Web Server Controls

Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The System.Web.UI.WebControl namespace can be divided following in four Groups:

- **Web Form Controls(HTML Intrinsic Controls):**

Button, TextBox, Label, Literal, Image, ImageButton, HyperLink, LinkButton, Panel, Placeholder, RadioButton, ChekBox, Table, XML.

- **List Controls:**

ListItem, CheckBoxList, DropDownList, ListBox, RadioButtonList, Repeater, DataGrid, GridView, DetailsView, FormView etc.

- **Rich Controls:**

Adrotator and Calender Controls.

- **Validation Controls:**

CompareValidator, RangeValidator, RegularExpressionValidator, RequiredFieldValidator, ValidationSummary and CustomValidation Controls.

Web Form Controls (HTML Intrinsic Controls)

Web Control Standard Properties

AccessKey, Attributes, BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Enabled, Font, EnableTheming, ForeColor, Height, IsEnabled, SkinID, Style, TabIndex, ToolTip, Width

Control Standard Properties

AppRelativeTemplateSourceDirectory, BindingContainer, ClientID, Controls, EnableTheming, EnableViewState, ID, NamingContainer, Page, Parent, Site, TemplateControl, TemplateSourceDirectory, UniqueID, Visible

➢ Label Control:

▪ Definition and Usage

The Label control is used to display text on a page. The text is programmable.

Note: This control lets you apply styles to its content!

▪ Syntax

```
<asp:Label id="label1" runat="server" />
```

▪ Properties

Property	**Description**
runat	Specifies that the control is a server control. Must be set to "server"
Text	The text to display in the label
AccessKey	Set the shortcut key for specified controls
AssociatedControlID	Id of the controls that you want to associate with label accesskey

Web Control Standard Properties

AccessKey, Attributes, BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Enabled, Font, EnableTheming, ForeColor, Height, IsEnabled, SkinID, Style, TabIndex, ToolTip, Width

Control Standard Properties

AppRelativeTemplateSourceDirectory, BindingContainer, ClientID, Controls, EnableTheming, EnableViewState, ID, NamingContainer, Page, Parent, Site, TemplateControl, TemplateSourceDirectory, UniqueID, Visible

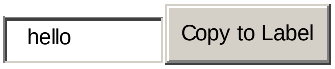
- ## Example:

In this example we declare one Label control, one TextBox control, and one Button control in an .aspx file. When the user clicks on the button, the submit subroutine is executed. The subroutine copies the content of the TextBox control to the Label control.

```
<script  runat="server">
Void submit( Object Sender, EventArgs e)
{
  label1.Text=txt1.Text
}
</script>

<html>
<body>

<form runat="server">
Write some text:
<asp:TextBox id="txt1" Width="200" runat="server" />
<asp:Button id="b1" Text="Copy to Label" OnClick="submit" runat="server" />
<p><asp:Label id="label1" runat="server" /></p>
</form>

</body>
</html>
```

- ## Output:

Write some text:



hello

Hello

➢ TextBox Control:

▪ Definition and Usage

The TextBox control is used to create a text box where the user can input text.

▪ Syntax:

```
<asp:TextBox id="txt1" TextMode=”multiline”  Autopostback=”true”
runat="server" />
```

▪ Properties

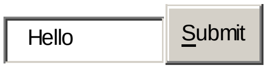
Property	Description	.NET
AutoCompleteType	Specifies the AutoComplete behavior of a TextBox	2.0
AutoPostBack	A Boolean value that specifies whether the control is automatically posted back to the server when the contents change or not. Default is false	1.0
CausesValidation	Specifies if a page is validated when a Postback occurs	2.0
Columns	The width of the textbox	1.0
MaxLength	The maximum number of characters allowed in the textbox	1.0
ReadOnly	Specifies whether or not the text in the text box can be changed	1.0
Rows	The height of the textbox (only used if TextMode="Multiline")	1.0
runat	Specifies that the control is a server control. Must be set to "server"	
TagKey		
Text	The contents of the textbox	1.0
TextMode	Specifies the behavior mode of a TextBox control (SingleLine, MultiLine or Password)	1.0
ValidationGroup	The group of controls that is validated when a Postback occurs	
Wrap	A Boolean value that indicates whether the contents of the textbox should wrap or not	1.0
OnTextChanged	The name of the function to be executed when the text in the textbox has changed	

Example:

In this example we declare one TextBox control, one Button control, and one Label control in an .aspx file. When the submit button is triggered, the submit subroutine is executed. The submit subroutine writes a text to the Label control.

```
<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>
```

Output:

Enter your name: 

Your name is Hello

Button Control:

Definition and Usage

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 Web 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.

- **Syntax:**

```
<asp:Button id=Button1 runat="server"Text="Button1"
onclick="Button1_Click" />
```

- **Properties**

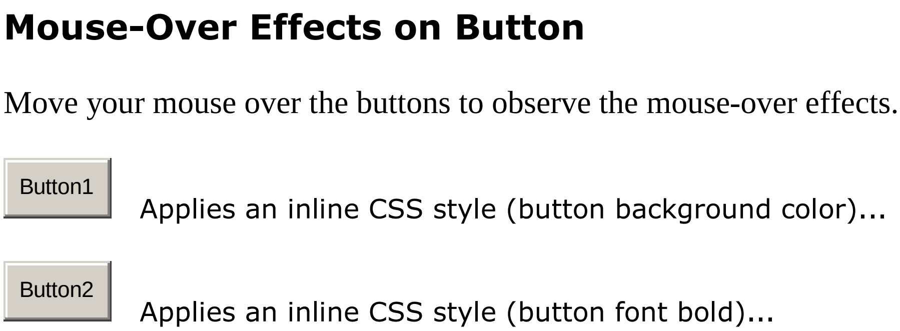
Property	Description	.NET
CausesValidation	Specifies if a page is validated when a Button control is clicked	1.0
CommandArgument	Additional information about the command to perform	1.0
CommandName	The command associated with the Command event	1.0
OnClientClick	The name of the function to be executed when the button is clicked	2.0
PostBackUrl	The URL of the page to post to from the current page when the Button control is clicked	2.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
Text	The text on the button	1.0
UseSubmitBehavior	A value indicating whether or not the Button control uses the client browser's submit mechanism or the ASP.NET postback mechanism	2.0
ValidationGroup	The group of controls for which the Button control causes validation when it posts back to the server	2.0

- **Example:**

```
<html>
<head>

    <script language="VB" runat="server">

       void Button1_Click(Object sender,EventArgs e)
        {
          Label1.Text="You clicked Button1"
       }

       void Button2_Click(Object sender,EventArgs e)
        {

          Label1.Text="You clicked Button2"
        }

    </script>

</head>
<body>
```

```
    <h3><font face="Verdana">Mouse-Over Effects on Button</font></h3>
    <p>Move your mouse over the buttons to observe the mouse-over
effects.</p>

    <form runat=server>
    <font face="Verdana" size="-1">

        <asp:Button id=Button1 runat="server"
            Text="Button1"
            Width="100px"
            onmouseover="this.style.backgroundColor='yellow'"
            onmouseout="this.style.backgroundColor='buttonface'"
            onclick="Button1_Click" />

            &nbsp;&nbsp;
            Applies an inline CSS style (button background color)...

        <p>

        <asp:Button id=Button2 runat="server"
            Text="Button2"
            Width="100px"
            onmouseover="this.style.fontWeight='bold'"
            onmouseout="this.style.fontWeight='normal'"
            onclick="Button2_Click" />

            &nbsp;&nbsp;
            Applies an inline CSS style (button font bold)...

        <p>

        <asp:Label id=Label1 runat=server />

    </font>
    </form>

</body>
</html>
```

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



Mouse-Over Effects on Button

Move your mouse over the buttons to observe the mouse-over effects.

Applies an inline CSS style (button background color)...

Applies an inline CSS style (button font bold)...

➢ LinkButton Control:

Definition and Usage

The LinkButton control is used to create a hyperlink button.

Note: This control looks like a HyperLink control but has the same functionality as the Button control.

Syntax

```
<asp:LinkButton Text="Click me!" OnClick="lblClick" runat="server" />
```

Properties

Property	Description	.NET
CausesValidation	Specifies if a page is validated when a LinkButton control is clicked	1.0
CommandArgument	Additional information about the command to perform	1.0
CommandName	The command associated with the Command event	1.0
OnClientClick	The name of the function to be executed when the LinkButton is clicked	2.0
PostBackUrl	The URL of the page to post to from the current page when the LinkButton control is clicked	2.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
Text	The text on the LinkButton	1.0
ValidationGroup	The group of controls for which the LinkButton control causes validation when it posts back to the server	2.0

Example:

In this example we declare one LinkButton control and one Label control in an .aspx file. When the user clicks on the link, the lbClick subroutine is executed. The subroutine sends the text "You clicked the LinkButton control" to the Label control.

```
<script  runat="server">
Sub lblClick(Object sender,EventArgs e)
{
  Label1.Text="You clicked the LinkButton control";
}
</script>
```

```
<html>
<body>

<form runat="server">
<asp:LinkButton Text="Click me!" OnClick="lblClick" runat="server" />
<p><asp:Label id="Label1" runat="server" /></p>
</form>

</body>
</html>
```

Output:

Click me!

You clicked the LinkButton control

ImageButton Control:

Definition and Usage

The ImageButton control is used to display a clickable image.

Syntax

```
<asp:ImageButton runat="server" ImageUrl="smiley.gif"
OnClick="getCoordinates"/>
```

Properties

Property	Description	.NET
CausesValidation	Specifies if a page is validated when an ImageButton control is clicked	1.0
CommandArgument	Additional information about the command to perform	1.0
CommandName	The command associated with the Command event	1.0
GenerateEmptyAlternateText	Specifies whether or not the control creates an empty string as an alternate text	2.0
OnClientClick	The name of the function to be executed when the image is clicked	2.0
PostBackUrl	The URL of the page to post to from the current page when the ImageButton control is clicked	2.0
runat	Specifies that the control is a server control. Must be	1.0

	set to "server"	
TagKey		1.0
ValidationGroup	The group of controls for which the ImageButton control causes validation when it posts back to the server	2.0

Note: The properties of the Image control can also be used on the ImageButton control.

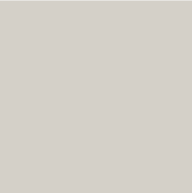
- **Example:**

In this example we declare one ImageButton control and one Label control in an .aspx file. When the user clicks on the image, the getCoordinates subroutine is executed. The subroutine sends the message "Coordinates: " and the x and y coordinates of the click to the Label control.

```
<script  runat="server">
void getCoordinates(Object sender,  ImageClickEventArgs e)

{
  mess.Text="Coordinates: " & e.x & ", " & e.y

}
End Sub
</script>

<html>
<body>

<form runat="server">
<p>Click on the image:</p>
<asp:ImageButton
runat="server"
ImageUrl="smiley.gif"
OnClick="getCoordinates"/>
<p><asp:label id="mess" runat="server"/></p>
</form>

</body>
</html>
```

- **Output:**

Click on the image:



Coordinates: 15, 20

➢ HyperLink Control:

▪ Definition and Usage

The HyperLink control is used to create a hyperlink.

▪ Syntax

```
<asp:HyperLink ImageUrl="/banners/w6.gif"
NavigateUrl="http://www.w3schools.com"
Text="Visit W3Schools!" Target="_blank"  runat="server" />
```

▪ Properties

Property	Description	.NET
ImageUrl	The URL of the image to display for the link	1.0
NavigateUrl	The target URL of the link	1.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
Target	The target frame of the URL	1.0
Text	The text to display for the link	1.0

▪ Example:

```
<html>
<body>

<form runat="server">
<asp:HyperLink
ImageUrl="/banners/w6.gif"
NavigateUrl="http://www.w3schools.com"
Text="Visit W3Schools!"
Target="_blank"
runat="server" />
</form>
```

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

- **Output:**

CheckBox Control:

- **Definition and Usage**

The CheckBox control is used to display a check box.

- **Syntax:**

```
<asp:CheckBox id="check1"
Text="some text" TextAlign="Right"
AutoPostBack="True" OnCheckedChanged="Check"
runat="server" />
```

- **Properties**

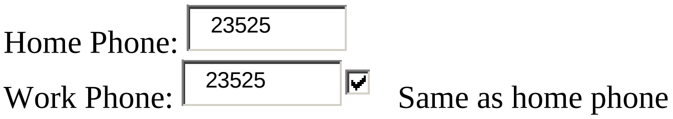
Property	**Description**	**.NET**
AutoPostBack	Specifies whether the form should be posted immediately after the Checked property has changed or not. Default is false	1.0
CausesValidation	Specifies if a page is validated when a Button control is clicked	2.0
Checked	Specifies whether the check box is checked or not	1.0
InputAttributes	Attribute names and values used for the Input element for the CheckBox control	2.0
LabelAttributes	Attribute names and values used for the Label element for the CheckBox control	2.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
Text	The text next to the check box	1.0
TextAlign	On which side of the check box the text should appear (right or left)	1.0
ValidationGroup	Group of controls for which the Checkbox control causes	2.0

	validation when it posts back to the server	
OnCheckedChanged	The name of the function to be executed when the Checked property has changed	

- **Example:**

```
<script  runat="server">
  void Check( Object sender,  EventArgs e)
    {
    if (check1.Checked)
      work.Text=home.Text;
    else
      work.Text="";

  }
</script>

<html>
<body>

<form runat="server">
<p>Home Phone:
<asp:TextBox id="home" runat="server" />
<br />
Work Phone:
<asp:TextBox id="work" runat="server" />
<asp:CheckBox id="check1"
Text="Same as home phone" TextAlign="Right"
AutoPostBack="True" OnCheckedChanged="Check"
runat="server" />
</p>
</form>

</body>
</html>
```

- **Output:**



Home Phone: 23525

Work Phone: 23525 ☑ Same as home phone

➢ RadioButton Control:

▪ Definition and Usage

The RadioButton control is used to display a radio button.

Tip: To create a set of radio buttons using data binding, use the RadioButtonList control!

▪ Syntax:

```
<asp:RadioButton id="red" Text="Red" Checked="True"
GroupName="colors" runat="server"/>
```

▪ Properties

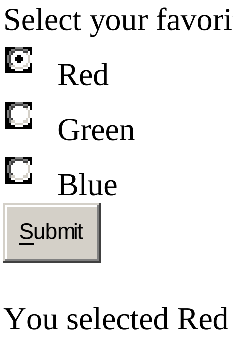
Property	**Description**
AutoPostBack	A Boolean value that specifies whether the form should be posted immediately after the Checked property has changed or not. Default is false
Checked	A Boolean value that specifies whether the radio button is checked or not
id	A unique id for the control
GroupName	The name of the group to which this radio button belongs
OnCheckedChanged	The name of the function to be executed when the Checked property has changed
runat	Specifies that the control is a server control. Must be set to "server"
Text	The text next to the radio button
TextAlign	On which side of the radio button the text should appear (right or left)

▪ Example:

In this example we declare three RadioButton controls, one Button control, and one Label control in an .aspx file. When the submit button is triggered, the submit subroutine is executed. The submit subroutine may respond in three ways: if the radiobutton with id="red" is selected, the server sends the message "You selected Red" to the Label control. If the radiobutton with id="green" is selected, the server sends the message "You selected Green" to the Label control. If the radiobutton with id="green" is selected, the server sends the message "You selected Blue" to the Label control.

```
<script  runat="server">
Sub submit(Sender As Object, e As EventArgs)
if red.Checked then
  Label1.Text="You selected " & red.Text
elseIf green.Checked then
  Label1.Text="You selected " & green.Text
elseIf blue.Checked then
  Label1.Text="You selected " & blue.Text
end if
End Sub
</script>

<html>
<body>

<form runat="server">
Select your favorite color:
<br />
<asp:RadioButton id="red" Text="Red" Checked="True"
GroupName="colors" runat="server"/>
<br />
<asp:RadioButton id="green" Text="Green"
GroupName="colors" runat="server"/>
<br />
<asp:RadioButton id="blue" Text="Blue"
GroupName="colors" runat="server"/>
<br />
<asp:Button text="Submit" OnClick="submit" runat="server"/>
<p><asp:Label id="Label1" runat="server"/></p>
</form>

</body>
</html>
```

- **Output:**



Select your favorite color:

● Red

○ Green

○ Blue

You selected Red

Image Control:

Definition and Usage

The Image control is used to display an image.

Syntax:

```
<asp:Image
runat="server"
AlternateText="W3Schools"
ImageUrl="/banners/w6.gif"/>
```

Properties

Property	Description	.NET
AlternateText	An alternate text for the image	1.0
DescriptionUrl	The location to a detailed description for the image	2.0
GenerateEmptyAlternateText	Specifies whether or not the control creates an empty string as an alternate text	2.0
ImageAlign	Specifies the alignment of the image	1.0
ImageUrl	The URL of the image to display for the link	1.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0

Example:

```
<html>
<body>

<form runat="server">
<asp:Image
runat="server"
AlternateText="W3Schools"
ImageUrl="/banners/w6.gif"/>
</form>

</body>
</html>
```

Output:

➢ ImageMap Control:

▪ Definition and Usage

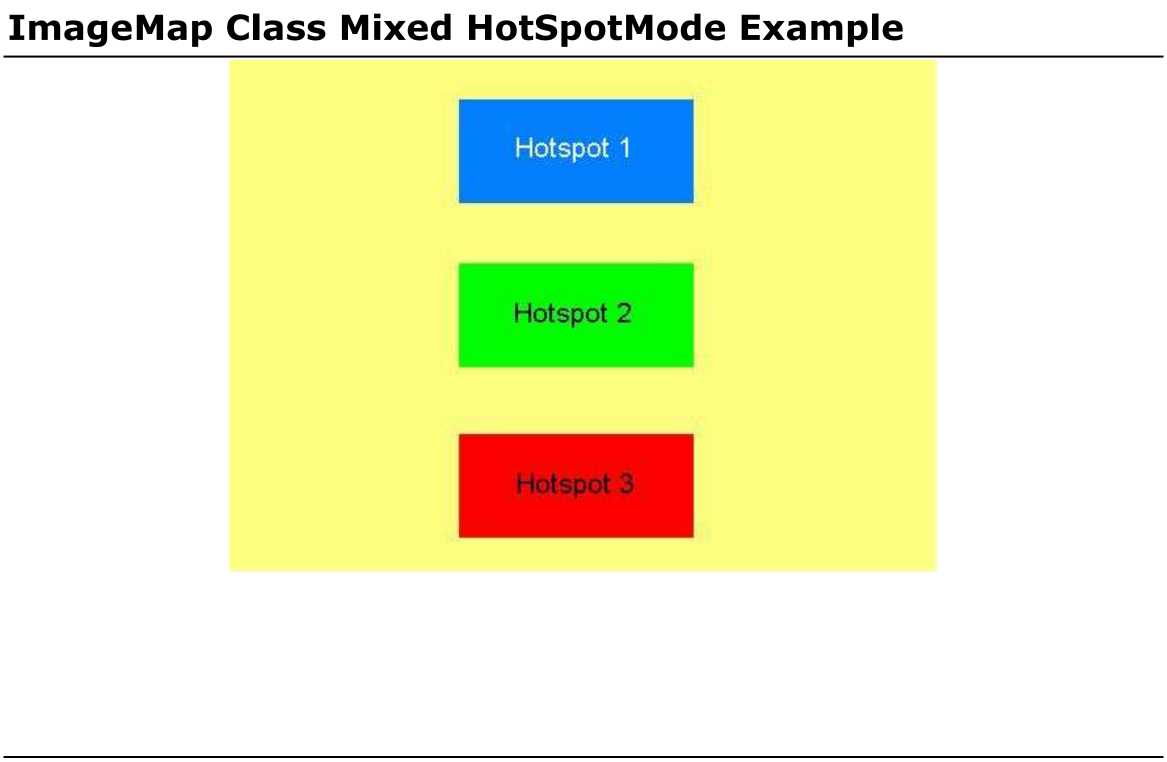
Use the ImageMap control to create an image that contains defined hotspot regions. When a user clicks a hot spot region, the control can either generate a post back to the server or navigate to a specified URL.

▪ Syntax:

▪ Example:

```
<%@ Page Language="C#" %>
<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>
            <h3><font face="Verdana">ImageMap Class Mixed HotSpotMode
Example</font></h3>

            <asp:imagemap id="Buttons" imageurl="hotspot.jpg"
alternatetext="Navigate buttons"
                runat="Server">

               <asp:RectangleHotSpot
               hotspotmode="Navigate"
                NavigateUrl="navigate1.htm"
                alternatetext="Button 1"
                top="30"
                left="175"
                bottom="110"
                right="355">
                </asp:RectangleHotSpot>

                <asp:RectangleHotSpot
                hotspotmode="Navigate"
                NavigateUrl="navigate2.htm"
                alternatetext="Button 2"
                top="155"
                left="175"
                bottom="240"
                right="355">
                </asp:RectangleHotSpot>

                <asp:RectangleHotSpot
                hotspotmode="Navigate"
                NavigateUrl="navigate3.htm"
                alternatetext="Button 3"
                top="285"
                left="175"
                bottom="365"
                right="355">
                </asp:RectangleHotSpot>

            </asp:imagemap>

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

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



ImageMap Class Mixed HotSpotMode Example

Table Control:

Definition and Usage

The Table control is used in conjunction with the TableCell control and the TableRow control to create a table.

Syntax:

```
<asp:Table id="Table1" BorderWidth="1" GridLines="Both" runat="server" />
```

Properties

Property	Description	.NET
BackImageUrl	A URL to an image to use as an background for the table	1.0
Caption	The caption of the table	2.0
CaptionAlign	The alignment of the caption text	2.0
CellPadding	The space between the cell walls and contents	1.0
CellSpacing	The space between cells	1.0
GridLines	The gridline format in the table	1.0
HorizontalAlign	The horizontal alignment of the table in the page	1.0
Rows	A collection of rows in the Table	1.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0

Example:

```
<script  runat="server">
void  Page_Load(Object sender, EventArgs e)
{
int rows,cells,j,I ;
rows=3 ;
cells=2 ;
For (j=0 ;j<rows-1;j++)
{
   TableRow r = new  TableRow();
   For (i=0 ;cells-1;i++)
   {
     TableCell c=New TableCell();
     c.Controls.Add(New LiteralControl("row " & j & ", cell " & i));
     r.Cells.Add(c);
   }
   Table1.Rows.Add(r)
}
}
```

```
</script>

<html>
<body>

<form runat="server">
<asp:Table id="Table1" BorderWidth="1" GridLines="Both" runat="server" />
</form>

</body>
</html>
```

- **Output:**

row 0, cell 0	row 0, cell 1
row 1, cell 0	row 1, cell 1
row 2, cell 0	row 2, cell 1

➢ Table Cell:

- **Definition and Usage**

The TableCell control is used in conjunction with the Table control and the TableRow control to create a cell in a table.

Tip: The cells of each row are stored in the Cells collection of the TableRow control.

- **Properties:**

Property	**Description**	**.NET**
AssociatedHeaderCellID	A list of table header cells associated with the TableCell control	2.0
ColumnSpan	The number of columns this cell should span	1.0
HorizontalAlign	The horizontal alignment of the contents in the table cell	1.0
RowSpan	The number of rows this cell should span	1.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
Text	Specifies the text of the table cell	1.0
VerticalAlign	The vertical alignment of the contents in the table cell	1.0
Wrap	Specifies whether the cell's contents should wrap or not	1.0

- ### Example:

Example is Given in Table Control.

➢ Table Row:

- ### Definition and Usage

The TableRow control is used in conjunction with the TableCell control and the Table control to create a row in a table.

Tip: The rows of a table are stored in the Rows collection of the Table control.

- ### Properties:

Property	Description	.NET
Cells		
HorizontalAlign	The horizontal alignment of the contents in the table row	1.0
TableSection	The location of a TableRow object in a Table control	2.0
VerticalAlign	The vertical alignment of the contents in the table row	1.0

- ### Example:

Example is Given in Table Control.

➢ Hidden Control:

- ### Definition and Usage

This control enables a developer to store a non-displayed value.

The HiddenField control is used to store a value that needs to be persisted across posts to the server. It is rendered as an <input type= "hidden"/> element. Normally view state, session state, and cookies are used to maintain the state of a Web Forms page.

- ### Syntax:

```
<asp:HiddenField ID="HiddenField1" runat="server" />
```

- ## Example:

Example illustrate t store some of the value in hidden field on button click.

```
<html>
<head>

    <script language="C#" runat="server">

       void Button1_Click(object sender, EventArgs e)
       {
            if (HiddenField1.Value == String.Empty)
               HiddenField1.Value = "0";

            //Increment the hidden field value by 1
            HiddenField1.Value = 
(Convert.ToInt32(HiddenField1.Value)+1).ToString();

            Label1.Text = HiddenField1.Value;
       }

    </script>

</head>
<body>

    <h3><font face="Verdana">HiddenField</font></h3>

    <form runat=server>
        <asp:HiddenField id=HiddenField1 runat=Server />

        <asp:Button id=Button1 Text="Click Me" onclick="Button1_Click" 
runat="server" />
        Clicked <asp:Label id=Label1 Text="0" runat=server /> times

    </form>

</body>
</html>
```

➢ Literal Control:

- ## Definition and Usage

The Literal control is used to display text on a page. The text is programmable.

Note: This control does not let you apply styles to its content!

Syntax

```
<asp:Literal id="ControlID" Text="I love ASP.NET!" runat="server" />
```

Properties

Property	Description	.NET
Mode		2.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
Text	Specifies the text to display	1.0

Example:

In this example we declare one Literal control and one Button control in an .aspx file. When the user clicks on the button, the submit subroutine is executed. The subroutine changes the text of the Literal control.

```
<script  runat="server">
void submit(Object sender,EventArgs e)
{
  Literal1.Text="I love ASP.NET!"
}
</script>

<html>
<body>

<form runat="server">
<asp:Literal id="Literal1" Text="I love ASP!" runat="server" />
<br /><br />
<asp:Button Text="Change Text" OnClick="submit" runat="server" />
</form>

</body>
</html>
```

Output:

I love ASP.NET!

➢ FileUpload Control:

▪ Definition and Usage

The FileUpLoad control enables you to upload file to the server. It displays a text box control and a browse button that allow users to select a file to upload to the server.

▪ Syntax

```
<asp:FileUpLoad id="FileUpLoad1" AlternateText="You cannot upload
files" runat="server" />
```

▪ Properties & Methods:

Property	Description
Accept	List of acceptable MIME types
Attributes	Returns all attribute name and value pairs of the element
Disabled	A Boolean value that indicates whether or not the control should be disabled. Default is false
id	A unique id for the element
MaxLength	The maximum number of characters allowed in this element
Name	The name of the element
PostedFile	Gets access to the posted file
runat	Specifies that the element is a server control. Must be set to "server"
Size	The width of the element
Style	Sets or returns the CSS properties that are applied to the control
TagName	Returns the element tag name
Type	The type of the element
Value	The value of the element
Visible	A Boolean value that indicates whether or not the control should be visible
Filename	Returns the name of file uploaded by user.

▪ Example:

```
<html>
<head>

    <script language="C#" runat="server">

    void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpLoad1.HasFile)
        {
            //Uncomment this line to Save the uploaded file
            //FileUpLoad1.SaveAs("C:\SomePhysicalPath" +
FileUpLoad1.Filename);
            Label1.Text = "Received " + FileUpLoad1.FileName + "
Content Type " + FileUpLoad1.PostedFile.ContentType + " Length " +
FileUpLoad1.PostedFile.ContentLength;
        }
        else
        {
            Label1.Text = "No uploaded file";
        }

    }

    </script>

</head>
<body>

    <h3><font face="Verdana">File Upload</font></h3>

    <form runat=server>

        <asp:FileUpLoad id="FileUpLoad1" AlternateText="You cannot
upload files" runat="server" />
        <asp:Button id="Button1" Text="Upload" OnClick="Button1_Click"
runat="server" />
        <asp:Label id="Label1" runat="server" />
    </form>

</body>
</html>
```

➢ Wizard Control:

▪ Definition and Usage

The Wizard control provides navigation through a series of steps that collect information incrementally from a user. Many websites include functionality that collects information from the end user (e.g. checking out on an ecommerce website). The Wizard consists of:

- **Collection of WizardSteps:** Each WizardStep contains a discrete piece of content to be displayed to the user. Only one WizardStep will be displayed at a time.
- **Navigation Area:** The navigation area below each WizardStep that contains the navigation buttons to go the next and pervious steps in the wizard.
- **SideBar:** An optional element that contains a list of all WizardSteps and provides a means to skip around the WizardSteps in a random order.

- **Header:** An optional element to provide consistent information at the top of the WizardStep.

The StepType associated with each WizardStep determines the type of navigation buttons that will be displayed for that step. The StepTypes are:

- **Start:** Displays a Next button.
- **Step:** Displays Next and Previous buttons.
- **Finish:** Displays a Finish button.
- **Complete:** Displays no navigation buttons and hides the SideBar if it is displayed.
- **Auto:**One of the step types listed above is selected based on the order of the step in the collection (e.g. the first step will have a Next button).

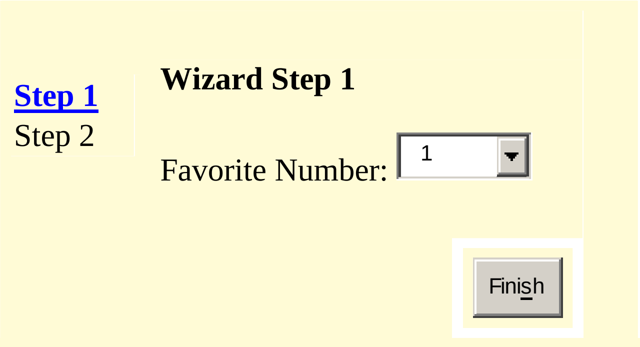
Syntax

```
<asp:Wizard ID="Wizard1" runat="server">
            <WizardSteps>
                <asp:WizardStep ID="WizardStep1" runat="server"
Title="Step 1">
                </asp:WizardStep>
                <asp:WizardStep ID="WizardStep2" runat="server"
Title="Step 2">
                </asp:WizardStep>
            </WizardSteps>

 </asp:Wizard>
```

Example:

```
<script language="C#" runat="server">
  void GetFavoriteNumberOnActiveStepIndex(Object Sender, EventArgs e)
  {
    Label1.Text = "Thank you for telling that your favorite number is:"
+ DropDownList1.SelectedItem.Text;
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Simple Single-Step Wizard Control</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <h2>Simple Single-Step Wizard Control</h2>
      <asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="0"
OnActiveStepChanged="GetFavoriteNumberOnActiveStepIndex"
        BackColor="#FFFBD6" BorderColor="#FFDFAD" BorderWidth="1px"
CellPadding="5" Font-Names="Verdana"
        Font-Size="0.8em" Width="322px">
        <WizardSteps>
        <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
            <strong>Wizard Step 1</strong>
            <br /><br />
            Favorite Number:
            <asp:DropDownList ID="DropDownList1" runat="server">
              <asp:ListItem>1</asp:ListItem>
              <asp:ListItem>2</asp:ListItem>
              <asp:ListItem>3</asp:ListItem>
              <asp:ListItem>4</asp:ListItem>
              <asp:ListItem>5</asp:ListItem>
              <asp:ListItem>6</asp:ListItem>
              <asp:ListItem>7</asp:ListItem>
              <asp:ListItem>8</asp:ListItem>
              <asp:ListItem>9</asp:ListItem>
              <asp:ListItem>10</asp:ListItem>
            </asp:DropDownList>
            <br />
          </asp:WizardStep>
      <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2"
StepType="Complete">
            <strong>Wizard Step 2</strong><br />
            <br />
            <asp:Label ID="Label1" runat="server"/>
          </asp:WizardStep>
        </WizardSteps>
        <SideBarStyle Width="75px" VerticalAlign="Top"/>
      </asp:Wizard>
    </div>
  </form>
</body>
</html>
```

- **Output:**



Step 1
Step 2

Wizard Step 1

Favorite Number: 1

➢ XML Control:

- **Definition and Usage**

The XML control is used to display an XML document or the results of an XSL Transform.

Note: At least one of the XML Document properties must be set or no XML document is displayed.

You can also specify an XSLT document that will format the XML document before it is written to the output. You can format the XML document with the Transform property or the TransformSource property.

Properties:

Property	Description	.NET
ClientID		
Controls		
Document	Deprecated. Specifies an XML document using a System.Xml.XmlDocument object	1.0
DocumentContent	Specifies an XML string	1.0
DocumentSource	Specifies a path to an XML file to display	1.0
EnableTheming		
runat	Specifies that the control is a server control. Must be set to "server"	1.0
SkinID		
Transform	Formats the XML document using a System.Xml.Xsl.XslTransform object	1.0
TransformArgumentList		
TransformSource	Specifies a path to an XSL Transform file	
XPathNavigator		

Example:

This example shows how to use the Xml control to display the result of an XSL Transform.

```
<html>
<body>

<form runat="server">
<asp:Xml DocumentSource="cdcatalog.xml" TransformSource="cdcatalog.xsl"
runat="server" />
</form>

<p><a href="cdcatalog.xml" target="_blank">View XML file</a></p>
<p><a href="cdcatalog.xsl" target="_blank">View XSL file</a></p>

</body>
</html>
```

• cdcatalog.xml

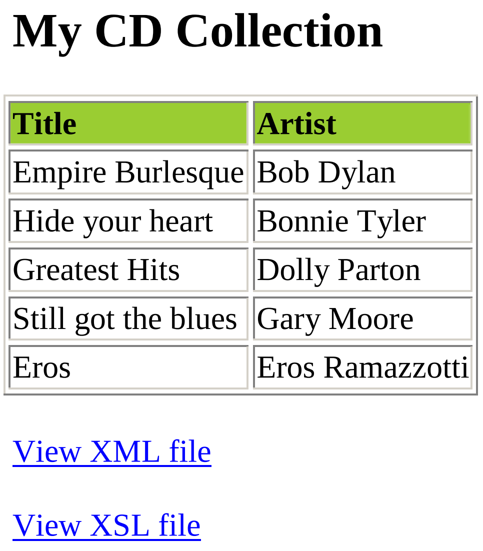
```
<catalog>
–
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
<year>1985</year>
</cd>
–
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
–
<cd>
<title>Greatest Hits</title>
<artist>Dolly Parton</artist>
<country>USA</country>
<company>RCA</company>
<price>9.90</price>
<year>1982</year>
</cd>
–
<cd>
<title>Still got the blues</title>
<artist>Gary Moore</artist>
<country>UK</country>
<company>Virgin records</company>
<price>10.20</price>
<year>1990</year>
</cd>
–
<cd>
<title>Eros</title>
<artist>Eros Ramazzotti</artist>
<country>EU</country>
<company>BMG</company>
<price>9.90</price>
<year>1997</year>
</cd>
</catalog>
```

cdcatalog.xs

```
<xsl:stylesheet version="1.0">
```

```
<xsl:template match="/">
–
<html>
–
<body>
<h2>My CD Collection</h2>
–
<table border="1">
–
<tr bgcolor="#9acd32">
<th align="left">Title</th>
<th align="left">Artist</th>
</tr>
–
<xsl:for-each select="catalog/cd">
–
<tr>
–
<td>
<xsl:value-of select="title"/>
</td>
–
<td>
<xsl:value-of select="artist"/>
</td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

- **Output:**



My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti

View XML file

View XSL file

➢ Panel Control:

▪ Definition and Usage:

The Panel control is used as a container for other controls.

Tip: This control is often used to generate controls by code and to display and hide groups of controls.

Note: This control renders as an HTML <div> element in IE and as a <table> tag in Mozilla.

▪ Syntax:

```
<asp:Panel id="panel1"
runat="server Height="100px" Width="100px">
Hello World!
</asp:Panel>
```

▪ Properties:

Property	Description	.NET
BackImageUrl	Specifies a URL to an image file to display as a background for this control	1.0
DefaultButton	Specifies the ID of the default button in the Panel	2.0
Direction	Specifies the content display direction of the Panel	2.0
GroupingText	Specifies the caption for the group of controls in the Panel	2.0
HorizontalAlign	Specifies the horizontal alignment of the content	1.0
runat	Specifies that the control is a server control. Must be set to "server"	1.0
ScrollBars	Specifies the position and visibility of scroll bars in the Panel	2.0
Wrap	Specifies whether the content should wrap or not	1.0

▪ Example:

Code:

```
<asp:Panel ID="Panel2" ScrollBars="Auto" runat="server" Height="112px"
Width="167px">
        <p>
            After the release of Internet Information Services
            4.0 in 1997, Microsoft began researching possibilities
            for a new web application model that would solve common
            complaints about ASP, especially with regard to separation
            of presentation and content and being able to write "clean"
            code.[1] Mark Anders, a manager on the IIS team, and Scott
            Guthrie,
            who had joined Microsoft in 1997 after graduating from Duke
            University, were tasked with determining what that model
            would look like. The initial design was developed over the
            course of two months by Anders and Guthrie, and Guthrie
            coded
            the initial prototypes during the Christmas holidays in
            1997.[2]
        </p>
 </asp:Panel>
```

➢ PlaceHolder Control:

▪ Definition and Usage:

The **PlaceHolder** control can be used as a container control within a document to dynamically load other controls. The **PlaceHolder** control has no HTML-based output and is used only to mark a spot for other controls that can be added to the **Controls** collection of the **PlaceHolder** during page execution. The following example illustrates adding controls to a **PlaceHolder**.

▪ Syntax:

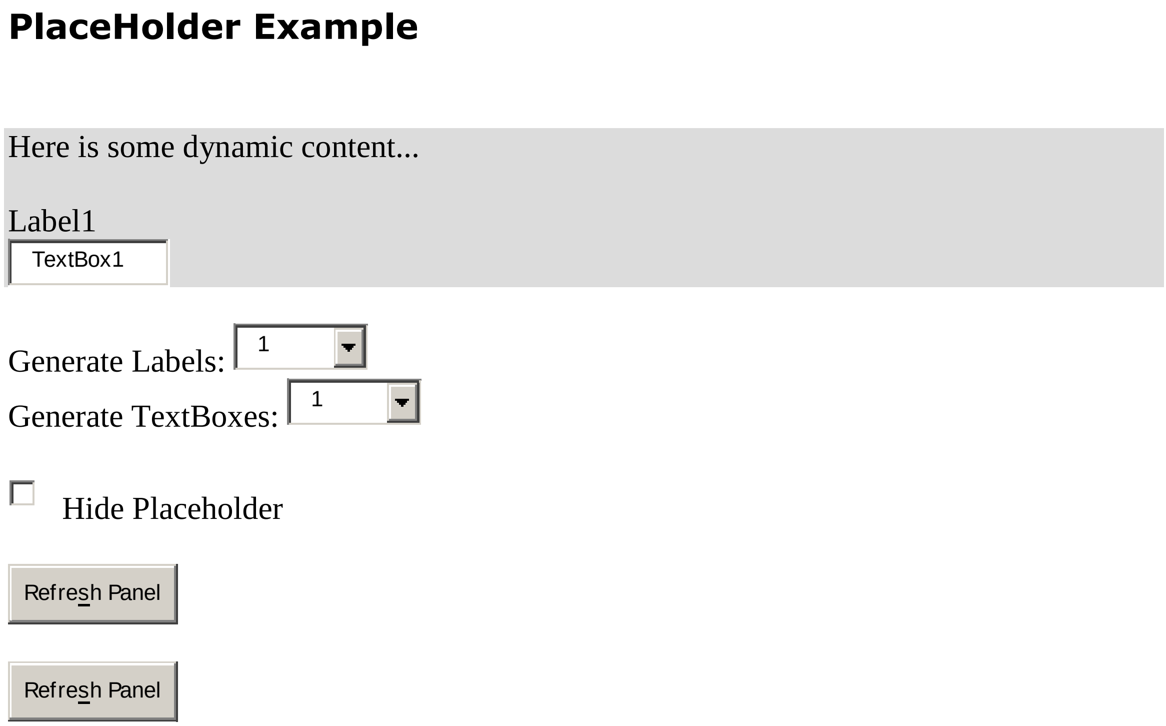
```
<asp:PlaceHolder runat="server"></asp:PlaceHolder>
```

▪ Example:

```
<html>
<head>

    <script language="C#" runat="server">

        void Page_Load(Object sender, EventArgs e) {

            // Show/Hide Panel Contents

            if (Check1.Checked) {
                PlaceHolder1.Visible=false;
            }
            else {
                PlaceHolder1.Visible=true;
            }

            // Generate label controls

            int numlabels = int.Parse(DropDown1.SelectedItem.Value);
            for (int i=1; i<=numlabels; i++) {
                System.Web.UI.WebControls.Label l = new
System.Web.UI.WebControls.Label();
                l.Text = "Label" + i.ToString();
                l.ID = "Label" + i.ToString();
               PlaceHolder1.Controls.Add(l);
               PlaceHolder1.Controls.Add(new LiteralControl("<br>"));
            }

            // Generate textbox controls

            int numtexts = int.Parse(DropDown2.SelectedItem.Value);

            for (int i=1; i<=numtexts; i++) {
                System.Web.UI.WebControls.TextBox t = new
System.Web.UI.WebControls.TextBox();
                t.Text = "TextBox" + i.ToString();
                t.ID = "TextBox" + i.ToString();
               PlaceHolder1.Controls.Add(t);
               PlaceHolder1.Controls.Add(new LiteralControl("<br>"));
            }
        }

    </script>

</head>
<body>

    <h3><font face="Verdana">Panel Example</font></h3>

    <form runat=server>

        <asp:PlaceHolder id="PlaceHolder1" runat="server"
            BackColor="gainsboro"
            Height="200px"
            Width="300px">

            Here is some static content...
            <p>

        </asp:PlaceHolder >

        <p>

        Generate Labels:
        <asp:DropDownList id=DropDown1 runat="server">
            <asp:ListItem Value="0">0</asp:ListItem>
            <asp:ListItem Value="1">1</asp:ListItem>
            <asp:ListItem Value="2">2</asp:ListItem>
            <asp:ListItem Value="3">3</asp:ListItem>
            <asp:ListItem Value="4">4</asp:ListItem>
        </asp:DropDownList>

        <br>

        Generate TextBoxes:
        <asp:DropDownList id=DropDown2 runat="server">
```

```
            <asp:ListItem Value="0">0</asp:ListItem>
            <asp:ListItem Value="1">1</asp:ListItem>
            <asp:ListItem Value="2">2</asp:ListItem>
            <asp:ListItem Value="3">3</asp:ListItem>
            <asp:ListItem Value="4">4</asp:ListItem>
        </asp:DropDownList>

        <p>
        <asp:CheckBox id="Check1" Text="Hide Panel" runat="server"/>

        <p>
        <asp:Button Text="Refresh Panel" runat="server"/>

    </font>
    </form>

</body>
</html>
```

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



PlaceHolder Example

Here is some dynamic content...

Label1

TextBox1

Generate Labels: 1

Generate TextBoxes: 1

□ Hide Placeholder

➢ MultiView And View Control:

▪ Definition and Usage:

The MultiView control represents a control that acts as a container for groups of View controls. It allows you to define a group of View controls, where each View control contains child controls, for example, in an online survey application. Your application can then render a specific View control to the client based on criteria such as user identity, user preferences, or information passed in a query string parameter.

The following sample illustrates a MultiView control hosting 3 View controls.

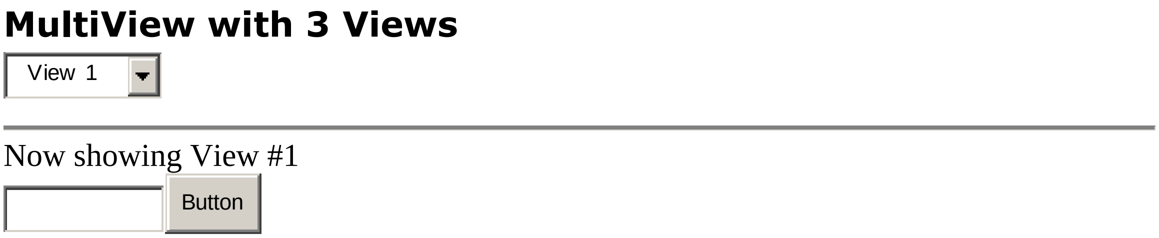
▪ Syntax:

```
    <asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
                <asp:View ID="View1" runat="server">
               </asp:View>
    </asp:MultiView>
```

▪ Example:

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">

    protected void DropDownList1_SelectedIndexChanged(object sender,
EventArgs e)
    {
        MultiView1.ActiveViewIndex =
Convert.ToInt32(DropDownList1.SelectedValue);
    }

</script>
<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>
            <h3>
                <font face="Verdana">MultiView with 3 Views</font>
            </h3>
            <asp:DropDownList ID="DropDownList1" runat="server"
AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
                <asp:ListItem Value="0">View 1</asp:ListItem>
                <asp:ListItem Value="1">View 2</asp:ListItem>
                <asp:ListItem Value="2">View 3</asp:ListItem>
            </asp:DropDownList><br />
            <hr />
            <asp:MultiView ID="MultiView1" runat="server"
ActiveViewIndex="0">
                <asp:View ID="View1" runat="server">
                    Now showing View #1<br />
                    <asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox><strong> </strong>
                    <asp:Button ID="Button1" runat="server"
Text="Button" /></asp:View>
                <asp:View ID="View2" runat="server">
                    Now showing View #2<br />
        <asp:HyperLink ID="HyperLink1" runat="server"
NavigateUrl="http://www.asp.net">HyperLink</asp:HyperLink>
                    <asp:HyperLink ID="HyperLink2" runat="server"
NavigateUrl="http://www.asp.net">HyperLink</asp:HyperLink></asp:View>
                <asp:View ID="View3" runat="server">
                    Now showing View #3<br />
          <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
                </asp:View>
            </asp:MultiView></div>
    </form>
</body>
</html>
```

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



MultiView with 3 Views

Now showing View #1