Unit -2 Standard controls, Rich controls, Navigational controls

1. Label Control

The label control is used to display text on a website.

It is primarily used to generate captions for other controls such as textboxes. Labels typically assist the user in entering data into text boxes by providing written instructions. Labels are controls on the server-side. The Label class can be found in the System.Web.UI.WebControls namespace.

Properties :

AccessKey: To add the keyboard shortcut for the label, Accesskey is used.

TabIndex: It determines the index of tab control for the webserver.

BackColor: To enhance the look and make it with different colors, we can use this property so that it will help to change the background color of the label.

BorderColor: If we want to set the color for the label border, we can use this property.

BorderWidth: This property will allow us to set a particular width for the label border.

Font: We need to set the font for the text of the label then we can use this property.

Forecolor: It is used to set the color for the label text.

Text: Text, which needs to be displayed for the label, is used by this property. ToolTip:

It provides the text to be displayed when we try to put the mouse over a label. Visible:

It will allow us to set the visibility of the control on the web form.

Height: It provides us to set the height of the label control.

AutoSize: If we need to resize the label control automatically, it will allow us to set the value for it.

BorderStyle: We can design the border of the label control as per the application

requirement. FlatStyle: It deals with the flat style appearance for the label control.

Font: It will determine the font for the text of label control. It will set the value for it.

TabStop: If the user can use the tab to the label control, then this property determines the value for it.

TextAlign: It will provide the alignment for the text in the label control.

Example :

```
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)
{
Label1.Test=" Welcome to homepage";
}
}
```

2.Literal Control

The Literal Control is similar to the Label Control as they both are used to display static text on a web page.

The Literal Control is not inherited from WebControl namespace. The Literal Control doesn't provide substantial functionality but Literal text is programmable. It doesn't add any HTML elements to the web page.

This control makes it possible to add HTML code directly in the code designer window without switching to design view and clicking the HTML button to edit the HTML. You cannot apply a style to a literal control.

Unlike Label control, there is no property like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. for Literal control. That makes it more powerful, you can even put a pure HTML contents into it.

Properties :

PassThrough : The contents of the control are not modified.

Encode : The contents of the control are converted to an HTML-encoded string.

Transform : Unsupported markup-language elements are removed from the contents of the control. If the Literal control is rendered on a browser that supports HTML or XHTMLcontrol's contents are not modified.

Example :

```
 <asp:Literal ID="Literal1" runat="server"></asp:Literal>
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)
          {
          Literal1.Text = "<h1><marquee>Welcome to ASP.NET</marquee></h1>";

          }
}
```

3.Bulleted List

The BulletedList control renders either an unordered (bulleted) or ordered (numbered) list. Each list item can be rendered as plain text, a LinkButton control, or a link to another web page. For example, the page given below uses the BulletedList control to render an unordered list of products.

Possible values are as follows:

Circle
CustomImage
Disc
LowerAlpha
LowerRoman
NotSet
Numbered
Square
UpperAlpha
UpperRoman

BulletedList control also supports the DisplayMode property that is used to modify the appearance of list items. Possible values are as follows:

HyperLink
LinkButton
Text

Example :

```
<script runat="server">
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Untitled Page</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:BulletedList
       id="blHyperLinkWebsites"
       DisplayMode="HyperLink"
       Target="_blank"
       Runat="server">
       <asp:ListItem
           Text="Itorian"
         Value="http://www.google.com" />
       <asp:ListItem
           Text="C# Corner"
         Value="http://www.amazon.com" />
       <asp:ListItem
           Text="VB.Net Haven"
         Value="http://www.flipcart.com" />
   </asp:BulletedList>
   </div>
   </form>
</body>
</html>
```

4. Textbox control :

A TextBox server control is a simple box that accepts some input text from the user. Why did I use the term server? Well, because like all other server controls, ASP.NET provides its own tag for the textbox control which is run at the server and the generated HTML code is returned as a response to the browser.
So, thinking from HTML perspective, the TextBox control generates the HTML text input element. It lets the users type in some text and perform required operations on it.

Properties :

ID : Identification name of textbox control.
Text : It is used to display text in a control.
BackColor : It is used to set background color of textbox control.
ForColor : It is used to set text color of the control.
ToolTip : It displays a text on control when mouse over on it.
TabIndex : It is used manage tab order of control.
CssClass : It is used to apply style on control.
Enable true/false : used to enable or disable control.
Enable Theming true/false : It is used to enable or disable effect of theme on control.
CausesValidation true/false : It is used to enable or disable validation effect on control
Visible true/false : It is used to hide or visible control on web page.

Important Properties of TextBox control

MaxLengh It is used to set maximum number of characters that can be input in TextBox.
TextMode Single / Multiline / Password
ReadOnly true/false : used to enable or disable control readonly.

Example :

```
<asp:TextBox ID="TextBox1" runat="server" Text="asp.net"></asp:TextBox>
TextBox1.Text = "ASP.Net with C#";
Label1.Text = TextBox1.Text;
protected void btndisplay_Click(object sender, EventArgs e)
{
TextBox1.Text = "asp.net";
}
protected void btnclear_Click(object sender, EventArgs e)
{
TextBox1.Text = "";
}
```

5. Radio button

Radio button allows the user to choose only one of a predefined set of options. When a user clicks on a radio button, it becomes checked, and all other radio buttons with same group become unchecked. The buttons are grouped logically if they all share the same GroupName property.

Properties :

AccessKey : It is used to set keyboard shortcut for the control.

TabIndex : The tab order of the control.

BackColor : It is used to set background color of the control.

BorderColor : It is used to set border color of the control.

BorderWidth : It is used to set width of border of the control.

Font : It is used to set font for the control text.

ForeColor : It is used to set color of the control text.

Text : It is used to set text to be shown for the control.

ToolTip : displays the text when mouse is over the control.

Visible : To set visibility of control on the form.

Height : It is used to set height of the control.

Width : It is used to set width of the control.

GroupName : It is used to set name of the radio button group.

Example :

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
        <title>Untitled Page</title>
</head>
<body>
        <form id="form1" runat="server">
        <div>
                <asp:RadioButton ID="RadioButton1" GroupName="language" runat="server"
Text="ASP.NET" /><br />
                <asp:RadioButton ID="RadioButton2" GroupName="language" runat="server"
Text="VB.NET" /><br />
                <asp:RadioButton ID="RadioButton3" GroupName="language" runat="server"
Text="C#" /><br />
                <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click"
/><br />
                 <asp:Label ID="Label1" runat="server" Text="Label1"></asp:Label>
        </div>
        </form>
</body>
</html>
```

6. Check box control

CheckBox control is an asp.net web server control. CheckBox control visually as square on web forms. The Checkbox control allow user to check square or uncheck square. In CheckBox control check and uncheck checkbox specify by the checked property of check box true or false. If checkbox control square ticked then checked = true and unchecked then checked=false. We can drag the checkbox control from toolbox and drop it to on web forms shows like below. Checkbox control allow user to do either checked(tick) or unchecked (untick) the checkbox control in asp.net.

Properties :

AccessKey : It is used to set keyboard shortcut for the control.
TabIndex : The tab order of the control.
BackColor : It is used to set background color of the control.
BorderColor : It is used to set border color of the control.
BorderWidth : It is used to set width of border of the control.
Font : It is used to set font for the control text.
ForeColor : It is used to set color of the control text.
Text : It is used to set text to be shown for the control.
ToolTip: displays the text when mouse is over the control.
Visible : To set visibility of control on the form.
Height : It is used to set height of the control.
Width : It is used to set width of the control.
Checked: It is used to set check state of the control either true or false.

Example :

```
<%@ Page Language="C#" AutoEventWireup="true"
 CodeBehind="WebControls.aspx.cs" Inherits="WebFormsControlls.WebControls" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:RadioButton ID="RadioButton1" runat="server" Text="Male" GroupName="gender"
/>
            <asp:RadioButton ID="RadioButton2" runat="server" Text="Female"
GroupName="gender" />
        </div>
        <p>
            <asp:Button ID="Button1" runat="server" Text="Submit" OnClick="Button1_Click"
style="width: 61px" />
```

```
        </p>
    </form>
    <asp:Label runat="server" id="genderId"></asp:Label>
</body>
</html>
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace WebFormsControlls
{
    public partial class WebControls : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            ShowCourses.Text = "None";
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            var message = "" ;
            if (CheckBox1.Checked)
            {
                message = CheckBox1.Text+" ";
            }
            if (CheckBox2.Checked)
            {
                message += CheckBox2.Text + " ";
            }
            if (CheckBox3.Checked)
            {
                message += CheckBox3.Text;
            }
            ShowCourses.Text = message;
        }
    }
}
```

7. The Button Control

The Button control is used to display a push button. The push button may be a submit button or a command button. By default, this control is a submit button.

A submit button does not have a command name and it posts the page back to the server when it is clicked. It is possible to write an event handler to control the actions performed when the submit button is clicked.

A command button has a command name and allows you to create multiple Button controls on a page. It is possible to write an event handler to control the actions performed when the command button is clicked.

Properties :

ID : Identification name of button control.

Text : It is used to display text in a control.

BackColor : It is used to set background color of button control.

ForColor : It is used to set text color of the control.

ToolTip : displays a text on control when mouse over on it.

TabIndex : It is used manage tab order of control.

CssClass : It is used to apply style on control.

Enable true/false : used to enable or disable control.

Enable Theming true/false :It is used to enable or disable effect of theme on control.

CausesValidation true/false : It is used to enable or disable validation effect on control

Visible true/false : It is used to hide or visible control on web page.

Important Properties of Button control

CommandArgument: A string value passed to code behind when button Command event is fired.

CommandName : A string value passed to code behind when button Command event is fired.

OnClientClick : The JavaScript function name or any client side script code function name. PostBackUrl The URL Path of the page to redirect, when button is clicked.

Example :

```
<html>
<body>

<form runat="server">
<asp:Button id="b1" Text="Submit" runat="server" />
</form>

</body>
</html>
```

```
<script runat="server">
Sub submit(sender As Object, e As EventArgs)
lbl1.Text="Your name is " & txt1.Text
End Sub
</script>

<html>
<body>

<form runat="server">
Enter your name:
<asp:TextBox id="txt1" runat="server" />
<asp:Button OnClick="submit" Text="Submit" runat="server" />
<p><asp:Label id="lbl1" runat="server" /></p>
</form>

</body>
</html>
```

8. LinkButton Control :

The LinkButton control is used to create a hyperlink-style button on the Web page. This control looks like a Hyperlink control but almost has the functionality of the Button control. Despite being a hyperlink, you can't specify the target URL. There is no UserSubmitBehavior property like the Button control with this control.

Properties :

CausesValidation : This indicates whether validation will be performed when this button will be clicked. Here, the Value can be set as true or false.

PostBackUrl : This Specifies the URL of the page to post to from the current page when the LinkButton control is clicked.

ValidationGroup : The group of controls for which the LinkButton control causes validation when it posts back to the server.

OnClick: Attach a server side method that will fire when this button will be clicked.
OnClientClick : Attach a client side (JavaScript) method that will fire when this button will be clicked.

Note:

- Usually, this control is used to give a uniform look and feel throughout the page or site if you are using a hyperlink. Also if you have less space and want to show a control that can fire a server-side event; placing a Button control will not work as this takes more space as well as its look and feel is completely different than the LinkButton control.
- ImageButton

- Its like an ASP Button control, the only difference is, you have the ability to place your own image as a button. You use an image Button when you want your button to look different than the plain rectangular button. Any image can be a button!. Some of the important properties of an ImageButton Control are:
- ImageUrl: Gets or Sets the location of the image to display as button control. CausesValidation: This indicates whether validation will be performed when this button will be clicked. Here, the Value can be set as true or false.
- PostBackUrl: This Specifies the URL of the page to post to from the current page when the LinkButton control is clicked.
- OnClick: Attach a server side method that will fire when this button will be clicked. OnClientClick: Attach a client side (JavaScript) method that will fire when this button will be clicked.

Example :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="buttons.aspx.cs"
 Inherits="buttons" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
   <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
   style="z-index: 1; left: 23px; top: 56px; position: absolute" Text="click!" />
   <asp:LinkButton ID="LinkButton1" runat="server" EnableTheming="True"
   onclick="LinkButton1_Click"
      style="z-index: 1; left: 24px; top: 194px; position: absolute"
      ToolTip="Link to another page">LinkButton</asp:LinkButton>
   <p style="height: 669px">
      <asp:ImageButton ID="ImageButton1" runat="server"
          ImageUrl="~/images/smiley-guy.jpg" onclick="ImageButton1_Click"
style="z-index: 1; left: 16px; top: 294px; position: absolute; height: 80px; width:
88px; right: 1177px; bottom: 421px" />
      <asp:TextBox ID="TextBox1" runat="server"
         style="z-index: 1; left: 20px; top: 383px; position: absolute"></asp:TextBox>
   </p>
   <p>&nbsp;</p>
   </form>
</body>
</html>
```

aspx.cs Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class buttons : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
         {
         }
   protected void Button1_Click(object sender, EventArgs e)
         {
         Button1.Text = "You clicked the button!";
         }
   protected void LinkButton1_Click(object sender, EventArgs e)
         {
         Response.Redirect("default.aspx");//move to the next link i.e default.aspx
         }
   protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
         {
         TextBox1.Text="keep smiling";
         }
}
```

9. Image button :

ImageButton control in ASP.Net is used in button formation by which we can use the images. It is like a button with an image on it. Generally, we have seen the images on the website and after clicking on it, certain activities performed. So, in this case, we need to use ImageButton control. It is used to fire an event after clicking on the ImageButton either on the client or server-side. We can set the image, which we want on the button. This image button will respond to the mouse click as soon as we click. When we click the image button control, it raises both the events that click and command events.

Properties :

ID : identification of ImageButton control

ImageUrl : set image path to display image on image button control.

AlternateText : AlternateText text display when image can not display on web

page. PostBackUrl : Path of the page when redirect while click the Image button.
OnClientClick : write JavaScript or any client side script code function name.

Example :

```
 ImageButton.aspx
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ImageButton.aspx.cs"
 Inherits="MyCalendar.ImageButton" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
<style type="text/css">
#form1 {
 height: 118px;
}
</style>
</head>
<body>
<form id="form1" runat="server">
Click image to visit Educba website<div>
<br />
</div>
<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="~\demo.png" Height="64px"
Width="158px" Imagealign="Left" PostBackUrl="https://www.google.com/" /> </form>
</body>
</html>
```

10. Hyperlink Control :

ASP.NET HyperLink control is used to create a hyperlink and this control is a server-side control. The main purpose of this control is to refer any web page on the server, it acts in response to the click event. For creating a Hyperlink either we can use drag and drop function which exists on Visual Studio IDE or we can use code for hyperlinking. Hyperlink control which generates the link allows users to navigate from one page to another page. It contains several properties like text property, imageUrl, navigateUrl and so on.

Properties :

AccessKey : Access Key is used for setting shortcuts of controls in the
TabIndex : TabIndex is used for setting the control to the tab order.
BackColor : This property is used for setting the background color of the control.

BorderColor: This property is used for setting the border color of the control.
BorderWidth :This property is used for setting the width of the border of the control
Font : This property is used to set the font for the control text.
ForeColor : This forecolor is used to set the color of the control text.
Text : This property is used to show the text to be shown for the control. ToolTip : This property is used to show the text to be displayed when the mouse is over the control.
Visible : This property is used to set the visibility of control on the form, it may be either true or false.
Height : This property is used to set the height of the control.
Width : To set the width of the control.
NavigateUrl : This property is used to set navigate URL, to navigate from one page to another.
Target : This property is the Target frame for the navigate URL.

Example:

```
<%@ Page Language="C#" AutoEventWireup="True" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>HyperLink Example</title>
</head>
<body>
<form id="Form1" runat="server">

  <h3>HyperLink Example</h3>

  Click on the HyperLink:<br />

  <asp:HyperLink id="hyperlink1"
             ImageUrl="images/pict.jpg"
            NavigateUrl="http://www.microsoft.com"
            Text=" Microsoft Official Site"
            Target="_new"
            runat="server"/>
</form>
</body>
</html>
```

11. Dropdownlist control :

Asp .net's Dropdownlist control is equivalent to HTML select box control. This control is very useful when you have a large collection of data and you want the user can only select a single item from the list of values. E.g. You have a collection of country names to choose from.

Properties :

DropDownList1.Items.Count:

it is used to provide the total number of options or items in the drop-down list.

DropDownList1.Items.Add("ItemName"):

suppose if we want to add some new item, then this property is useful for adding the item in the drop-down list.

DropDownList1.Items.Remove("ItemName"):

It will help to remove the item from the drop-down list.

DropDownList1.Items.Insert(int index, "ItemName"):

if we want the item to be added at a specific position, this property helps to add a new item at a specific position in the drop-down list control.

DropDownList1.Items.RemoveAt(int index):

It will remove the specific item from a specific position (index) from the drop-down list control.

DropDownList1.Items.Clear():

if we don't want all the items and now we want to add another or maybe we want to change the items from the drop-down list.so it's better to clear all the items first. This property is used to clear all the provided items from the drop-down list.

DropDownList1.SelectionItem.Text:

this is one of the important properties because it will return the text value which is in the selected items in the drop-down list.

DropDownList1.SelectedIndex:

the index will always start from zero. When we select any item from the drop-down list, it is associated with the index. This property will return the position of the selected item which is its index value.

DropDownList1.DataSource:

it is mostly the DataTable or DataSet.

DropDownList1.DataValueField: It will bind the value to the drop-down list which will be visible in the dropdown list.

Item:

it will provide the collection of the items from the drop-down list.

AutoPostBack:

Its value is ‘true’ or ‘false’. True represents that form is posted back automatically to the server when the user changes the dropdown list selection.

DataTextField:

values will be visible to the end users. it is used to set the text in the Dropdown list control.

DataValueFeild:

This is used to set the name of the column as a value in the drop-down list. This value is not visible to the end user.

Example :

```
 <asp:dropdownlist runat="server" id="ddlTest">
     <asp:listitem text="Red" value="1"></asp:listitem>
     <asp:listitem text="Black" value="2"></asp:listitem>
     <asp:listitem text="Blue" value="3"></asp:listitem>
     <asp:listitem text="Green" value="4"></asp:listitem>
     <asp:listitem text="Yellow" value="5"></asp:listitem>
</asp:dropdownlist>
protected void ddlTest_SelectedIndexChanged(object sender, EventArgs e)
   {
       string selectedText = ddlTest.SelectedItem.Text;
       string selectedValue = ddlTest.SelectedItem.Value;

      //--- Show results on page.
       Response.Write("Selected Text is " + selectedText + " and selected value is :" +
selectedValue);
   }
```

12. Listbox control :

The list box is like a set of radio buttons except that the list box is scrollable and can contain more items than a set of Radio Buttons. Additionally, the number of entries that can be added to the list box can be at the run time. The list box is a web server control. The web control class provides the properties, methods, and events that are common to all web server controls. The appearance and behavior of these controls can be controlled with this web control class. Example: The font color and background color can be controlled using the ‘ForeColor’ and ‘BackColor’ properties respectively.

Properties :

Items.Count: Returns the total number of items in the list box.
Items.Clear: Clears all the items from the list box.
SelectedItem.Text: Returns the text of the selected item.
Items.Remove(“name”): Removes the item with text “name”.
Items.RemoveAt(int index): Removes the item present at the given Index.
Items.Insert(int index, “text”): Inserts the “text” at the given index.
SelectedItem: Returns the index of the selected item.
SelectionMode: Specifies the selection mode “single or multiple”.

Example :

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title> An example of ASP.Net ListBox</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h2>Choose a color:</h2>
<asp: ListBox ID ='ListBox1’ runat = 'server' AutoPostBack = 'true' Font-Size = 'X-Large' Rows =
'5'
 ForeColors = 'Tomato' Width = '350' >
<asp: ListItem> Dark Grey </asp: ListItem>
<asp: ListItem> Red </asp: ListItem>
<asp: ListItem> Green </asp: ListItem>
<asp: ListItem> Blue </asp:ListItem>
<asp: ListItem> Yellow </asp: ListItem>
<asp: ListItem> Black </asp: ListItem>
</asp: ListBox>
</div>
</form>
</body>
</html>
```

13. Image map control :

The ImageMap control in ASP.NET 2.0 and onward versions can be used to create an image that contains defined hot spot regions. When a user clicks a hot spot region, the control can either generate a post back to the server or navigate to a specified URL. There are three kinds of hot spot regions defined in ImageMap control.

- RectangleHotSpot
- CircleHotSpot
- PolygonHotSpot

The RectangleHotSpot defines rectangular hot spot regions. The CircleHotSpotdefines circle-shaped ones and the PolygonHotSpot is used for irregularly shaped hot spot area.

Example :

```
<%@ page language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html
xmlns="http://www.w3.org/1999/xhtml" >
<head id="head1" runat="server">
  <title>ImageMap Class Navigate Example</title>
</head>
  <body>
    <form id="form1" runat="server">
      <h3>ImageMap Class Navigate Example</h3>

      <h4>Shopping Choices:</h4>

      <asp:imagemap id="Shop"
        imageurl="Images/ShopChoice.jpg"
       width="150"
        height="360"
        alternatetext="Shopping choices"
        runat="Server">

        <asp:circlehotspot
```

```
          navigateurl="http://www.tailspintoys.com"
         x="75"
          y="290"
          radius="75"
          hotspotmode="Navigate"
          alternatetext="Shop for toys">
        </asp:circlehotspot>

        <asp:circlehotspot
          navigateurl="http://www.cohowinery.com"
         x="75"
          y="120"
          radius="75"
          hotspotmode="Navigate"
          alternatetext="Shop for wine">
        </asp:circlehotspot>
      </asp:imagemap>

    </form>
  </body>
</html>
```

14.Image control :

An ASP.NET Image server control is a control that displays an image on the web page. The image is sourced from either a location on the server or from a URL on the internet. Why did I use the term server control? Well, because like all other server controls, ASP.NET provides its own tag for the Image control which is run at the server and the generated HTML code is returned as a response to the browser. So, thinking from the HTML perspective, the Image control generates the HTML <img> tag along with the attributes such as source, height, width, styles, etc.

Properties :

AlternateText: Enables you to provide alternate text for the image (required for accessibility).

DescriptionUrl :Enables you to provide a link to a page that contains a detailed description of the image (required to make a complex image accessible).

GenerateEmpty

AlternateText : Enables you to set the AlternateText property to an empty string.

ImageAlign : Enables you to align the image relative to other HTML elements in the page.

Possible values are AbsBottom, AbsMiddle, Baseline, Bottom, Left, Middle, NotSet, Right, TextTop, and Top.

ImageUrl : Enables you to specify the URL to the image.

Example :

```
<%@ Page Language="C#" %>
<!DOCTYPE html>

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Asp.Net Image Control Example</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/Bird1.jpg" />
    </div>
    </form>
</body>
</html>
```

15. Panel control :

Panel control is a container of other control. We can place multiple web server control in Panel control and make same effect or action on all controlby applying effect of Panel Control.

Exmaple: if we want to Hide or visible more then one Control as same time, so we can place them to in Panle control and Just Visible or Hide Panle control the all the control will be Visible or Hide according to Panel Control.

The Panel Control is automatically exhaust the Space of web form.

Properties:

Direction: Text direction in the panel.
GroupingText: Allows the grouping of text as a field.
HorizontalAlign: Horizontal alignment of the content in the panel.
ScrollBars: Specifies visibility and location of scrollbars within the panel.

Example :

```
<asp:Panel ID="Panel1" runat="server" BorderColor="#990000" BorderStyle="Solid"
Borderstyle="width:1px" Height="116px" ScrollBars="Both" style="width:278px">
This is a scrollable panel.
<br />
<br />
<asp:Button ID="btnpanel" runat="server" Text="Button" style="width:82px" />
</asp:Panel>
```

16. Sitemap path control :

The **SiteMapPath** control basically is used to access web pages of the website from one webpage to another. It is a navigation control and displays the map of the site related to its web pages. This map includes the pages in the particular website and displays the name of those pages. You can click on that particular page in the Site Map to navigate to that page. We can say that the SiteMapPath control displays links for connecting to URLs of other pages. The SiteMapPath control uses a property called **SiteMapProvider** for accessing data from databases and it stores the information in a data source. The SiteMapProvider internally utilizes the SiteMapProvider abstract class defined under the **System.Web** namespace. The representation of the SiteMapPath control is as follows:

Root Node->Child Node

Public Properties of SiteMapPath class :

ParentLevelsDisplayed: It specifies the number of levels of parent nodes and then displays the control accordingly related to the currently displayed node.

RenderCurrentNodeAsLink: It specifies whether or not the site navigation node that represents the currently displayed page is rendered as a hyperlink.

PathSeperator: It specifies the string that displays the SiteMapPath nodes in the rendered navigation path.

Style properties of the SiteMapPath class :

CurrentNodeStyle: It specifies the style used for the display text for the current node.

RootNodeStyle: It specifies the style for the root node style text.

NodeStyle: It specifies the style used for the display text for all nodes in the site navigation path.

Example :

```
<?xml version=”1.0″ encoding=”utf-8″ ?>
<siteMap xmlns=”http://schemas.microsoft.com/AspNet/SiteMap-File-1.0″ >
<siteMapNode url=”Default.aspx” title=”HOME PAGE” description=”Home page of site”>
<siteMapNode url=”Laptop.aspx” title=”LAPTOP PAGE” description=”Laptop page of
site” >
<siteMapNode url=”Sony.aspx” title=”SONY LAPTOP” description=”Sony laptop page” >
<siteMapNode url=”Sonyvio.aspx” title=”SONY VIO” description=”Sony Vio page” />
<siteMapNode url=”Sonypc.aspx” title=”SONY PC” description=”Sony Pc page” />
</siteMapNode>
<siteMapNode url=”Samsung.aspx” title=”SAMSUNG
LAPTOP” description=”Samsung laptop page” />
</siteMapNode>
</siteMapNode>
</siteMap>
```

17. Menu control:

The Menu control is used to create a menu of hierarchical data that can be used to navigate through the pages. The Menu control conceptually contains two types of items. First is StaticMenu that is always displayed on the page, Second is DynamicMenu that appears when opens the parent item.

Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. are implemented through style properites of <table, tr, td/> tag.

Following are some important properties that are very useful.

Properties :

DataSourceID : Indicates the data source to be used (You can use .sitemap file as datasource).

Text : Indicates the text to display in the menu.

Tooltip : Indicates the tooltip of the menu item when you mouse over.

Value : Indicates the nondisplayed value (usually unique id to use in server side events)

NavigateUrl : Target Selectable Indicates the target location to send the user when menu item is clicked. If not set you can handle MenuItemClick event to decide what to do.If NavigationUrl property is set, it indicates where to open the target location (in new window or same window).

true/false : If false, this item can't be selected. Usually in case of this item has some child.

ImageUrl, ImageToolTip, PopOutImageUrl : Indicates the image that appears next to the menu item. Indicates the tooltip text to display for image next to the item.

Target : If NavigationUrl property is set, it indicates where to open the target location (in new window or same window).

Styles of Menu Control :

StaticMenuStyle : Sets the style of the parent box in which all menu items appears.

DynamicMenuStyle : Sets the style of the parent box in which dynamic menu items appears. **Example :**

```
<div class="frame">
<ej:Menu ID="menu" runat="server" Width="615px">
</ej:Menu>
</div>
<Items>
<ej:MenuItem Id="Products" Text="Products">
</ej:MenuItem>
<ej:MenuItem Id="Support" Text="Support">
</ej:MenuItem>
<ej:MenuItem Id="Purchase" Text="Purchase"> </ej:MenuItem>
<ej:MenuItem Id="Downloads" Text="Downloads">
</ej:MenuItem>
<ej:MenuItem Id="Resources" Text="Resources"> </ej:MenuItem>
<ej:MenuItem Id="Company" Text="Company"> </ej:MenuItem>
</Items>
```

18. TreeView control :

The TreeView control contains a hierarchy of TreeViewItem controls. You can use the TreeView control to display information from a wide variety of data sources such as an XML file, site-map file, string, or from a database. It provides a way to display information in a hierarchical structure by using collapsible nodes . The top level in a tree view are root nodes that can be expanded or collapsed if the nodes have child nodes.

Properties:

BackColor : Gets or sets the background color of the tree node.

Checked : Gets or sets a value indicating whether the tree node is in a checked state.

ContextMenu : Gets the shortcut menu that is associated with this tree node.

ContextMenuStrip : Gets or sets the shortcut menu associated with this tree node.

FirstNode: Gets the first child tree node in the tree node collection.

FullPath: Gets the path from the root tree node to the current tree node.

Index: Gets the position of the tree node in the tree node collection.

IsEditing: Gets a value indicating whether the tree node is in an editable state.

IsExpanded:Gets a value indicating whether the tree node is in the expanded state.

IsSelected:Gets a value indicating whether the tree node is in the selected state.

IsVisible: Gets a value indicating whether the tree node is visible or partially visible.

LastNode: Gets the last child tree node.

Level: Gets the zero-based depth of the tree node in the TreeView control.

Name: Gets or sets the name of the tree node.

NextNode: Gets the next sibling tree node.

Nodes: Gets the collection of TreeNode objects assigned to the current tree node.

Parent: Gets the parent tree node of the current tree node.

PrevNode: Gets the previous sibling tree node.

PrevVisibleNode: Gets the previous visible tree node.

Tag :Gets or sets the object that contains data about the tree node. Text: Gets or sets the text displayed in the label of the tree node.

ToolTipText :Gets or sets the text that appears when the mouse pointer hovers over a TreeNode.

TreeView: Gets the parent tree view that the tree node is assigned to.

Example :

```
 <%@ Page Language="C#"
AutoEventWireup="true" CodeBehind="TreeView.aspx.cs"
 Inherits="TreeViewControl.TreeView" %>

 <!DOCTYPE html>

 <html xmlns="http://www.w3.org/1999/xhtml">
 <head runat="server">
 <title></title>
 </head>
<body>
 <form id="form1" runat="server">
 <div style="font-family: Arial">
 <asp:TreeView runat="server"
ID="TreeView1">
 <Nodes>
 <asp:TreeNode Text="Home"
```

```
NavigateUrl="~/Home.aspx" Target="_blank"/>
                                    <asp:TreeNode Text="Employee"
NavigateUrl="~/Employee.aspx" Target="_blank">
                                                 <asp:TreeNode Text="Upload Resume"
         NavigateUrl="~/Upload_Resume.aspx" Target="_blank" /> <asp:TreeNode
                                                 Text="Edit Resume"
          NavigateUrl="~/Edit_Resume.aspx" Target="_blank" /> <asp:TreeNode
          Text="View Resume"
NavigateUrl="~/View_Resume.aspx" Target="_blank" /> </asp:TreeNode>
                              <asp:TreeNode Text="Employer"
NavigateUrl="~/Employer.aspx" Target="_blank">
                                                    <asp:TreeNode Text="Upload Job"
          NavigateUrl="~/Upload_Job.aspx" Target="_blank" /> <asp:TreeNode
                                                  Text="Edit Job"
                                              NavigateUrl="~/Edit_Job.aspx" Target="_blank" />
                                                    <asp:TreeNode Text="View Job"
                                           NavigateUrl="~/View_Job.aspx" Target="_blank" />
 </asp:TreeNode>
 <asp:TreeNode Text="Admin"
NavigateUrl="~/Admin.aspx" Target="_blank">
                                                 <asp:TreeNode Text="Add User"
                                          NavigateUrl="~/Add_User.aspx" Target="_blank" />
                                                  <asp:TreeNode Text="Edit User"
                                           NavigateUrl="~/Edit_Use.aspx" Target="_blank" />
 <asp:TreeNode Text="View User"
NavigateUrl="~/View_User.aspx" Target="_blank" /> </asp:TreeNode>
 </Nodes>
</asp:TreeView>
 </div>
 </form>
</body>
 </html>
```

19. File upload control :

ASP.NET has two controls that allow users to upload files to the web server. Once the server receives the posted file data, the application can save it, check it, or ignore it. The following controls allow the file uploading:

- HtmlInputFile - an HTML server control
- FileUpload - and ASP.NET web control

Both controls allow file uploading, but the FileUpload control automatically sets the encoding of the form, whereas the HtmlInputFile does not do so.

In this tutorial, we use the FileUpload control. The FileUpload control allows the user to browse for and select the file to be uploaded, providing a browse button and a text box for entering the filename.

Once, the user has entered the filename in the text box by typing the name or browsing, the SaveAs method of the FileUpload control can be called to save the file to the disk.

Properties :

FileBytes :Returns an array of the bytes in a file to be uploaded.

FileContent: Returns the stream object pointing to the file to be uploaded.

FileName :Returns the name of the file to be uploaded.

PostedFile: Returns a reference to the uploaded file.

Example :

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

```
<div>
<h3> File Upload:</h3>
<br />
<asp:FileUpload ID="FileUpload1" runat="server" />
<br /><br />
<asp:Button ID="btnsave" runat="server" onclick="btnsave_Click" Text="Save"
style="width:85px" />
<br /><br />
<asp:Label ID="lblmessage" runat="server" /> </div>
</form> </body>
```

```
 Protected void UploadBtn_Click(object sender, EventArgs e)
        {
        if (FileUpLoad1.HasFile)
        {

       FileUpLoad1.SaveAs(@"C:\temp\" +
       FileUpLoad1.FileName);
        Label1.Text = "File Uploaded: " +
       FileUpLoad1.FileName;
       }
        else
        {
        Label1.Text = "No File Uploaded.";
        }
        }
```

20. Calendar control :

Birthdays, anniversaries, appointments, holidays, bill payments, and project deadlines. All these have one thing in common. Can you guess? It's a date. It is difficult to remember dates. That is where the calendar comes to rescue.

ASP.NET provides a Calendar control that is used to display a calendar on a Web page. ASP.NET Calendar control displays a month calendar that allows user to select dates and move to the next and previous months.

By default, this control displays the name of the current month, day headings for the days of the weeks, days of the month and arrow characters for navigation to the previous or next month.

Properties

DayNameFormat: style of the days of the week. Possible values are FirstLetter, FirstTwoLetters, Full, Short, and Shortest.

NextMonthText: the text that appears for the next month link.

NextPrevFormat: style of the next month and previous month link.

Possible values are

CustomText, FullMonth, and ShortMonth.

PrevMonthText: the text for the previous month link.

SelectedDate: get / set the selected date.

SelectedDates: get / set a collection of selected dates.

SelectionMode: how dates are selected. Possible values are Day, DayWeek, DayWeekMonth, and None.

SelectMonthText: text for selecting a month.

SelectWeekText: text for selecting a week.

ShowDayHeader: hide or display the day names.

ShowNextPrevMonth: hide or display the links for the next and previous months.

ShowTitle: hide or display the title bar displayed at the top of the calendar.

TitleFormat: format the title bar.

Possible values are Month and MonthYear.

TodaysDate: set current date. This property defaults to the current date.

VisibleDate:	set the month displayed. defaults to the month for TodaysDate.
DayRender:	Raised as each day is rendered.
SelectionChanged:	Raised when a new day, week, or month is selected
VisibleMonthChanged:	Raised when the next or previous month link is clicked.

Example :

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="calendardemo._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html
xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>
Untitled Page
</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h3> Your Birthday:</h3>
<asp:Calendar ID="Calendar1" runat="server
SelectionMode="DayWeekMonth"
onselectionchanged="Calendar1_SelectionChanged">
</asp:Calendar>
</div>
<p>Todays date is:
<asp:Label ID="lblday" runat="server"></asp:Label>
</p>
<p>Your Birday is:
```

```
<asp:Label ID="lblbday" runat="server"></asp:Label> </p>
</form>
</body>
</html>
protected void Calendar1_SelectionChanged(object sender,
EventArgs e) {
lblday.Text = Calendar1.TodaysDate.ToShortDateString();
lblbday.Text = Calendar1.SelectedDate.ToShortDateString(); }
```