UNIT-II

Anatomy of an ASP.NET Application:

- ASP.NET applications are generally divided into multiple Web pages. Every Web page in an ASP.NET application shares a common set of resources and configuration settings.
- Each ASP.NET application is executed inside a separate application domain. These application domains ensure that even if a Web application causes a fatal error, it does not affect other applications that are currently running on the same computer.
- Each Web application is a separate entity that has its own set of data. It can be described as a combination of files, pages, handlers, modules, and executable code that can be invoked from a virtual directory on a Web server.

Directories Used in ASP.NET Web Applications:

- **Bin:** Stores all the compiled .NET components (DLLs) that an ASP.NET Web application uses.
- **App_Code:** Stores source code files that are dynamically compiled to be used in the Web application.
- **App_Browsers:** Stores browser definition files.
- **App_GlobalResources:** Stores global resources that are accessible to every page in the Web application.
- **App_LocalResources:** Stores .resx files that are accessible to a specific page only.
- **App_Data:** Stores data such as database files and XML files that are used by the application.
- **App_Themes:** Stores the themes that are used in the Web application.

Files used in ASP.NET Web Applications:

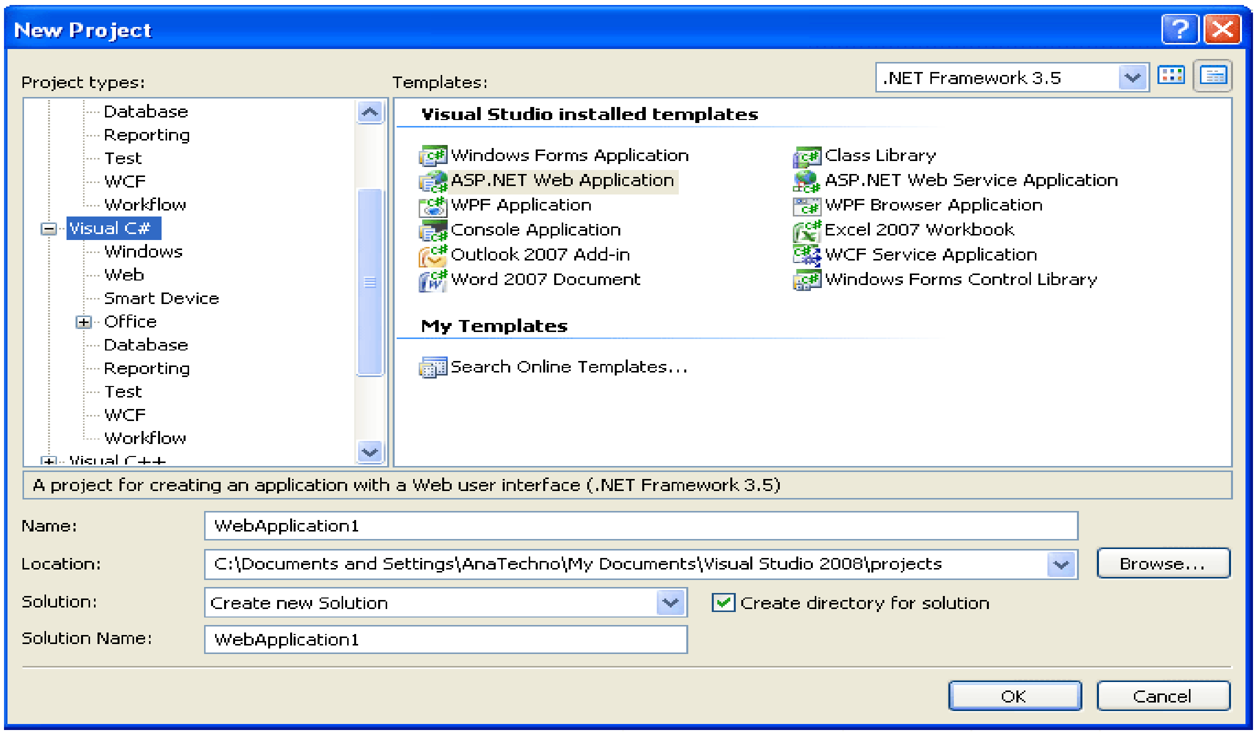
- **Ends with .aspx:** These are ASP.NET Web pages that contain the user interface and, optionally, the underlying code.
- **Ends with .ascx:** These are ASP.NET user controls.
- **Ends with .asmx:** These are ASP.NET Web Services.
- **web.config:** This is the XML-based configuration file for ASP.NET applications.
- **Global.asax:** This is the global application file.
- **Ends with .cs:** These are code-behind files that contain C# code.

Event & Event Handler:

- An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification.
- Events in ASP.NET raised at the client machine, and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.
- The server has a subroutine describing what to do when the event is raised; it is called the event-handler. Therefore, when the event message is transmitted to the server, it checks whether the Click event has an associated event handler. If it has, the event handler is executed.

Adding Event Handlers

Step:1: Create a web application as File->New ->Project :



A project for creating an application with a Web user interface (.NET Framework 3.5)

Name: WebApplication1

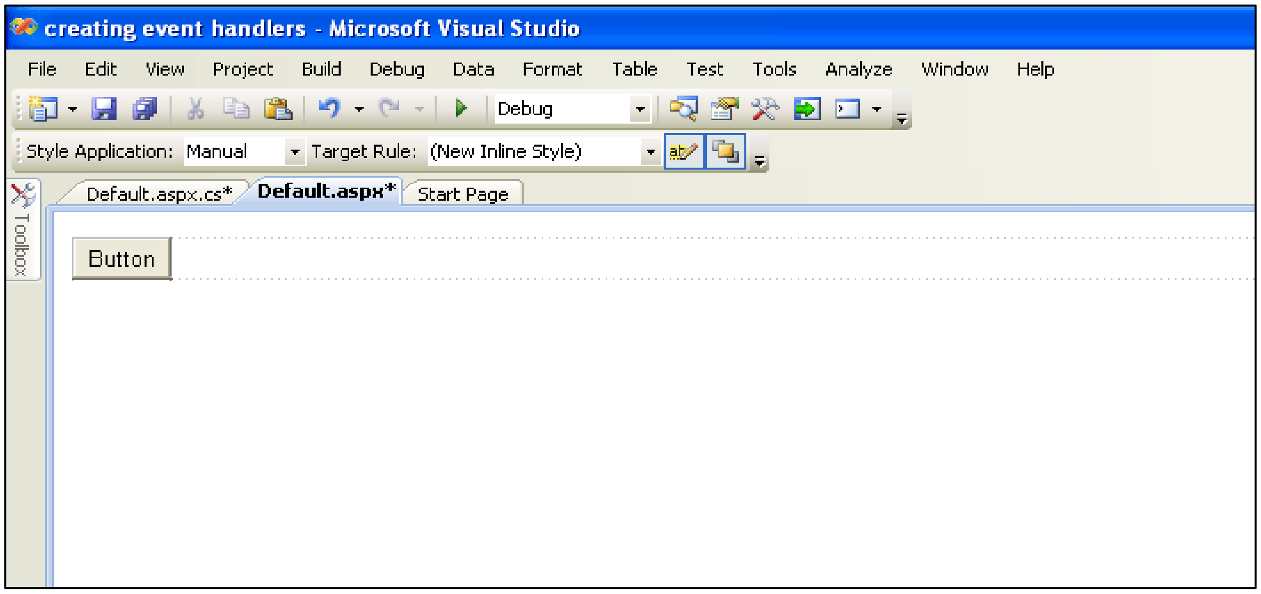
Location: C:\Documents and Settings\AnaTechno\My Documents\Visual Studio 2008\projects

Solution: Create new Solution

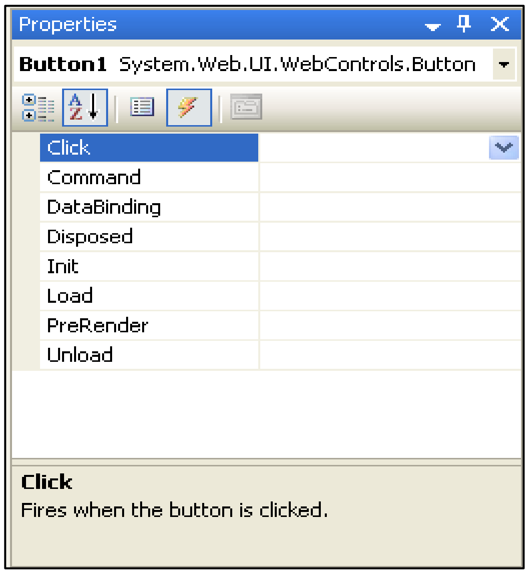
☑ Create directory for solution

Solution Name: WebApplication1

Step 2: Go to the design view and add a button control as:



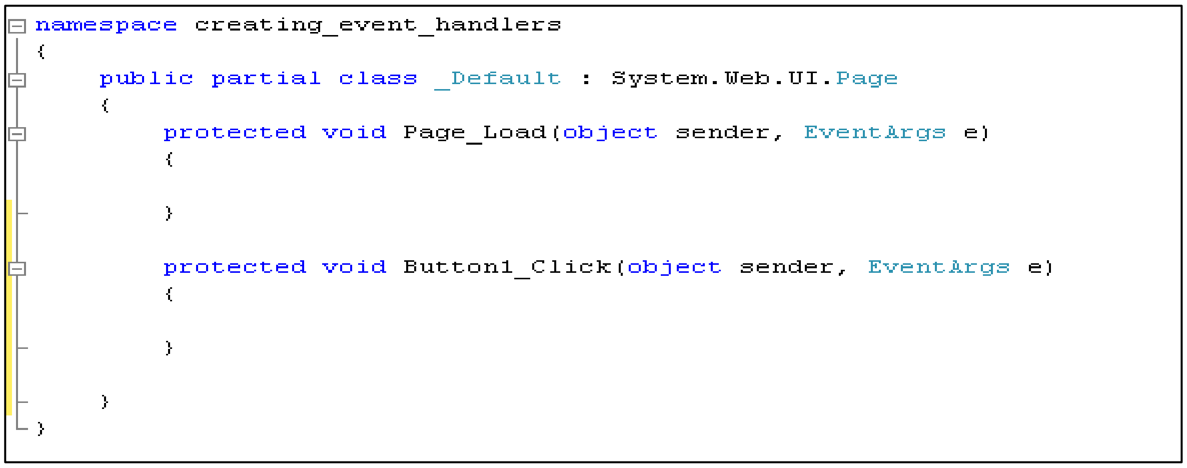
Step 3: Select the button and right click and click on properties. In the properties window click the yellow icon event. The window will change like the figure below:



Click
Command
DataBinding
Disposed
Init
Load
PreRender
Unload

Click
Fires when the button is clicked.

You can double click the the button, it will take you to code behind file where you can write the code if the user click button what action has to be performed. See the figure below:



```
namespace creating_event_handlers
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {

        }

    }
}
```

ASP.NET Controls:

- Controls are small building blocks of the graphical user interface, which include text boxes, buttons, check boxes, list boxes, labels, and numerous other tools.
- Using these tools, the users can enter data, make selections and indicate their preferences.
- Controls are also used for structural jobs, like validation, data access, security, creating master pages, and data manipulation.

ASP.NET uses 2 main types of web controls, which are:

- HTML controls
- Server controls

HTML CONTROLS:

- HTML server controls are HTML elements that contain attributes to accessible at server side.

- By default, HTML elements on an ASP.NET Web page are not available to the server.
- These components are treated as simple text and pass through to the browser.
- We can convert an HTML element to server control by adding a **runat= "server"** and an **id** attribute to the component.
- All the HTML Server controls can be accessed through the **Request** object.

Now, we can easily access it at code behind.

Example:
<input id="UserName" type="text" size="50"runat="server" **/>**

The following table contains commonly used HTML components.

Controls Name	Description
Button	It is used to create HTML button.
Reset Button	It is used to reset all HTML form elements.
Submit Button	It is used to submit form data to the server.
Text Field	It is used to create text input.
Text Area	It is used to create a text area in the html form.
File	It is used to create a input type = "file" component which is used to upload file to the server.
Password	It is a password field which is used to get password from the user.
CheckBox	It creates a check box that user can select or clear.
Radio Button	A radio field which is used to get user choice.
Table	It allows us to present information in a tabular format.
Image	It displays an image on an HTML form
ListBox	It displays a list of items to the user. You can set the size from two or more to specify how many items you wish to show.
Dropdown	It displays a list of items to the user in a dropdown list.
Horizontal Rule	It displays a horizontal line across the HTML page.

Example
Here, we are implementing an HTML server control in the form.

// htmlcontrolsexample.aspx

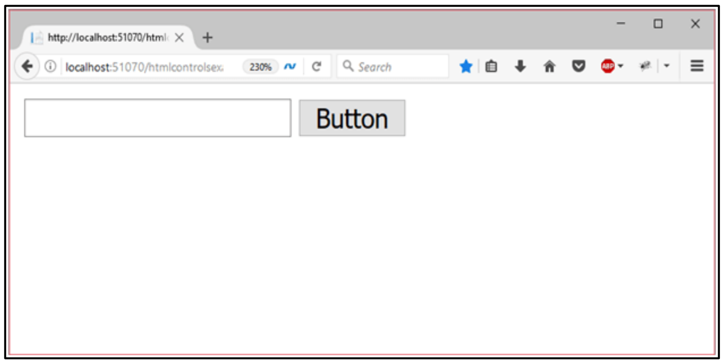
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="htmlcontrolsexample.aspx.cs"           Inherits="asp.n
etexample.htmlcontrolsexample" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<input id="Text1" type="text" runat="server"/>
<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click"/>
</div>
</form>
</body>
</html>
```

This application contains a code behind file.

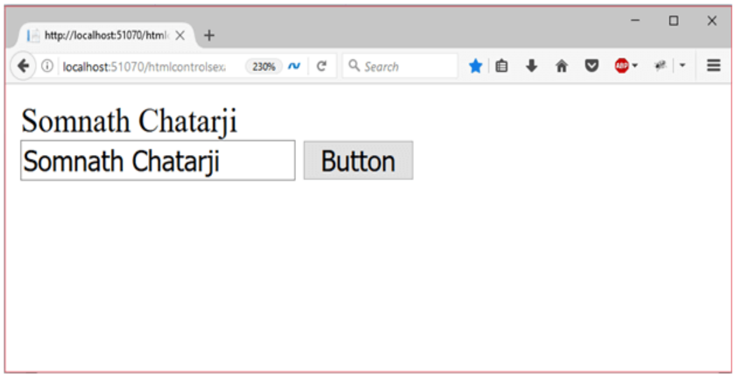
// htmlcontrolsexample.aspx.cs

```
using System;
namespace asp.netexample
{
public partial class htmlcontrolsexample : System.Web.UI.Page
   {
protected void Page_Load(object sender, EventArgs e)
       {
       }
protected void Button1_Click(object sender, EventArgs e)
       {
          string a = Request.Form["Text1"];
           Response.Write(a);
       }
   }
}
```

<u>**Output:**</u>



When we click the button after entering text, it responses back to client.



Somnath Chatarji

Somnath Chatarji

ASP.NET Server Controls

- 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.

The syntax for using server controls is:

```
<asp:controlType  ID ="ControlID" runat="server" Property1=value1  [Property2=value2] />
```

The following table contains the server controls for the web forms.

Control Name	Applicable Events	Description

Label	None	It is used to display text on the HTML page.
TextBox	TextChanged	It is used to create a text input in the form.
Button	Click, Command	It is used to create a button.
LinkButton	Click, Command	It is used to create a button that looks similar to the hyperlink.
ImageButton	Click	It is used to create an imagesButton. Here, an image works as a Button.
Hyperlink	None	It is used to create a hyperlink control that responds to a click event.
DropDownList	SelectedIndexChanged	It is used to create a dropdown list control.
ListBox	SelectedIndexCnhaged	It is used to create a ListBox control like the HTML control.
CheckBox	CheckChanged	It is used to create checkbox.
CheckBoxList	SelectedIndexChanged	It is used to create a group of check boxes that all work together.
RadioButton	CheckChanged	It is used to create radio button.
RadioButtonList	SelectedIndexChanged	It is used to create a group of radio button controls that all work together.
Image	None	It is used to show image within the page.
Panel	None	It is used to create a panel that works as a container.
PlaceHolder	None	It is used to set placeholder for the control.
Calendar	SelectionChanged, VisibleMonthChanged, DayRender	It is used to create a calendar. We can set the default date, move forward and backward etc.

AdRotator	AdCreated	It allows us to specify a list of ads to display. Each time the user re-displays the page.
Table	None	It is used to create table.
XML	None	It is used to display XML documents within the HTML.

Difference between HTML control and Web Server control.

HTML control	Web Server control
HTML control runs at client side.	ASP.Net controls run at server side.
We can run HTML controls at server side by adding attribute runat=”server”.	We cannot run ASP.Net Controls on client side as these controls have this attribute runat=”server” by default.
HTML controls are client sidecontrols , so it does not provide STATE management.	ASP.Net Controls are Server side controls, provides STATE management.
HTML control does not require rendering.	HTML control does not require rendering.
As HTML controls runs on client side,execution is fast.	As ASP.Net controls run on server side, execution is slow.
HTML controls does not support Object Oriented paradigm.	With ASP.Net controls, you have full support of Object oriented paradigm.
HTML control have limited set of properties and/or methods.	ASP.Net controls have rich set of properties and/or methods.
<input type="text" ID="txtName">	<asp:TextBoxId="txtName"runat="server"> </asp:TextBox>

LIST CONTROL

ASP.NET provides the following controls

1. Drop-down list,
2. List box,
3. Radio button list,
4. Check box list,
5. Bulleted list.

These controls display list of options to select. You can select one or more options, the choice depends upon control. They all derive from the System.Web.UI.WebControls.ListControl class

Common Properties of list control

Property	**Description**
Items	The collection of ListItem objects that represents the items in the control. This property returns an object of type ListItemCollection.
Rows	Specifies the number of items displayed in the box. If actual list contains more rows than displayed then a scroll bar is added.
SelectedIndex	The index of the currently selected item. If more than one item is selected, then the index of the first selected item. If no item is selected, the value of this property is -1.
SelectedValue	The value of the currently selected item. If more than one item is selected, then the value of the first selected item. If no item is selected, the value of this property is an empty string ("").
SelectionMode	Indicates whether a list box allows single selections or multiple selections.
Text	The text displayed for the item.
Selected	Indicates whether the item is selected.
Value	A string value associated with the item.

Method of list Controls

Methods	**Description**
Add(string)	Adds a new item at the end of the collection and assigns the string parameter to the Text property of the item.
Add(ListItem)	Adds a new item at the end of the collection.
Insert(integer, string)	Inserts an item at the specified index location in the collection, and assigns string parameter to the text property of the item.
Insert(integer, ListItem)	Inserts the item at the specified index location in the collection.
Remove(string)	Removes the item with the text value same as the string.
Remove(ListItem)	Removes the specified item.
RemoveAt(integer)	Removes the item at the specified index as the integer.
Clear	Removes all the items of the collection.

FindByValue(string)	Returns the item whose value is same as the string.
FindByValue(Text)	Returns the item whose text is same as the string.
Item(integer)	A ListItem object that represents the item at the specified index.
Count	The number of items in the collection.

1. **Drop down list:**

- The DropDownList is a web server control which is used to create an HTML Select component. It allows us to select an option from the dropdown list. It can contain any number of items
- ASP.NET provides a tag to create DropDownList for web application.
- The following is the Syntax of DropDownList tag.

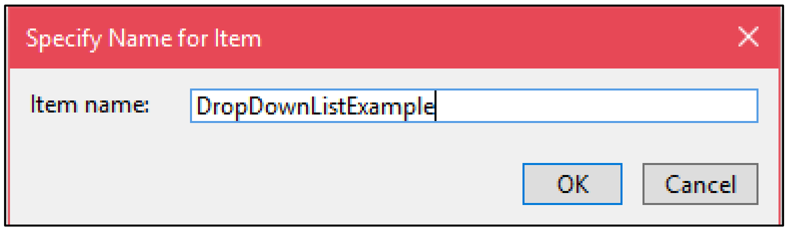
```
<asp:DropDownList id="DropDownList1" runat="server" >
  <asp:ListItem value="value" selected="True|False">
    Text
  </asp:ListItem>
</asp:DropDownList>
```

ASP.NET DropDownList Example

We are creating DropDownList by using Visual Studio 2017. This example includes the following steps.

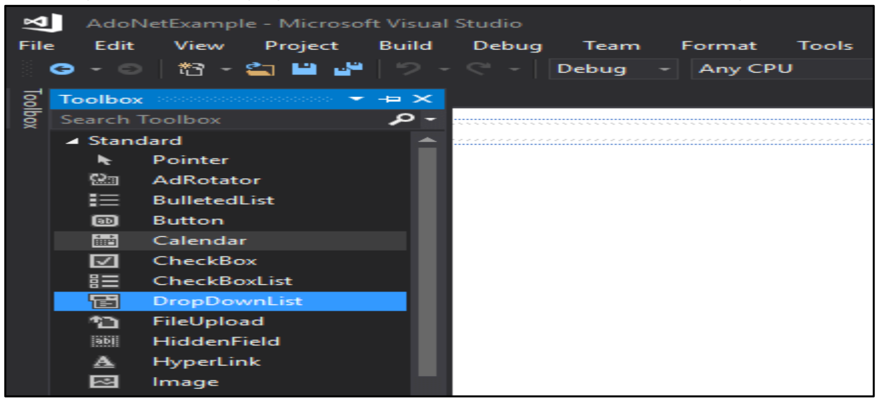
Create a Web Form

Add a new form by specifying its name.

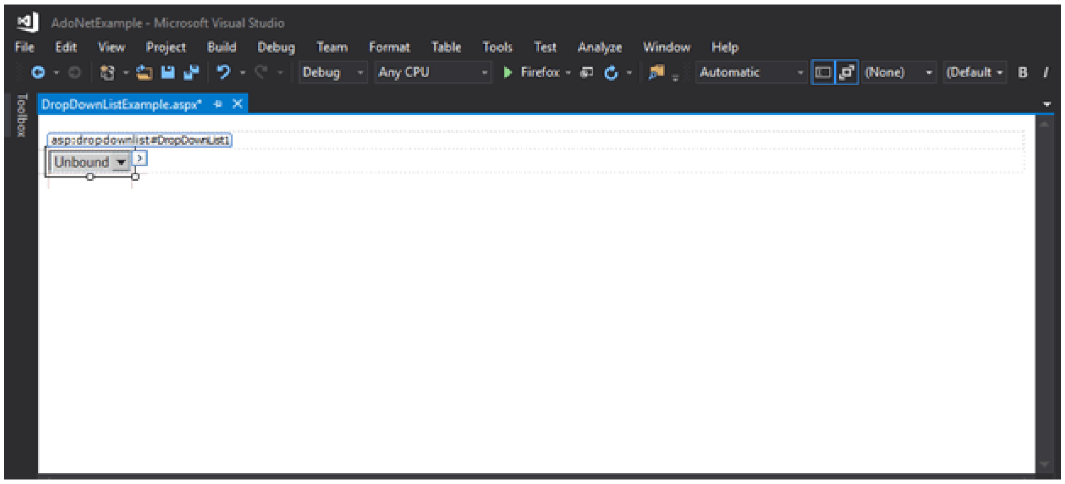


Item name: DropDownListExample

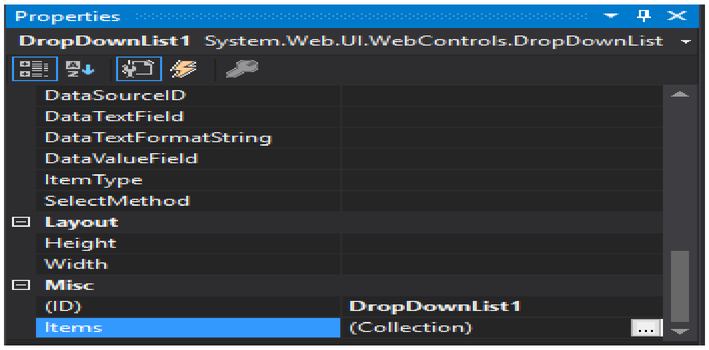
Initially, it is an empty form. Now, we will add a new DropDownList by dragging it from the toolbox.



After dragging, our web form looks like the below.

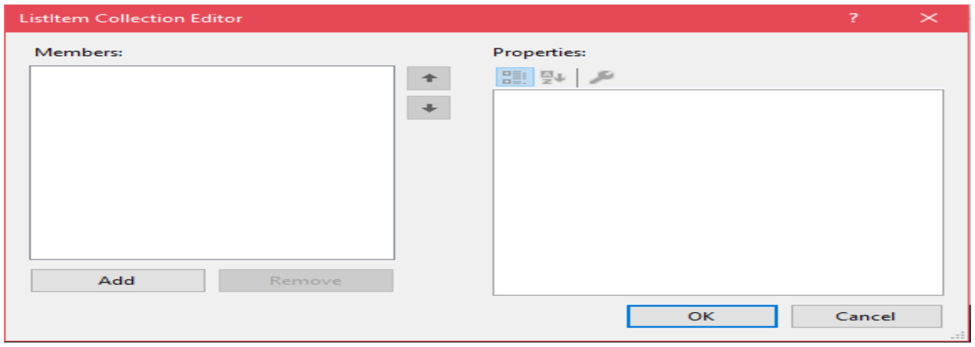

AdoNetExample - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Table Tools Test Analyze Window Help
Debug Any CPU Firefox Automatic (None) (Default B I
Toolbox
DropDownListExample.aspx*
asp:dropdownlist#DropDownList1
Unbound

Now, to add items to the list, visual studio provides **Items** property where we can add items. The property window looks like this.

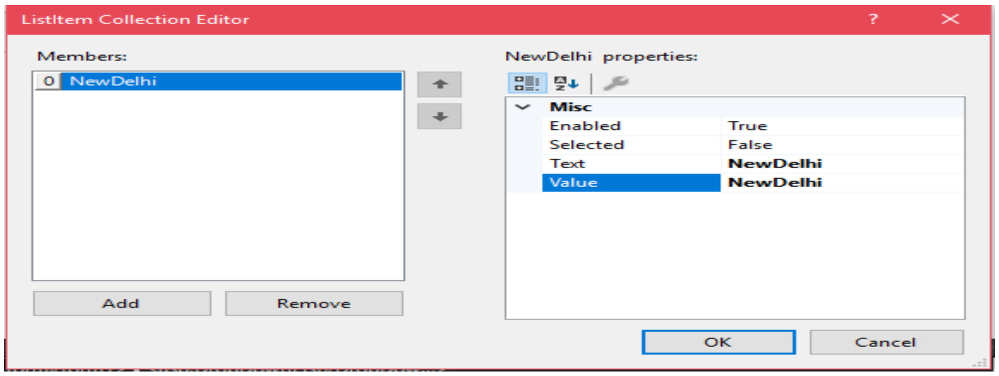


DataSourceID	
DataTextField	
DataTextFormatString	
DataValueField	
ItemType	
SelectMethod	
Layout	
Height	
Width	
Misc	
(ID)	DropDownList1
Items	(Collection)

Click on the items (collection) and it will pop up a new window as given below. Initially, it does not have any item. It provides **add** button to add new items to the list.



Adding item to the DropDownList, by providing values to the Text and Value properties.

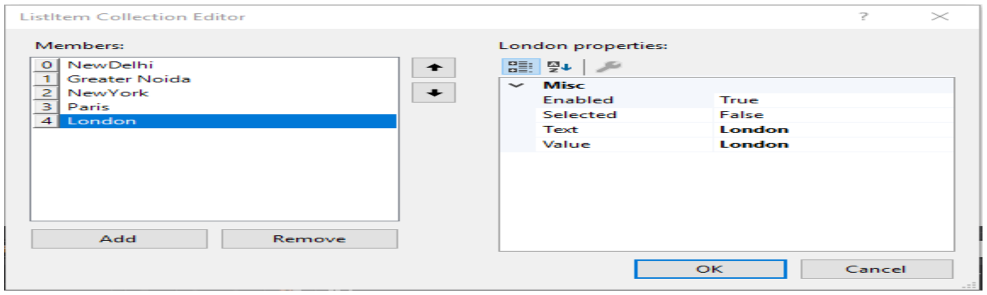


0 NewDelhi

NewDelhi properties:

Misc	
Enabled	True
Selected	False
Text	NewDelhi
Value	NewDelhi

We have added more items to it and now, it looks like the following.



0 NewDelhi
1 Greater Noida
2 NewYork
3 Paris
4 London

London properties:

Misc	
Enabled	True
Selected	False
Text	London
Value	London

DropDownListExample.aspx

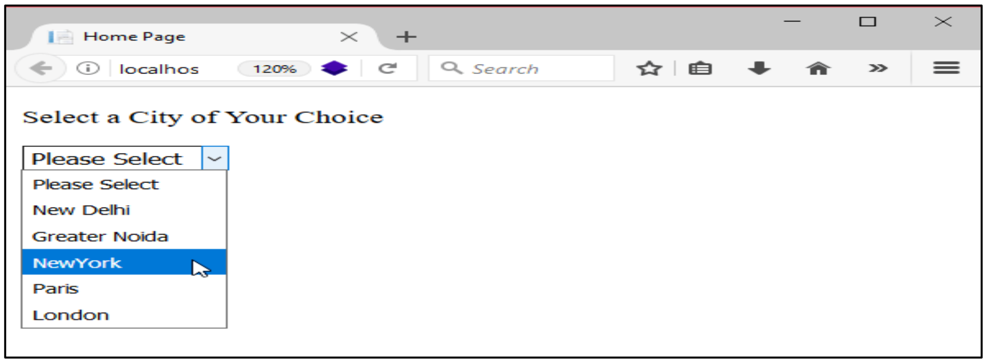
```
<%@ Page Title="Home Page" Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="DropDownListExample._Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <p>Select a City of Your Choice</p>
    <div>
      <asp:DropDownList ID="DropDownList1" runat="server" >
      <asp:ListItem Value="">Please Select</asp:ListItem>
      <asp:ListItem>New Delhi </asp:ListItem>
      <asp:ListItem>Greater Noida</asp:ListItem>
      <asp:ListItem>NewYork</asp:ListItem>
      <asp:ListItem>Paris</asp:ListItem>
      <asp:ListItem>London</asp:ListItem>
    </asp:DropDownList>
    </div>
    <br />
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Submit" />
    <br />
    <br />
```

```
        <asp:Label ID="Label1" runat="server" EnableViewState="False"></asp:Label>
    </form>
</body>
</html>
```

DropDownListExample.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace DropDownListExample
{
public partial class _Default : Page
{
protected void Page_Load(object sender, EventArgs e)
{
}
protected void Button1_Click(object sender, EventArgs e)
{
if (DropDownList1.SelectedValue == "")
{
Label1.Text = "Please Select a City";
}
else
Label1.Text = "Your Choice is: " + DropDownList1.SelectedValue;
}
}
}
```

__Output:__



Select a City of Your Choice

Please Select
Please Select
New Delhi
Greater Noida
NewYork
Paris
London

At server side, selected city is fetched and display to the user.

2. List Box:

- List box control in ASP.Net is a web control derived from the List Control Class, which in turn is derived from System.Web.UI.WebControls class.
- It allows the selection of single and multiple items in the list, unlike the dropdown list control, which enables the selection of a single item at a time; this list can also be bound to the data source.

Syntax:
The list box control can be dragged and dropped from the ASP.Net toolbox on the web form created, following the code in the .cs file.

```
<asp: ListBox id="ListBox1" Rows="6" Width="100px" SelectionMode="Single" runat="server"> </asp: ListBox>
```

Vice versa, writing the above code in a .cs file creates a list box control on the web form. The following code can be used to add the elements to the list.

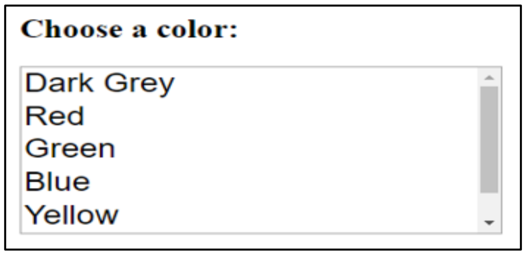
```
<asp: ListItem>Item 1</asp: ListItem>
```

Example
Code:

```
<%@ 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>
```

You can access the list and make necessary changes using the ID value “ListBox1” in the code.

Output:



```
Choose a color:
Dark Grey
Red
Green
Blue
Yellow
```

3. Radiobutton list:

- RadioButton List class is derived from the Web Control class which groups the radio button controls.
- This immensely helps programmers by providing a single selection radio button group.
- This group can also be generated dynamically with the help of data binding.

Syntax:

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
</asp:RadioButtonList>
```

The following code can be used to add the elements to the list.

```
<asp: ListItem>Item 1</asp: ListItem>
```

Example:
Radiobutton.Aspx

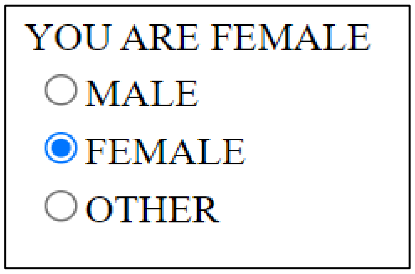
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="radiobuttonList.aspx.cs"
Inherits="controls_2024.radiobuttonList" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged">
            <asp:ListItem>MALE</asp:ListItem>
            <asp:ListItem>FEMALE</asp:ListItem>
            <asp:ListItem>OTHER</asp:ListItem>
```

```
        </asp:RadioButtonList>
        <br />
      </div>
   </form>
</body>
</html>
```

Radiobutton.aspx.cs

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
        Response.Write("YOU ARE " + RadioButtonList1.SelectedValue);
}
```

Output:



```
YOU ARE FEMALE
 ○ MALE
 ● FEMALE
 ○ OTHER
```

4. Checkbox list:

- The CheckBoxList class is derived from the class System.Web.UI.WebControls.ListControls. ASP.NET CheckBoxList is a web control that can be used to collate the items that can be checked, thus giving the user the ability to select multiple items simultaneously.
- This list of items in the CheckBoxList can be dynamically generated using the Data Binding functions.
- This immensely helps programmers by providing a single selection radio button group.
- This group can also be generated dynamically with the help of data binding.

Syntax:

The checkboxlist can be created using the design section by dragging and dropping the control from the ASP.NET toolbar window, or else it can also create from the markup section using the following code.

```
<asp:CheckBoxList id= "checkboxlist1" AutoPostBack = "True" TextAlign = "Right" OnSelectedIndexChanged =
"CheckList_Clicked" runat= "server">

<asp:ListItem> Item 1 </asp:ListItem>

<asp:ListItem> Item 2 </asp:ListItem>

<asp:ListItem> Item 3 </asp:ListItem>
```

```
</asp:checkboxlist>
```

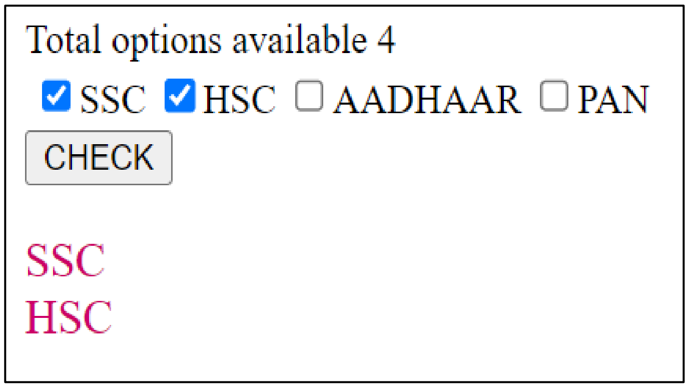
Example:
Checkboxlist_control.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="checkboxlist_control.aspx.cs"
Inherits="controls_2024.checkboxlist_control" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <asp:CheckBoxList ID="CheckBoxList1" runat="server" RepeatDirection="Horizontal">
            <asp:ListItem>SSC</asp:ListItem>
            <asp:ListItem>HSC</asp:ListItem>
            <asp:ListItem>AADHAAR</asp:ListItem>
            <asp:ListItem>PAN</asp:ListItem>
         </asp:CheckBoxList>
         <asp:Button ID="Button1" runat="server" Text="CHECK" OnClick="Button1_Click" />
         <br />
         <br />
         <asp:Label ID="Label1" runat="server" Font-Size="Larger" ForeColor="#CC0066"
Text="Label"></asp:Label>
         <br />
      </div>
   </form>
</body>
</html>
```

Checkboxlist_control.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
      {
         Response.Write("Total options available " + CheckBoxList1.Items.Count);

         string n = " ";
         for(int i=0;i< CheckBoxList1.Items.Count;i++)
         {
            if (CheckBoxList1.Items[i].Selected)
            {

              n = n + CheckBoxList1.Items[i].Text + "</br>";
            }
        }
        Label1.Text = n;
    }
```

OUTPUT



```
Total options available 4
[✓] SSC [✓] HSC [ ] AADHAAR [ ] PAN
CHECK

SSC
HSC
```

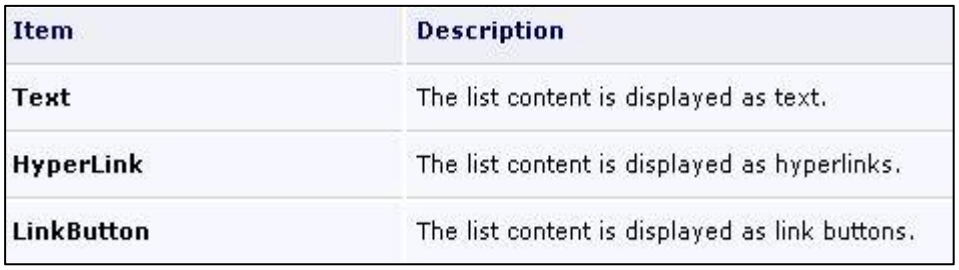
5. Bulleted List

- BulletedList control is very rich in displaying the items in different styles.
- It displays the list either in unordered or ordered list.
- Each list item can be rendered as plain text, a LinkButton control, or a link to another web page.
- BulletedList control supports the BulletStyle property.
- The default value of BulletStyle property is NotSet and rendered as in list of bulleted items.

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:



Item	Description
Text	The list content is displayed as text.
HyperLink	The list content is displayed as hyperlinks.
LinkButton	The list content is displayed as link buttons.

The following code creates a BulletedList with quare bullet.

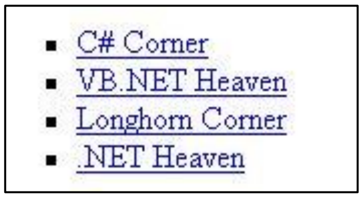
```
<asp:BulletedList ID=BulletedList1 BulletStyle=Square runat=server>

<asp:ListItem>C# Corner</asp:ListItem>
<asp:ListItem>VB.NET Heaven</asp:ListItem>
<asp:ListItem>Longhorn Corner</asp:ListItem>
<asp:ListItem>.NET Heaven</asp:ListItem>

</asp:BulletedList>
```

BulletedListDisplayMode enumeration

The following code creates a bulleted list with hyperlinks.

```
<asp:BulletedList ID=BulletedList1 DisplayMode=HyperLink BulletStyle=Square runat=server>

<asp:ListItem Value="www.c-sharpcorner.com">C# Corner</asp:ListItem>
<asp:ListItem Value="www.vbdotnetheaven.com">VB.NET Heaven</asp:ListItem>
<asp:ListItem Value="www.longhorncorner.com">Longhorn Corner</asp:ListItem>
<asp:ListItem Value="www.dotnetheaven.com">.NET Heaven</asp:ListItem>

</asp:BulletedList>
```

Output



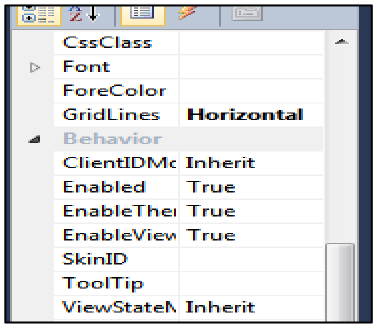
▪ C# Corner
▪ VB.NET Heaven
▪ Longhorn Corner
▪ .NET Heaven

Table Controls:

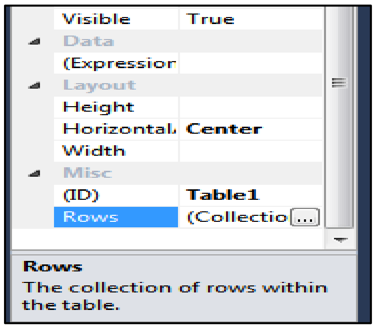
- Table control are used to divide a page into several rows and columns to arrange the information, or The Table control is used in conjunction with the TableCell control and the TableRow control to create a table.

Properties: These are the following properties of the table control.

- Caption: The caption of the table.
- CaptionAlign: The alignment of the caption text. CellPadding: The space between the cell walls and contents. CellSpacing: The space between cells.
- GridLines: The gridline format in the table.
- ForeColor: Color of the text within the control.
- runat: Specifies that the control is a server control. Must be set to "server" .



```
CssClass
Font
ForeColor
GridLines      Horizontal
Behavior
ClientIDMc     Inherit
Enabled        True
EnableThe      True
EnableView     True
SkinID
ToolTip
ViewStateN     Inherit
```



```
Visible        True
Data
(Expressior
Layout
Height
Horizontal     Center
Width
Misc
(ID)           Table1
Rows           (Collectio [...]

Rows
The collection of rows within
the table.
```

Table: ASP.NET table and its Helper Control Classes

Manages a collection of table cells such as adding a cell to a row or removing a cell from it.

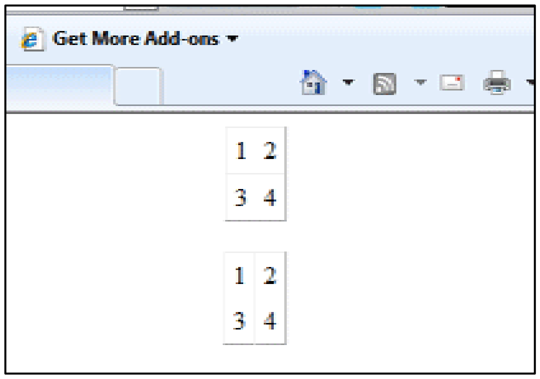
CONTROL	CODE	DESCRIPTION
Table	<asp:Table>	The System.Web.UI.Table class encapsulates an HTML table. An HTML table control, used to creates a table with the help of TableRow and TableCell.
TableRow	<asp:TableRow>	The System.Web.UI.TableRow class encapsulates a row within a table, which later can be used to get or set row's cells values using TableCell.
TableCell	<asp:TableCell>	The System.Web.UI.TableCell class encapsulates a cell within a table.
TableRowCollection	<asp:TableRowCollection>	The System.Web.UI.TableCell class encapsulates a TableRowCollection and is used to manage a collection of table a collection or removing a row from it.

TableCellCollection	<asp:TableCellCollection>	Manages a collection of table cells such as adding a cell to a row or removing a cell from it.
TableHeaderCollection	<asp:TableHeaderCell>	Encapsulate a table header cell.

Program:

drag two table control from the toolbox.

```
<form id="form1" runat="server">
<asp:Table ID="Table1" runat="server" CellPadding="5" GridLines="horizontal" HorizontalAlign="Center">
<asp:TableRow>
<asp:TableCell>1</asp:TableCell>
<asp:TableCell>2</asp:TableCell>
</asp:TableRow>
<asp:TableRow>
<asp:TableCell>3</asp:TableCell>
<asp:TableCell>4</asp:TableCell>
</asp:TableRow>
</asp:Table>
<br />
<asp:Table ID="Table2" runat="server" CellPadding="5" GridLines="vertical" HorizontalAlign="Center">
<asp:TableRow>
<asp:TableCell>1</asp:TableCell>
<asp:TableCell>2</asp:TableCell>
</asp:TableRow>
<asp:TableRow>
<asp:TableCell>3</asp:TableCell>
<asp:TableCell>4</asp:TableCell>
</asp:TableRow>
</asp:Table>
</form>

</body>
</html>
```



1	2
3	4

1	2
3	4

Web Control Events:

- An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification. A process communicates through events. For example, interrupts are system-generated events. When events occur, the application should be able to respond to it and manage it.
- Events in ASP.NET raised at the client machine, and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.
- The server has a subroutine describing what to do when the event is raised; it is called the event-handler. Therefore, when the event message is transmitted to the server, it checks whether the Click event has an associated event handler. If it has, the event handler is executed.

Event Arguments

ASP.NET event handlers generally take two parameters and return void. The first parameter represents the object raising the event and the second parameter is event argument.
The general syntax of an event is:

```
private void EventName (object sender, EventArgs e);
```

Page and Control Events

Common page and control events are:

- **DataBinding** - It is raised when a control binds to a data source.
- **Disposed** - It is raised when the page or the control is released.
- **Error** - It is a page event, occurs when an unhandled exception is thrown.
- **Init** - It is raised when the page or the control is initialized.
- **Load** - It is raised when the page or a control is loaded.
- **PreRender** - It is raised when the page or the control is to be rendered.
- **Unload** - It is raised when the page or control is unloaded from memory.

The common control events are:

Event	**Attribute**	**Controls**
Click	OnClick	Button, image button, link button, image map
Command	OnCommand	Button, image button, link button
TextChanged	OnTextChanged	Text box
SelectedIndexChanged	OnSelectedIndexChanged	Drop-down list, list box, radio button list, check box list.
CheckedChanged	OnCheckedChanged	Check box, radio button

Some events cause the form to be posted back to the server immediately, these are called the postback events.
For example, the click event such as, Button.Click.
Some events are not posted back to the server immediately, these are called non-postback events.
For example, the change events or selection events such as TextBox.TextChanged or CheckBox.CheckedChanged. The nonpostback events could be made to post back immediately by setting their AutoPostBack property to true.

Default Events

The default event for the Page object is Load event. Similarly, every control has a default event. For example, default event for the button control is the Click event.
The default event handler could be created in Visual Studio, just by double clicking the control in design view.
The following table shows some of the default events for common controls:

Control	**Default Event**
AdRotator	AdCreated
BulletedList	Click
Button	Click
Calender	SelectionChanged
CheckBox	CheckedChanged
CheckBoxList	SelectedIndexChanged
DataGrid	SelectedIndexChanged
DataList	SelectedIndexChanged
DropDownList	SelectedIndexChanged
HyperLink	Click
ImageButton	Click
ImageMap	Click
LinkButton	Click
ListBox	SelectedIndexChanged
Menu	MenuItemClick
RadioButton	CheckedChanged
RadioButtonList	SelectedIndexChanged

AUTOPOSTBACK EVENT

In ASP.NET, **AutoPostBack** is a property available in certain web controls, such as TextBox, DropDownList, CheckBox, and others. When the AutoPostBack property is set to true, it causes the page to automatically post back to the server when the user interacts with the control, triggering the appropriate server-side event.

- **Without AutoPostBack:** When AutoPostBack is set to false (the default), changes made to the control will not immediately trigger a postback. The user would typically have to submit the form (e.g., by clicking a button) to send the changes to the server.
- **With AutoPostBack:** When AutoPostBack is set to true, the control automatically sends the form data to the server as soon as its value changes. For example, if you have a DropDownList with AutoPostBack enabled, selecting a different item will cause the page to post back immediately, executing any server-side event handlers associated with that control.

Here's a simple example with a DropDownList: asp

```
<asp:DropDownList ID="ddlExample" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ddlExample_SelectedIndexChanged">
   <asp:ListItem Text="Option 1" Value="1"></asp:ListItem>
   <asp:ListItem Text="Option 2" Value="2"></asp:ListItem>
   <asp:ListItem Text="Option 3" Value="3"></asp:ListItem>
</asp:DropDownList>
```

- In this example, whenever the user selects a different item from the DropDownList, the page will automatically post back to the server,
- and the ddlExample_SelectedIndexChanged event handler will be executed.

Validation:

ASP.NET provides several types of validation controls that can be used to validate user input in web forms. Some of the common validation controls are:

The below table describes the controls and their use.

Validation Control	Description
RequiredFieldValidation	This control ensures that a field is not left empty or blank. It can be used for textboxes, dropdown lists, checkboxes, and other input controls.
CompareValidator	This control compares the value of one input control to another. It can validate passwords, confirm email addresses, and other scenarios where two values must match.
RangeValidator	This control checks if a value falls within a specific range. For example, it can be used to validate a user's age, income, or date of birth.
RegularExpressionValidator	This control ensure that user input matches a specified regular expression pattern. This validator is particularly useful for

	enforcing formats like email addresses, phone numbers, postal codes, and other pattern-based input requirements.
ValidationSummary	This control displays a report of all validation errors that occurred on a Web page.

Important points for validation controls

- ControlToValidate property is mandatory for all validate controls.
- One validation control will validate only one input control, but multiple validation control can be assigned to the input control.
- To use any validation control following code must be added in web.config file.

```
<configuration>
        <appSettings>
        <add key="ValidationSettings:UnobtrusiveValidationMode" value="None"> </add>
      </appSettings>
</ configuration>
```

1. ASP.NET RequiredFieldValidation Control

- The RequiredFieldValidator control is simple validation control that checks to see if the data is entered for the input control. You can have a RequiredFieldValidator control for each form element you wish to enforce the mandatory field rule. It has foolowing properties
  - ControlToValidate
  - ErrorMessage

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"

Style="top: 98px; left: 367px; position: absolute; height: 26px; width: 162px"

ErrorMessage="password required" ControlToValidate="TextBox2">

</asp:RequiredFieldValidator>
```

2. ASP.NET CompareValidator Control

- The CompareValidator control allows you to make comparisons to compare data entered in an input control with a constant value or a value in a different control.
- It can most commonly be used when you need to confirm the password entered by the user at registration time. The data is always case-sensitive.
- It has following properties
  - ControlToValidate
  - ControlToCompare
  - ErrorMessage

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ErrorMessage="password
required" ControlToValidate="TextBox3" ControlToCompare="Textbox2">
</asp:RequiredFieldValidator>
```

3. ASP.NET RangeValidator Control

- The RangeValidator Server Control is another validator control that checks to see if a control value is within a valid range.
- The attributes necessary for this control are MaximumValue, MinimumValue, and Type.

```
<asp:RangeValidator ID="RangeValidator1" runat="server" ErrorMessage="RangeValidator"
ControlToValidate="TextBox4" MaximumValue="100" MinimumValue="18" Type="Integer">

</asp:RangeValidator>
```

4. RegularExpressionValidator

- This control ensure that user input matches a specified regular expression pattern. This validator is particularly useful for enforcing formats like email addresses, phone numbers, postal codes, and other pattern-based input requirements.

  Key Properties of RegularExpressionValidator
- **ControlToValidate**: Specifies the ID of the input control that the validator is associated with (e.g., TextBox).
- **ValidationExpression**: The regular expression pattern that the input value must match.
- **ErrorMessage**: The message displayed to the user if the input does not match the specified pattern.

Rich Controls:

ASP.NET provides large set of controls. These controls are divided into different categories, depends upon their functionalities. The followings control comes under the rich controls category.

1. FileUpload control
2. Calendar control
3. AdRotator control
4. MultiView control

1. FileUpload Control:

- The FileUpload control in ASP.NET is used to allow users to upload files from their local computer to the web server.
- It is a simple and powerful control that handles the file upload process, including selecting the file, sending it to the server, and saving it.

Key Properties of FileUpload Control

- **FileName**: Gets the name of the file that the user selected for upload.
- **HasFile**: Returns true if a file is selected for upload; otherwise, false.
- **PostedFile**: Gets a reference to the uploaded file, providing access to properties like ContentLength, ContentType, and InputStream.

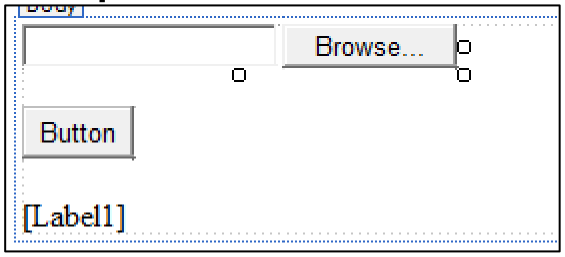
Key Methods

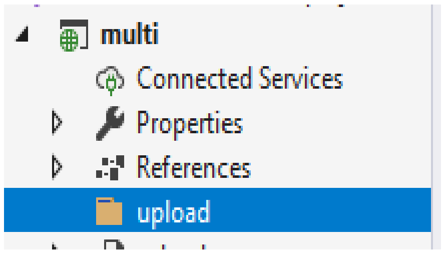
- **SaveAs(string filename):** Saves the uploaded file to the specified path on the server.

This is a server side control and ASP.NET provides own tag to create it. The example is given below.

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

Example:





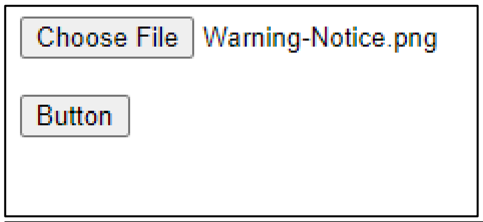
File_upload.aspx file

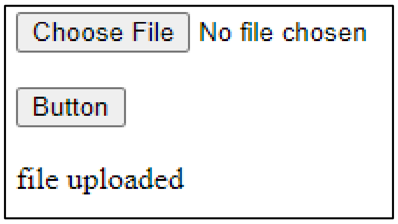
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="file uplaod.aspx.cs" Inherits="multi.file_uplaod" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:FileUpload ID="FileUpload1" runat="server" />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
        </div>
    </form>
</body>
</html>
```

File_upload.aspx.cs file:

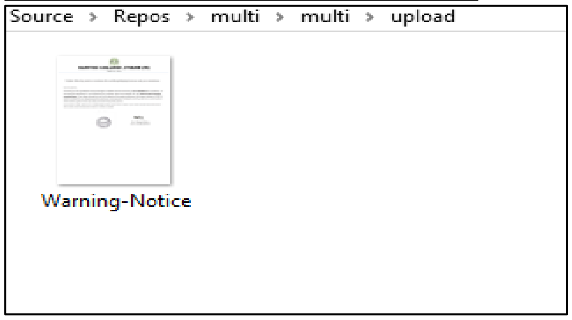
```
protected void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpload1.HasFile)
        {
            //Label1.Text = FileUpload1.FileName;
            string ex = System.IO.Path.GetExtension(FileUpload1.FileName);
            if(ex.ToLower() != ".png" || ex.ToLower() != ".png")
            {
                Label1.Text = "file type not supported";
            }
            else
            {
                FileUpload1.SaveAs(Server.MapPath("~/upload/" + FileUpload1.FileName));
                Label1.Text = "file uploaded";
            }
        }
            else
            Label1.Text = "Upload a file";
    }
```

Output:





file uploaded



2. Calendar Control:

The calendar control is a functionally rich web control, which provides the following capabilities:

- Displaying one month at a time
- Selecting a day, a week or a month
- Selecting a range of days
- Moving from month to month
- Controlling the display of the days programmatically

The basic syntax of a calendar control is:

```
<asp:Calender ID = "Calendar1" runat = "server"> </asp:Calender>
```

Properties and Events of the Calendar Control

The calendar control has many properties and events, using which you can customize the actions and display of the control. The following table provides some important properties of the Calendarcontrol:

Properties	Description
Caption	Gets or sets the caption for the calendar control.
CaptionAlign	Gets or sets the alignment for the caption.
CellPadding	Gets or sets the number of spaces between the data and the cell border.
CellSpacing	Gets or sets the space between cells.
DayHeaderStyle	Gets the style properties for the section that displays the day of the week.
DayNameFormat	Gets or sets format of days of the week.
DayStyle	Gets the style properties for the days in the displayed month.
FirstDayOfWeek	Gets or sets the day of week to display in the first column.
NextMonthText	Gets or sets the text for next month navigation control. The default value is >.
NextPrevFormat	Gets or sets the format of the next and previous month navigation control.
OtherMonthDayStyle	Gets the style properties for the days on the Calendar control that are not in thedisplayed month.

PrevMonthText	Gets or sets the text for previous month navigation control. The default value is <.
SelectedDate	Gets or sets the selected date.
SelectedDates	Gets a collection of DateTime objects representing the selected dates.
SelectedDayStyle	Gets the style properties for the selected dates.
SelectionMode	Gets or sets the selection mode that specifies whether the user can select asingle day, a week or an entire month.
SelectMonthText	Gets or sets the text for the month selection element in the selector column.
SelectorStyle	Gets the style properties for the week and month selector column.
SelectWeekText	Gets or sets the text displayed for the week selection element in the selectorcolumn.
ShowDayHeader	Gets or sets the value indicating whether the heading for the days of the weekis displayed.
ShowGridLines	Gets or sets the value indicating whether the gridlines would be shown.
ShowNextPrevMonth	Gets or sets a value indicating whether next and previous month navigationelements are shown in the title section.
ShowTitle	Gets or sets a value indicating whether the title section is displayed.
TitleFormat	Gets or sets the format for the title section.
Titlestyle	Get the style properties of the title heading for the Calendar control.

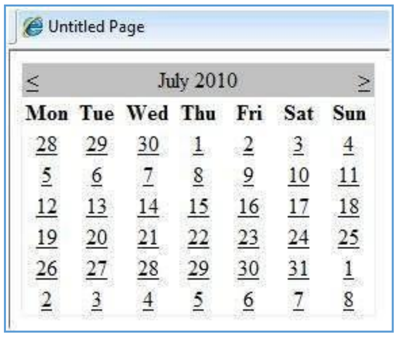
TodayDayStyle	Gets the style properties for today's date on the Calendar control.
TodaysDate	Gets or sets the value for today's date.
VisibleDate	Gets or sets the date that specifies the month to display.
WeekendDayStyle	Gets the style properties for the weekend dates on the Calendar control.

The Calendar control has the following three most important events that allow the developers to program the calendar control. They are:

Events	**Description**
SelectionChanged	It is raised when a day, a week or an entire month is selected.
DayRender	It is raised when each data cell of the calendar control is rendered.
VisibleMonthChanged	It is raised when user changes a month.

Working with the Calendar Control

Putting a bare-bone calendar control without any code behind file provides a workable calendar to a site, which shows the months and days of the year. It also allows navigation to next and previous months.



< July 2010 >

Mon	Tue	Wed	Thu	Fri	Sat	Sun
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

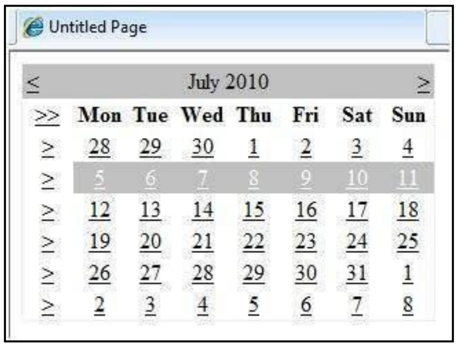
Calendar controls allow the users to select a single day, a week, or an entire month. This is done by using the SelectionMode property. This property has the following values:

Properties	Description
Day	To select a single day.
DayWeek	To select a single day or an entire week.
DayWeekMonth	To select a single day, a week, or an entire month.
None	Nothing can be selected.

The syntax for selecting days:

```
<asp:Calender ID = "Calendar1" runat = "server"SelectionMode="DayWeekMonth">
</asp:Calender>
```

When the selection mode is set to the value DayWeekMonth, an extra column with the > symbol appears for selecting the week, and a >> symbol appears to the left of the days name for selecting the month.



Example

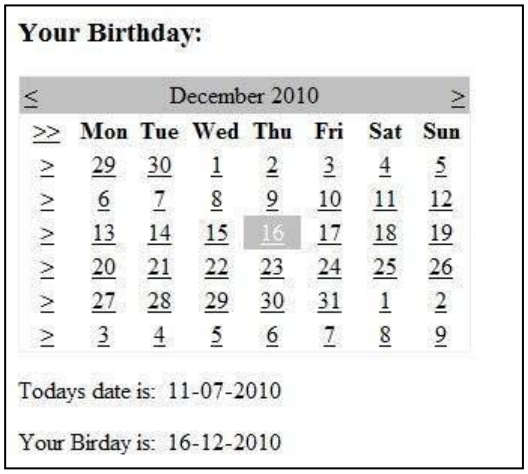
The following example demonstrates selecting a date and displays the date in a label: The content file code is as follows:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="calendardemo._Default" %>

<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 Birthday is:
        <asp:Label ID="lblbday" runat="server">
        </asp:Label>
      </p>
    </form>
  </body>
</html>
```

The event handler for the event SelectionChanged:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
  lblday.Text = Calendar1.TodaysDate.ToShortDateString();
  lblbday.Text = Calendar1.SelectedDate.ToShortDateString();
}
```

When the file is run, it should produce the following output:



```
Your Birthday:

<            December 2010            >
>>  Mon Tue Wed Thu Fri Sat Sun
>   29  30  1   2   3   4   5
>   6   7   8   9   10  11  12
>   13  14  15  16  17  18  19
>   20  21  22  23  24  25  26
>   27  28  29  30  31  1   2
>   3   4   5   6   7   8   9

Todays date is: 11-07-2010

Your Birday is: 16-12-2010
```

3. Adrotator Control:

- The AdRotator control randomly selects banner graphics from a list, which is specified in an external XML schedule file. This external XML schedule file is called the advertisement file.
- The AdRotator control allows you to specify the advertisement file and the type of window that the link should follow in the AdvertisementFile and the Target property respectively.

The basic syntax of adding an AdRotator is as follows:

```
<asp:AdRotator  runat = "server" AdvertisementFile = "adfile.xml"  Target =  "_blank" />
```

Before going into the details of the AdRotator control and its properties, let us look into the construction of the advertisement file.

The Advertisement File

- The advertisement file is an XML file, which contains the information about the advertisements to be displayed.
- Extensible Markup Language (XML) is a W3C standard for text document markup. It is a text-based markup language that enables you to store data in a structured format by using meaningful tags. The term 'extensible' implies that you can extend your ability to describe a document by defining meaningful tags for the application.
- XML is not a language in itself, like HTML, but a set of rules for creating new markup languages. It is a meta-markup language. It allows developers to create custom tag sets for special uses. It structures, stores, and transports the information.

Following is an example of XML file:

```
<BOOK>
  <NAME> Learn XML </NAME>
  <AUTHOR> Samuel Peterson </AUTHOR>
  <PUBLISHER> NSS Publications </PUBLISHER>
  <PRICE> $30.00</PRICE>
</BOOK>
```

Like all XML files, the advertisement file needs to be a structured text file with well-defined tags delineating the data. There are the following standard XML elements that are commonly used in the advertisement file:

Element	Description
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.

NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

The following code illustrates an advertisement file ads.xml.Add an xml file from solution explorer and write following code into that.

```
<Advertisements>
  <Ad>
    <ImageUrl>rose1.jpg</ImageUrl>
    <NavigateUrl>http://www.1800flowers.com</NavigateUrl>
    <AlternateText>
      Order flowers, roses, gifts and more
    </AlternateText>
    <Impressions>20</Impressions>
    <Keyword>flowers</Keyword>
  </Ad>

  <Ad>
    <ImageUrl>rose2.jpg</ImageUrl>
    <NavigateUrl>http://www.babybouquets.com.au</NavigateUrl>
    <AlternateText>Order roses and flowers</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>gifts</Keyword>
  </Ad>

  <Ad>
    <ImageUrl>rose3.jpg</ImageUrl>
    <NavigateUrl>http://www.flowers2moscow.com</NavigateUrl>
    <AlternateText>Send flowers to Russia</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>russia</Keyword>
  </Ad>

  <Ad>
    <ImageUrl>rose4.jpg</ImageUrl>
    <NavigateUrl>http://www.ediblebblooms.com</NavigateUrl>
    <AlternateText>Edible Blooms</AlternateText>
    <Impressions>20</Impressions>
    <Keyword>gifts</Keyword>
  </Ad>
</Advertisements>
```

Working with AdRotator Control

Create a new web page and place an AdRotator control on it and set its advertisement file property.

```
<form id="form1" runat="server">
  <div>
    <asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile  ="~/ads.xml" />
  </div>
</form>
```

4. Multiple Views:

- MultiView and View controls allow you to divide the content of a page into different groups, displaying only one group at a time. Each View control manages one group ofcontent and all the View controls are held together in a MultiView control.
- The MultiView control is responsible for displaying one View control at a time. The View displayed is called the active view.
- We can see single view at a time on web page by specify the ActiveViewIndex property of multiview control.

- All view control assign automatically index to all it, the index always start from zero. The first view1 index is zero, second is one and so on..
- If we want to display first view on web page, then we need to write MultiView1.ActiveViewIndex=0.

The syntax of MultiView control is:

```
<asp:MultView ID= "MultiView1" runat= "server"> </asp:MultiView>
```

The syntax of View control is:

```
<asp:View ID= "View1" runat= "server"> </asp:View>
```

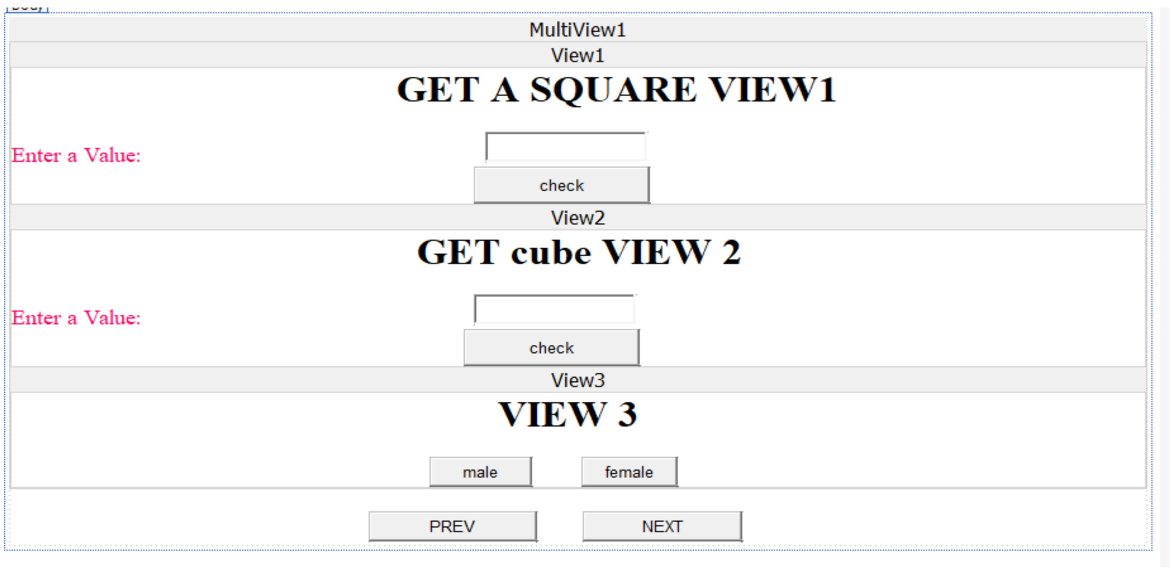
However, the View control cannot exist on its own. It would render error if you try to use it stand-alone. It is always used with a Multiview control as:

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

Every time a view is changed, the page is posted back to the server and a number of events areraised. Some important events are:

Events	**Description**
ActiveViewChanged	Raised when a view is changed
Activate	Raised by the active view
Deactivate	Raised by the inactive view

Here we take an example to understand how to use multiview control in asp.net c#.


MultiView1

View1

GET A SQUARE VIEW1

Enter a Value:

check

View2

GET cube VIEW 2

Enter a Value:

check

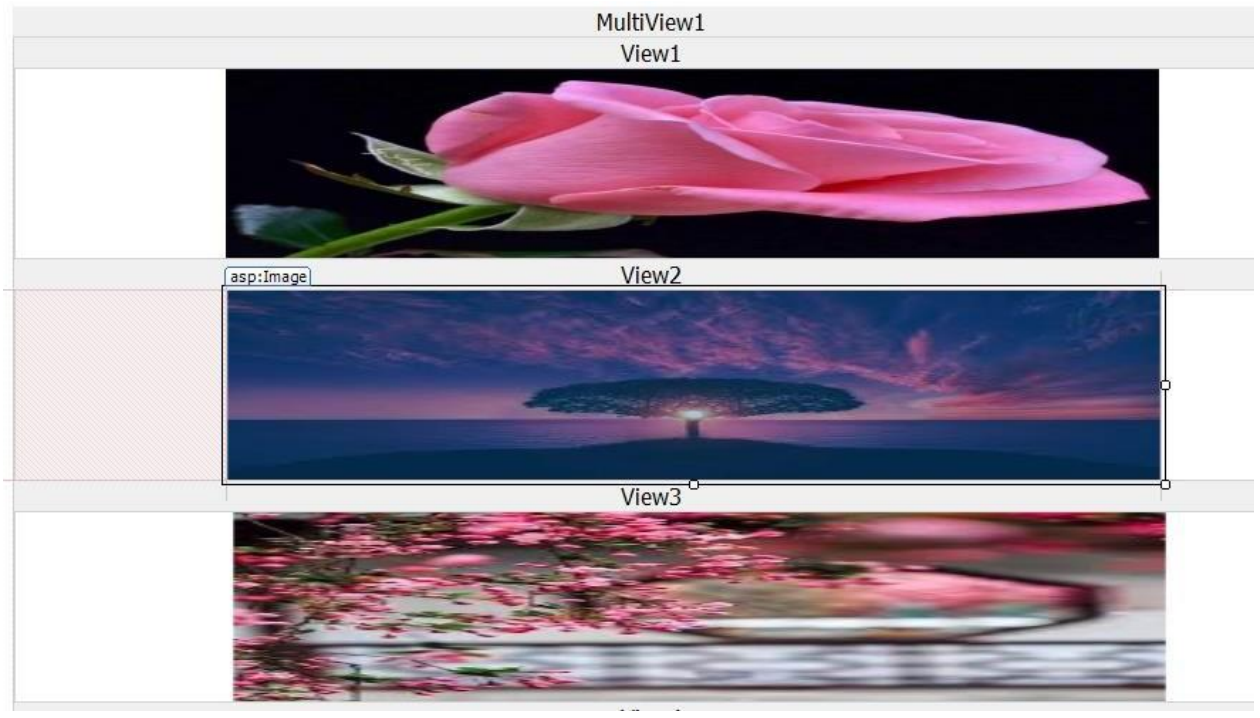
View3

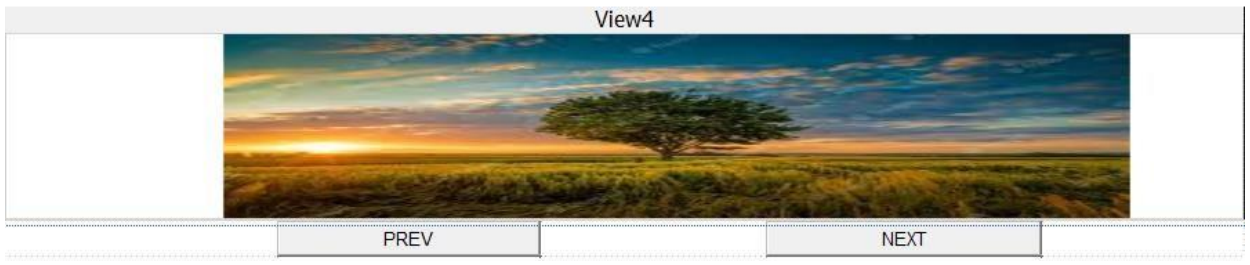
VIEW 3

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Web; usingSystem.Web.UI;
using System.Web.UI.WebControls;

namespace multivieww
{
public partial class multi : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{

}
protected void PREV_Click(object sender, EventArgs e)
{
if (MultiView1.ActiveViewIndex <= 0)
MultiView1.ActiveViewIndex = 0;
else
MultiView1.ActiveViewIndex -= 1;
}

protected void NEXT_Click(object sender, EventArgs e)
{
if (MultiView1.ActiveViewIndex >= 2)
MultiView1.ActiveViewIndex = 0;
else
MultiView1.ActiveViewIndex += 1;
}
}
}
```

Create image slider using Multiview control

α. Drag a Multiview and add 4 view control inside that
β. In each view take an image control and set image into that.





.aspx File:

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

<!DOCTYPE html>

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

```
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">

<asp:View ID="View1" runat="server">
<asp:Image ID="Image1" runat="server" CssClass="auto-style1" Height="133px" Width="657px"
ImageUrl="~/image/hi.jpg" />
</asp:View>

<asp:View ID="View2" runat="server">
<asp:Image runat="server" Height="133px" Width="657px" CssClass="auto-style2"
ImageUrl="~/image/tree1.jfif"></asp:Image>
</asp:View>

<asp:View ID="View3" runat="server">
<asp:Image runat="server" Height="133px" Width="657px" CssClass="auto-style3"
ImageUrl="~/image/tree3.jfif"></asp:Image>
</asp:View>

<asp:View ID="View4" runat="server">
<asp:Image ID="Image2" runat="server" Height="133px" Width="657px" CssClass="auto-style4"
ImageUrl="~/image/download (2).jpg"/>
</asp:View>

</asp:MultiView>

</div>

<asp:Button ID="Button1" runat="server" Text="PREV" CssClass="auto-style6"
OnClick="Button1_Click" Width="191px" />
<asp:Button ID="Button2" runat="server" Text="NEXT" CssClass="auto-style5"
OnClick="Button2_Click" Width="200px" />
</form></body>
</html>
```

.cs file

```
using System;
using System.Collections.Generic;
 using System.Linq;
using System.Web;
 usingSystem.Web.UI;
```

```
using System.Web.UI.WebControls;

protected void PREV_Click(object sender, EventArgs e)
{
if (MultiView1.ActiveViewIndex == 0)
MultiView1.ActiveViewIndex = 3;
else
MultiView1.ActiveViewIndex -= 1;
}

protected void NEXT_Click(object sender, EventArgs e)
{
if (MultiView1.ActiveViewIndex ==3)
MultiView1.ActiveViewIndex = 0;
else
MultiView1.ActiveViewIndex += 1;
}
```

USER CONTROL:

- A User Control is a reusable page or control with an extension of .ascx and created similar to an .aspx page but the difference is that a User Control does not render on its own, it requires an .aspx page to be rendered.
- User Controls are very useful to avoid repetition of code for similar requirements. Suppose I need a calendar control in my application with some custom requirements in multiple pages, then instead of creating the control repetitively you can create it once and use it on multiple pages.

Key points

- The User Control page structure is similar to the .aspx page but a User Control does not need to add an entire HTML structure such as body, head and form.
- A User Control has an .ascx extension.
- A User Control is derived from the UserControl class whereas an .aspx page is derived from the Page class.
- A User Control does not render on its own, it needs an .aspx page.
- To use a User Control in an .aspx page you need to register the control in the .aspx page.

How to create a User Control

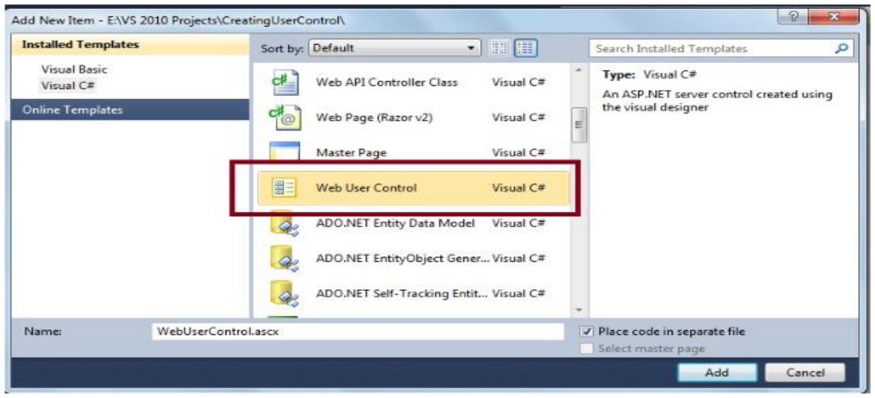
Step 1: Create Web Application

Open Visual Studio.

1. "File" -> "New" -> "Project..." then select ASP.NET Webform Application.
2. Add a new web form.

Step 2: Create the User Control

1. Then right-click on the project in the Solution Explorer then select "Add New Item" then select Web User Control template as in the following:


Add New Item - E:\VS 2010 Projects\CreatingUserControl\
Installed Templates
Visual Basic
Visual C#
Online Templates
Sort by: Default
Search Installed Templates
Web API Controller Class Visual C#
Web Page (Razor v2) Visual C#
Master Page Visual C#
Web User Control Visual C#
ADO.NET Entity Data Model Visual C#
ADO.NET EntityObject Gener... Visual C#
ADO.NET Self-Tracking Entit... Visual C#
Type: Visual C#
An ASP.NET server control created using the visual designer
Add
Cancel

Name: WebUserControl.ascx

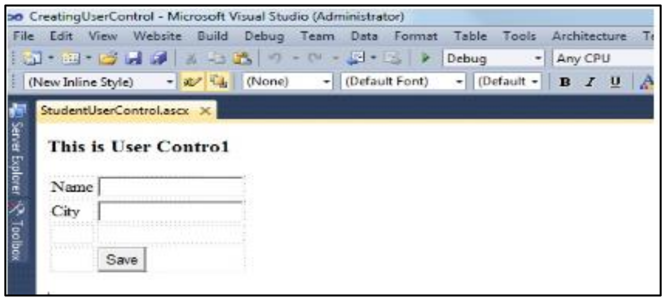
[✓] Place code in separate file

[ ] Select master page

2. Now click on Add then User Control will be added into the solution of the application. Now open the design mode and add the two textboxes, one label, one button and after adding the studentcontrol.ascx the source code will look as follows:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="StudentUserControl.a
scx.cs" Inherits="StudentUserControl" %>

<h3>This is User Contro1   </h3>
Name:
<asp:TextBox ID="txtName" runat="server"></asp:TextBox> <br>

City <asp:TextBox ID="txtcity" runat="server"></asp:TextBox> <br>

<asp:Button ID="txtSave" runat="server" Text="Save" onclick="txtSave_Click" />
 <asp:Label ID="Label1" runat="server" ForeColor="White" Text=" ">
</asp:Label>
```

In the preceding code you have noticed that there is no whole HTML code in User Control such as head, body and form even then it will create the server control. Now switch to design mode then the control will look such as follows:


CreatingUserControl - Microsoft Visual Studio (Administrator)
File Edit View Website Build Debug Team Data Format Table Tools Architecture
StudentUserControl.ascx
This is User Control
Name
City
Save

Now double-click on the save button and write the following code in the studentusercontrol.ascx.cs file as:

```
protected void txtSave_Click(object sender, EventArgs e)
{
   Label1.Text="Your Name is "+txtName.Text+"  and you are  from  "+txtcity.Text;
}
```

Step 3: Adding User Control into .aspx page
To use a User Control in an .aspx we need to register it using the Register page directive in .ascx file, the register page directive has the following properties as:

- Src: Used to set the source of User Control.
- TagName: Used to provide the name for the User Control used on a page similar to a TextBox or label, you can define any name.
- TagPrefix: This is used to specify the prefix name of User Control which is similar to ASP: You can define any prefix name.

Add a web form names "default.aspx" and add following code to use user control.

```
<%@ Register Src="~/StudentUserControl.ascx" TagPrefix="uc" TagName="Student"%>
```

Now the whole code of the default .aspx code will look as in the following:

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

<%@ Register Src="~/StudentUserControl.ascx" TagPrefix="uc" TagName="Student"%>

<html>
<head id="Head1" runat="server">
   <title>Article by Vithal Wadje</title>
</head>
<body bgcolor="blue">
```

```
   <form id="form2" runat="server">
   <div style="color: White;">

     <uc:Student ID="studentcontrol" runat="server" />

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

Navigation Controls:

- Navigation controls in ASP.NET are server-side controls designed to help developers create and manage navigation structures in a web application.
- These controls are typically used to build menus, site maps, and breadcrumb trails, which guide users through the website's hierarchy and content.

Types of Navigation Controls in ASP.NET

1. SiteMapPath Control
2. Menu Control
3. TreeView Control

1. SiteMapPath Control:

- The SiteMapPath Control is a Navigation Control of ASP.Net.
- SiteMapPath Control define the Current location of page with Respect to the Home page.
- SiteMapPath control is used to show Hierarchical path of the Navigation. Simply we can say SiteMapPath control display the current location of the Page from Home page. The Navigation link is clickable to previous page.

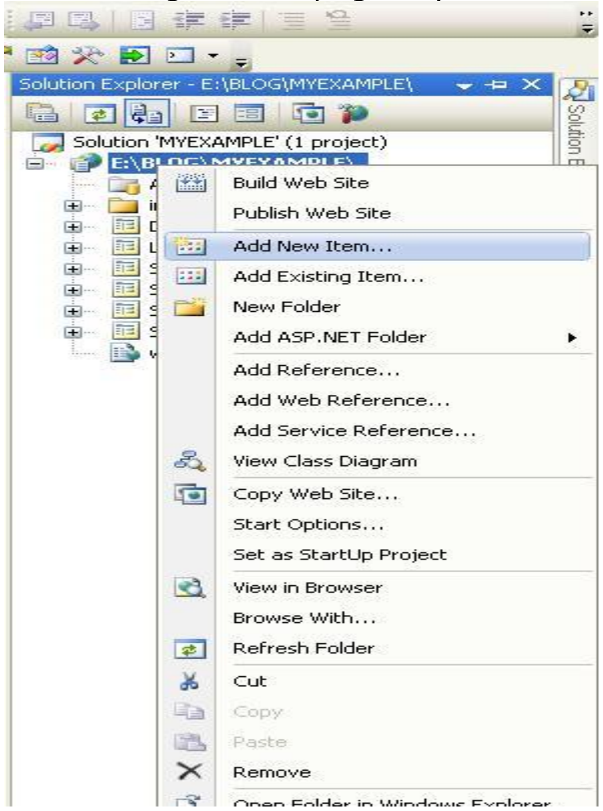
- **lets take an Example to understand SiteMapPath control in asp.net.**

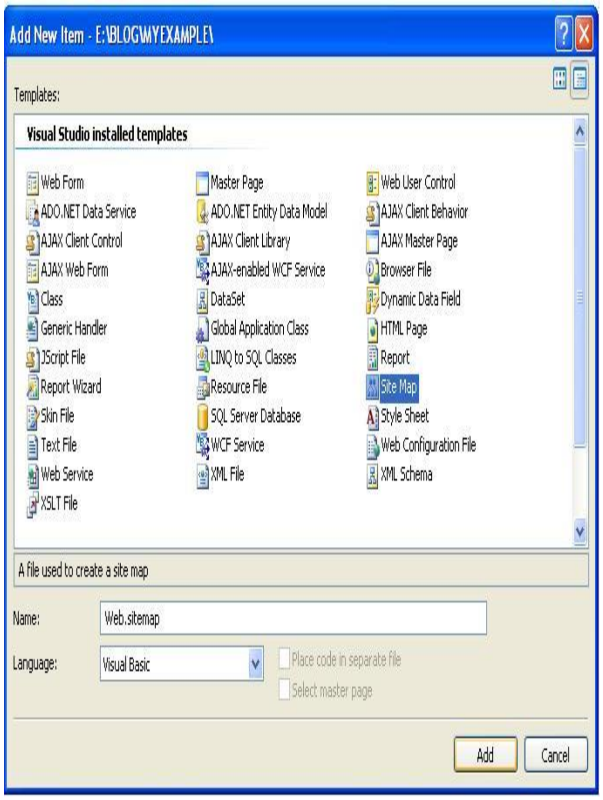
  Here, we have example to show the path like :

  - “Home Page –> Laptop Page –> Sony Laptop –> Sony vio”
  - “Home Page –> Laptop Page –> Sony Laptop –> Sony PC”
  - “Home Page –> Laptop Page –> SamSung Laptop”.

For doing above example of navigation we need to take 6 (six) asp.net web for named like: **Default.aspx, laptop.aspx, sony.aspx, samsung.aspx, sonyvio.aspx, sonypc.aspx.**

After taking six web page in your website add **web.sitemap** file int Solution Explorer.





A file used to create a site map

Name: Web.sitemap

Language: Visual Basic

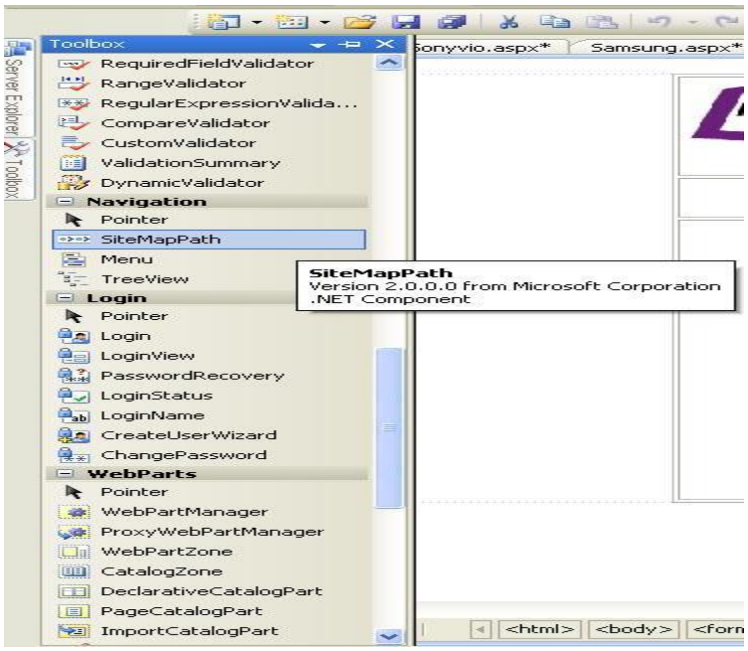
☐ Place code in separate file

☐ Select master page

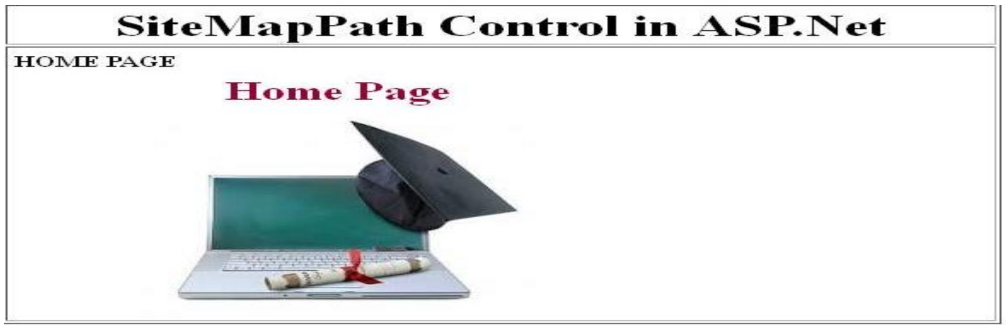
Now, write below code in web.sitemap file.

```
<?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>
```

Add SiteMapPath control in all the web form.



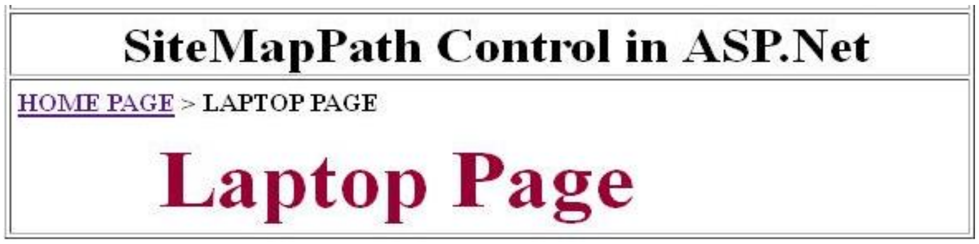
The Default.aspx home page output like :


SiteMapPath Control in ASP.Net

HOME PAGE

Home Page

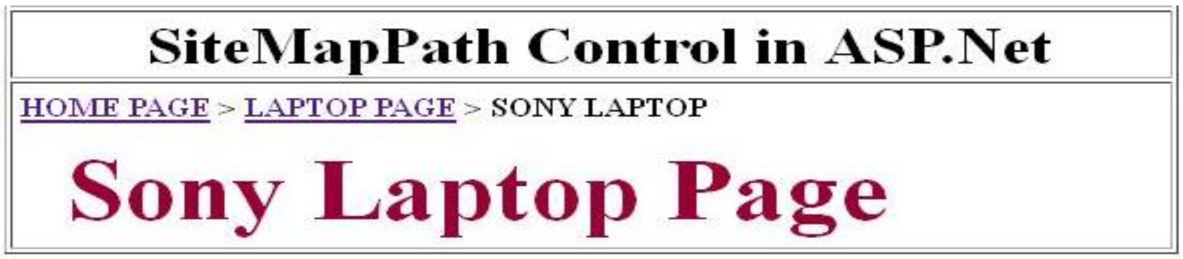
The Laptop.aspx laptop page out put is :


SiteMapPath Control in ASP.Net

HOME PAGE > LAPTOP PAGE

Laptop Page

The Sony.aspx Sony laptop page output is :

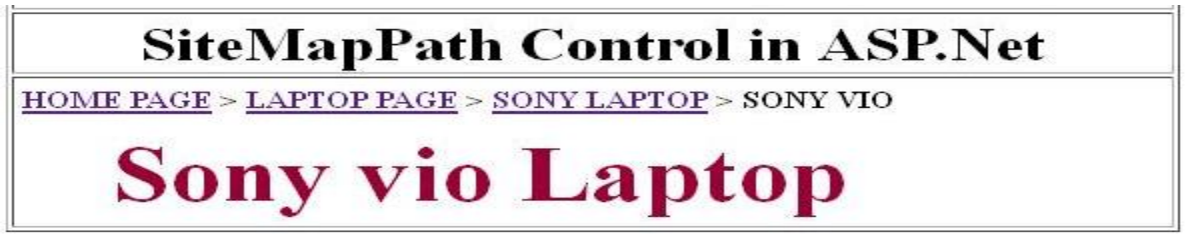


SiteMapPath Control in ASP.Net

HOME PAGE > LAPTOP PAGE > SONY LAPTOP

Sony Laptop Page

The sonyvio.aspx Sony vio laptop page output is:

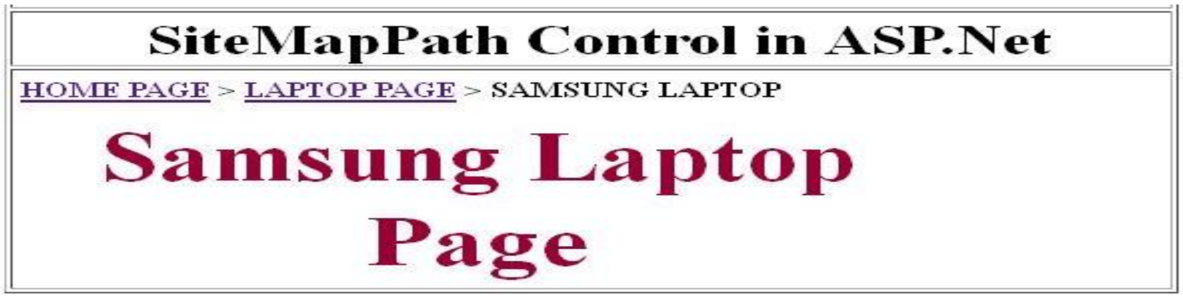


SiteMapPath Control in ASP.Net

HOME PAGE > LAPTOP PAGE > SONY LAPTOP > SONY VIO

Sony vio Laptop

The samsung.aspx Samsung laptop page out put is :



SiteMapPath Control in ASP.Net

HOME PAGE > LAPTOP PAGE > SAMSUNG LAPTOP

Samsung Laptop Page

2. TreeView Control:

In ASP.NET, a TreeView control is used to display hierarchical data in a tree-like structure. This can be useful for navigation menus, organizational charts, or any data that has a parent-child relationship. The TreeView control allows users to expand and collapse nodes to view more or less of the hierarchy.

Here's a basic guide on how to use the TreeView control in an ASP.NET Web Forms application:

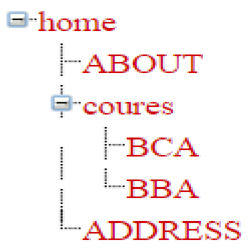
Adding the TreeView Control

You can add the TreeView control to your ASP.NET page by including it in your `.aspx` file. Here's an example of how to add a TreeView control to your page:

```
<asp:TreeView ID="TreeView1" runat="server">
</asp:TreeView>
```

Example:

```
<asp:TreeView ID="TreeView1" runat="server" ShowLines="True" ForeColor="#CC0000"
PathSeparator="&lt;">
  <Nodes>
<asp:TreeNode Text="home" Value="New Node" NavigateUrl="~/HOME.aspx">
<asp:TreeNode NavigateUrl="~/ABOUT.aspx" Text="ABOUT" Value="ABOUT"></asp:TreeNode>
<asp:TreeNode Text="coures" Value="New Node" NavigateUrl="~/course.aspx">
<asp:TreeNode Text="BCA" Value="New Node" NavigateUrl="~/BCA.aspx">
</asp:TreeNode>
<asp:TreeNode NavigateUrl="~/BBA.aspx" Text="BBA" Value="BBA"></asp:TreeNode>
</asp:TreeNode>
 <asp:TreeNode Text="ADDRESS" Value="ADDRESS"
NavigateUrl="~/ADDRESS.aspx"></asp:TreeNode>
</asp:TreeNode>
</Nodes>
</asp:TreeView>
```



3. Menu control

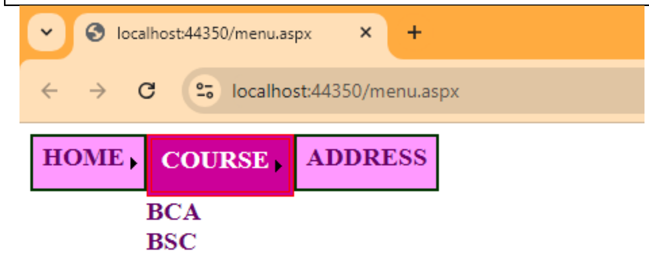
- In ASP.NET Web Forms, the Menu control provides a way to create navigation menus that can help users move between different pages of your website.
- It allows you to create hierarchical menus with support for submenus, dynamic binding, and various customization options.
- To add a Menu control to your `.aspx` page, include it in the page markup:

```
<asp:Menu ID="Menu1" runat="server">       </asp:Menu>
```

Define menu items directly in the markup:

```
<asp:Menu ID="Menu1" runat="server" DisappearAfter="1000" Orientation="Horizontal"
ForeColor="#660066" Font-Bold="True" Font-Italic="False" Font-Size="Larger" Font-
Underline="False" MaximumDynamicDisplayLevels="1" Target="blank" PathSeparator="-">
<Items>
 <asp:MenuItem NavigateUrl="~/HOME.aspx" Text="HOME" ToolTip="H" Value="HOME">
```

```
<asp:MenuItem NavigateUrl="~/ABOUT.aspx" Text="ABOUT"
Value="ABOUT"></asp:MenuItem>
</asp:MenuItem>
  <asp:MenuItem NavigateUrl="~/course.aspx" Text="COURSE" Value="COURSE">
    <asp:MenuItem NavigateUrl="~/BCA.aspx" Text="BCA" Value="BCA"></asp:MenuItem>
     <asp:MenuItem NavigateUrl="~/BBA.aspx" Text="BSC" Value="BSC"></asp:MenuItem>
    </asp:MenuItem>
<asp:MenuItem NavigateUrl="~/ADDRESS.aspx" Text="ADDRESS"
Value="ADDRESS"></asp:MenuItem>
</Items>
 <LevelMenuItemStyles>
<asp:MenuItemStyle Font-Underline="False" />
</LevelMenuItemStyles>
<StaticHoverStyle BackColor="#CC0099" ForeColor="White" BorderStyle="Double"
BorderWidth="4px" BorderColor="Red" />
 <StaticMenuItemStyle BackColor="#FF99FF" BorderColor="#003300" BorderStyle="Solid"
BorderWidth="2px" ForeColor="#660066" Height="30px" HorizontalPadding="7px"
VerticalPadding="4px" />
 <StaticMenuStyle BackColor="Red" Height="0px" />
  <StaticSelectedStyle HorizontalPadding="120px" ItemSpacing="120px"
VerticalPadding="120px" BackColor="#FF0066" />
</asp:Menu>
```


localhost:44350/menu.aspx
localhost:44350/menu.aspx
HOME
COURSE
ADDRESS
BCA
BSC