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/redirect-http-to-https-in-iis-part-102.html) [Part 103 - Redirect http to https in iis using custom errors](http://csharp-video-tutorials.blogspot.com/2012/12/redirect-http-to-https-in-iis-using.html)   **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.  http://4.bp.blogspot.com/-svJ1SKwwQ4k/UOIY_Az7GEI/AAAAAAAAA9w/0J8rphDI_Gw/s1600/User%2Bcontrols%2Bin%2Basp.net.png    **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. [We discussed about this in Part 32 of the asp.net video series.](http://csharp-video-tutorials.blogspot.com/2012/11/aspnet-calendar-control-part-32.html)  **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.  In this demo, 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**. In the next video, we will discuss about using this calendar control on a web form. Adding and using user controls on a webform - Part 105 **Suggested Videos** [Part 102 - Redirect http to https in IIS](http://csharp-video-tutorials.blogspot.com/2012/12/redirect-http-to-https-in-iis-part-102.html) [Part 103 - Redirect http to https in iis using custom errors](http://csharp-video-tutorials.blogspot.com/2012/12/redirect-http-to-https-in-iis-using.html) [Part 104 - Creating user controls](http://csharp-video-tutorials.blogspot.com/2012/12/user-controls-in-aspnet-part-104.html)   **In the previous video**, we discussed about creating a calendar user control. Please watch Part 104, before proceeding with this video. In this video 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.  We will be discussing about these in later video sessions.  In the next video session we will discuss about adding "events" to our "CalendarUserControl" Raising custom events from user controls - Part 106 **Suggested Videos** [Part 103 - Redirect http to https in iis using custom errors](http://csharp-video-tutorials.blogspot.com/2012/12/redirect-http-to-https-in-iis-using.html) [Part 104 - Creating user controls](http://csharp-video-tutorials.blogspot.com/2012/12/user-controls-in-aspnet-part-104.html) [Part 105 - Using user controls on a webform](http://csharp-video-tutorials.blogspot.com/2012/12/adding-and-using-user-controls-on.html)   **In this video we will discuss about**  **1.** Adding events to UserControls **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. To get the most out of this video, I strongly recommend to [watch parts 36, 37 , 38 and 39 from C# Video series](http://www.youtube.com/playlist?list=PLAC325451207E3105), and [parts 104 and 105 from asp.net video series](http://www.youtube.com/playlist?list=PL6n9fhu94yhXQS_p1i-HLIftB9Y7Vnxlo), before proceeding with this video.   **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 **overridde** 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. This will be much clear in the next video session, when we discuss about consuming CalendarVisibilityChanged event. 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 and instance of "CalendarVisibilityChangedEventArgs" and pass event data, that is "true" or "false" to the contrustor 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 calendarVisibilityChangedEventData =  new CalendarVisibilityChangedEventArgs(true); OnCalendarVisibilityChanged(calendarVisibilityChangedEventData); } }  protected void Calendar1\_SelectionChanged(object sender, EventArgs e) { txtDate.Text = Calendar1.SelectedDate.ToShortDateString(); Calendar1.Visible = false; CalendarVisibilityChangedEventArgs calendarVisibilityChangedEventData =  new CalendarVisibilityChangedEventArgs(false); OnCalendarVisibilityChanged(calendarVisibilityChangedEventData); }  **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 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(); 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);  **In the next video, we will discuss about consuming CalendarVisibilityChangedEvent.** Consuming user control custom events - Part 107 **Suggested Videos** [Part 104 - Creating user controls](http://csharp-video-tutorials.blogspot.com/2012/12/user-controls-in-aspnet-part-104.html) [Part 105 - Using user controls on a webform](http://csharp-video-tutorials.blogspot.com/2012/12/adding-and-using-user-controls-on.html) [Part 106 - Raising custom events from user controls](http://csharp-video-tutorials.blogspot.com/2013/01/raising-custom-events-from-user.html)   In Part 106 of this video series, we discussed about raising custom events from a user control. Please watch part 106, before proceeding.     **In this video, we will discuss about**  **1.** Consuming custom events of the user control **2.** Understanding the importance of, checking if the event is null, before raining the event. We skipped discussing this, when we were discussing about raising custom events in Part 106. 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.   **In the next video, we will discuss about raising another custom event from** CalendarUserControl |  |

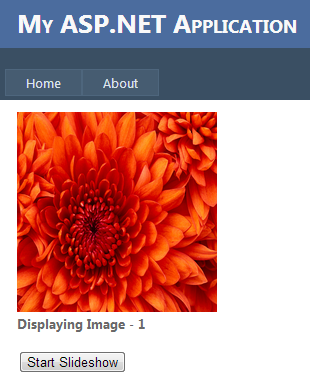
# Add image slideshow to your website using asp.net ajax and c# - Part 134

**Suggested Videos**  
[Part 131 - Cache dependency on sql server database table](http://csharp-video-tutorials.blogspot.com/2013/02/cache-dependency-on-sql-server-database.html)   
[Part 132 - Reload data into cache automatically when data in the table changes](http://csharp-video-tutorials.blogspot.com/2013/02/reload-data-into-cache-automatically.html)   
[Part 133 - What is AutoEventWireup in asp.net](http://csharp-video-tutorials.blogspot.com/2013/03/what-is-autoeventwireup-in-aspnet-part.html)  
  
In this video, 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 adavantage 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. We will discuss fixing this in our next video.

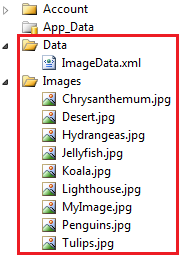
# Display images in sequence in an image slideshow - Part 135

**Suggested Videos**  
[Part 132 - Reload data into cache automatically when data in the table changes](http://csharp-video-tutorials.blogspot.com/2013/02/reload-data-into-cache-automatically.html)  
[Part 133 - What is AutoEventWireup in asp.net](http://csharp-video-tutorials.blogspot.com/2013/03/what-is-autoeventwireup-in-aspnet-part.html)   
[Part 134 - Add image slideshow to your website using asp.net ajax and c#](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html)   
  
  
  
In [Part 134](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html), we discussed adding image slideshow to a website or a web application. the problem with slideshow 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, we want to display the number of the image that is being displayed. Please watch [Part 134](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html), before proceeding. We will be modifying the example, that we started in [Part 134](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html).   
  
  
  
**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. We will discuss this in our next video

# Provide capability to start and stop image slideshow - Part 136

**Suggested Videos**  
[Part 133 - What is AutoEventWireup in asp.net](http://csharp-video-tutorials.blogspot.com/2013/03/what-is-autoeventwireup-in-aspnet-part.html)  
[Part 134 - Add image slideshow to your website using asp.net ajax and c#](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html)   
[Part 135 - Display images in sequence in an image slideshow](http://csharp-video-tutorials.blogspot.com/2013/07/display-images-in-sequence-in-image.html)  
  
Please watch [Parts 134](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html) & [135](http://csharp-video-tutorials.blogspot.com/2013/07/display-images-in-sequence-in-image.html) before proceeding. We will be continuing with the example, that we started in [Part 135](http://csharp-video-tutorials.blogspot.com/2013/07/display-images-in-sequence-in-image.html). At the moment, there is no mechanism in place to start or stop the slideshow.   
  
  
  
**What we want to achieve in this video?**  
  
  
  
  
**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 - Part 137

**Suggested Videos**  
[Part 134 - Add image slideshow to your website using asp.net ajax and c#](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html)  
[Part 135 - Display images in sequence in an image slideshow](http://csharp-video-tutorials.blogspot.com/2013/07/display-images-in-sequence-in-image.html)  
[Part 136 - Provide capability to start and stop image slideshow](http://csharp-video-tutorials.blogspot.com/2013/07/part-136-provide-capability-to-start.html)   
  
**There are 2 problems with the image slideshow**, that we have built in [Parts 134](http://csharp-video-tutorials.blogspot.com/2013/07/add-image-slideshow-to-your-website.html), [135](http://csharp-video-tutorials.blogspot.com/2013/07/display-images-in-sequence-in-image.html), & [136](http://csharp-video-tutorials.blogspot.com/2013/07/part-136-provide-capability-to-start.html). 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.   
  
  
  
**There are two ways to fix the above 2 issues.**  
**1.** Using an XML file  
**2.** Using a database table  
  
In this video, we will discuss using an XML file and in our next video, we will discuss 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.**   
  
  
**Deafult.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 - Part 138

**Suggested Videos**  
[Part 135 - Display images in sequence in an image slideshow](http://csharp-video-tutorials.blogspot.com/2013/07/display-images-in-sequence-in-image.html)  
[Part 136 - Provide capability to start and stop image slideshow](http://csharp-video-tutorials.blogspot.com/2013/07/part-136-provide-capability-to-start.html)   
[Part 137 - Add images to slideshow using xml file](http://csharp-video-tutorials.blogspot.com/2013/07/add-images-to-slideshow-using-xml-file.html)  
  
In [Part 137](http://csharp-video-tutorials.blogspot.com/2013/07/add-images-to-slideshow-using-xml-file.html), we discussed storing the image data in an xml file. In this video, we will be using a database table to store image data. So, we can safely delete ImageData.xml file. Please watch [Part 137](http://csharp-video-tutorials.blogspot.com/2013/07/add-images-to-slideshow-using-xml-file.html), before proceeding.   
  
  
  
**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.