ASP.NET Tutorial

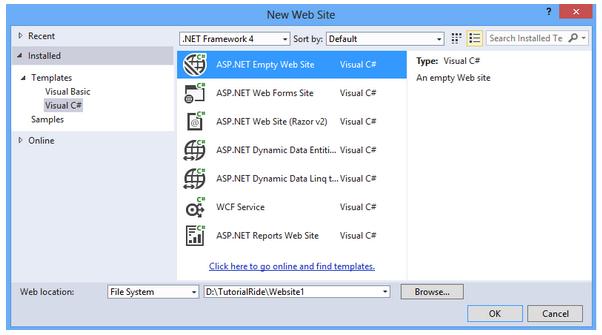
## Introduction

ASP.NET is a web development technology and a part of Microsoft .Net platform. Web application is totally different from standard Windows application. A web application is deployed on a web server.  Web browser on a client machine is access the web application using HTTP. Request from the web browser and response by the web server is done through HTTP.  
  
When any user writes website address on address bar of browser and hit the OK button then the following steps are performed.

* A GET request is sent to web server through HTTP.
* The Web server processes the GET request and sends a response back to the Web browser.
* Web browser then processes the response.
* If the user again submit the page to web server by clicking button or by some event, then HTTP post the data back to server.
* Again server processes the request and sends HTML to client (Browser)
* Web browser again processes the response, which came from server, and displays the Web page to the user on the browser.

This process continues till the user explicitly close the application.  
  
In earlier version of Visual Studio .NET (as example 2002/2003 IDEs), developers had to use IIS for all Web applications. In later version visual studio includes a built - in Web server that you can use for development. Based on location, In Visual Studio 2008 and later you can create file-system–based server on your computer, an IIS server, or an FTP server web application.  
  
**File system:** You can store all the files of your web application in desired folder. It uses inbuilt server of visual studio. If you want to run and debug your Web site locally but do not want to use local IIS, then you can use file system.   
  
**FTP:** If Web site is hosted on a remote computer and you want to connect via FTP to manage your files on a remote server then FTP is good choice.  
  
**HTTP:** If a site deployed inside IIS, then HTTP-based Web site is used.

#### Creating a new website

Start Visual Web Developer and select File, New Web Site. The New Web Site dialog box appears as given below. Provide the location and name of your website. If you don’t want inbuilt master pages along with other web pages then choose ASP.NET empty web site otherwise choose ASP.NET forms web site. We have chosen ASP.NET empty web site.  
  
  
  
Click on OK button. It will create new empty website. Right click on solution explorer and new web form. You will get Default.aspx and Default.aspx.cs page.   
  
**ASP.NET provides two types of coding model.**

* Code-inline model.
* Code-behind model.

In code inline model, your logic and presentation code are written in a single aspx page.  
  
Code-behind model provides clean code separation of the page’s business logic from its presentation logic. Your presentation logic goes in .aspx file and your cs or vb code (business logic) in .aspx.cs file.

#### Default.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>  
  
<!DOCTYPE html>   
<html xmlns="http://www.w3.org/1999/xhtml">  
  <head runat="server">  
    <title></title>  
  </head>  
  <body>  
    <form id="form1" runat="server">  
        <div>      
        </div>  
    </form>  
  </body>  
</html>

#### Default.aspx.cs

using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
  
public partial class \_Default : System.Web.UI.Page  
{  
    protected void Page\_Load(object sender, EventArgs e)  
    {  
    }  
}

Default.aspx page is starts with page directive.  
  
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>  
  
In ASP.NET a directive always begins with the special characters <%@ and ends with the characters %>. When the page is compiled, directive provides the important information to compiler.   
  
General syntax of a directive is as follows.  
  
<%@ [Directive] [Attribute=Value] %>  
  
**Following are the important directive in ASP.NET 4**

* Assembly
* Control
* Implements
* Import
* Master
* MasterType
* OutputCache
* Page
* PreviousPageType
* Reference
* Register

**@Page directive:** This directive is used by every .aspx page and controls the behavior of asp.net page. It provides the important information to compiler.   
  
**@Master directive:** It similar to page directive but mainly used with master page. The content page uses master page as template. @Master directive has less attributes than @Page directive.  
  
**@Control directive:** When you create custom/user ASP.NET control, then control directive is used. It allows you to create the properties to be inherited by the user control.  
  
**Example:**  
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs" Inherits="WebUserControl" %>  
  
**@Import directive:** The @Import directive is used to import a namespace into the ASP.NET page or user control. By using particular namespace with the help of import directive, all the classes and interfaces of the namespace are available to the page or user control. This directive supports only a single attribute called Namespace.  
  
**Example:**  
<%@ Import Namespace="System.Data" %>  
<%@ Import Namespace="System.Data.SqlClient" %>  
  
**@Implements directive:** The @Implements directive is used to implement .NET Framework interface. This directive supports only a single attribute name as Interface.  
  
**@Register directive:** This directive comes in picture when you drag and drop a user control onto any of your .aspx pages. When you drag a user control onto the .aspx page, @Register directive creates automatically at the top of the page.  
  
**@Assembly directive:**The @Assembly directive attaches assemblies in web pages. It supports Name and Src Attributes.  
  
**Example:**  
<%@ Assembly Name="MyWebAssembly" %>  
<%@ Assembly Src="MyWebAssembly.cs" %>  
  
**@PreviousPageType directive:** It is used for Cross-page posting between ASP.NET pages. It supports TypeName and VirtualPath attributes.  
  
**@MasterType directive:** This directive is used, when the master page is accessed from the Master property.   
  
**@OutputCache directive:** It is used for Page Output Caching. It catches the page for given time duration.  
  
**@Reference directive:** As it name suggests it refer to another ASP.NET page or user control. Referred page is compiled along with ASP.NET page.

#### ASP.NET folders

ASP.NET provides different folders for containing different types of folder. Some folders added implicitly in your website but you can also add these folders explicitly.

* **App\_Code:** This folder contains class files (.cs,.vb, and .jsl files)
* **App\_Data:** It contains databases such as Microsoft SQL Server Express Edition .mdf files.
* **App\_Browsers:** It Contains browser definition files (.browser) that uses to identify browsers.
* **App\_GlobalResources:** Contains resources files such as .resx and .resources files.
* **App\_Themes:** It contains specific theme for your site. App\_Themes folder has SkinFile as subfolder that contains .skin file.
* **App\_WebReferences:** Contains Web reference files such as .wsdl, .xsd, .disco, and .discomap files.
* **Bin:** This folder contains the assembly or .DLL file. These files are referenced in your application automatically.

**ASP.NET supports three types of controls.**

* HTML Controls
* HTML Server Controls
* Web Server Controls

## HTML Controls

HTML controls are client side controls and parsed by browser. Because these controls are runs on clients, therefore it improves the performance of the web page. In ASP.NET HTML controls are not available to the web server. HTML controls are mainly used in classic ASP. In classic ASP, developers have to write everything (Presentation and business logic) in single page. These controls do not provide state management. HTML controls cannot be accessed form code behind files in ASP.NET.

#### Classic ASP Web Page

<html>  
    <head><title>Employee Page</title></head>  
         <body>  
               <form name="EmployeeForm" method="post" action="Empupdate.asp"         
           id="EmployeeForm" >  
                      <input type="text" name="EmpName" id="EmpName" >  
                      <input type="submit" name="SubmitButton" value="Submit" id="SubmitButton" >  
             </form>  
        </body>  
</html>

## HTML Server Controls

HTML server controls mainly used for converting classic ASP pages into ASP.NET pages. There is no separate HTML server controls in ASP.NET but if you apply the attribute runat="server", then plain HTML controls is converted to HTML server controls.

#### Converted ASP.NET code

<html>  
<head><title>Employee Page</title></head>  
            <body>  
                       <form name="EmployeeForm" method="post" id="EmployeeForm"    
  runat="server">  
                            <input type="text" name="EmpName" id="EmpName"  runat="server">  
                            <input type="submit" name="SubmitButton" value="Submit"            
id="SubmitButton"  runat="server">  
</form>  
</body>  
</html>

Look at the above code, We only add the runat="server" attribute to controls and form tag. The action attribute is removed from the "form" tag. All HTML server control should be placed inside a form tag with runat="server" attribute to operate properly.

## Web Server Controls

ASP.NET provides a rich set of web server controls. They provide more functionality compare to HTML server controls.  
  
**Properties of web server control:**

* Runs on the web server.
* All ASP.NET server control have runat="server" attribute, by default.
* Server controls provide state management.
* Its execution is slow compare to HTML control.
* You can access these controls from code-behind.
* Server controls have predefined classes.
* Controls are inherited from System.Web.UI.WebControls name space.

**Types of server control:**

* Standard Controls
  + Buttons, input fields, and labels etc.
* Validation Controls
  + RequiredFieldValidator
  + RangeValidator
  + RegularExpressionValidator
  + CompareValidator
  + CustomValidator
  + ValidationSummary
* Rich Controls
  + Calendars, file upload, Ad-rotator, and multistep wizards etc.
* Data Controls
  + GridView, DetailView, FormView, ListView, Chart, etc.
  + Navigation Controls
  + Menu, TreeView, SiteMapPath.
* Login Controls
  + Login, CreateUserWizard, LoginName, etc.

When a user request a web page, then web server process the page by executing different events. The following are the various events of ASP.Net page life cycle.

#### Page Life Cycle Events

**PreInit**  
This event is responsible for following task:

* Set a master page dynamically.
* Set the Theme property dynamically.
* Creates dynamic controls.

**Init**  
This event fires after each control has been initialized. You can use this event to  
read or initialize control properties.  
  
**InitComplete**  
This event is raised after initializations of the page and controls have been completed.  
  
**PreLoad**  
This event is raised before PostBack is processed.  
  
**Load**  
It is an important event of page life cycle. You can use IsPostBack property to avoid unnecessary code to execute during postback.  
  
**Control (PostBack) event(s)**  
Fires those events which caused the PostBack to occur.  
  
**LoadComplete**  
All controls are loaded and initialized completely.  
  
**PreRender**  
This event fires before saving ViewState.  You can do final changes to the page or its control in PreRender.  
  
**SaveStateComplete**  
View state and control state have been saved for the page and for all controls in this event. Any changes to the page’s controls at this stage, is ignored.  
  
**Render**  
HTML, DHTML, and scripts are generated in render event that are necessary to properly display a control at the browser.    
  
**UnLoad**  
This event is mainly used for cleanup code.

## Common Server Control

#### Label control

Label control and the Literal control are used to display text in a page. Literal control only supports text property, but the Label control has a number of formatting properties.  
By default, a Label control renders its content in an HTML <span> tag. After executing your application, you can see the respective label <span> tag by using view source on the browser.   
  
The Label control displays text at a particular position on a Web page. Generally Label control is used as the caption of a TextBox.  
  
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

#### Literal control

It is light weight control. The Literal Control is similar to the Label Control but it does not support properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, etc. Also you cannot apply a style to a literal control. It is used to display static text on a Web page without adding additional properties or HTML tags.  
  
<asp:Literal ID="Literal1" runat="server">It is a Literal Control</asp:Literal>

#### TextBox control

The TextBox control is frequently used and one of the most important control. It is used to collect information from a user. It is an input control which is used to input the data.  
  
The TextBox control contains an important property called TextMode. By using this property you can set textbox as

* SingleLine
* MultiLine
* Password

**SingleLine** mode is default mode of textbox control and allows the user to enter data in a single line of text.  
  
**Password** mode masks (text is hidden) the values entered by the user.  
  
**MultiLine** mode enables user to enter text in more than one line. You can use MultiLine mode, in combination with the Columns and Rows properties. Column specifies the number of columns (Width) and rows specify the number of rows (Height) to display.  
  
Another important property of textbox control is MaxLength. It sets the limit on the number of character that a user can enter into textbox.  
  
**TextBox control supports the following event:**  
  
**TextChanged:** It fires, when you change the content or text of the textbox control. TextChanged event only fire, when AutoPostBack property has the value True. By default AutoPostBack property has the value false.   
  
If you change the text of the TextBox control and press tab key from the keyboard, the form is automatically posted back to the server.

#### Button Controls

**There are three types of button control available in ASP.NET:**

* Button
* Link Button
* Image Button

#### Button control

Button control postbacks the web page to webserver, when user clicks the button. A Button control can be used as a submit button (default) or a command button.   
  
You can also use Button control as a command button. It is useful when you want to group a set of buttons. Button control has property called CommandName. Assign unique CommandName value for each button. Create a single command event handler explicitly, that will handle the event of all command buttons.  
Suppose that you have four buttons as given below and want to create all buttons as a command button.  
  
command button  
  
Provide a unique CommandName to each button. As example First, Last, Previous, Next. 

#### Example: Default.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>  
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml">  
    <head runat="server">  
        <title></title>  
    </head>  
    <body>  
        <form id="form1" runat="server">  
        <div>      
        </div>                 
            <asp:Button ID="Button1" runat="server" OnCommand="Navigate\_Command" Text="<<" CommandName="First" Width="65px"/>  
            <asp:Button ID="Button2" runat="server" OnCommand="Navigate\_Command" Text="<" CommandName="Previous" Width="65px"/>  
            <asp:Button ID="Button3" runat="server" OnCommand="Navigate\_Command" Text=">" CommandName="Next" Width="65px"/>  
            <asp:Button ID="Button4" runat="server" OnCommand="Navigate\_Command" Text=">>" CommandName="Last" Width="65px"/>  
        </form>  
    </body>  
</html>

#### Default.aspx.cs

using System;  
using System.Web.UI.WebControls;  
using System.Drawing;  
public partial class \_Default : System.Web.UI.Page  
{  
    protected void Page\_Load(object sender, EventArgs e)  
    {  
    }  
    protected void Navigate\_Command(object sender, CommandEventArgs e)  
    {  
        switch (e.CommandName)  
        {  
            case "First":  
                Response.Write("First");  
                break;  
            case "Previous":  
                Response.Write("Previous");  
                break;  
            case "Next":  
                Response.Write("Next");  
                break;  
            case "Last":  
                Response.Write("Last");  
                break;  
        }  
    }  
}

**CausesValidation property:** It checks the page validation. By default this property is true for button control. If you set CauseValidation value to false, then it will bypass the page validation.

#### LinkButton

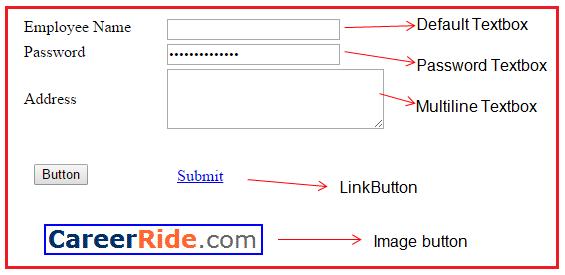
LinkButton control displays a link instead of a push button. By default, a LinkButton control is a Submit button.  
  
**Some of the important properties of LinkButton Control are:**  
  
**CausesValidation:** If this property is set as true then validation will be performed when Linkbutton is be clicked. Otherwise it will bypass the validation.  
  
**PostBackUrl:** It posts a form to a particular page when the LinkButton control is clicked.  
  
**ValidationGroup:** The group of controls when the LinkButton control causes posts back to the server.  
  
**OnClick:**Attach a server side method that will fire when this button will be clicked.  
  
**OnClientClick:** You can attach a client-side script that executes when the LinkButton is clicked.

#### ImageButton

The ImageButton control is similar to Button and LinkButton controls but it always displays an image. It works as a clickable image. Most of the properties are same as Button or LinkButton.  
The main difference is ImageUrl and AlternateText property. ImageUrl Gets or Sets the location of the image. AlternateText property provides alternate text for the image.  
  
Let us take an example that will show the use of TextBox and Different type of Button.

#### Example: Default.aspx

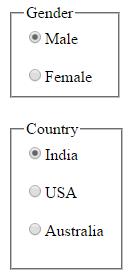
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="\_Default" %>  
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head runat="server">  
    <title></title>  
    <style type="text/css">  
        .auto-style1   
        {  
            width: 103px;  
        }  
        .auto-style3   
        {  
            width: 165px;  
        }  
        .auto-style4   
        {  
            width: 143px;  
        }  
    </style>  
</head>  
<body>  
    <form id="form1" runat="server">              
              <table style="width: 76%;">  
            <tr>  
                <td class="auto-style1">      
        <asp:Label ID="Label1" runat="server" Text="Label">EmployeeName</asp:Label>  
                </td>  
                <td class="auto-style4">  
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>  
                </td>  
                <td class="auto-style3"> </td>  
            </tr>  
            <tr>  
                <td class="auto-style1">  
        <asp:Label ID="Label2" runat="server" Text="Label">Password</asp:Label>  
                </td>  
                <td class="auto-style4">  
        <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"> </asp:TextBox>  
                </td>  
                <td class="auto-style3"> </td>  
            </tr>  
            <tr>  
                <td class="auto-style1">  
        <asp:Label ID="Label3" runat="server" Text="Label">Address</asp:Label>  
                </td>  
                <td class="auto-style4">  
        <asp:TextBox ID="TextBox3" runat="server" Height="54px" TextMode="MultiLine" Width="211px"></asp:TextBox>  
                </td>  
                <td class="auto-style3"> </td>  
            </tr>  
            <tr>  
                <td class="auto-style1">  
        <asp:Button ID="Button1" runat="server" Text="Button" />  
                </td>  
                <td class="auto-style4">  
        <asp:LinkButton ID="LinkButton1" runat="server">Submit</asp:LinkButton>  
                </td>  
                <td class="auto-style3">  
        <asp:ImageButton ID="ImageButton1" runat="server" AlternateText="This is Image Button" ImageUrl="~/logo.gif" BorderColor="Blue" BorderStyle="Solid" BorderWidth="2px" PostBackUrl="~/Default2.aspx" />  
                </td>  
            </tr>  
        </table>       
    </form>  
</body>  
</html>



#### CheckBox Control

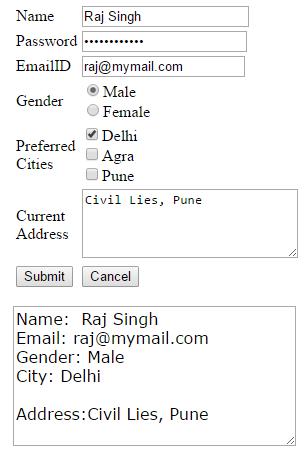
A CheckBox control is used to select a single or multiple options from the available choices. For example a person can select more than one cities for travelling.  
  
**Important Properties of the CheckBox Control**  
  
**Checked:** It is used to check if the check box is checked or not. This is a boolean property.  
  
**Text:**It is used to get or set the text associated with the check box control. This is a string property.  
  
**TextAlign:** It enables you to set the text right or left of the check box. By default TextAlign property is set to right.  
  
**AutoPostBack:** If you want that when you change the status of checkbox (check or uncheck), then set AutoPostBack property true. By default this property is set to false.  
  
**Methods:**  
**Focus():** It sets 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 checkbox control is changed. AutoPostBack property should be set as true to fire this event.

#### RadioButton Control

Radio Button control is used, when you want to select only one option from the available choices. RadioButton control works in a group. There may be more than one group of radio button. You can select only one RadioButton control from particular radio button group.   
  
  
  
Most of the properties and events are same as CheckBox control but GroupName property is different.  
  
When you use RadioButton, they are not in a group. You can select more than one RadioButton, they are not mutually exclusive. You have to form the group of radio button as above given example. You can create group by assigning similar GroupName to RadioButton.  
  
Let us take one example and we will use all the controls that we have discussed till now.  
  
In this example, when user fill all the details and click on submit button, the information will came in last multiline textbox. When user clicks on cancel button, all fields will be empty.

#### Example

using System;  
public partial class EmployeeForm : System.Web.UI.Page  
{  
    protected void Page\_Load(object sender, EventArgs e)  
    {  
    }  
    protected void btnSubmit\_Click(object sender, EventArgs e)  
    {  
        string Name ="Name:  "+ txtName.Text;  
        string Email ="Email: "+ txtEmailID.Text;  
        string Gender="" ;  
        if(RDBmale.Checked)  
        {  
            Gender ="Gender: "+ RDBmale.Text;  
        }  
        if (RDBfemale.Checked)  
        {  
            Gender = "Gender: " + RDBfemale.Text;  
        }          
        string Cities = "";  
        if(chbAgra.Checked)  
        {  
            Cities +=chbAgra.Text+"\n";  
        }  
        if (chbDelhi.Checked)  
        {  
            Cities += chbDelhi.Text + "\n";  
        }  
        if (chbPune.Checked)  
        {  
            Cities += chbPune.Text + "\n";  
        }  
        string Address ="Address:"+ txtAddress.Text;  
        txtResult.Text = Name + "\n" + Email+"\n"+Gender+"\n"+"City: "+Cities +"\n" +Address;  
    }  
    protected void btnCancel\_Click(object sender, EventArgs e)  
    {  
        txtName.Text = String.Empty;  
        txtPassword.Text = String.Empty;  
        txtAddress.Text = String.Empty;  
        txtEmailID.Text = String.Empty;  
        RDBfemale.Checked = false;  
        RDBmale.Checked = false;  
        chbPune.Checked = false;  
        chbAgra.Checked = false;  
        chbDelhi.Checked = false;  
        txtResult.Text= String.Empty;  
    }      
}

**Output:**  
  


|  |  |  |
| --- | --- | --- |
| **Field** | **Control** | **ID** |
| Name | TextBox | txtName |
| Password | TextBox | txtPassword |
| Email | TextBox | txtEmailID |
| Gender | RadioButton | RDBmale |
| Prefered Cities | CheckBox | chbDelhi |
| Current Address | Multiline TextBox | txtAddress |
| Submit | Button | btnSubmit |
| Cancel | Button | btnCancel |