INTRODUCTION TO HTML CONTROLS, SERVER CONTROLS AND VALIDATION CONTROLS

There are three types of the controls:

- HTML Controls
- Web Server Controls
- Validation Controls

HTML Controls

- HTML Forms are required when you want to collect some data from the site visitor.
- **For example:** During user registration you would like to collect information such as name, email address, credit card details etc.
- A form will take input from the site visitor and then will post it to a back-end application such as ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.
- There are various form elements or controls available like text fields, text area fields, drop-down menus, radio buttons, checkboxes etc.
- By default, HTML elements on an ASP.NET Web page are not available to the server.

Web Server Controls

- Web Controls are very similar to the HTML Server controls such as Button, Textbox and Hyper link except the web controls have a standardized set of property names.
- Web controls provide the same functionality as their HTML server control. However, Web control includes additional methods, events and properties.

Validation Controls

- Validation controls are used to validate the values that are entered into controls of the page.
- Validation controls perform client-side validation, server-side validation or both. Validation controls offer the following advantages:
  - You can associate one or more validation controls with each control that you want to validate.
  - You can specify programmatically whether validation should occur which is useful if you want to provide a cancel button so that the user can exit without having to fill valid data in all of the fields.
  - The validation controls automatically detect whether validation should be performed on the client side or the server side.

Difference between HTML Control & ASP.NET Web Control

HTML Control	ASP.NET Web Control
• Runs at client side	• Runs at server side
• Can run on server side by adding runat="server"	• Cannot run on client side
• Does not provide state management	• Provides state management
• Executes fast	• Executes slow
• Do not have separate class for controls	• Have separate class for controls
• Does not support object-oriented paradigm	• Supports object-oriented paradigm
• Limited set of properties and/or methods	• Rich set of properties and/or method
• Example `<input type="text" ID="txtname">`	• Example `<asp:TextBox ID="txtname" runat="server"></asp:TextBox>`

WORKING WITH PROPERTIES, EVENTS & METHODS OF SERVER CONTROLS*

- Asp.net provides many built-in server controls that can be easily drag and drop to the web pages.

Common Properties for all controls

Property	Description
BackColor	Background color.
BorderColor	Border color.
CssClass	CSS class.
Height	Gets or sets the Height of the Web server control.
ID	Identifier for the control.
ToolTip	Gets or sets the text displayed when the mouse pointer hovers over the web server control.
Visible	It indicates whether a server control is visible.
Width	Gets or sets the width of the Web server control.
Runat	runat="server" Property.
Text	Gets or sets the text caption displayed on the control.

Common Event for all controls

Event	Description
Click	The Button, HyperLink, ImageButton, and LinkButton controls send this event when users click them.
TextChanged	The TextBox control send this event when text in the TextBox Changed.
Init	Called when the control is initialized. This is the first event called for every control.
Load	Called when the Page object loads the control.
PreRender	Called right before the control is rendered into the HTML result stream.
Disposed	Called when the control is released from memory. This call can happen at any time after the page is fully rendered, when the .NET garbage collector runs.

Button

- Displays a push button.
- Buttons in an ASP.NET Web page allow users to send a command.
- Buttons submit the page to the server and cause it to be processed along with any pending events.
- ASP.NET provides three types of button control.
  - Button: It displays text within a rectangular area.
  - Link Button: It displays text that looks like a hyperlink.
  - Image Button: It displays an image.
- Special Attributes:
  - ✓ **ImageUrl** - For image button control only. The image to be displayed for the button.
  - ✓ **PostBackUrl** - The URL of the page that is requested when the user clicks the button.
  - ✓ **OnClick** - Attach server side method that will fire when button will be clicked
  - ✓ **OnClientClick** - Attach a client side (javascript) event that will fire when button will be clicked
  - ✓ **CauseValidation** –This indicates whether validation will be performed when a button is clicked
- **Example**

```
<asp:Button ID="Button1" runat="server" Text="Click Me" />
```

TextBox

- The TextBox control is used to create a text box where the user can input text.
- The TextBox Web server control provides a way for users to type information into an ASP.NET Web page including text, numbers and dates.
- This control is used to take input from user into a default string format.
- Special Attributes:
  - **TextMode** - Specifies the type of text box. SingleLine creates a standard text box, MultiLine creates a text box that accepts more than one line of text and the Password causes the characters that are entered to be masked. The default is SingleLine.
  - **MaxLength** - The maximum number of characters that can be entered into the text box.
  - **ReadOnly** - Determines whether the user can change the text in the box; default is false, i.e., the user can change the text.
- **Example**

```
<asp:TextBox ID="TextBox1" runat="server" MaxLength="10"
TextMode="MultiLine"></asp:TextBox>
```

Label

- The Label Web server control provides a way to display text programmatically control in an ASP.NET Web page.
- This Control is used to display information on web page.

- **Example**

```
<asp:Label ID=" Label1" runat="server" Visible="true" Text="Hi this is darshan"></asp:Label>
```

Hi this is darshan

Compare asp:label and asp:literal

asp:label	asp:literal
We can apply CSS	We cannot apply CSS
Label control wraps the text in a tag when rendered	Literal control does not wrap the text in a tag when rendered
By default, text property of label has "label"	By default, literal text has blank

CheckBox

- This control is used to select multiple values from the list.
- It displays a check box that allows the user to select value, if it is selected then its value will become TRUE otherwise its value will become FALSE.
- Special Attributes:
  - **Checked** - Specifies whether it is selected or not, default is false.
  - **TextAlign** –Side of the checkbox where the text will be displayed
- **Example: -**

```
<asp:CheckBox ID=" CheckBox1" runat="server" AutoPostBack="true" Visible="true" ></asp:CheckBox>
```

Apple

Radio Button

- The RadioButton control is used to display a radio button.
- RadioButton present a group of options from which the user can select just one option.
- Special Attributes:
  - **GroupName**- Name of the group the control belongs to.
  - **Checked** - Specifies whether it is selected or not, default is false.
  - **TextAlign** – Side of the RadioButton where the text will be displayed
- **Example**

```
<asp:RadioButton ID="RadioButton1" runat="server" GroupName="Gender" Text="Male" />
<asp:RadioButton ID="RadioButton2" runat="server" GroupName="Gender" Text="Female" />
```

Male Female

Link Button

- The LinkButton control is used to create a hyperlink button.
- Displays a hyperlink-style button control on a Web page.

- This control looks like a HyperLink control but has the same functionality as the Button control.
- **Example**

```
<asp:LinkButton  ID="LinkButton1" runat="server"
PostBackUrl="~/Home.aspx">Click Here!</asp:LinkButton>
```

Click Here!

Image

- The Image control is used to display an image on the web page.
- Special Attributes:
  - **AlternateText**- Alternate text to be displayed in absence of the image.
  - **ImageAlign** - Alignment options for the control.
  - **ImageUrl** - Path of the image to be displayed by the control.
- **Example**

```
<asp:Image ID="Image1" runat="server" AlternateText="Image Not
Available!" ImageAlign="Left" ImageUrl="~/App_Data/bird.jpg" />
```

Hyperlink

- The HyperLink control is used to create a hyperlink.
- This control provides easy navigation between various Pages.
- Special Attributes:
  - **NavigateUrl** - The target URL of the link.
- **Example**

```
<asp:HyperLink ID="hlDarshan" runat="server"
NavigateUrl=https://www.darshan.ac.in
Target="_blank">HyperLink</asp:HyperLink>
```

Panel

- The Panel control works as a container for other controls on the page.
- It controls the appearance and visibility of the controls it contains.
- Special Attributes:
  - **BackImageUrl** - URL of the background image of the panel.
  - **Direction** - Text direction in the panel.
  - **HorizontalAlign** - Horizontal alignment of the content in the panel.
  - **ScrollBars** - Specifies visibility and location of scrollbars within the panel.
  - **Wrap** - Allows text wrapping.

- **Example**

```
<asp:Panel ID="panele" runat="server" HorizontalAlign="Left"
ScrollBars="Auto" Direction="LeftToRight" Wrap="true" Visible="false">
Hey This is example of Panel Control <br />It can contains more than one
control <br /></asp:Panel>
```

PlaceHolder

- PlaceHolder is an asp.net web server control which used to store dynamically added web server controls on the web page. by using a PlaceHolder control we can dynamically add Label, TextBox, Button, RadioButton, Image and many more web server controls in an asp.net web page.
- PlaceHolder server control act as a container control to store server controls that are dynamically added to the web page.
- PlaceHolder control does not provide any visible output. We only can see the dynamically added server controls inside a PlaceHolder control as child controls. We can add and remove server controls programmatically in the PlaceHolder control.
- **Example**

```
<h2 style="color:Navy">PlaceHolder Example</h2>
<asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
<asp:Button ID="Button1" runat="server" Text="Add Button Control"
     />

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim myButton As HtmlButton = New HtmlButton()
        myButton.InnerText = "Button 1"
        PlaceHolder1.Controls.Add(myButton)
 End Sub
```

FileUpload

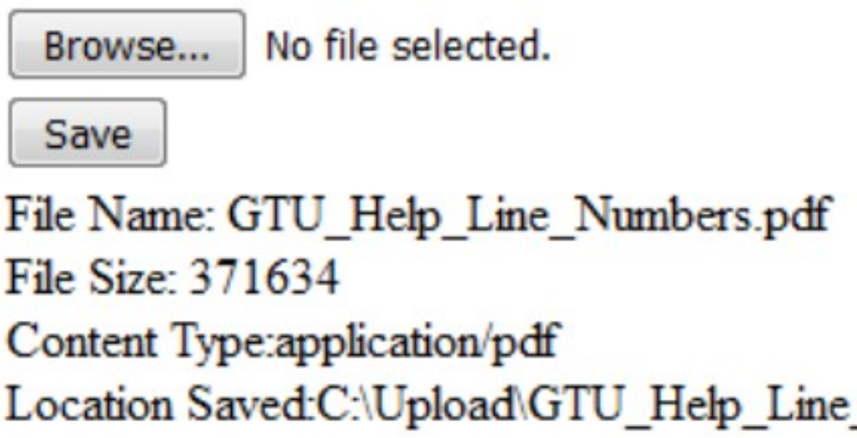
- The FileUpload control allows the user to browse and select the file to be uploaded, providing a browse button and a text box for entering the filename.
- Once, the user has entered the filename in the text box by typing the name or browsing, the SaveAs method of the FileUpload control can be called to save the file to the disk.
- **Example**

```
<h2 style="text-align:center"> File Upload Control Example </h2>
<br />
```

```
<table><tr><td>
<asp:FileUpload ID="fule" runat="server" />
</td></tr>
<tr><td>
<asp:Button ID="btnSave" Text="Save" runat="server" />
</td></tr>
<tr><td>
<asp:Label ID="lblDisplay" runat="server"></asp:Label>
</td></tr>
</table>
```

```
Protected Sub btnSave_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles btnSave.Click
        Dim sb As StringBuilder = New StringBuilder
                fule.SaveAs("C:\Upload\" & fule.FileName)
                lblDisplay.Text = "File Name: " & fule.PostedFile.FileName
& "<br>" & "File Size: " & fule.PostedFile.ContentLength & "<br>" &
"Content Type:" & fule.PostedFile.ContentType & "<br>" & "Location
Saved:C:\Upload\" & fule.FileName

End Sub
```



LIST CONTROLS

- Asp.Net provides many built in List controls to show the data in the form of list.
- User can select one or more value from the list.
- List controls are many and they are used as per user's requirement

CheckBoxList

- Provides a checkbox with each data and user can select more than one item.
- A check box list presents a list of independent options.
- These controls contain a collection of ListItem objects that could be referred to through the Items property of the control.
- Special Attributes:
  - **RepeatLayout** - This attribute specifies whether the table tags or the normal html flow to use while formatting the list when it is rendered. The default is Table.

- **RepeatDirection** - It specifies the direction in which the controls to be repeated. The values available are Horizontal and Vertical. Default is Vertical.
- **RepeatColumns** - Specifies the number of columns to use when repeating the controls; default is 0.

- **Example**

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
<asp:ListItem>Cricket</asp:ListItem>
<asp:ListItem>Football</asp:ListItem>
<asp:ListItem>Hockey</asp:ListItem>
</asp:CheckBoxList>
```

Cricket
Football
Hockey

ListBox

- Provides all the data in a box format and user can select one and more data.
- List boxes contain one or more list items.
- These lists can be loaded either by code or by the ListItem objects.
- Special Attributes:
  - **Items** - The collection of ListItem objects that represents the items in the control.
  - **Rows** - Specifies the number of items displayed in the box. If actual list contains more rows than displayed then a scroll bar is added.
  - **SelectionMode** - Indicates whether a list box allows single selections or multiple selections.
  - **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 ("").
- **Example**

```
<asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
<asp:ListItem>Apple</asp:ListItem>
<asp:ListItem>Banana</asp:ListItem>
<asp:ListItem>Mango</asp:ListItem>
</asp:ListBox>
```

Apple
Banana
Mango

DropDownList

- Provides a list of items with only one item visible at a time and user can select single or multiple items.
- Special Attributes:
  - **Items** - The collection of ListItem objects that represents the items in the control.
- **Example**

```
<asp:DropDownList ID="DropDownList1" runat="server">
<asp:ListItem>Rajkot</asp:ListItem>
<asp:ListItem>Morbi</asp:ListItem>
<asp:ListItem>Jamnagar</asp:ListItem>
</asp:DropDownList>
```

Rajkot

- **DropDownList.Items.Add** : It allows you to add new ListItem into the DropDownList. This item will be added as the Last item of the DropDownList.
  `Example: DropDownList.Items.Add(new ListItem("Default Panel","0"))`
- **DropDownList.Items.Insert :** It allows you to specify the index of the item within the DropDownList where you want to insert the ListItem.
  `Example: DropDownList.Items.Insert(0, new ListItem("Default Panel","0"))`

VALIDATION CONTROLS

- Asp.Net provides various validation controls for error free access for the web page. This validation Controls validates the input for controls. So that any mismatching and faulty data cannot be inserted to the data storage.
- Various Validation controls are:
  - Required Field Validator
  - Range Validator
  - Regular Expression Validator
  - Compare Validator
  - Custom Validator
  - Validation Summary

Common Properties of Validation controls

Property	Description
ID	Identifier for the control.
Runat	Server side Controls are useless without runat="server" Property.
ControlToValidate	Indicates the input control to validate.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Display	The display behavior for the validation control. Legal values are: ▪ **None**-the control is not displayed. Used to show the error message only in the Validation Summary control. ▪ **Static**-the control displays an error message if validation fails. Space is reserved on the page for the message even if the input passes validation. ▪ **Dynamic**-the control displays an error message if validation fails. Space is not reserved on the page for the message if the input passes validation.
ValidationGroup	The logical group of multiple validators, where this control belongs.

Required Field Validator*

- It provides validation for emptiness of any control.
- This Validator is applied to such controls where input is necessary.

- **Example**

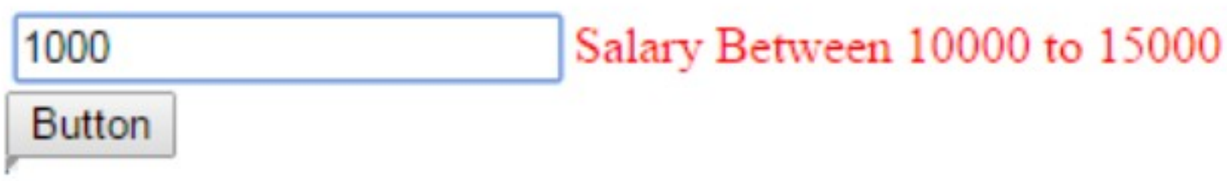
```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1" ErrorMessage="Text Box
Must Not Empty!">
</asp:RequiredFieldValidator>
<asp:Button ID="Button1" runat="server" Text="Button" />
```

Range Validator*

- The RangeValidator control verifies that the input value falls within a predetermined range.
- It provides validation for range of an input by setting minimum value and maximum value.
- This Validator is used when input is taken between certain type of values.
- Special Attributes:
  - **Type** - It defines the type of the data. The values are: Currency, Date, Double, Integer, and String.
  - **MinimumValue** - It specifies the minimum value of the range.
  - **MaximumValue** - It specifies the maximum value of the range.

- **Example**

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RangeValidator ID="RangeValidator1" runat="server"
ControlToValidate="TextBox1" ErrorMessage="Salary Between 10000 to
15000"
MaximumValue="15000" MinimumValue="10000"
Type="Integer"></asp:RangeValidator>
<asp:Button ID="Button1" runat="server" Text="Button" />
```



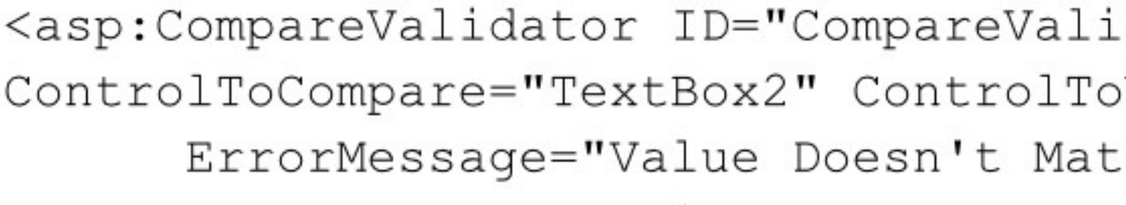
Regular Expression Validator*

- The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.
- Special Attributes:
  - ✓ **ValidationExpression** - Specifies the expression used to validate input control.

- **Example**

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server"
```

```
ControlToValidate="TextBox1" ErrorMessage="Please Enter Valid Email!"
validationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)*"></asp:RegularExpressionValidator>
<asp:Button ID="Button1" runat="server" Text="Button" />
```



Compare Validator*

- The CompareValidator control compares a value in one control with a fixed value or a value in another control.
- Special Attributes:
  - **Type** - It specifies the data type.
  - **ControlToCompare** - It specifies the value of the input control to compare with.
  - **ValueToCompare** - It specifies the constant value to compare with.
  - **Operator** - It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.
- **Example**

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
<asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToCompare="TextBox2" ControlToValidate="TextBox1"
      ErrorMessage="Value Doesn't Match! "></asp:CompareValidator>
<asp:Button ID="Button1" runat="server" Text="Button" />
```

Custom Validator

- The CustomValidator control allows you to write a method to handle the validation of the value entered.
- Special Attributes:
  ClientValidationFunction - Specifies the name of the client-side validation script function to be executed.
- **Example**

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:CustomValidator ID="CustomValidator1" runat="server"
            ClientValidationFunction="usernamevalidation"
ControlToValidate="TextBox1"
```

```
            ErrorMessage="Username between 8 to 16
Character!"></asp:CustomValidator>
<asp:Button ID="Button1" runat="server" Text="Button" />


Protected Sub CustomValidator1_ServerValidate(ByVal source As Object,
ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs)
Handles CustomValidator1.ServerValidate
        If Len(args.Value) < 8 Or Len(args.Value) > 16 Then
            args.IsValid = False
        Else
            args.IsValid = True
        End If
    End Sub
```

abc Username between 8 to 16 Character!

Button

Validation Summary

- The ValidationSummary control is used to display a summary of all validation errors occurred in a Web page.
- The error message displayed in this control is specified by the ErrorMessage property of each validation control. If the ErrorMessage property of the validation control is not set, no error message is displayed for that validation control.
- Special Attributes:
  - **DisplayMode** - How to display the summary. Legal values are: BulletList, List, SingleParagraph.
  - **HeaderText** - A header in the ValidationSummary control.
  - **ShowMessageBox** - A Boolean value that specifies whether the summary should be displayed in a message box or not.
  - **ShowSummary** - A Boolean value that specifies whether the ValidationSummary control should be displayed or hidden.
- **Example**

```
<asp:ValidationSummary ID="VS1" runat="server"
HeaderText="Following fields must be correct! "ShowMessageBox="false"
DisplayMode="BulletList" ShowSummary="true" ForeColor="Red" />
```

abc Username between 8 to 16 Character!

abc.gmail.com Enter Valid Email!

Following fields must be correct!

- Username between 8 to 16 Character!
- Enter Valid Email!

Button

Validation Group

- The ValidationGroup property specifies which group of controls is validated on validation.
- This property is mostly used when there are several buttons in a form.
- **Example**

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
ErrorMessage="Required" ControlToValidate="TextBox1"
ValidationGroup="gr1"></asp:RequiredFieldValidator>

<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
runat="server"
ErrorMessage="Required" ControlToValidate="TextBox2"
ValidationGroup="gr1"></asp:RequiredFieldValidator>
<asp:Button ID="Button1" runat="server" Text="Button"
ValidationGroup="gr1" />

<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator3"
runat="server"
ErrorMessage="Required" ControlToValidate="TextBox3"
ValidationGroup="gr2"></asp:RequiredFieldValidator>

<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator4"
runat="server"
ErrorMessage="Required" ControlToValidate="TextBox4"
ValidationGroup="gr2"></asp:RequiredFieldValidator>
<asp:Button ID="Button2" runat="server" Text="Button"
ValidationGroup="gr2" />
```