ASP.NET – Standard Controls

Intrinsic Controls

Rakesh Singh

NARESH I TECHNOLOGIES Hyderabad

Standard Controls:

Using the Standard Controls:

The ASP.Net standard web server controls provides the wide range of in-built features over HTML controls. Likewise HTML, the ASP.Net standard controls also contain textbox, label and button controls but they also include several advanced server controls such as ImageMap, Calendar, Adrotator, View, MultiView, Wizard etc. These advanced controls have their simple building block that renders the HTML markup code at the time of page rendering.

In this document, you learn how to use the core controls contained in the ASP.NET 4.5 Web Application Framework. These are controls that you'll use in just about any ASP.NET web application that you build. You learn how to display information to users by using the Label and Literal controls. You learn how to accept user input with the TextBox, CheckBox, and RadioButton controls. You also learn how to submit forms with the button controls and many more.

ASP.NET controls are the heart of ASP.NET Web Application Framework. An ASP.NET control is a .NET class that executes on the server and renders certain content to the browser.

In ASP.NET there are several standard web server controls that can be used to create ASP.Net web pages. Some of the standard server controls map to the single HTML element for example textbox, label or button but they differ from their basic HTML markup modal. The ASP.NET standard web server controls provide much more rich customizable properties that enable you to modify the behaviour or appearance programmatically. Some server controls may render with group of HTML controls along with complex HTML markup elements for example Tables to represent the final output.

Following are the ASP.NET Standard web server controls that are used to build simple ASP.NET Web Forms:

I) Intrinsic Controls (Basic Controls):

- Label
- TextBox
- Button
- LinkButton
- ImageButton
- CheckBox
- RadioButton
- DropDownList
- ListBox
- CheckBoxList
- RadioButtonList
- BulletedList
- Image
- Hyperlink
- FileUpload
- Literal
- HiddenField
- PlaceHolder
- Panel
- Table

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Label Control:

This control is mostly used to display texts on a web form. Programmatically we can hide/unhide or set the text for label, and this feature helps the developer to display some message when any task completed.
OR
The Label control is used to display the static text at specific locations on the web page. You can also set the text property of the Label control dynamically (programmatically).

The **ASP.Net Label control** enables you to display the static text on the web page. You can use the text property of the Label Control to display the text on the web page that can be specified at the time of designing or programmatically at runtime. The Label web server control allows you to set the value as **plain text** as well as **HTML code** or scripts. You can customize the style of displayed text by adding HTML tags and inline CSS styles. You can also attach the client side events to the Label web server control as it renders as HTML tag on the ASP.Net web page. Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc. are implemented through style properties of . You can set its Text property either by setting Text properties in the .aspx page or from server side code.

Properties of ASP.Net Label Control:

The Label web server controls consists of two types of properties such as CSS based style properties that enable you to customize the look and feel of the text displayed and the second type of properties allow you specify the text value and its association with other controls on the web page such as textbox control.
Following are the two commonly used properties of ASP.Net Label control:
1. AssociatedControlID: It accepts the ID of the input type control used on the web page as a form object for which you want the Label control to appear as caption.
2. AccessKey: Define the keyboard shortcut key used by the control.
3. Enabled: It is used to gets or sets a value indicating the enabled state of the control. By default is set to true.
4. EnableTheming: It indicates whether the control can be themed. By default is set to true.
5. EnableViewState: It indicates whether the control automatically saves its state for use in round-trip. By default is set to true.
6. SkinID: It is used to specify the SkinId of the control skin that provides the skin of the control.
7. Text: It accepts the string type value that appears as static text on the web page. You can set the value of text property of Label control in the HTML source code as well as dynamically.
8. ToolTip: It is used to gets or set the text to display tooltip when the mouse is over the control.
9. Visible: It indicates whether the control is visible and rendered. By default is set to true.

Other than the above properties it also contains CSS based properties that allow you to customize the style of the text displayed. These properties include BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Font-Bold, Font-Italic, Font-Names, Font-Overline, Font-Size, Font-Strikeout, Font-Underline, ForeColor, Height and Width.

Label Control Examples:

Label Control Text Property:
HTML Code:

```
<asp:Label ID="Label1" runat="server"  Text="This is Label Control" />
```

OR

```
<asp:Label ID="Label1" runat="server">This is Label Control</asp:Label>

<asp:Button ID="Button1" runat="server" Text="Submit" OnClick="Button1_Click" />
```

C# Code:

```
protected void Button1_Click(object sender, EventArgs e)
{
   Label1.Text = "Label text changed dynamically.";
}
```

In the above sample code we have used a Label Control and the ASP.Net Button control. We have attached the click event handler to the button control to raise the postback event. When the user will click the button control it will execute the provided code at server side and will assign the specified string value to the Text property of the ASP.Net Label control dynamically.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Using ASP.Net Label Control as Caption:

The **ASP.Net Label control** also provides the functionality to display it as an active **caption** in front of another web server control such as ASP.Net Textbox control. You can assign the text value to the Label control to show the description or title for the input type control placed inline to it. The Label control cannot act as an input control to accept the user input on the web page but you can associate it to the other web server control. You can assign the access key to the Label control working as a caption for other server control such as Textbox that enables you to set the focus to the caption associated Textbox control by pressing **ALT** and the access key simultaneously on the Keyboard.

Steps to Use the ASP.Net Label control as Caption:

1. The Label web server control has the AssociatedControlID property. You can assign the ID of the other web server control to this property for which you want to use the Label control as caption.

2. The other property of ASP.Net Label control i.e. Accesskey allows you to specify a single letter or number that can be used to set focus to the associated input type control for which you are setting the Label control as caption. You can use the ALT key and the specified letter or number simultaneously to set the focus to the input control. The access key feature work only in Internet Explorer 4.0 or later.

3. Last you can assign the caption text to the Text property of the Label control. You can underline the access key character in the Text property value to indicate the user that it is the access key for its associated input control.

```
<asp:Label ID="lblFirstName"
runat="server" Text="First <u>N</u>ame:" AccessKey="N" AssociatedControlID="txtFirstName"></asp:Label>

<asp:TextBox ID="txtFirstName" runat="server"></asp:TextBox>

<br />

<asp:Label ID="lblLastName" runat="server" Text="<u>L</u>ast Name:" AccessKey="L"

AssociatedControlID="txtLastName"> </asp:Label>

<asp:TextBox ID="txtLastName" runat="server"></asp:TextBox>

<br />
```

In the above code we have used 2 Label controls and 2 TextBox controls to illustrate the use of AssociatedControlID and AccessKey property of Label control. For the first Label control we have specified the letter "N" to the AccessKey and the ID of the textbox control to the AsscocitedControlID property. Similarly we have assigned the letter "L" to the AccessKey property of second Label control and ID of the second textbox control to the AssociatedControlID property of the second Label control. You can set the focus to first TextBox control by pressing ALT + N keys simultaneously and similarly ALT + L key for second textbox control.

Output:

First Name:

Last Name:

TextBox Control:

This control is used to enter information for processes. Like if we are wishing to create a web page where user can enter his username and password, surely take the advantage of text box here. We can change the TextBox TextMode property by SingleLine, MultiLine or Password.
OR
Text box controls are typically used to accept input from the user. A text box control can accept one or more lines to text depending upon the setting of the TextMode attribute.
OR
The TextBox control enables the user to enter text. By default, the TextMode of TextBox is SingleLine, but you can modify the behaviour of TextBox by setting the TextMode to Password or MultiLine.

The display width of TextBox is determined by its Columns property. If TextMode is MutliLine, the display height of TextBox is determined by the Rows property.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

The **ASP.Net TextBox control** enables you to display an input type textbox control on the ASP.net web page that allows the users to enter their input in text mode and submit it to the server side using **ASP.Net Button control**. You can get or set the value to or from the ASP.Net Textbox control using its Text property programmatically. It support both **plain text** as well as **HTML** code or scripts that can be submitted to the server side or can be displayed in the multiline enabled textbox control. The HTML code and CSS support of textbox control enables you to customize the styles of the text entered through it. This feature also leads to security vulnerabilities. That is why it is recommended to validate the input entered by the user into the ASP.Net textbox control to prevent the execution of malicious client side scripts. You can also attach the client side events to the textbox control that can be used to validate the user input.

Properties of ASP.Net TextBox Control:

The TextBox control consists of two types of properties such as CSS based style properties that allow you to customize the look and feel of the control as well as the input text entered by the users and the other type of properties handles the behaviour of textbox.

Following are the commonly used properties of ASP.Net Textbox control:

1. AutoCompleteType: It accepts the predefined keyword for example FirstName, LastName, Email etc. that provides the functionality to display the autocomplete list under the textbox control to enter the input from the previously entered values for the textboxes with same AutoCompleteType.

2. AutoPostBack: It is a Boolean type property that enables or disables the auto postback functionality of text box control that posts the web page to server when user enters the input into the textbox control jumps to the another control. By default is set to false.

3. CausesValidation: It is also a Boolean type property that enables or disables the validation controls associated with the textbox control to validate the input. By default is set to false.

4. Columns: It accepts the integer value to set the width of the multiline textbox. The integer value enables the control to accommodate the specified number of characters horizontally.

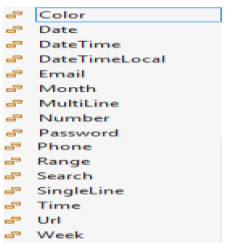
5. MaxLength: It also accepts the integer value that restricts the control to reject the characters exceeding the specified value. This property works with single line textbox control

6. ReadOnly: It is a Boolean property that enables or disables the textbox control whether to allow the user to change its value or not.

7. Rows: It accepts the integer value to set the height of the multiline textbox. The integer value enables the control to accommodate the specified number of characters vertically.

8. Text: It accepts the text value that you want to display in the textbox control initially. You can assign the value dynamically as well as get the value entered by the user through the Text property of ASP.Net textbox control.

9. TextMode: It accepts the predefined keyword such as **SingleLine**, **MultiLine** or **Password** that adjusts the behaviour of the textbox control on the web page. ASP.NET 4.5 supports many text modes based on HTML5 supports. These are following:



10. ValidationGroup: It accepts the name of the validation group to which it belongs that validates the user input when a postback occurs.

11. Wrap: It indicates whether the text should wrap or not. By default is set to true.

Other than the above properties it also contains CSS based properties that allow you to customize the style of the textbox and the input text. These properties include BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Font-Bold, Font-Italic, Font-Names, Font-Overline, Font-Size, Font-Strikeout, Font-Underline, ForeColor, Height and Width.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net TextBox Events

The Textbox control provides OnTextChanged event that can be used to read and validate the user input entered into the textbox.

TextBox Control Examples:

- **TextBox Control Text Property**
- **TextBox Control TextMode Property**
- **TextBox Control OnTextChanged Event**

Code for ASP.Net TextBox Control Text Property:

HTML Code:

```
<table width="400" border="0" cellpadding="5" cellspacing="0" align="center">
        <tr width="100">
          <td>
            <asp:Label ID="Label1" runat="server" Font-Size="13px" Text="Enter text: "></asp:Label></td>
          <td>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:Button ID="Button1" runat="server" Text="Submit" />
          </td>
        </tr>
        <tr>
          <td>
            <asp:Label ID="Label2" runat="server" Font-Bold="true" Font-Size="13px" Text="Output: "></asp:Label>
          </td>
          <td></td>
        </tr>
        <tr>
          <td colspan="2">
            <asp:Label ID="Label3" runat="server" Font-Size="13px"></asp:Label>
          </td>
        </tr>
</table>
```

C# Code:

```
protected void Page_Load(object sender, EventArgs e)
{
    Button1.Click += new EventHandler(Button1_Click);
}

protected void Button1_Click(object sender, EventArgs e)
{
    Label3.Text = string.Format("You Entered: <b>{0}</b>", TextBox1.Text);
}
```

Output:

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net TextBox TextMode Property:

The **TextMode property** of **ASP.Net TextBox control** sets the rendering behaviour of input type textbox on the web page. There are few predefined set of values that changes the behaviour of textbox control. Each value renders the control using different type of HTML source code such as **text**, **password** or **textarea**. You can set the appearance of textbox control according to the behaviour specified for it using its TextMode Property.

Following are the predefined values for TextMode property of ASP.Net textbox control:

1. SingleLine: It renders the textbox as input type text html form control that accepts text as a single line. It does not allow entering the newline character. It is the default TextMode for the textbox control.

2. MutliLine: It renders the textbox as textarea html form control that accepts the text as multiple lines. It allows entering the newline characters in the input string.

3. Password: It renders the textbox as input type password html form control that accepts the text as single line string but does not shows the input string as a plain text. It masks the text input with special dots or asterisk according the web browser.

Example for TextMode Property of ASP.Net TextBox:

HTML Code:

```
<b>Singleline TextBox</b>
<br />
<asp:TextBox ID="TextBox1" runat="Server" Width="300"></asp:TextBox>
<br /><br />
<b>Multiline TextBox</b>
<br />
<asp:TextBox ID="TextBox2" runat="Server" TextMode="multiLine"
Rows="4" Columns="50"
Text="TextBox with TextMode as Multiline">
</asp:TextBox>
<br /><br />
<b>Password TextBox</b>
<br />
<asp:TextBox ID="TextBox3" runat="Server" TextMode="Password"></asp:TextBox>
```

Output:

Singleline TextBox

Multiline TextBox

TextBox with TextMode as Multiline

Password TextBox

Output HTML source code:

```
<b>Singleline TextBox</b>
<br />
<input name="TextBox1" type="text" id="TextBox1" />
<br /><br />
<b>Multiline TextBox</b>
<br />
<textarea name="TextBox2" rows="2" cols="20" id="TextBox2">
</textarea>
<br /><br />
<b>Password TextBox</b>
<br />
<input name="TextBox3" type="password" id="TextBox3" />
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net TextBox Control OnTextChanged Event:

The **ASP.Net TextBox control** provides **OnTextChanged event** that enables you to get the input value entered by the user immediately after when he leaves the textbox control. The textbox control raises the TextChanged event when user jumps to another control on the web page after filling the information in the textbox control. It does not raise any event for each keystroke made by the user it occurs only when the user leaves the associated textbox control. You can handle the OnTextChanged event at server side that can be used to get the textbox value and validate it before submitting to the data store or further processing.

There are two different types of behaviours of OnTextChanged event of ASP.Net TextBox control that are based on the value of its **AutoPostBack property** as follow:

1. By default the AutoPostBack property value is false that does not allow the TextBox control to raise the TextChanged event immediately after the user leaves the control. It fires the event and executes the server side OnTextChanged handler code at the time the form is posted to the server.

2. The second type of behaviour is executing the TextChanged event immediately when user leaves the textbox control that posts the web form to the server. This instant behaviour can be set by setting the AutoPostBack property of textbox control to true.

Attaching the OnTextChanged Event to ASP.Net TextBox:

You can attach the TextChanged event of textbox control in two ways as follows:

1. Double clicking the textbox control for which you want to generate the server side OnTextChanged event handler. In this document we have also used the same approach to handle the TextChanged event of textbox control.

2. You can also attach the TextChanged event programmatically in the page load method of ASP.Net web page E.g.:

```
protected void Page_Load(object sender, EventArgs e)
{
    TextBox1.TextChanged += new EventHandler(TextBox1_TextChanged);
}

protected void TextBox1_TextChanged(object sender, EventArgs e)
{
}
```

When you will write the ID of the textbox control inside the Page_Load method for example TextBox1, and will type the "." Dot operator it will display the list of properties and methods of textbox control. Select the TextChanged event from the intellisense and press "+=" and then tab key from the keyboard. It will generate the event handler server side code automatically.

Example of TextChanged Event:

HTML Code

```
<asp:Label ID="Label1" runat="server" Text="Enter Text: "></asp:Label>
<asp:TextBox ID="TextBox1" runat="server" AutoPostBack="true" OnTextChanged="TextBox1_TextChanged"></asp:TextBox>
 
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Submit" />
<br />
<asp:Label ID="Label2" runat="server" Text="Output: "></asp:Label>
```

C# Code:

```
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    Label2.Text += "TextChanged Event: " + TextBox1.Text;
    Label2.Text += "<br />";
}

protected void Button1_Click(object sender, EventArgs e)
{
    Label2.Text += "Click Event: " + TextBox1.Text;
}
```

Enter Text: Submit
Output:

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Button Control:

Very widely used control. Writing some lines about this control is very tinny thing. It is used everywhere like have to submit form, login, delete, create user, etc.
The Button control provides a command button-style control that is used to post a Web Forms page back to the server.
OR
The button server control sends the page to the PostBack and executes the action specified in the associated event handler for it.
OR
ASP .Net provides three types of button controls: buttons, link buttons and image buttons. As the names suggest a button displays text within a rectangular area, a link button displays text that looks like a hyperlink. And an Image Button displays an image.
When a user clicks a button control, two events are raised Click and Command.

The **ASP.Net Button control** provides the functionality to display a push button control on the ASP.Net web page that enables you to post the data back to server side and perform some specific action over there. The Button control belongs to the family of ASP.Net standard web server controls that allows you to send the input data of web form to the server. The ASP.Net Button control can perform in two ways as:

1. Submit button

2. Command button

Both the behaviours of Button control are special in their own way. By default the button control sends the form data to the server side using HTML **POST** method that can be processed there under the associated click event handler. The command button performs this action along with its associated command name that enables you to create single event handler for multiple command buttons and execute the server code based on the command name passed by the button being clicked by the user for example Sort, Order, Edit, Delete etc. But the submit button does not have a command name associated to it that directly performs the specified server side action.

Properties of ASP.Net Button Control:

Following are the properties of Button control that allows you to control and customize its behaviour:

1. CausesValidation: It accepts the Boolean values as true/false that performs the web page validation check when user clicks the button control.

2. CommandArgument: You can specify the additional information as a value for CommandArgument property of Button control that can be accessed at server side when it raises the command based click event.

3. CommandName: You can specify the command name for the Button control that enables you to perform the server side action based on it such as edit, update, delete etc.

4. OnClientClick: This property allows you to call the client side script function that will execute before sending the data to the server.

5. PostBackUrl: You can specify the URL of the other web page to which you want to post the input data of the current web form.

6. Text: It enables you to display the caption text value of button control.

7. UseSubmitBehaviour: It changes the way of sending the data to the server based on its true/false Boolean type value. If UseSubmitBehaviour is true then button control uses the client browser's submit mechanism otherwise it uses the ASP.Net postback mechanism. By default its value is true.

8. ValidationGroup: It enables you to specify the name of the validation group that allows the button to perform the validation check (CausesValidation) only for the controls that belong to the specified validation group.

Event Handlers for ASP.Net Button Control:

You can attach the following events with Button control that are raised at the time of button click:

1. OnClick: It raises the server side click event of the Button control.

2. OnCommand: It raises the command based event of the Button control.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Button Control Examples:

- **Button Control**
- **Button Click Event**
- **Button Command Event**
- **Button OnClientClick Property**
- **Button Confirm Message Box**

Sample Code for Button Control:

HTML Code:

```
<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
```

C# Code:

```
protected void Button1_Click(object sender, EventArgs e)
  {
     Button1.Text = "Hey!!! You clicked me.";

  }
```

ASP.Net Button Click Event:

The ASP.Net Button click event is raised when it is being clicked by the user. This event of Button control submits the web form data to the server and performs the specified action there. The click event is commonly used when the ASP.Net Button has no command name associated with it and is placed as a submit button on the web page to send the user input to the server using HTML Post method. In ASP.Net each web server control has its own set of events that can be raised and handled at server side. The Button server control has OnClick method that can be used to trap the click event. It allows you to handle the event without attaching a delegate.

Attaching Click Event with Button Control:

There are 2 ways for attaching the OnClick event handler with ASP.Net Button control:

1. Double clicking the button control in design view of web page while working in Visual Studio. By default it will generate the code for OnClick event of that Button as follows:

```
<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />

protected void Button1_Click(object sender, EventArgs e)
{

}
```

2. You can also attach the event programmatically using server code at Page_Load event of web page as follows:

```
protected void Page_Load(object sender, EventArgs e)
{
     Button1.Click += new EventHandler(Button1_Click);

}
protected void Button1_Click(object sender, EventArgs e)
{
        //Write the appropriate code
}
```

For attaching the event programmatically write the ID of the button control inside the Page_Load event, as "**Button1**" used in above sample code. Next use the dot operator to open the intellisense showing properties and methods available for it and select the **Click event** from the list. Then add **"+="** and press tab to auto generate the handler code for it. You will get the code similar to the sample code shown above.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net Button Command Event:

The **ASP.Net Button Command event** is used when there is a command name associated with it. This event of Button control also fires when it is being clicked by the user. You can use the **Command event** of Button control for multiple buttons placed on the ASP.Net web page. Each button control will have its own Command Name associated with it that will direct the command event handler to execute the server code based on the name of command for example edit, delete, sort etc. The ASP.Net Button control has **OnCommand method** that can be used to trap the click and perform the action at server side according to the value of **CommandName property** associated with the clicked button. You can handle this event without attaching any delegate.

The Command event handler of Button control uses **CommandEventArgs** class object as parameter that provides the access to the two properties of Button Control:

1. CommandName

2. CommandArgument

You can specify the values in the HTML markup code block of ASP.Net button control. Both CommandName and CommandArgument property accept string type value that can also be specified programmatically.

Attaching Command Event with Button Control

There are 2 ways for attaching the **OnCommand event** handler with ASP.Net Button control:

1. Click to select the Button control in the Design view of web page in Visual Web Developer. Then press F4 to open its property fly-out window. Change the property window's view to events as shows in the figure below:

Double click on the Command method to generate the command event handler code for the selected Button control.

```
<asp:Button ID="Button1" runat="server" Text="Button 1"      OnCommand="Button1_Command" />
```

It will generate the following code for the Button Control:

```
protected void Button1_Command(object sender, CommandEventArgs e)
   {
   }
```

2. You can also attach the event programmatically using server code at Page_Load event of web page as follow:

```
   protected void Page_Load(object sender, EventArgs e)
   {
      Button1.Command += new CommandEventHandler(Button1_Command);
   }

   protected void Button1_Command(object sender, CommandEventArgs e)
   {

   }
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

For attaching the event programmatically write the ID of the button control inside the Page_Load event, as "**Button1**" used in above sample code. Next use the dot operator to open the intellisense showing properties and methods available for it and select the **Command event** from the list. Then add **"+="** and press tab to auto generate the handler code for it. You will get the code similar to the sample code shown above.

Example for ASP.Net Button Command

HTML Code:

```
<asp:Button ID="Button1"
 runat="server"
 CommandArgument="1"
 CommandName="Insert"
 Text="Button 1"
 OnCommand="Button1_Command" />
```

C# Code:

```
protected void Button1_Command(object sender, CommandEventArgs e)
{
   Response.Write("Command Name = <b>" + e.CommandName + "</b>");
   Response.Write("<br />Command Argument: <b>" + e.CommandArgument + "</b>");
}
```

ASP.Net Button OnClientClick Property:

The **OnClientClick property** of **ASP.Net Button** provides the functionality to execute the client script before sending the data to server. If you will specify any client script code for this property of Button control it will be executed at the time of Click event whether it is handled by Click event handler or Command event handler at the server. You can create the JavaScript functions as a client side script that can be used as a value for **OnClientClick property** of the Button Control.

In ASP.Net you can specify the value for Button Control's OnClientClick property in two ways. First way is to assign the value at the time of design time by adding the OnClientClick property as an attribute of Button control in the HTML markup code block.

```
<asp:Button ID="Button1"
 runat="server"
 Text="Button"
 OnClientClick="javascript:return ClientClick();" />
```

Using Code Section:

Button1.OnClientClick = "javascript:return ClientClick();";

You can add the above code at the Page_Load event of ASP.Net web page that will render the specified client script value for OnClientClick property along with the associated ASP.Net Button control.

As shown in the code above we have used "**javascript**" and colon "**:**" before specifying the client script function name. These attributes are required only if you are specifying the javascript function that will return the Boolean value as true/false. The javascript functions that return true/false enable/disable the associated Button control to submit the form data to the server.

Example for ASP.Net Button Control OnClientClick Property:

In the example below we have created a javascript function that sets the value of ASP.Net Button control at client side. It will indicate the user that the server code is executing.

HTML Code:

```
<script type="text/javascript">
    function ClientClick() {
       document.getElementById('<%=Button1.ClientID %>').value = 'Processing please wait...';
       return true;

    }
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
<asp:Button ID="Button1"
    runat="server"
    Text="Button"
    OnClientClick="javascript:return ClientClick();" />
```

C# Code:

```
protected void Button1_Click(object sender, CommandEventArgs e)
  {
    // Server code will go here.
    Response.Write("Welcome to ASP.NET Web Application!!!");

  }
```

Above example shows that you can execute both client-side script and server-side event handler code on a single button click using OnClientClick property and Click method of ASP.Net Button Control.

ASP.Net Button Confirm Message Box:

The Confirm message box can be attached to an ASP.Net Button control using OnClientClick property that allows you to specify the value as a client script. The "confirm" is a javascript method, it pop-ups a confirmation message box when user hits the push button control placed on the ASP.Net web. The ASP.Net Button Control's OnClientClick property, proves to be a beneficial property while working with CRUD (create, read, update and delete) methods of database driven web pages. It is beneficial because it allows you to specify the client side method that is javascript Confirm message box that confirms the action taken by the user. When user clicks on the Button control it opens up a confirm box showing some information along with two command buttons "Ok" and "Cancel" on it. If user hits the "Cancel" button of the confirm box then it returns "false" and stops the web form to submit the form data to the server. In actual it cancels the action taken by the user. In the same way if user clicks on the "Ok" button of confirmation box then it returns "true" and allows the web form to post the input web form data to the server side and perform the action specified there.

Example for ASP.Net Button Confirm:

JavaScript Code:

```
<script type="text/javascript">
    function confirmAction() {
        if (confirm('Are you sure, you want to execute this action???')) {
            // you clicked the OK button.
            // you can allow the form to post the data.
            return true;
        }
        else {
            document.getElementById('<%=lblMessage.ClientID %>').innerHTML = "You clicked the Cancel button.";

            // you clicked the Cancel button.
            // you can disallow the form submission.
            return false;
        }
    }
</script>
```

HTML Code:

```
<asp:Label ID="lblMessage" runat="server" />
<br />
<asp:Button ID="btnConfirm"
    Text="Confirm Box"
    runat="server"
    OnClientClick="javascript:return confirmAction();"
    OnClick="btnConfirm_Click" />
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

C# Code:

```
protected void btnConfirm_Click(object sender, EventArgs e)
   {
      lblMessage.Text = "You clicked the Ok button.";

   }
```

The above sample code will display the message for Cancel button without sending the web page to the PostBack. When you will click the OK button of the confirmation box it will return true and allow the page to execute the server end event handler code.

ASP.Net Button Alert Message Box:

The **Alert message box** can also be attached to an **ASP.Net Button control** using **OnClientClick property** that enables you to specify the client side script that is to be executed before submitting the form data to the server. The **"alert"** is a JavaScript method, and it pop-ups a message box when user hits the push button control placed on the web page. The **OnClientClick** property of ASP.Net Button control is also beneficial when you are using input fields on your web page that will send the information into the database tables. You can validate the fields and display the warning alert messages using **JavaScript alert box** with the help of **OnClientClick property**. You can also display the necessary information using JavaScript alert box, for example if you want to redirect the web page after processing the data then you can intimate the user by displaying the alert box as soon as user hits the button. The JavaScript alert box displays only "Ok" button with text information on it and always return true when user clicks the **"Ok"** button.

Example for ASP.Net Button Alert Box:

JavaScript Code:

```
<script type="text/javascript">
      function alertBox() {
         if (document.getElementById('<%=TextBox1.ClientID %>').value == "") {
      alert('TextBox is empty. \nPlease enter the value in TextBox.');

      // TextBox is empty.
      // Disallow the form submission.
      return false;
   }
   else {
      alert('You entered: \n\n' + document.getElementById('<%=TextBox1.ClientID %>').value);

      // TextBox has value.
      // You can allow the form to submit the data.
      return true;
   }
}
   </script>
```

HTML Code:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      <asp:Button ID="Button1"
         Text="Alert Box"
         runat="server"
         OnClientClick="javascript:return alertBox()"
         OnClick="Button1_Click" />
```

C# Code:

```
protected void Button1_Click(object sender, EventArgs e)
   {
      Response.Write("You enetered: " + TextBox1.Text);

   }
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

The above sample code will display an alert message box if user will click the button control leaving the TextBox control empty and will not allow the web form to submit the form data to the server. If user will click the ASP.Net Button control after entering a value in TextBox control then it will display alert box showing the input value and will allow the form submission that will execute the code provided for click event at server side.

Button PostBackUrl Property:

The **ASP.Net Button PostBackUrl property** provides the functionality to post the web form data to the specified web page URL. By default **PostBackUrl property** has no value that sets its string value as an empty string ("") that causes the web page to post back the form data to itself. You can perform the cross-page post using the **PostBackUrl** property of the Button control.

Set the URL of the other page as a value of **PostBackUrl** of the Button control to post the form data to the specified web page. In ASP.Net you can use **PreviousPage property** of Page class which represents the web page that transferred the control to the current page. The **PreviousPage** property provides the request information to the Post Back targeted web page. For example if you have 2 web pages say default1.aspx and default2.aspx then you can specify the **PostBackUrl** property value as "default2.aspx" for the submit Button control placed inside the default1.aspx. When user will hit the Button control of default1.aspx, it will post the information to the default2.aspx that can be accessed using **PreviousPage** property there.

Example for ASP.Net Button PostBackUrl

HTML Code:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
<asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
<br />
<asp:Button ID="Button1"
   runat="server"
   Text="PostBackURL"
   UseSubmitBehaviour="true"
   PostBackUrl="Default2.aspx" />
```

C# Code:

Place the following C# code inside the Page_Load event handler of Default2.aspx web page:

```
protected void Page_Load(object sender, EventArgs e)
  {
     Response.Write(((TextBox)PreviousPage.FindControl("TextBox1")).Text);

     Response.Write("<br />");

     Response.Write(((TextBox)PreviousPage.FindControl("TextBox2")).Text);

  }
```

The above code will display the values of TextBox1 and TextBox2 controls of Default1.aspx web page.

Multiple Button Controls with Single Event Handler:

The **Command method of ASP.Net Button control** enables you to handle the click event of multiple button controls with a single command event handler. The clicks on ASP.Net Button control can be handled by command event handler using the command method that will execute the specified action based on the values of **CommandName** and **CommandArgument** properties. In the command event handler code you can create the conditional sets of server side code that will perform the action according the specified CommandName for the Button control that is being clicked by the user. The **CommandArgument** property can also be used to specify any additional parameter value that can be utilized to perform the desired action. The Command Event handler of ASP.Net Button control

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

also allows you to handle the multiple Button controls on a web page whether they are openly placed on the page or nested inside the data bound controls such as DataList, GridView, and Repeater etc.

HTML Code

```
<asp:Label ID="Label1" runat="server" /><br />
<asp:Label ID="Label2" runat="server" /><br />
<br />
<asp:Button ID="Button1"
   Text="Red"
   runat="server"
   OnCommand="btnCommand"
   CommandArgument="#CC0000"
   CommandName="Color" />
<asp:Button ID="Button2"
   Text="Blue"
   runat="server"
   OnCommand="btnCommand"
   CommandArgument="#0000CC"
   CommandName="Color" />
<asp:Button ID="Button3"
   Text="Arial"
   runat="server"
   OnCommand="btnCommand"
   CommandArgument="Arial"
   CommandName="Font" />
<asp:Button ID="Button4"
   Text="MS Serif"
   runat="server"
   OnCommand="btnCommand"
   CommandArgument="MS Serif"
   CommandName="Font" />
```

C# Code

```
protected void btnCommand(object sender, CommandEventArgs e)
  {
    Label1.Text = "Command Name = <b>" + e.CommandName + "</b>";
    Label2.Text = "Command Argument = <b>" + e.CommandArgument + "</b>";

    if (e.CommandName == "Color")
    {
       Label1.ForeColor = System.Drawing.Color.FromName(e.CommandArgument.ToString());
       Label2.ForeColor = System.Drawing.Color.FromName(e.CommandArgument.ToString());
    }
    else if (e.CommandName == "Font")
    {
       Label1.Font.Name = e.CommandArgument.ToString();
       Label2.Font.Name = e.CommandArgument.ToString();
    }

  }
```

In the above example two command names have been used for multiple ASP.Net Button control as **"Color"** and **"Font"**. Only one Command event handler has been used to handle the click event of multiple Button controls that contains conditional code blocks for the both types of Command names used in the Code. The "Color" command name will set the specified font color of the output text in the Label controls whereas the "Font" command name will set the specified font family for the output text.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Button UseSubmitBehaviour Property:

The **ASP.Net Button UseSubmitBehaviour Property** enables you to specify the submit behaviour of the associated Button control. It specifies that whether the Button control will use the client browser's submit mechanism to submit the web form data to the server or ASP.Net PostBack mechanism. If you will not declare the **UseSubmitBehaviour property** then it will use its default value **"true"** that will use the client browser's submit mechanism to post the web form data to the server.
When **UseSubmitBehaviour** property of ASP.Net button control is specified with **"false"** then it adds client-side script to the page that enables the button control to post the form to the server. You can see the difference between rendered controls by viewing the source code of the ASP.Net web page.
The ASP.Net Button control having UseSubmitBehaviour property with "false" will render through the following generated HTML markup code:

```
<input type="button"
        name="Button1"
        value="Submit Behaviour"
        onclick="javascript: __doPostBack('Button1', '')"
        id="Button1" />
```

The ASP.Net Framework will add the following client-side JavaScript code that will enable the above type of Button control to post the form to the server:

```
<script type="text/javascript">
    //<![CDATA[
    var theForm = document.forms['form1'];
    if (!theForm) {
        theForm = document.form1;
    }
    function __doPostBack(eventTarget, eventArgument) {
        if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
            theForm.__EVENTTARGET.value = eventTarget;
            theForm.__EVENTARGUMENT.value = eventArgument;
            theForm.submit();
        }
    }
    //]]>
</script>
```

Example Code for ASP.Net Button UseSubmitBehaviour Property:

HTML Code:

```
<asp:Button ID="Button1"
        Text="Submit Behaviour"
        OnClick="SubmitBtn_Click"
        UseSubmitBehaviour="false"
        runat="server" />
```

C# Code:

```
protected void SubmitBtn_Click(object sender, EventArgs e)
  {
    Response.Write("Hello RakeshDotNet!!!");

  }
```

The above HTML markup code block will generate the following HTML code while rendering the Button control:

```
<input type="button"
        name="Button1"
        value="Submit Behaviour"
        onclick="javascript: __doPostBack('Button1', '')"
        id="Button1" />
```

The onclick event of the Button control will execute the **__doPostBack** client-side function of JavaScript generated by the ASP.Net framework that will perform the page PostBack.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

LinkButton Control:

This controls works like hyperlink style button. Like the Button control, LinkButton is used to post a Web Forms page back to the server.
OR
It renders the HTML anchor tag with a combination of complex functionality of ASP.Net Button server control. The LinkButton control appears as hyperlinked text but also raises the server side event that sends the page to the PostBack state to execute the specified action provided in the associated event handler.

The **ASP.Net LinkButton control** provides the functionality to display a hyper link type button control on the ASP.Net web page that enables you to post the data back to server side and perform some specific action over there. The LinkButton control also belongs to the family of ASP.Net standard web server controls that allows you to send the input data of web form to the server. The LinkButton control renders like a HyperLink control but performs its actions like the Button control.

By default LinkButton works as a submit button. You can attach its click event to control its server side actions when it is being clicked. The LinkButton can also be used as a command button by specifying its **CommandName** that enables you to create a single event handler for multiple command LinkButton and execute the server code based on the command name passed by the LinkButton being clicked by the user for example Sort, Order, Edit, Delete etc. But the by default the submit behaviour of LinkButton does not have a command name associated to it that directly performs the specified server side action.

The LinkButton control renders using **HTML anchor <a> tag** in the web browser that uses JavaScript based ASP.Net postback method to perform the server post backs.

Properties of ASP.Net LinkButton Control

Following are the properties of LinkButton control that allows you to control and customize its behaviour:

1. CausesValidation: It accepts the Boolean values as true/false that performs the web page validation check when user clicks the linkbutton control.

2. CommandArgument: You can specify the additional information as a value to CommandArgument property of LinkButton control that can be accessed at server side when it raises the command based click event.

3. CommandName: You can specify the command name for the LinkButton control that enables you to perform the server side action based on its value such as sort, edit, update etc.

4. OnClientClick: This property allows you to call the client side JavaScript function that will execute before sending the data to the server.

5. PostBackUrl: You can specify the URL of the other web page to which you want to post the input data of the current web form when user clicks the LinkButton to submit the form.

6. Text: It enables you to display the caption text value of LinkButton control.

7. ValidationGroup: It enables you to specify the name of the validation group that allows the LinkButton to perform the validation check (CausesValidation) only for the controls that belong to the specified validation group.

Event Handlers for ASP.Net LinkButton Control

You can attach the following events with LinkButton control that are raised at the time of click:

1. OnClick: It raises the server side click event of the LinkButton control.

2. OnCommand: It raises the command based event of the LinkButton control.

LinkButton Control Examples:

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

- **LinkButton Control Click Event**
- **LinkButton Control Command Event**
- **LinkButton OnClientClick Property**
- **LinkButton JavaScript Confirm Message Box**
- **LinkButton JavaScript Alert Box**

Sample Code for LinkButton Control

HTML Code:

```
<asp:LinkButton ID="LinkButton1"
        runat="server"
        OnClick="LinkButton1_Click"
        Text="LinkButton1">
    </asp:LinkButton>
```

C# Code:

```
protected void LinkButton1_Click(object sender, EventArgs e)
  {
    LinkButton1.Text = "Hey!!! You clicked me.";
  }
```

ASP.Net LinkButton Click Event:

The **ASP.Net LinkButton click event** is raised when it is being clicked by the user. This event of LinkButton control submits the web form data to the server and performs the specified action there. The **click event** is commonly used when the**ASP.Net LinkButton** has no command name associated with it and is placed as a submit button on the web page to send the user input to the server using ASP.Net PostBack method. In ASP.Net each web server control has its own set of events that can be raised and handled at server side programmatically. The LinkButton server control has **OnClick** method that can be used to handle the click event. It allows you to handle the event without attaching a delegate.

Attaching Click Event with LinkButton Control:

There are 2 ways for attaching the OnClick event handler with ASP.Net LinkButton control:

1. Double clicking the linkbutton control in design view of web page while working in Visual Studio. By default it will generate the code for OnClick event of that Button as follows:

```
<asp:LinkButton ID="LinkButton1" runat="server" Text="Button" OnClick="LinkButton1_Click" />

protected void LinkButton1_Click(object sender, EventArgs e)
{
        //Write the appropriate code
}
```

2. You can also attach the event programmatically using server code at Page_Load event of web page as follows:

```
protected void Page_Load(object sender, EventArgs e)
{
    LinkButton1.Click += new EventHandler(LinkButton1_Click);
}

protected void LinkButton1_Click(object sender, EventArgs e)
{
        //Write the appropriate code
}
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

For attaching the event programmatically write the ID of the button control inside the Page_Load event, as "**LinkButton1**" used in above sample code. Next use the dot operator to open the intellisense showing properties and methods available for it and select the **Click event** from the list. Then add **"+="** and press tab to auto generate the handler code for it. You will get the code similar to the sample code shown above.

ASP.Net LinkButton Command Event:

The **ASP.Net LinkButton Command event** is used when there is a command name associated with it. This event of LinkButton control also fires when it is being clicked by the user. You can use the **Command event** of LinkButton control for multiple LinkButton placed on the ASP.Net web page. Each LinkButton control will have its own Command Name associated with it that will direct the command event handler to execute the server code based on the name of command for example edit, delete, sort etc. The ASP.Net LinkButton control has **OnCommand method** that can be used to trap the click and perform the action at server side according to the value of **CommandName property** associated with the clicked LinkButton. You can handle this event without attaching any delegate.

The Command event handler of LinkButton control uses **CommandEventArgs** class object as parameter that provides the access to the two properties of LinkButton Control:

1. CommandName

2. CommandArgument

You can specify the values in the HTML markup code block of ASP.Net LinkButton control. Both CommandName and CommandArgument property accept string type value that can also be specified programmatically.

Attaching Command Event with LinkButton Control

There are 2 ways for attaching the **OnCommand event** handler with ASP.Net LinkButton control:

1. Click to select the Button control in the Design view of web page in Visual Web Developer. Then press F4 to open its property fly-out window. Change the property window's view to events as shows in the figure below:

Double click on the Command method to generate the command event handler code for the selected LinkButton control.

```
<asp:LinkButton ID="LinkButton1" runat="server" Text="LinkButton 1"      OnCommand="LinkButton1_Command" />
```

It will generate the following code for the LinkButton Control:

```
protected void LinkButton1_Command(object sender, CommandEventArgs e)
   {

   }
```

2. You can also attach the event programmatically using server code at Page_Load event of web page as follow:

```
protected void Page_Load(object sender, EventArgs e)
{
   LinkButton1.Command += new CommandEventHandler(LinkButton1_Command);
}
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
protected void LinkButton1_Command(object sender, CommandEventArgs e)
{

}
```

For attaching the event programmatically write the ID of the LinkButton control inside the Page_Load event, as "**LinkButton1**" used in above sample code. Next use the dot operator to open the intellisense showing properties and methods available for it and select the **Command event** from the list. Then add **"+="** and press tab to auto generate the handler code for it. You will get the code similar to the sample code shown above.

Example for ASP.Net LinkButton Command

HTML Code:

```
<asp:LinkButton ID="LinkButton1"
 runat="server"
 CommandArgument="1"
 CommandName="Insert"
 Text="LinkButton 1"
 OnCommand="LinkButton1_Command" />
```

C# Code:

```
protected void LinkButton1_Command(object sender, CommandEventArgs e)
{
   Response.Write("Command Name = <b>" + e.CommandName + "</b>");
   Response.Write("<br />Command Argument: <b>" + e.CommandArgument + "</b>");
}
```

ASP.Net LinkButton OnClientClick Property:

The **OnClientClick property** of **ASP.Net LinkButton** provides the functionality to execute the client script before sending the data to server. If you will specify any client script code for this property of LinkButton control it will be executed at the time of Click event whether it is handled by Click event handler or Command event handler at the server. You can create the JavaScript functions as a client side script that can be used as a value for **OnClientClick property** of the LinkButton Control.

In ASP.Net you can specify the value for LinkButton Control's OnClientClick property in two ways. First way is to assign the value at the time of design time by adding the OnClientClick property as an attribute of LinkButton control in the HTML markup code block.

```
<asp:LinkButton ID="LinkButton1"
 runat="server"
 Text="LinkButton"
 OnClientClick="javascript:return ClientClick();" />
```

Using Code Section:

LinkButton1.OnClientClick = "javascript:return ClientClick();";

You can add the above code at the Page_Load event of ASP.Net web page that will render the specified client script value for OnClientClick property along with the associated ASP.Net LinkButton control.

As shown in the code above we have used "**javascript**" and colon "**:**" before specifying the client script function name. These attributes are required only if you are specifying the javascript function that will return the Boolean value as true/false. The javascript functions that return true/false enable/disable the associated LinkButton control to submit the form data to the server.

Example for ASP.Net LinkButton Control OnClientClick Property:

In the example below we have created a javascript function that sets the value of ASP.Net LinkButton control at client side. It will indicate the user that the server code is executing.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

HTML Code:

```
<script type="text/javascript">
      function ClientClick() {
         document.getElementById('<%=LinkButton1.ClientID %>').value = 'Processing please wait...';
         return true;

       }
      <asp:LinkButton ID="LinkButton1"
          runat="server"
          Text="LinkButton"
          OnClientClick="javascript:return ClientClick();" />
```

C# Code:

```
protected void LinkButton1_Click(object sender, CommandEventArgs e)
   {
      // Server code will go here.
      Response.Write("Welcome to ASP.NET Web Application!!!");

   }
```

Above example shows that you can execute both client-side script and server-side event handler code on a single LinkButton click using OnClientClick property and Click method of ASP.Net LinkButton Control.

ASP.Net LinkButton Confirm Message Box:

The Confirm message box can be attached to an ASP.Net LinkButton control using OnClientClick property that allows you to specify the value as a client script. The "confirm" is a javascript method, it pop-ups a confirmation message box when user hits the LinkButton control placed on the ASP.Net web. The ASP.Net LinkButton Control's OnClientClick property, proves to be a beneficial property while working with CRUD (create, read, update and delete) methods of database driven web pages. It is beneficial because it allows you to specify the client side method that is javascript Confirm message box that confirms the action taken by the user. When user clicks on the LinkButton control it opens up a confirm box showing some information along with two command buttons "Ok" and "Cancel" on it. If user hits the "Cancel" button of the confirm box then it returns "false" and stops the web form to submit the form data to the server. In actual it cancels the action taken by the user. In the same way if user clicks on the "Ok" button of confirmation box then it returns "true" and allows the web form to post the input web form data to the server side and perform the action specified there.

Example for ASP.Net LinkButton Confirm:

JavaScript Code:

```
<script type="text/javascript">
      function confirmAction() {
         if (confirm('Are you sure, you want to execute this action???')) {
            // you clicked the OK button.
            // you can allow the form to post the data.
            return true;
         }
         else {
            document.getElementById('<%=lblMessage.ClientID %>').innerHTML = "You clicked the Cancel button.";

            // you clicked the Cancel button.
            // you can disallow the form submission.
            return false;
         }
      }
</script>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

HTML Code:

```
<asp:Label ID="lblMessage" runat="server" />
<br />
<asp:LinkButton ID="btnConfirm"
   Text="Confirm Box"
   runat="server"
   OnClientClick="javascript:return confirmAction();"
   OnClick="btnConfirm_Click" />
```

C# Code:

```
protected void btnConfirm_Click(object sender, EventArgs e)
  {
     lblMessage.Text = "You clicked the Ok button.";

  }
```

The above sample code will display the message for Cancel button without sending the web page to the PostBack. When you will click the OK button of the confirmation box it will return true and allow the page to execute the server end event handler code.

ASP.Net LinkButton Alert Message Box:

The **Alert message box** can also be attached to an **ASP.Net LinkButton control** using **OnClientClick property** that enables you to specify the client side script that is to be executed before submitting the form data to the server. The **"alert"** is a JavaScript method, and it pop-ups a message box when user hits the LinkButton control placed on the web page. The **OnClientClick** property of ASP.Net LinkButton control is also beneficial when you are using input fields on your web page that will send the information into the database tables. You can validate the fields and display the warning alert messages using **JavaScript alert box** with the help of **OnClientClick property**. You can also display the necessary information using JavaScript alert box, for example if you want to redirect the web page after processing the data then you can intimate the user by displaying the alert box as soon as user hits the LinkButton. The JavaScript alert box displays only "Ok" button with text information on it and always return true when user clicks the **"Ok"** button.

Example for ASP.Net LinkButton Alert Box:

JavaScript Code:

```
<script type="text/javascript">
      function alertBox() {
         if (document.getElementById('<%=TextBox1.ClientID %>').value == "") {
      alert('TextBox is empty. \nPlease enter the value in TextBox.');

      // TextBox is empty.
      // Disallow the form submission.
      return false;
   }
   else {
      alert('You entered: \n\n' + document.getElementById('<%=TextBox1.ClientID %>').value);

      // TextBox has value.
      // You can allow the form to submit the data.
      return true;
   }
}
   </script>
```

HTML Code:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      <asp:LinkButton ID="LinkButton1"
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
        Text="Alert Box"
        runat="server"
        OnClientClick="javascript:return alertBox()"
        OnClick="LinkButton1_Click" />
```

C# Code:

```
protected void LinkButton1_Click(object sender, EventArgs e)
  {
    Response.Write("You enetered: " + TextBox1.Text);
  }
```

The above sample code will display an alert message box if user will click the LinkButton control leaving the TextBox control empty and will not allow the web form to submit the form data to the server. If user will click the ASP.Net LinkButton control after entering a value in TextBox control then it will display alert box showing the input value and will allow the form submission that will execute the code provided for click event at server side.

LinkButton PostBackUrl Property:

The **ASP.Net LinkButton PostBackUrl property** provides the functionality to post the web form data to the specified web page URL. By default **PostBackUrl property** has no value that sets its string value as an empty string ("") that causes the web page to post back the form data to itself. You can perform the cross-page post using the **PostBackUrl** property of the LinkButton control.

Set the URL of the other page as a value of **PostBackUrl** of the LinkButton control to post the form data to the specified web page. In ASP.Net you can use **PreviousPage property** of Page class which represents the web page that transferred the control to the current page. The **PreviousPage** property provides the request information to the Post Back targeted web page. For example if you have 2 web pages say default1.aspx and default2.aspx then you can specify the **PostBackUrl** property value as "default2.aspx" for the LinkButton control placed inside the default1.aspx. When user will hit the LinkButton control of default1.aspx, it will post the information to the default2.aspx that can be accessed using **PreviousPage** property there.

Example for ASP.Net LinkButton PostBackUrl

HTML Code:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
<asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
<br />
<asp:LinkButton ID="LinkButton1"
   runat="server"
   Text="PostBackURL"
   UseSubmitBehaviour="true"
   PostBackUrl="Default2.aspx" />
```

C# Code:

Place the following C# code inside the Page_Load event handler of Default2.aspx web page:

```
protected void Page_Load(object sender, EventArgs e)
  {
    Response.Write(((TextBox)PreviousPage.FindControl("TextBox1")).Text);

    Response.Write("<br />");

    Response.Write(((TextBox)PreviousPage.FindControl("TextBox2")).Text);
  }
```

The above code will display the values of TextBox1 and TextBox2 controls of Default1.aspx web page.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Multiple LinkButton Controls with Single Event Handler:

The **Command method of ASP.Net LinkButton control** enables you to handle the click event of multiple LinkButton controls with a single command event handler. The clicks on ASP.Net LinkButton control can be handled by command event handler using the command method that will execute the specified action based on the values of **CommandName** and **CommandArgument** properties. In the command event handler code you can create the conditional sets of server side code that will perform the action according the specified CommandName for the LinkButton control that is being clicked by the user. The **CommandArgument** property can also be used to specify any additional parameter value that can be utilized to perform the desired action. The Command Event handler of ASP.Net LinkButton control also allows you to handle the multiple LinkButton controls on a web page whether they are openly placed on the page or nested inside the data bound controls such as DataList, GridView, and Repeater etc.

HTML Code

```
<asp:Label ID="Label1" runat="server" /><br />
<asp:Label ID="Label2" runat="server" /><br />
<br />
<asp:LinkButton ID="LinkButton1"
   Text="Red"
   runat="server"
   OnCommand="lnkBtn_Command"
   CommandArgument="#CC0000"
   CommandName="Color" />
<asp:LinkButton ID="LinkButton2"
   Text="Blue"
   runat="server"
   OnCommand=" lnkBtn_Command"
   CommandArgument="#0000CC"
   CommandName="Color" />
<asp:LinkButton ID="LinkButton3"
   Text="Arial"
   runat="server"
   OnCommand=" lnkBtn_Command"
   CommandArgument="Arial"
   CommandName="Font" />
<asp:LinkButton ID="LinkButton4"
   Text="MS Serif"
   runat="server"
   OnCommand=" lnkBtn_Command"
   CommandArgument="MS Serif"
   CommandName="Font" />
```

C# Code

```
protected void btnBtn_Command(object sender, CommandEventArgs e)
  {
    Label1.Text = "Command Name = <b>" + e.CommandName + "</b>";
    Label2.Text = "Command Argument = <b>" + e.CommandArgument + "</b>";

    if (e.CommandName == "Color")
    {
       Label1.ForeColor = System.Drawing.Color.FromName(e.CommandArgument.ToString());
       Label2.ForeColor = System.Drawing.Color.FromName(e.CommandArgument.ToString());
    }
    else if (e.CommandName == "Font")
    {
       Label1.Font.Name = e.CommandArgument.ToString();
       Label2.Font.Name = e.CommandArgument.ToString();
    }
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

}

In the above example two command names have been used for multiple ASP.Net LinkButton control as **"Color"** and **"Font"**. Only one Command event handler has been used to handle the click event of multiple LinkButton controls that contains conditional code blocks for the both types of Command names used in the Code. The "Color" command name will set the specified font color of the output text in the Label controls whereas the "Font" command name will set the specified font family for the output text.

ImageButton Control:

This control look like button control but it has image. We simply click on image and works like button control. Like the Button control, ImageButton is used to post back to the server. We can set its image using ImageUrl property.

OR

The ImageButton control displays the specified image using ImageUrl property instead of text value using Text property and provides the functionality same as the ASP.Net Button server control.

The **ASP.Net ImageButton Control** is a web server control that enables you to display a clickable image that performs server **PostBack** when user clicks on it. The ImageButton consists of the combination of properties that are available in **ASP.Net Image control** and **Button Control**. The properties that provide the functionality to display and set the image on web page have been inherited from the Image control and the properties that provide the functionality to attach the server side events and post backs have been inherited from the ASP.Net Button control. You can also attach the client side events to the ImageButton control execute the client side scripts before sending the page to the server.

Properties of ASP.Net ImageButton Control

As we discussed above that the ImageButton control is a combination of image and button control it contains two types of properties. First types of properties allow you to manage the display image and its attributes. Following are the image related properties:

1. AlternateText: It allows you to specify the alternate text for the display image that appears in case the image is unavailable.

2. DescriptionUrl: It accepts the URL of the web page containing the full length description of the specified image that you want to display on the web page.

3. ImageAlign: It enables you to set the alignment of the specified image with respect to the other elements available inline to the ImageButton control. It supports predefined values such as Right, Left, Bottom, Top, AbsMiddle, AbsBottom, Middle, Baseline and TextTop to align the image in relation to other elements.

4. ImageUrl: It accepts the URL of the image that you want to display as button on the web page.

The rest of the properties enable you to attach the server side events to the ImageButton control that can be handled at server side using VB or C# code. Following are the properties that provide the Button type behaviour in the ASP.Net web page:

1. CommandArgument: It gets or sets the additional values that can be accessed at server side along with CommandName when the Command event occurs.

2. CommandName: It gets or sets the command name associated the ImageButton Control that provides the functionality to perform the server action based on it.

3. PostBackUrl: It gets or sets the URL of the other web page to which you want to post the form data of the current web page.

The Button type behaviour of ImageButton control supports the following events that can be attached to it:

Server Side Events

1. OnClick Event: It occurs when user clicks on the ImageButton Control. You can attach the server side click handler to handle the click event of ImageButton.

2. OnCommand Event: It raises the command event of ImageButton that can be handled at server side directly by using its command event handler.

Client Side Event

1. OnClientClick Event: It enables you to get or set client-side script that can be executed before sending the page to the server side when user clicks the ImageButton.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ImageButton Control Examples:

- **ImageButton Control Sample**
- **ImageButton OnClick Event**
- **ImageButton OnCommand Event**
- **ImageButton OnClientClick Property**
- **ImageButton PostBackUrl Property**
- **Determine the Image Coordinates Using ImageButton Control**

ImageButton Control Sample:

```
<asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="Images/imgButton1.jpg" />
```

ASP.Net ImageButton OnClick Event:

The **ASP.Net ImageButton click event** is raised when it is being clicked by the user. This event of ImageButton control submits the web form data to the server and performs the specified action there. The **click event** is commonly used when the**ASP.Net ImageButton** has no command name associated with it and is placed on the web page to send the user input to the server using HTML Post method. In ASP.Net each web server control has its own set of events that can be raised and handled at server side. The ImageButton server control has **OnClick method** that can be used to trap the click event. It allows you to handle the event without attaching a delegate.

Attaching Click Event with ImageButton Control

There are 2 ways for attaching the OnClick event handler with ASP.Net ImageButton control:
1. Double clicking the button control in design view of web page while working in Visual Web Developer. By default it will generate the code for OnClick event of that ImageButton as follows:

HTML Code:

```
<asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="Images/imgButton1.jpg"
        OnClick="ImageButton1_Click" />
```

C# Code:

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
  {

  }
```

2. You can also attach the event programmatically using server code at Page_Load event of web page as follows:

```
  protected void Page_Load(object sender, EventArgs e)
  {
     ImageButton1.Click += new ImageClickEventHandler(ImageButton1_Click);
  }

  protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
  {

  }
```

For attaching the event programmatically write the ID of the ImageButton control inside the Page_Load event, as "ImageButton1" used in above sample code. Next use the dot operator to open the intellisense showing properties and methods available for it and select the Click event from the list. Then add "+=" and press tab to auto generate the handler code for it. You will get the code similar to the sample code show above.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Microsoft .NET Solutions
Robust, Scalable & Secure Applications
Developed for Diverse Business Needs
Develop your Microsoft skills and knowledge through great training
NARESH i technologies
Microsoft .net

Example for ASP.Net ImageButton Click

HTML Code:

```
<asp:Label ID="Label1" runat="server"></asp:Label>

<asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="images/imgButton1.jpg"
        OnClick="ImageButton1_Click" />
```

C# Code:

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
  {
     Label1.Text = "You clicked the image button";
  }
```

ASP.Net ImageButton OnCommand Event

The **Command event of ASP.Net ImageButton** is used when there is a command name associated with it. This event of ImageButton control also occurs when it is being clicked by the user. You can use the **Command event** of ImageButton control for multiple ImageButton placed on the ASP.Net web page. Each ImageButton will have its own Command Name associated with it that will direct the command event handler to execute the server code based on the name of command, for example edit, delete, sort etc. The ASP.Net ImageButton control has **OnCommand method** that can be used to trap the click and perform the action at server side according to the value of CommandName property associated with the clicked ImageButton. You can handle this event without attaching any delegate to it.

The Command event handler of ImageButton control uses **CommandEventArgs class** object as parameter that provides the access to the two properties of ImageButton Control:

1. CommandName

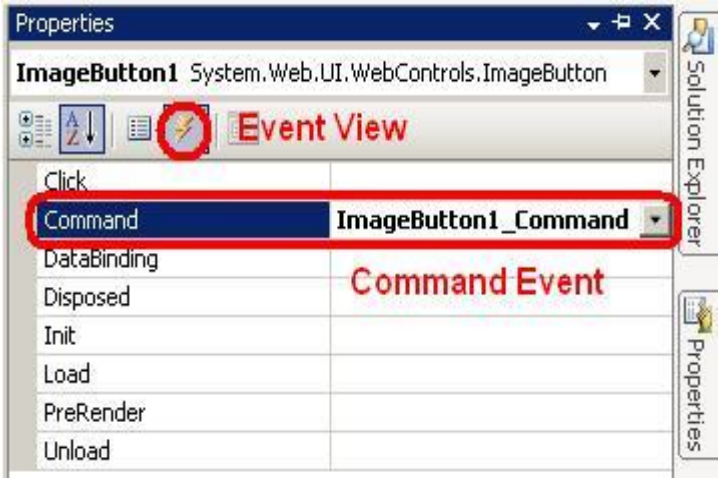
2. CommandArgument

You can specify the values in the HTML markup code block of ASP.Net ImageButton control. Both CommandName and CommandArgument property accept string type value that can also be specified programmatically.

Attaching Command Event with ImageButton Control

There are 2 ways for attaching the OnCommand event handler with ASP.Net ImageButton control:

1. Click to select the ImageButton control in the Design view of web page in Visual Studio/Visual Web Developer. Then press F4 to open its property fly-out window. Change the property window's view to events as shows in the figure below:



Double click on the Command method to generate the command event handler code for the selected ImageButton control.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
<asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="images/imgButton1.jpg"
        OnCommand="ImageButton1_Command" />
```

It will generate the following code for the ImageButton Control:

```
protected void ImageButton1_Command(object sender, CommandEventArgs e)
  {

  }
```

2. You can also attach the event programmatically using server code at Page_Load event of web page as follow:

```
protected void Page_Load(object sender, EventArgs e)
  {
    ImageButton1.Command += new CommandEventHandler(ImageButton1_Command);
  }

  void ImageButton1_Command(object sender, CommandEventArgs e)
  {

  }
```

For attaching the event programmatically write the ID of the button control inside the Page_Load event, as **"ImageButton1"** used in above sample code. Next use the dot operator to open the intellisense showing properties and methods available for it and select the **Command event** from the list. Then add **"+="** and press tab to auto generate the handler code for it. You will get the code similar to the sample code show above.

Example for ASP.Net ImageButton Command

HTML Code:

```
<asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="images/imgButton1.jpg"
        CommandArgument="1"
        CommandName="insert"
        OnCommand="ImageButton1_Command" />
```

C# Code:

```
protected void ImageButton1_Command(object sender, CommandEventArgs e)
  {
    Label1.Text = "Command Name = <b>" + e.CommandName + "</b>";
    Label1.Text += "<br />Command Argument: <b>" + e.CommandArgument + "</b>";
  }
```

ASP.Net ImageButton OnClientClick Property:

The **OnClientClick property of ASP.Net ImageButton** provides the functionality to execute the client side script before sending the form data to the server. If you will specify any client side code for this property of ImageButton control it will be executed at the time of Click event whether it is being handled by the Click event handler or Command event handler at the server. You can create the JavaScript functions as a client side script that can be used as a value for **OnClientClick property** of the ImageButton Control.

In ASP.Net you can specify the value for OnClientClick property of ImageButton in two ways.

First way is to assign the value at the time of designing by adding the OnClientClick property as an attribute of ImageButton control in the HTML markup code block.

```
 <asp:ImageButton ID="ImageButton1" runat="server"
        ImageUrl="images/imgButton1.jpg"
        OnClientClick="javascript:return ClientClick();" />
 <asp:Label ID="lblMessage" runat="server"></asp:Label>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

The Second way is to add the OnClientClick property value programmatically at the server side.

ImageButton1.OnClientClick = "javascript:return ClientClick();";

You can add the above code at the Page_Load event of ASP.Net web page that will render the specified client script value for OnClientClick property along with the associated ASP.Net ImageButton.

As shown in the code above we have used **"javascript"** and colon **":"** before specifying the Javascript function name. These attributes are required only if you are specifying the javascript function that will return the Boolean value such as true/false. The javascript functions that return true/false value provide the functionality to enable/disable the associated ImageButton to submit the form data to the server.

Example for ASP.Net ImageButton Control OnClientClick Property

In the example below we have created a javascript function that will open up a confirm message box displaying a message along with OK and Cancel button:

JavaScript Code

```
<script type="text/javascript">
      function ClientClick() {
         if (confirm('Are you sure, you want to execute this action???')) {
            // you clicked the OK button.
            // you can allow the form to post the data.
            return true;
         }
         else {
            document.getElementById('<%=lblMessage.ClientID %>').innerHTML = "You clicked the Cancel button.";
            // you clicked the Cancel button.
            // you can disallow the form submission.
            return false;
         }
      }
   </script>
```

C# Code:

```
protected void ImageButton1_Click1(object sender, ImageClickEventArgs e)
   {
      // you can add the server code here
      lblMessage.Text = "Welcome to Server Side!!!";
   }
```

Above example shows that you can execute both client-side script and server-side event handler code on a single click using OnClientClick property and Click method of ASP.Net ImageButton Control. When user will click the imagebutton it will pop-up the**JavaScript confirm message box** displaying a message along with OK and Cancel buttons. When user will hit Cancel button it will return false that will disable the web page to submit the form data to the server. When user will hit the OK button then it will return true allowing the web page to post the form data to the server side thereby it also executes the server side Click event handler code.

ASP.Net ImageButton PostBackUrl Property

The **ASP.Net ImageButton PostBackUrl property** provides the functionality to post the form data to the specified web page URL. By default **PostBackUrl property**has no value that sets its string value as an empty string ("") which causes the web page to postback the form data to current web page. You can perform the cross-page post using the PostBackUrl property of the ImageButton control. Set the URL of the other page as a value of PostBackUrl of the ImageButton to post the form data to the specified web page.

In ASP.Net you can use **PreviousPage property** of Page class which represents the web page that transferred the control to the current page. The PreviousPage property provides the request information to the targeted web page that loads when user clicks on the associated ImageButton. For example if you have 2 web pages say default1.aspx and default2.aspx then you can specify the PostBackUrl property value as "default2.aspx" for the ImageButton placed inside the default1.aspx. When user will hit the ImageButton control of default1.aspx, it will post the information to the default2.aspx that can be accessed using **PreviousPage** property there.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Example for ASP.Net ImageButton PostBackUrl Property

HTML Code:

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

    <asp:TextBox ID="TextBox2"
        runat="server"
        TextMode="Password">
    </asp:TextBox><br />

    <asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="images/imgButton1.jpg"
        PostBackUrl="Default2.aspx" />
```

Place the above HTML code inside the Default1.aspx web page.

C# Code:

Place the following C# code inside the Page_Load event handler of Default2.aspx web page:

```
protected void Page_Load(object sender, EventArgs e)
  {
    Response.Write(((TextBox)PreviousPage.FindControl("TextBox1")).Text);

    Response.Write("<br />");
    Response.Write(((TextBox)PreviousPage.FindControl("TextBox2")).Text);
  }
```

The above code will display the values of TextBox1 and TextBox2 controls of Default1.aspx web page.

Getting Coordinates of Image Using ASP.Net ImageButton Control:

The Server Side **Click event handler** of **ASP.Net ImageButton control** also provides the functionality to find out the coordinates of the image where the user clicked on it. This feature of determining the coordinates of an image enables you to provide the functionality to perform the task based on the coordinate values. You can define the different sets of ranges along with their associated tasks that will be executed based on the values of clicked coordinates of the image by the user. You can also use this feature to find out the coordinates that can used to define the clickable area for ASP.Net ImageMap control that we will learn later on. The coordinate's information is sent as a part of event argument object of Click event of ImageButton control.

Steps to Determine the Coordinates using Click Event

1. First of all drag and drop an ImageButton control on your ASP.Net web page in the design view and set the image that you want to display as button.

2. Double click on ImageButton to generate its click event handler. It will generate the following C# handler in the server code of the web page:

```
protected void ImageButton1_Click1(object sender, ImageClickEventArgs e)
  {

  }
```

You can see that in the above code of Click event there is an ImageClick EventArgs object i.e. "e". It provides the access to the coordinates properties of the ImageButton control. You can access the value of X and Y coordinate of the imagebutton that returns the integer type values when users clicks it.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Sample Code for Getting Image Coordinates

HTML Code:

```
<asp:ImageButton ID="ImageButton1"
        runat="server"
        ImageUrl="images/fishes.jpg"
        OnClick="ImageButton1_Click1" />
```

C# Code:

```
protected void ImageButton1_Click1(object sender, ImageClickEventArgs e)
  {
    // You can add the server code here
    Response.Write(e.X.ToString() + " " + e.Y.ToString());
  }
```

The above code shows how to get the values of X and Y coordinates of the image. They return the values based on the position on image of ImageButton where the user clicked to raise the click event.

Note: In ASP.NET, Button, LinkButton, and ImageButton Controls are same in the form of functionality to perform the postback. They are just differ in the form of appearances. So any one control can be used in place of each other to perform the postback depends on requirement in asp.net web application development.

CheckBox Control:

This control is used to accept the data in true/false or say in yes/no Boolean format. Like if we checked it then it says true or yes. By default its checked property is false but we can set it to true.
OR
The CheckBox server control accepts Boolean (true or false) input. When selected, its Checked property is true. Typically a check box is processed as one of several fields in a web form.
OR
The CheckBox control works on Boolean values i.e. true or false. When user hits the blank CheckBox control then it changes its state to Checked and returns its value as true. When the CheckBox control is clicked again then it unchecks the CheckBox and returns the value as false. It works similar to HTML checkbox control.

CheckBox cannot be bind with database. It can be also used to trigger postback to the server if its AutoPostBack property is true.

The **ASP.Net Checkbox control** provides the functionality to display check box in the Web form that returns the true or false type Boolean value as a result. If the **Checkbox control** is in **checked state** then it returns true and if it is in unchecked state then the checkbox control returns false. You can use the ASP.Net checkbox web server control to get the user input as Yes or No. The Checkbox control enables you to provide the functionality to enable or disable any feature, for choosing any option, and the most commonly used for accepting the agreements while registering on any web site.

Properties of CheckBox Control:

The ASP.Net Checkbox control has following properties that are commonly used to get or set the user input through it:

1. Checked: It gets or sets the checkbox state. If the check box control is in checked state then checked property returns true otherwise it returns false. The Checked property of checkbox control accepts and returns Boolean value.

2. AutoPostBack: If you want to execute the PostBack event of checkbox control when user changes the checkbox state to checked or unchecked. The AutoPostBack property also accepts the Boolean value i.e. true or false to allow or disallow the PostBack event of checkbox control.

3. Text: It allows you to display the label with a text string next to the checkbox control. You specify a string value for the text property of checkbox control.

4. TextAlign: The alignment of the text label with respect to each item. By default it is set to Right.

5. AccessKey: Keyboard shortcut used by the control.

6. CausesValidation: It indicates whether the control causes the validation to fire or not. By default it is set to false.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

7. ValidationGroup: Specify the group that should be validated when control causes a postback.

The Checkbox control consists of following PostBack event that can be handled at server side to execute the server code and perform the desired action according to the checked or unchecked state of checkbox:

CheckedChanged: It enables you to handle the PostBack event of checkbox control when user changes its state from checked to unchecked or vice-versa. When user checks or unchecks the checkbox control then it raises the CheckedChanged event that is raised only if the **AutoPostBack property** of checkbox control is set to true.

The checkbox control is an ASP.Net web server control when it is rendered on the web page it splits up into two parts:

1. **HTML input control** having **"type"** attribute value as **"checkbox"**.
2. **HTML label control** to display the string specified for the text property of checkbox server control.

CheckBox Control Examples:

- **CheckBox Control Sample**
- **CheckBox Checked Property**
- **CheckBox AutoPostBack Property**
- **CheckBox OnCheckedChanged Event**

ASP.Net HTML Code for Checkbox control

```
<asp:CheckBox ID="CheckBox1" runat="server" Text="Checkbox control" />
```

The HTML code that is generated by ASP.Net framework when it renders the control on the browser:

```
<input id="CheckBox1" type="checkbox" name="CheckBox1" />
<label for="CheckBox1">Checkbox control</label>
```

ASP.Net CheckBox Checked Property:

The **Checked property** of **ASP.Net Checkbox control** enables you to get or set the checked state of checkbox. You can also set the checked property of checkbox control dynamically using server code or manually when user clicks the checkbox control to change its checked state. The **checked property** of ASP.Net checkbox control accepts the Boolean value to set its state. It also returns the Boolean value to represent the state of the checkbox control placed on the web page. When user clicks the checkbox control to set its state as checked then it returns **"true"** and when user un-checks the checkbox control then checked property returns **"false"** as a resulting value to represent the state of the checkbox control.

Sample Code for ASP.Net Checkbox Checked Property

HTML Code:

```
<asp:CheckBox ID="CheckBox1" runat="server" Text="Checkbox control 1" Checked="True" />
<br />
<br />
<asp:CheckBox ID="CheckBox2" runat="server" Text="Checkbox control 2" />
<br />
<br />
<asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Submit" />
<br />
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
```

In the above HTML source code for Checkbox control we have declared the **Checked property** as **"true"** for the first Checkbox control. It will render the checkbox in checked state initially.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

C# Code:

```
protected void Page_Load(object sender, EventArgs e)
   {
      if (!IsPostBack)
         CheckBox2.Checked = true;
   }

   protected void Button1_Click(object sender, EventArgs e)
   {
      Label1.Text = "Checkbox1.Checked = " + CheckBox1.Checked.ToString() + "<br />";
      Label1.Text += "Checkbox2.Checked = " + CheckBox2.Checked.ToString();
   }
```

In the C# code above we declared the value for **Checked property** of Checkbox2 dynamically that will render the Checkbox2 in checked state initially. When you will click the button control it will display the value Checked property of both the checkbox controls.

ASP.Net CheckBox AutoPostBack Property:

The **AutoPostBack property** of the **ASP.Net CheckBox control** provides the functionality to allow the server side post back event of the web page when user clicks the checkbox control to change its checked state. You can get the value of checked state of the checkbox control at the Page Load event of ASP.Net web page to perform the server side action based on the checked state of checkbox. The **AutoPostBack** property is a Boolean type property that accepts value as true/false to enable or disable the auto post back functionality of the checkbox control to the server side when user clicks it. You can declare its value in two ways such as in HTML code block as an attribute of Checkbox control or by declaring it programmatically when page loads.

Sample Code for ASP.Net CheckBox AutoPostBack Property

HTML Code:

```
<asp:CheckBox ID="CheckBox1"
        runat="server"
        AutoPostBack="True"
        Text="Checkbox control" />
    <br />
    <br />
    <asp:Label ID="Label1"
        runat="server">
    </asp:Label>
```

In the above ASP.Net HTML code for Checkbox control we have declared the value for **AutoPostBack property** of checkbox control as an inline attribute. We have specified the value of AutoPostBack property as **"true"** to allow the web page to post back when user will click the checkbox control to change its checked state.

C# Code:

```
protected void Page_Load(object sender, EventArgs e)
   {
      Label1.Text = "CheckBox1.Checked = " + CheckBox1.Checked.ToString();
   }
```

The C# code shows that you can get the value of **checked property** of checkbox control at the Page Load event of ASP.Net web page when the **AutoPostBack property** of checkbox control is specified with value **"true"**. The above sample code will display the value of checked property of checkbox control when user will click it to change its checked state. The AutoPostBack property will allow the auto post back of web form as soon as the user will hit the checkbox control.

ASP.Net CheckBox OnCheckedChanged Event

The **OnCheckedChanged event** handler of **ASP.Net Checkbox control** enables you to handle the click event of checkbox at server side that is raised at the time when user clicks the checkbox control to change its checked state. The **CheckedChanged event** of Checkbox control raises only if the **AutoPostBack property** of checkbox is specified with value **"true"**. You can handle the click events of multiple checkbox controls separately and perform the server side action for each checkbox using their dedicate OnCheckedChaneged event handlers. The CheckedChanged event occurs immediately when the checked state of the checkbox control changes and it submits the web form to the server only if the AutoPostBack property has value "true".

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Sample Code for ASP.Net CheckBox OnCheckedChanged Event

HTML Code:

```
<asp:CheckBox ID="CheckBox1"
        runat="server"
        AutoPostBack="True"
        Text="Checkbox control"
        OnCheckedChanged="CheckBox1_CheckedChanged" />
      <br />
      <br />
      <asp:Label ID="Label1"
        runat="server">
      </asp:Label>
```

In the above sample code for Checkbox control we have specified the AutoPostBack property value as "true" and also OnCheckedChanged server side event is used to handle the click event of checkbox when user clicks the control to change its checked state.

You can attach the CheckedChanged Event of checkbox control is two ways as follows:

1. Double clicking the checkbox control for which you want to generate the server side OnCheckedChanged event handler. In this document we have also used the same approach to handle the CheckedChanged event of checkbox control.

2. You can also attach the CheckedChanged event programmatically in the page load method of ASP.Net web page E.g.:

```
protected void Page_Load(object sender, EventArgs e)
  {
    CheckBox1.CheckedChanged += new EventHandler(CheckBox1_CheckedChanged);
  }

  void CheckBox1_CheckedChanged(object sender, EventArgs e)
  {

  }
```

When you will write the ID of the checkbox control inside the Page_Load method for example CheckBox1, and will type the **"."** Dot operator it will display the list of properties and methods of checkbox control. Select the CheckedChanged event from the intellisense and press **"+="** and then **tab key** from the keyboard. It will generate the event handler automatically.

C# Code:

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
  {
    Label1.Text = CheckBox1.Checked.ToString();
  }
```

Example1: Working with CheckBox Control as toggle option to determine true or false condition based operation:

HTML Code:

```
<asp:CheckBox ID="CheckBox1"
        runat="server"
        AutoPostBack="True"
        Text="Show Image"
        OnCheckedChanged="CheckBox1_CheckedChanged" />
      <br />
    <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/Ads2.jpg" Width="300" Height="300" Visible="false" />
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Microsoft .NET Solutions
Robust, Scalable & Secure Applications
Developed for Diverse Business Needs
Develop your Microsoft skills and knowledge through great training
NARESH i technologies
Microsoft .net

C# Code:

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
  {
    if (CheckBox1.Checked == true)
    {
      Image1.Visible = true;
      CheckBox1.Text = "Hide Image";
    }
    else
    {
      Image1.Visible = false;
      CheckBox1.Text = "Show Image";
    }
  }
```

Example2: Working with multiple checkbox controls to select multiple options and identify selected options on button's click event as following:

HTML Code:

```
<b>Select Options:</b> 
  <asp:CheckBox ID="chk1" runat="server" Text="Option1" />
    <asp:CheckBox ID="chk2" runat="server" Text="Option2" />
    <asp:CheckBox ID="chk3" runat="server" Text="Option3" />
    <br />
    <asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
    <br />
    <asp:Label ID="lblResult" runat="server" />
```

C# Code:

```
protected void btnSubmit_Click(object sender, EventArgs e)
  {
    string strSelectedOptions = string.Empty;

    if (chk1.Checked == true)
      strSelectedOptions += chk1.Text + ",";
    if (chk2.Checked == true)
      strSelectedOptions += chk2.Text + ",";
    if (chk3.Checked == true)
      strSelectedOptions += chk3.Text + ",";

    if (strSelectedOptions != string.Empty)
    {
      strSelectedOptions = strSelectedOptions.Remove(strSelectedOptions.Length - 1, 1);

      lblResult.Text = "<b>Selected Options: </b>" + strSelectedOptions;
    }
    else
    {
      lblResult.Text = "<b style='color:red'>No Option Selected</b>";
    }
  }
```

Output:

Select Options: ☑ Option1 ☐ Option2 ☑ Option3

Submit

Selected Options: Option1,Option3

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net Multiple CheckBox Validation using JavaScript:

In **ASP.Net** the **CheckBox control** is generally used to display the number of features and users have to select the single or multiple features among multiple choices of checkboxes. By showing **multiple checkbox controls** you can provide the functionality to select one or more checkboxes from a group of options. This functionality requires a **JavaScript validation** to control the number of selected checkboxes that are being checked by the user to specify their selected options. In this document we will learn how to create a group of multiple ASP.Net checkbox controls and will apply the validation using JavaScript client end script to control the number of selections made by the user. You can use the **ASP.Net Panel control** to create the group of checkboxes. In Panel control you can add multiple checkbox controls to display the list of options.

Following HTML Source code can be used to arrange the multiple checkboxes inside the ASP.Net Panel control:

```
<asp:Panel ID="Panel1" runat="server">
  <asp:CheckBox ID="CheckBox1" runat="server" Text="Item 1" onclick="chkCount(this);"  />
  <br />
  <asp:CheckBox ID="CheckBox2" runat="server" Text="Item 2" onclick="chkCount(this);"  />
  <br />
  <asp:CheckBox ID="CheckBox3" runat="server" Text="Item 3" onclick="chkCount(this);"  />
  <br />
  <asp:CheckBox ID="CheckBox4" runat="server" Text="Item 4" onclick="chkCount(this);"  />
  <br />
  <asp:CheckBox ID="CheckBox5" runat="server" Text="Item 5" onclick="chkCount(this);"  />
</asp:Panel>
<asp:HiddenField ID="hiddenChkCount" runat="server" Value="0" />
```

We just need to add **onclick="chkCount(this);"** to each checkbox to call the javascript function for validating the number of checkbox controls that can be selected by the user.

Next we need a JavaScript function to validate the number of checkbox controls checked by the user to specify his/her selected options. The JavaScript validation function requires a resource that could hold the count of number of checkbox controls checked by the user. So we have used the **ASP.Net HiddenField control** to store the count of checked checkbox controls.

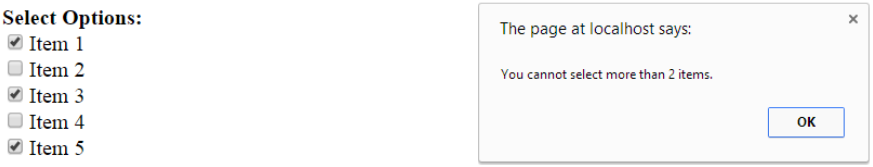
Following JavaScript client side validation function can be used to validate the multiple checkbox selection:

```
<script type="text/javascript">
    function chkCount(obj) {
        if (obj.checked == true) {
            if (document.getElementById('<%=hiddenChkCount.ClientID %>').value >= 2) {
                alert('You cannot select more than 2 items.');
                obj.checked = false;
            }
            else {
                document.getElementById('<%=hiddenChkCount.ClientID %>').value =
parseInt(document.getElementById('<%=hiddenChkCount.ClientID %>').value) + 1;
            }
        }
        else {
            document.getElementById('<%=hiddenChkCount.ClientID %>').value =
parseInt(document.getElementById('<%=hiddenChkCount.ClientID %>').value) - 1;
        }
    }
</script>
```

In the above JavaScript code we have created a simple logic to store the count of selected checkbox controls. When user will click the checkbox control it will call this client side function. First of all it will verify that whether the user has checked the checkbox or un-checked. For checked action of checkbox it will look for the count of checked controls from the HiddenField control. In this "if condition" You can specify the limit for the number of checkbox controls that can be selected by the user. After this "if condition" test, JavaScript code will increase or decrease the counter value in HiddenField based on the user action.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Output:



ASP.Net C# code to add Checkbox onclick event programmatically:

The ASP.Net CheckBox onclick event **can also be added using** C# code dynamically **if there are number of checkbox controls. You can add the** onclick event **as an inline attribute simply if there are one, two or three checkbox controls as we discussed in the previous example. But if there is a group of multiple checkbox controls which require to execute the same client side click event then you can use this second way to attach the event using C# code programmatically. Here we will use the** Add method **of** Attributes collection property **of the Checkbox web server control.**

The Add method of Attributes collection property accepts two string type parameters:

1. key: it accepts the name of the attribute that you want to add.

2. value: it accepts the value for the specified attribute.

You can use the following C# code to add the Checkbox onlcick event dynamically:

```
CheckBox1.Attributes.Add("onclick", "alert('This is a JavaScript alert message box.')");
```

Above C# code is the simplest way to add the attribute to the associated checkbox control programmatically. But for the previous example: **ASP.Net Multiple CheckBox Validation using JavaScript** we need the C# code to add the onclick event to the group of multiple checkboxes. For that purpose you can use the following C# code to attach the same client side onlcick event to each checkbox control placed inside the ASP.Net panel control:

```
protected void Page_Load(object sender, EventArgs e)
  {
    foreach (Control control in Panel1.Controls)
    {
      if (control is CheckBox)
      {
        ((CheckBox)(control)).Attributes.Add("onclick", "chkCount(this)");
      }
    }
  }
```

In the above sample code we have used **C# foreach loop** over the Controls collection of Panel control. Inside the loop, we have used the "if condition" to check whether the control is checkbox or not. If it returns true then it will execute the C# code placed inside the "if condition" block that will add the onclick event to each checkbox control. In C# typecast is required to access the properties of the CheckBox control.

ASP.Net CheckBox Multiple Selection to Check All using C#

You can use a single **ASP.Net Checkbox control** to raise the server side event that can be handled by **C# code** for **multiple selection** of all the checkbox controls. You can create a group of multiple checkbox controls by placing them inside the **ASP.Net Panel control**. It enables you to draw a server based boundary around the controls that can be used to perform the server side action within that Panel control. You can use the C# code to loop over the child controls of the Panel control to change the checked state of all the checkbox controls placed inside it. A single decision making checkbox control must be placed outside the Panel control to reduce the complexity of C# code. Otherwise you will need to apply one extra if condition test to filter the child controls other than the decision making control.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Requirements for Select All Checkbox Control

The decision making Checkbox control that will be used to check all the checkbox controls placed inside the Panel control must be declared with **AutoPostBack property** having value as **"true"** to allow the control to raise the PostBack event of the web page. The other thing is to attach the server side **OnCheckedChanged event handler** to it for handling the click event that will occur at the time when user will click the checkbox control labeled as "Select All".

You can use the following HTML code to display the multiple checkbox controls as a single group of options along with a separate Select All checkbox control:

```
<asp:CheckBox ID="chkSelect"
        runat="server"
        Text="Select All"
        AutoPostBack="True"
        OnCheckedChanged="chkSelect_CheckedChanged" />
    <br />
    <br />
    <asp:Panel ID="Panel1" runat="server">
        <asp:CheckBox ID="CheckBox1" runat="server" Text="Item 1" />
        <br />
        <asp:CheckBox ID="CheckBox2" runat="server" Text="Item 2" />
        <br />
        <asp:CheckBox ID="CheckBox3" runat="server" Text="Item 3" />
        <br />
        <asp:CheckBox ID="CheckBox4" runat="server" Text="Item 4" />
        <br />
        <asp:CheckBox ID="CheckBox5" runat="server" Text="Item 5" />
    </asp:Panel>
```

The following C# code can be used to handle the CheckedChanged server side event handler to check all the checkbox controls placed inside the panel:

```
protected void chkSelect_CheckedChanged(object sender, EventArgs e)
  {
    // C# Multiple Selection
    foreach (Control control in Panel1.Controls)
    {
      if (control is CheckBox)
      {
        ((CheckBox)(control)).Checked = chkSelect.Checked;
      }
    }
  }
```

In the above sample code we have used **C# foreach loop** over the Controls collection of Panel control that provides the reference to each child control. Inside the loop, we have used the **"if condition"** to check whether the control is checkbox or not. If it returns true then it will execute the C# code placed inside the "if condition" block that will set the checked state of the referenced Checkbox control same as the checked state of the decision making "Select All" labelled checkbox that raised the OnCheckedChanged event. In C# typecast is required to access the properties of the CheckBox control.

ASP.Net Check All CheckBox Controls using JavaScript:

You can also the **JavaScript code** to check all the **ASP.Net Checkbox controls** without refreshing the web page. In the previous example: **ASP.Net CheckBox Multiple Selection to Check All using C#** we learnt the use of server side C# code to select all the checkbox controls placed inside the Panel control that sends the web page to PostBack and executes the code specified for **CheckedChanged event** handler of checkbox. Here we will use client side JavaScript function that will check all the checkbox control without sending the web page to PostBack. To develop the functionality we created a group of multiple checkbox controls inside the Panel control and added one more checkbox control outside the Panel. We added the separate Checkbox control for decision making that will all the JavaScript function to select call the checkbox controls placed inside the Panel control.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

You can use the following HTML code to display the group of checkbox controls along with their decision making "Select All" labelled checkbox:

```
<asp:CheckBox ID="chkSelect"
        runat="server"
        Text="Select All"
        onclick="checkAll(this);" />
    <br />
    <br />
    <asp:Panel ID="Panel1" runat="server">
        <asp:CheckBox ID="CheckBox1" runat="server" Text="Item 1" />
        <br />
        <asp:CheckBox ID="CheckBox2" runat="server" Text="Item 2" />
        <br />
        <asp:CheckBox ID="CheckBox3" runat="server" Text="Item 3" />
        <br />
        <asp:CheckBox ID="CheckBox4" runat="server" Text="Item 4" />
        <br />
        <asp:CheckBox ID="CheckBox5" runat="server" Text="Item 5" />
    </asp:Panel>
```

In the above sample html markup code we have added the **checkbox onclick event**as an inline attribute that will call the JavaScript client side function. You can also attach the onclick event to the outer checkbox control using following C# code:

```
chkSelect.Attributes.Add("onclick", "checkAll(this);");
```

Last we need the JavaScript code to execute the logic for selecting the multiple checkbox controls in a single click. You can use the following JavaScript code to check all checkbox controls:

```
<script type="text/javascript">
    function checkAll(obj1) {
        var checkboxCollection = document.getElementById('<%=Panel1.ClientID %>').getElementsByTagName('input');

        for (var i = 0; i < checkboxCollection.length; i++) {
            if (checkboxCollection[i].type.toString().toLowerCase() == "checkbox") {
                checkboxCollection[i].checked = obj1.checked;
            }
        }
    }
</script>
```

In the above JavaScript code we have used the **document.getElementById function** to get the reference to Panel control and **document.getElementsByTagName function** to retrieve the collection of all the HTML input controls rendered inside the Panel control. Next we have used the "if condition" to detect the type of the control whether it is checkbox or not. It will change the checked state of all the checkbox controls stored in the collection of input controls placed inside the Panel control.

RadioButton Control:

This control is used to accept the data in true/false or say in yes/no Boolean format. Like if we select it then it says true or yes. By default its checked property is false but we can set it to true.
OR
The RadioButton control provides the functionality to create a group of mutually exclusive items so that the user may select only one option at a time.
OR
The RadioButton control permits you to intersperse the radio buttons in a group with other content in the page. The buttons in the sample are grouped logically because they all share the same **GroupName**.

RadioButton cannot be bind with database. It can be also used to trigger postback to the server if its AutoPostBack property is true.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

The **ASP.Net RadioButton control** provides the functionality to display an individual radio button on the Web form that returns the true or false type Boolean value as a result. If the **RadioButton control** is in **selected state** then it returns true and if it is in unselected state then the radio button returns false. You can display a group of multiple radio buttons on the web page to provide a mutually exclusive set of options to the users. Users can select one of the radio buttons from the group to specify their choice. You can perform the server side action based on the selection made by the user that can detected by getting the value of checked property of each radio button displayed as a member of the group.

Properties of ASP.Net RadioButton Control

Following are the properties of ASP.Net RadioButton control to get or set its value or behaviour on the web page:

1. Checked: It gets or sets the RadioButton state. If the radio button control is in selected state then checked property returns true otherwise it returns false. The Checked property of radio button control accepts and returns Boolean value.

2. AutoPostBack: If you want to execute the PostBack event of radio button control when user changes its state to selected state. The AutoPostBack property also accepts the Boolean value i.e. true or false to allow or disallow the PostBack event of radio button control.

3. GroupName: It accepts the string value as a name for the group of radio buttons to create a list of mutually exclusive set of options. Multiple radio buttons specified with same GroupName allows to select only one radio button option.

4. Text: It allows you to display the label with a text string next to the RadioButton control. You can specify a string value for the text property of radio button control.

5. TextAlign: It enables you to adjust the alignment of the text value specified to the Text property of the radio button. You can align the specified text to the left or right side of the radio button.

6. AccessKey: Keyboard shortcut used by the control.

7. CausesValidation: It indicates whether the control causes the validation to fire or not. By default it is set to false.

7. ValidationGroup: Specify the group that should be validated when control causes a postback.

The RadioButton control consists of following PostBack event that can be handled at server side to execute the server code and perform the specified action according to the selected radio button control:

CheckedChanged Event: It enables you to handle the PostBack event of RadioButton control when user changes its state from selected state. When user clicks the radio button it raises the CheckedChanged event only if the **AutoPostBack property** of RadioButton control is set to true.

The radio button control is an ASP.Net web server control when it is rendered on the web page it splits up into two parts:

1. HTML input control having **"type"** attribute value as **"radio"**.

2. HTML label control to display the string specified for the text property of radio button server control.

RadioButton Control Examples:

- **RadioButton Control Sample**
- **RadioButton Control Checked and GroupName Property**
- **RadioButton Control AutoPostBack Property**
- **RadioButton Control OnCheckedChanged Event**

ASP.Net HTML Code for RadioButton control

```
<asp:RadioButton ID="RadioButton1"
        runat="server"
        Text="Radio Button 1" />
```

The HTML code that is generated by ASP.Net framework when it renders the control on the browser:

```
<input id="RadioButton1" type="radio" name="RadioButton1" value="RadioButton1" />
    <label for="RadioButton1">Radio Button 1</label>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net RadioButton Checked and GroupName Property:

The **Checked property** of **ASP.Net RadioButton control** enables you to get or set the selected state of RadioButton. You can also set the checked property of radio button control dynamically using server code or manually when user clicks the RadioButton control to change its selected state. The **checked property** of ASP.Net RadioButton control accepts the **Boolean value** to set its state. It also returns the Boolean value to represent the state of the radio button placed on the web page. When user clicks the RadioButton control to set its state as checked then it returns **"true"** and when user selects other radio button from the same group then it sets the first radio button to un-checked state and clicked radio button to checked state.

Sample Code for ASP.Net RadioButton Checked Property

HTML Code

```
<asp:RadioButton ID="RadioButton1"
        runat="server"
        Text="radio button 1"
        Checked="true"
        GroupName="group1" />
    <br />
    <asp:RadioButton ID="RadioButton2"
        runat="server"
        Text="radio button 2"
        GroupName="group1" />
    <br />
    <asp:Button ID="Button1"
        runat="server"
        Text="Submit"
        OnClick="Button1_Click" />
    <br />
    <asp:Label ID="Label1"
        runat="server">
    </asp:Label>
```

In the above HTML source code for RadioButton control we have declared the**Checked property** as "true" for the first radio button. It will render the radio button with checked state initially. To create the mutually exclusive set of choices we have used the **GroupName property** of radio buttons assigned with same value as**"group1"**. It will provide the functionality to select the single radio button within the same **group of radio buttons**.

C# Code

```
protected void Button1_Click(object sender, EventArgs e)
  {
     Label1.Text = "RadioButton1.Checked = " + RadioButton1.Checked.ToString() + "<br />";
     Label1.Text += "RadioButton2.Checked = " + RadioButton2.Checked.ToString();
  }
```

We have also used the Button Control to get the value of Checked property of radio button controls. In the attached click event handler of Button control we have retrieved the values to display them on the web page with the help of Label control.

ASP.Net RadioButton AutoPostBack Property:

The **AutoPostBack property** of the **ASP.Net RadioButton control** provides the functionality to allow the server side post back event of the web page when user clicks the radio button to change its inactive state to selected state. You can get the value of checked state of the radio button control at the Page Load event of ASP.Net web page to perform the server side action based on the checked state of radio button. The **AutoPostBack** property is a Boolean type property that accepts value as true/false to enable or disable the auto post back functionality of the radio button control to the server side when user selects it. You can declare its value in two ways such as in HTML code block as an inline attribute of radio button control or by declaring it programmatically when page loads.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Sample Code for ASP.Net RadioButton AutoPostBack Property

HTML Code

```
<asp:RadioButton ID="RadioButton1" runat="server" Text="Radio Button 1" GroupName="group1" AutoPostBack="True"
        Checked="True" />
 <br />
<asp:RadioButton ID="RadioButton2" runat="server" Text="Radio Button 2" GroupName="group1" AutoPostBack="True" />
 <br />
<asp:Panel ID="Panel1" runat="server" Visible="true"  BorderStyle="Solid" BorderWidth="1px" BorderColor="Red"
 ForeColor="Maroon" Width="200px" HorizontalAlign="Justify">
<h3>Content and Other Controls associated to first RadioButton Control</h3>
</asp:Panel>
<asp:Panel ID="Panel2" runat="server" Visible="false" BorderStyle="Solid" BorderWidth="1px" BorderColor="Red"
        ForeColor="Maroon" Width="200px" HorizontalAlign="Justify">
 <h3>Content and Other Controls associated to second RadioButton Control</h3>
</asp:Panel>
```

In the above HTML source code we have used 2 ASP.Net radio button controls declared with their **GroupName property** and **AutoPostBack property**. The GoupName property enables you to create a group of radio buttons so that the user could select only one radio button among **multiple set of options**. The AutoPostBack property is used in the sample to allow the controls for posting the web form data to the server when user will click any of the radio button control. Other than the radio button controls we have used ASP.Net Panel controls also to illustrate the additional functionality that can be provided along with the AutoPostBack feature of radio button controls. When the user will select the first radio button then it will set the first panel control to be visible on the web page whereas the second radio button will hide the first panel control and display the second panel control associated to it.

C# Code

```
protected void Page_Load(object sender, EventArgs e)
  {
     Panel1.Visible = RadioButton1.Checked;
     Panel2.Visible = !RadioButton1.Checked;
  }
```

The C# code shows that you can get the value of **checked property** of radio button controls at the Page Load event of ASP.Net web page when the **AutoPostBack property** of radio button controls is specified with value "true".

ASP.Net RadioButton OnCheckedChanged Event

The **OnCheckedChanged event** handler of **ASP.Net RadioButton control** enables you to handle the click event of radio button at server side that is raised at the time when user clicks the control to change its in-active state. The **CheckedChanged** event of radio button control occurs only if its **AutoPostBack property** is specified with value "true". You can handle the click events of multiple radio button controls separately and perform the server side action for each control using their dedicate OnCheckedChanged event handlers. The CheckedChanged event occurs immediately when user clicks the radio button control for selecting the item and submits the web form to the server only if the AutoPostBack property has value "true".

Sample Code for ASP.Net RadioButton OnCheckedChanged Event

HTML Code

```
<asp:RadioButton ID="RadioButton1" runat="server" Text="Radio Button 1" GroupName="group1" AutoPostBack="True"
        Checked="True" OnCheckedChanged="RadioButton1_CheckedChanged" />
 <br />
<asp:RadioButton ID="RadioButton2" runat="server" Text="Radio Button 2" GroupName="group1" AutoPostBack="True"
OnCheckedChanged="RadioButton2_CheckedChanged" />
 <br />
<asp:Panel ID="Panel1" runat="server" Visible="true"  BorderStyle="Solid" BorderWidth="1px" BorderColor="Red"
 ForeColor="Maroon" Width="200px" HorizontalAlign="Justify">
<h3>Content and Other Controls associated to first RadioButton Control</h3>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.


```
</asp:Panel>
<asp:Panel ID="Panel2" runat="server" Visible="false" BorderStyle="Solid" BorderWidth="1px" BorderColor="Red"
        ForeColor="Maroon" Width="200px" HorizontalAlign="Justify">
 <h3>Content and Other Controls associated to second RadioButton Control</h3>
</asp:Panel>
```

In the above sample code we have used two radio buttons specified with their AutoPostBack property value as "true" and also OnCheckedChanged server side event is used to handle the click event of each radio button separately when user clicks the control to change its checked state.

You can attach the **Checked Event of radio button control** in two ways as follows:

1. Double clicking the radio button control for which you want to generate the server side OnCheckedChanged event handler. In this document we have also used the same approach to handle the CheckedChanged event of radio buttons.

2. You can also attach the CheckedChanged event programmatically in the page load method of ASP.Net web page E.g.:

```
protected void Page_Load(object sender, EventArgs e)
  {
    RadioButton1.CheckedChanged += new EventHandler(RadioButton1_CheckedChanged);
  }

  void RadioButton1_CheckedChanged(object sender, EventArgs e)
  {

  }
```

When you will write the ID of the radio button control inside the Page_Load method for example RadioButton1, and will type the "." Dot operator it will display the list of properties and methods of RadioButton control. Select the CheckedChanged event from the intellisense and press "+=" and then tab key from the keyboard. It will generate the event handler automatically.

C# code

```
protected void RadioButton1_CheckedChanged(object sender, EventArgs e)
  {
    Panel1.Visible = RadioButton1.Checked;
    Panel2.Visible = !RadioButton1.Checked;
  }

  protected void RadioButton2_CheckedChanged(object sender, EventArgs e)
  {
    Panel1.Visible = !RadioButton2.Checked;
    Panel2.Visible = RadioButton2.Checked;
  }
```

The above server side event handlers will handle the checked changed event of both radio button controls individually that we placed on the web page as discussed above. Other than radio button controls we have used two ASP.Net Panel controls also to illustrate the additional functionality that you can combine with the checked changed event of radio button controls. When you will click the first radio button then it will hide the second panel control and will display only first Panel control as an associated control to it. Whereas when you will click the second radio button control then its CheckedChanged handler code will execute and will set the visible state of the second panel control to true and first panel control to false.

DropDownList Control:

The DropDownList control provides the user to select a single item from a drop-down list.
OR
This control is used to select any one item from list of items. We cannot select multiple items from DropDownList.
OR
It renders a drop down menu containing a list of items. You can select a single item at a time and associate an event handler that will get executed at the time selected item changes. It has also database binding property.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

The **ASP.Net DropDownList control** provides the functionality to display a single select list of items in the form of drop down menu. It does not allow the user to select multiple items at different indexes and enables you to get or set the single selected index of the DropDownList control to store it into the data store according to the user input. You can display a long list of items in the ASP.Net DropDownList control and adjust its height and width using CSS styles or its inbuilt properties that enable you to specify its height and width. Some browsers do not support the height parameter and display the length of DropDownList based on their instruction sets.

The ASP.Net DropDownList control consists of following properties that are commonly used to get or set the list item values or user input. It also includes the two types of property sets that allow you to customize the look of the **DropDownList control** and bind the list items to it for displaying the single select list of items on the web page. Following properties allow you to get or set the list items of ASP.Net DropDownList control:

AppendDataBoundItems: It allows you to add the list items of DropDownList along with the data-bound list items retrieved from any data source. It is a Boolean type property allows or disallows the append feature of list items based on its true or false value.

AutoPostBack: It is also a Boolean type property that enables the control to send the web page to PostBack when user clicks the DropDownList list item to select it from the drop down menu.

DataSource: This property allows you to specify the name of the data source dynamically.

DataSourceID: It accepts the ID of the DataSource control such as SqlDataSource, XmlDataSource control etc. used to retrieve the data from the database.

DataMember: It accepts the name of one of the data table stored in the data source, if there is a collection of data tables in it. For example, using SQL queries or stored procedures you can fetch the data in the form of multiple data tables that can be stored into a DataSet class object. This DataSet object can be used to bind the data to different types of Data-bound controls. The DataMember property enables you to specify the name of the table that you want to bind with any data-bound control or DropDownList control that also support data binding feature.

DataTextField: It accepts the name of the Field/Column of the data table stored in the DataSource object that you want to display as the text content of the DropDownList items.

DataValueField: It accepts the name of the field of the data table stored in the DataSource object that you want to set as the value attribute of each list item of dropdownlist control. This value part does not appear as text content.

DataTextFormatString: It allows you to format the text property of the DropDownList item. For example you can use {0:C} to format the numeric value as currency. There are different types of formats such as {0:X}, {0:E}, {0:N}, {0:C}.

Items: The Items property provides an easy way to add the list items manually. It allows you to generate the collection of list items using a dialog box that auto generates the HTML markup code at the back ground.

It represents the collection of items in the list control, in which each item called ListItem and each ListItem has its own properties.

ListItem Properties

Each list item has following properties that can be declared as inline attribute of ListItem of DropDownList control:

1. Text
2. Value
3. Selected
4. Enabled

The **text property** renders the text content for each list item and the **value property** provides the back end value associated to them individually. The **Enabled property** accepts the Boolean type true/false value that enables or disables the visibility of list item. The **Selected property** is also a Boolean type property of ListItem that gets or sets the selected state of the dropdownlist items.

Other than above properties of ASP.Net DropDownList control that allow you to populate it with list items, it also contains few **CSS based style properties** that allow you to customize the appearance of control and its fonts. The DropDownList control supports BackColor, CssClass, Font style properties, ForeColor, Height and Width property. The Font style properties include Bold, Italics, Name, Underline, Size, and Strikeout like properties that enable you to change the appearance of text displayed as list items of DropDownList.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

DropDownList Control Event

It supports **OnSelectIndexChanged event** that occurs when user clicks the list item to select it. You can attach the server side SelectedIndexChanged event handler for DropDownList control to handle the click event raised by the user using C# code.

DropDownList Control Examples:

- **DropDownList Control Sample**
- **DropDownList Control AutoPostBack Property**
- **DropDownList Control OnSelectedIndexChanged Event**
- **Populate DropDownList Items Dynamically**

DropDownList Control Sample:

ASP.Net HTML Code for DropDownList control

```
<asp:DropDownList ID="DropDownList1" runat="server">
        <asp:ListItem Text="Select" Value="Select" />
        <asp:ListItem Text="Option1" Value="Opt1" />
        <asp:ListItem Text="Option2" Value="Opt2" />
        <asp:ListItem Text="Option3" Value="Opt3" />
 </asp:DropDownList>

OR

<asp:DropDownList ID="DropDownList1" runat="server">
        <asp:ListItem Value="Select">Select</asp:ListItem>
        <asp:ListItem Value="Opt1">Option1</asp:ListItem>
        <asp:ListItem Value="Opt2">Option2</asp:ListItem>
        <asp:ListItem Value="Opt3">Option3</asp:ListItem>
 </asp:DropDownList>
```

The HTML code that is generated by ASP.Net framework when it renders the control on the browser:

```
<select name="DropDownList1" id=" DropDownList1">
        <option value="Select">Select</option>
        <option value="Opt1">Option1</option>
        <option value="Opt2">Option2</option>
        <option value="Opt3">Option3</option>
</select>
```

ASP.Net DropDownList AutoPostBack Property:

The **AutoPostBack property** of the **ASP.Net DropDownList control** provides the functionality to allow the server side post back event of the web page when user clicks on a list item to select it. You can get the value of selected item of the DropDownList control at the **Page Load event** of ASP.Net web page to execute the server side code. The **AutoPostBack property** is a Boolean type property that accepts value as true/false to enable or disable the auto post back functionality of the dropdownlist control that submits the web page data to server side when user selects any list item. You can declare the value of AutoPostBack property in any of your preferred way, either by declaring it as an inline attribute in the HTML markup code block or by using C# code in the code behind file programmatically.

Sample Code for ASP.Net DropDownList AutoPostBack

HTML Code

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="true">
        <asp:ListItem Value="Select">Select</asp:ListItem>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.


```
        <asp:ListItem Value="Opt1">Option1</asp:ListItem>
        <asp:ListItem Value="Opt2">Option2</asp:ListItem>
        <asp:ListItem Value="Opt3">Option3</asp:ListItem>
</asp:DropDownList>
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
```

In the above ASP.Net HTML code for DropDownList control we have declared the value for AutoPostBack property of as an inline attribute. We have specified the value of **AutoPostBack property** as **"true"** to allow the web page to post back when user will select any list item of dropdownlist menu.

C# Code

```
protected void Page_Load(object sender, EventArgs e)
  {
     Label1.Text ="Selected Item: <br />Text: "  + DropDownList1.SelectedItem.Text + "<br />Value: " +
DropDownList1.SelectedItem.Value;
  }
```

The above C# code shows that you can get the value of select list item of dropdownlist control at the Page Load event of ASP.Net web page when the AutoPostBack property of dropdownlist is specified with value "true". The above sample code will display the text & value of list item selected by the user. The AutoPostBack property will allow the auto post back of web form as soon as the user will click the item.

ASP.Net DropDownList OnSelectedIndexChanged Event:

The **OnSelectedIndexChanged event** handler of **ASP.Net DropDownList control** enables you to handle the click event of list item at server side that is raised at the time when user clicks to select a list item of dropdownlist control. The **SelectedIndexChanged event** of DropDownList control occurs only if the **AutoPostBack property** of dropdownlist is specified with value "true". The SelectedIndexChanged event occurs immediately when the index of the selected list item of the dropdown list item changes and it submits the web form to the server only if its AutoPostBack property has value "true". You can handle the event at server side using C# code and apply the custom logic there to execute any specific action based on the value of select item of the dropdownlist menu.

Sample Code for ASP.Net DropDwonList OnSelectedIndexChanged Event

HTML Code

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="true"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
        <asp:ListItem Value="Select">Select</asp:ListItem>
        <asp:ListItem Value="Opt1">Option1</asp:ListItem>
        <asp:ListItem Value="Opt2">Option2</asp:ListItem>
        <asp:ListItem Value="Opt3">Option3</asp:ListItem>
    </asp:DropDownList>
    <br />
    <asp:Label ID="Label1" runat="server"></asp:Label>
```

In the above sample code for DropDownList control we have specified the AutoPostBack property value as "true" and also OnSelectedIndexChanged server side event is used to handle the click event of dropdown list item when user clicks an item to select it. You can attach the SelectedIndexChanged of dropdownlist control in two ways as follows:

1. Double clicking the dropdownlist control for which you want to generate the server side OnSelectedIndexChanged event handler. In this document we have also used the same approach to handle the SelectedIndexChanged event of dropdownlist control.

2. You can also attach the SelectedIndexChanged event programmatically in the page load method of ASP.Net web page E.g.:

```
protected void Page_Load(object sender, EventArgs e)
  {
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
    DropDownList1.SelectedIndexChanged += new EventHandler(DropDownList1_SelectedIndexChanged);
  }

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
  {

  }
```

When you will write the ID of the dropdownlist control inside the Page_Load method for example DropDownList1, and will type the "." Dot operator it will display the list of properties and methods of dropdownlist control. Select the SelectedIndexChanged event from the intellisense and press "+=" and then tab key from the keyboard. It will generate the event handler automatically.

C# code

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
  {
    if (DropDownList1.SelectedIndex > 0)
    {
      Label1.Text = "Selected Item: <br />Text: " + DropDownList1.SelectedItem.Text + "<br />Value: " +
DropDownList1.SelectedItem.Value;
    }
    else
    {
      Label1.Text = "<b style='color:red'>Please Select Option!!!</b>";
    }
  }
```

In the above C# sample code for SelectIndexChanged event handler of ASP.Net DropDownList control we have used the SelectedIndex property to check the selected item index value and if it is >0 then only displays the text & value of selected item using the following properties such as SelectedItem.Text & SelectedItem.Value. When user will click on any list item of dropdownlist menu it will raise the SelectedIndexChanged event that will execute the above specified code to get the value of selected item.

Populate ASP.Net DropDownList by Adding Items Dynamically:

The **ASP.Net DropDownList control** also provides the method to populate it by adding the list items dynamically using C# code. You can use the Add method of Items collection property of DropDownList control to populate the items programmatically. The **Items property** gets the collection of list items stored in the DropDownList control. It belongs to the **ListItemCollection class** and provides the read-only member property of this collection class. The Items collection property further enables you to access the public methods of ListItemCollection class that allow you to add or remove the list items. Here we will use the **Add method** to populate the list items into the DropDownList control.
Following are the two overloaded ways for using the Add method of Items collection property of DropDownList:

1.
[DropDownList].Items.Add(ListItem item);

It allows you to add the specified list item at the end of DropDownList's Item collection. You can use the well suited overloaded method of ListItem class constructor to add a new item.

2.
[DropDownList].Items.Add(**string** text);

It appends the collection of list items of DropDownList control by adding the specified string at the end of items collection.

For adding the ListItem using first overloaded Add method you can use the following overloaded constructor of ListItem class:

new ListItem(**string** text, **string** value);

It allows you to add the ListItem with its text that will appear in the DropDownList control along with its value that will work at the back end.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Sample Code to Populate DropDownList Control Dynamically

HTML Code

```
<asp:DropDownList ID="ddl1" runat="server">
</asp:DropDownList>
```

C# Code

There are different ways to add the List Items into the DropDownList control. In the following samples we have shown 6 different ways to add the ListItems:

1.

```
protected void Page_Load(object sender, EventArgs e)
   {
     //Items as String Item
      if (!Page.IsPostBack)
      {
         //Adding
         ddl1.Items.Add("Option1");
         ddl1.Items.Add("Option2");
         ddl1.Items.Add("Option3");

         //Inserting Item as String at Zero Index Position
         ddl1.Items.Insert(0, "Select");
      }
   }
```

2.

```
protected void Page_Load(object sender, EventArgs e)
   {
     //Items as List Item
      if (!Page.IsPostBack)
      {
         //Creating the ListItems
         ListItem li1 = new ListItem("Option1", "Opt1");
         ListItem li2 = new ListItem("Option2", "Opt2");
         ListItem li3 = new ListItem("Option3", "Opt3");
         //Adding
         ddl1.Items.Add(li1);
         ddl1.Items.Add(li2);
         ddl1.Items.Add(li3);

         //Inserting Item as ListItem at Zero Index Position
         ListItem liSelect = new ListItem("Select", "Select");
         ddl1.Items.Insert(0,liSelect);
      }
   }
```

3.

```
protected void Page_Load(object sender, EventArgs e)
   {
     //Items as List Item
      if (!Page.IsPostBack)
      {
         //Adding
         ddl1.Items.Add(new ListItem("Option1", "Opt1"));
         ddl1.Items.Add(new ListItem("Option2", "Opt2"));
         ddl1.Items.Add(new ListItem("Option3", "Opt3"));
```

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
        //Inserting Item as ListItem at Zero Index Position
         ddl1.Items.Insert(0, new ListItem("Select", "Select"));
      }
   }
```

4.

```
protected void Page_Load(object sender, EventArgs e)
   {
     //Items as List Item
      if (!Page.IsPostBack)
      {
         //Adding Items as an Array of ListItems using AddRange() Method
         ListItem[] li = new ListItem[3]
         {
            new ListItem("Option1", "Opt1"),
            new ListItem("Option2", "Opt2"),
            new ListItem("Option3", "Opt3")
         };
         ddl1.Items.AddRange(li);

         //Inserting Item as ListItem at Zero Index Position
          ddl1.Items.Insert(0, new ListItem("Select", "Select"));
      }
   }
```

5.

```
protected void Page_Load(object sender, EventArgs e)
   {
     //Items as List Item
      if (!Page.IsPostBack)
      {
         //Adding Items as an Array of ListItems using AddRange() Method
         ddl1.Items.AddRange(new ListItem[3]
         {
            new ListItem("Option1", "Opt1"),
            new ListItem("Option2", "Opt2"),
            new ListItem("Option3", "Opt3")
         });

         //Inserting Item as ListItem at Zero Index Position
          ddl1.Items.Insert(0, new ListItem("Select", "Select"));
      }
   }
```

6.

```
protected void Page_Load(object sender, EventArgs e)
   {
     //Items as List Item
      if (!Page.IsPostBack)
      {
         ListItemCollection lic = ddl1.Items;
         lic.Add(new ListItem("Option1", "Opt1"));
         lic.Add(new ListItem("Option2", "Opt2"));
         lic.Add(new ListItem("Option3", "Opt3"));

         //Inserting Item as ListItem at Zero Index Position
          lic.Insert(0, new ListItem("Select", "Select"));
      }
   }
```

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net DropDownList Validation using JavaScript:

You can also **validate** the selected value of **ASP.Net DropDownList control** using **JavaScript** client side function by calling it when user clicks the list item of dropdownlist control to select it. The **JavaScript validation function** executes at the click event of DropDownList control that occurs at client side and executes the client side script before sending the ASP.Net web page to post back. The validation is required when the field is **mandatory** to send the input received from the user side. You can check the user input at the time of web form post back but **client side validation script** enables you to display the warning messages without reloading the web page again and again for different input fields. You can display the **alert message box** showing the instructions for the user to specify the valid information using web form fields. In this document we have also used the JavaScript function to check the selected value of **ASP.Net DropDownList control** whether the user has selected the valid item or not. If a user will select invalid list item then it will pop-up an alert message box showing the warning message.

Sample Code for ASP.Net DropDownList Validation

HTML Code

```
<asp:DropDownList ID="DropDownList1"
        runat="server">
        <asp:ListItem Value="">[Please select any group item]</asp:ListItem>
        <asp:ListItem Value="" style="font-weight: bold;">Group1</asp:ListItem>
        <asp:ListItem Value="g11">item 1</asp:ListItem>
        <asp:ListItem Value="g12">item 2</asp:ListItem>
        <asp:ListItem Value="g13">item 3</asp:ListItem>
        <asp:ListItem Value="g14">item 4</asp:ListItem>
        <asp:ListItem Value="g15">item 5</asp:ListItem>
        <asp:ListItem Value="" style="font-weight: bold;">Group2</asp:ListItem>
        <asp:ListItem Value="g21">item 1</asp:ListItem>
        <asp:ListItem Value="g22">item 2</asp:ListItem>
        <asp:ListItem Value="g23">item 3</asp:ListItem>
        <asp:ListItem Value="g24">item 4</asp:ListItem>
        <asp:ListItem Value="g25">item 5</asp:ListItem>
</asp:DropDownList>
<asp:Button ID="Button1"
        runat="server"
        Text="Submit"
        OnClick="Button1_Click" />
    <br />
<asp:Label ID="Label1" runat="server"></asp:Label>
```

In the above HTML markup code for DropDownList control we have created two groups of list items. There is no value specified for the ListItems having text content "Group1" and "Group2". We have also not specified the value for first item. By default the DropDownList control will display the first list item as a selected value but according to the validation criteria the list items having no value are invalid items as a user input. The user is supposed to select only list items under the group names otherwise it will show a JavaScript alert message box.

Use the following JavaScript client side function to validation ASP.Net dropdownlist control:

```
<script type="text/javascript">
    function validateDropDown(obj) {
        if (document.getElementById(obj).value == '') {
            alert('please select an item value under any group name.');
            return false;
        }
    }
</script>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Now you have to attach the above JavaScript function with DropDownList control and a Button control that will submit the form data to the server. In the JavaScript function code we have used an "if condition" to check the value of selected item of DropDownList control. It will show an alert message for invalid blank values. You can use the following C# code to attach the client side events to the DropDownList and Button web server controls:

C# Code

```
protected void Page_Load(object sender, EventArgs e)
  {
     DropDownList1.Attributes.Add("onchange", "javascript:return validateDropDown('" + DropDownList1.ClientID + "');");

     Button1.Attributes.Add("onclick", "javascript:return validateDropDown('" + DropDownList1.ClientID + "');");
  }

  protected void Button1_Click(object sender, EventArgs e)
  {
     Label1.Text = "ASP.NET From Server Side!!!";
  }
```

In the above C# code we have used the **onchange client side event** of DropDownList control that will call the JavaScript function when the user will change the selected item of dropdownlist menu. The onclick client side event has been used to call the same JavaScript validation function to validate the selected value of dropdownlist control before submitting the form data to the server.

ListBox Control:

This control is used to select one or more items from a list of items. It has same properties list as combo box but combo box can't select multiple items. Change SelectionMode property to multiple in ListBox for multiple selections.
OR
The ListBox control provides a single-selection or multiple-selection list. To enable multiple selection, set the SelectionMode property to Multiple.
OR
The ListBox control renders with a predefined list of items and allows selecting one or more than one item from the list. You can also attach the PostBack event that is to be executed when user changes the selected item.

The **ASP.Net ListBox control** provides the functionality to display a list of items in the form of menu that allows single or multiple items selection. You can select the multiple list items of ListBox control by changing their selected state. The selected items of ListBox return the Boolean type result as true or false that determine the selected state of an individual list item or multiple list items based on the C# code logic applied on it. If the list item is in **selected state** then it returns true and if it is in un-selected state then the associated list item returns false. You can use the ASP.Net ListBox web server control to get the user input as multiple items selected under a specific group. A simple example of ListBox control is displaying a list of items to display country, state or cities that can be used to select multiple locations to search any desired object based on filter criteria specified by using multiple selection of items of ListBox controls.
The ASP.Net ListBox control consists of following properties that are commonly used to get or set the values or user input. There are two sets of properties below; the first set includes the properties that enable you to get or set the values of list items of ListBox and the second set includes the properties that enable you to customize its appearance:

AppendDataBoundItems: It allows you to add the list items of ListBox in a static way along with the data-bound list items retrieved from any data source. It is a Boolean type property that accepts value as true or false for allowing or disallowing the append feature of list items.

AutoPostBack: It is also a Boolean type property that enables the control to send the web page to PostBack when user clicks the list item of ListBox to select or un-select it.

DataSource: This property allows you to specify the name of the data source dynamically.

DataSourceID: It accepts the ID of the DataSource control used to retrieve the data from the database.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

DataMember: It accepts the name of one of the data table stored in the data source, if there is a collection of data tables in it. For example, while using SQL queries or stored procedures you can retrieve data in the form of multiple data tables that can be stored into the DataSet class object. This DataSet object can be used to bind the data to different types of Data-bound controls on the ASP.Net web page. The DataMember property enables you to specify the name of the table that you want to bind with any particular data-bound control.

DataTextField: It accepts the name of the Field/Column of the data table stored in the DataSource object that you want to display as the text content of the list item.

DataValueField: It accepts the name of the field of the data table stored in the DataSource object that you want to set as the value attribute of each list item. This value part does not appear as text content.

DataTextFormatString: It allows you to format the text property of the list item. For example you can use {0:C} to format the numeric value as currency. There are different types of formats such as {0:X}, {0:E}, {0:N}, {0:C}.

Items: The Items property provides an easy way to add the list items manually. It allows you to generate the collection of list items using a dialog box that auto generates the HTML markup code at the back ground.

It represents the collection of items in the list control, in which each item called ListItem and each ListItem has its own properties.

ListItem Properties

Each list item has following properties that can be declared as inline attribute of ListItem of DropDownList control:

1. Text
2. Value
3. Selected
4. Enabled

The **text property** renders the text content for each list item and the **value property** provides the back end value associated to them individually. The **Enabled property** accepts the Boolean type true/false value that enables or disables the visibility of list item. The **Selected property** is also a Boolean type property of ListItem that gets or sets the selected state of the dropdownlist items.

SelectionMode: It allows you to set the selection mode of ListBox control to enable it for single or multiple selections of items. It accepts the following two types of predefined values to set the selection mode:

1. Single

2. Multiple

Other than above properties of ASP.Net ListBox control that allow you to populate it with list items, it also contains few **CSS based style properties** that allow you to customize the appearance of control and its fonts. The ListBox control supports BackColor, CssClass, Font style properties, ForeColor, Height and Width property. The Font style properties include Bold, Italics, Name, Underline, Size, and Strikeout like properties that enable you to change the appearance of text displayed as list items of ListBox.

ListBox Control Event

It supports **OnSelectIndexChanged event** that occurs when user clicks the list item to select it. You can attach the server side SelectedIndexChanged event handler for ListBox control to handle the click event raised by the user using C# code.

ListBox Control Examples:

- **ListBox Control Sample**
- **ListBox Control AutoPostBack Property**
- **ListBox Control OnSelectedIndexChanged Event**
- **ListBox Control SelectionMode Property**

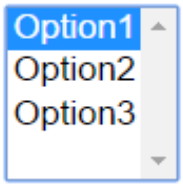
Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Microsoft .NET Solutions
Robust, Scalable & Secure Applications
Developed for Diverse Business Needs
Develop your Microsoft skills and knowledge through great training
NARESH i technologies
Microsoft .net

ListBox Control Sample:

ASP.Net HTML Code for ListBox control

SelectionMode: Single

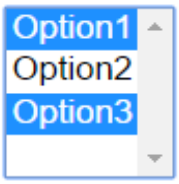



```
<asp:ListBox ID="ListBox1" runat="server">
        <asp:ListItem Text="Option1" Value="Opt1" />
        <asp:ListItem Text="Option2" Value="Opt2" />
        <asp:ListItem Text="Option3" Value="Opt3" />
</asp:ListBox>
OR
<asp:ListBox ID="ListBox1" runat="server">
        <asp:ListItem Value="Opt1">Option1</asp:ListItem>
        <asp:ListItem Value="Opt2">Option2</asp:ListItem>
        <asp:ListItem Value="Opt3">Option3</asp:ListItem>
 </asp:ListBox>
```

The HTML code that is generated by ASP.Net framework when it renders the control on the browser:

```
<select size="4" name="ListBox1" id="ListBox1">
        <option value="Opt1">Option1</option>
        <option value="Opt2">Option2</option>
        <option value="Opt3">Option3</option>
</select>
```

SelectionMode: Multiple




```
<asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
        <asp:ListItem Text="Option1" Value="Opt1" />
        <asp:ListItem Text="Option2" Value="Opt2" />
        <asp:ListItem Text="Option3" Value="Opt3" />
</asp:ListBox>
OR
<asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
        <asp:ListItem Value="Opt1">Option1</asp:ListItem>
        <asp:ListItem Value="Opt2">Option2</asp:ListItem>
        <asp:ListItem Value="Opt3">Option3</asp:ListItem>
 </asp:ListBox>
```

The HTML code that is generated by ASP.Net framework when it renders the control on the browser:

```
<select size="4" name="ListBox1" multiple="multiple" id="ListBox1">
        <option value="Opt1">Option1</option>
        <option value="Opt2">Option2</option>
        <option value="Opt3">Option3</option>
</select>
```

Note: All the above ListBox Control Examples are same as DropDownList Control except of SelectionMode property option.

ASP.Net ListBox SelectionMode Property:

The **SelectionMode property** of the **ASP.Net ListBox control** allows you to enable or disable the multiple selections of list items displayed in the form list menu. It accepts the predefined enumeration value to specify the selection mode for the ListBox. The **ListSelectionMode enumeration** provides the predefined values for the SelectionMode property of the ASP.Net ListBox control. For selecting the multiple items of ListBox you can use the Ctrl key of the keyboard and click button of mouse. To select more than one list items you must keep pressing the **Ctrl key** and click on multiple list items to select than. You can also select all the list items of ListBox control by select and drag method by pressing down the mouse button over the first list item and then moving the mouse up to the last list item for selecting all the items.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net ListBox SelectionMode Values

The SelectionMode property of ListBox control supports the following two types of predefined enumerated values provided by the ListSelectionMode enumerator:

1. Single: It disables the multiple selections of list items but allows you to select at least one item of the ListBox.

2. Multiple: It enables the single as well as multiple selection of items.

Sample Code ASP.Net ListBox with SelectionMode

1. ASP.Net ListBox Control As Single SelectionMode

HTML Code:

```
<b>Select Item:</b>
   <br />
   <asp:ListBox ID="ListBox1" runat="server" AutoPostBack="true" OnSelectedIndexChanged="ListBox1_SelectedIndexChanged">
      <asp:ListItem Value="Item1">Item1</asp:ListItem>
      <asp:ListItem Value="Item2">Item2</asp:ListItem>
      <asp:ListItem Value="Item3">Item3</asp:ListItem>
   </asp:ListBox>
   <br />
   <asp:Label ID="lblResult" runat="server" />
```

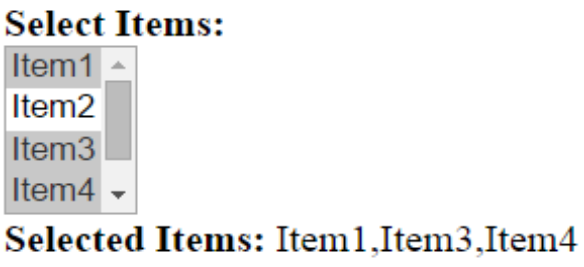
C# Code:

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
   {
      lblResult.Text = "<b>Slected Item: </b>" + "<br />" + "Text: " + ListBox1.SelectedItem.Text + "<br />" + "Value: " +
ListBox1.SelectedItem.Value;
   }
```

2. ASP.NET ListBox Control As Multiple SelectionMode

HTML Code:

```
<b>Select Items:</b>
   <br />
   <asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple"
AutoPostBack="true"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged">
      <asp:ListItem Value="Item1">Item1</asp:ListItem>
      <asp:ListItem Value="Item2">Item2</asp:ListItem>
      <asp:ListItem Value="Item3">Item3</asp:ListItem>
      <asp:ListItem Value="Item4">Item4</asp:ListItem>
      <asp:ListItem Value="Item5">Item5</asp:ListItem>
   </asp:ListBox>
   <br />
   <asp:Label ID="lblResult" runat="server" />
```



C# Code:

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
   {
      string strSelectedItems = string.Empty;

      for (int index = 0; index < ListBox1.Items.Count; index++)
      {
         if (ListBox1.Items[index].Selected == true)
         {
            strSelectedItems += ListBox1.Items[index].Text + ",";
         }
      }
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
    if (strSelectedItems != string.Empty)
    {
        strSelectedItems = strSelectedItems.Substring(0, strSelectedItems.Length - 1);
        lblResult.Text = "<b>Selected Items: </b>" + strSelectedItems;
    }
    else
    {
        lblResult.Text = "No Item Selected!!!";
    }
}
```

OR Using foreach:

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    string strSelectedItems = string.Empty;

    foreach (ListItem li in ListBox1.Items)
    {
        if (li.Selected == true)
            strSelectedItems += li.Text + ",";
    }

    if (strSelectedItems != string.Empty)
    {
        strSelectedItems = strSelectedItems.Substring(0, strSelectedItems.Length - 1);
        lblResult.Text = "<b>Selected Items: </b>" + strSelectedItems;
    }
    else
    {
        lblResult.Text = "No Item Selected!!!";
    }
}
```

CheckBoxList Control:

This control also works as checkbox as given above but it has some more properties like, it can have collection of check boxes, and one great advantage, it has database binding property.
OR
The CheckBoxList control provides a multiple-selection checked list. Like other list controls, CheckBoxList has an Items collection with members that correspond to each item in the list. To determine which items are selected, test the selected property of each item.

You can control the rendering of the list with the RepeatLayout and RepeatDirection properties. If RepeatLayout is Table, the list is rendered within a table. If it is set to Flow, the list is rendered without any table structure. By default, RepeatDirection is Vertical. Setting this property to Horizontal causes the list to be rendered horizontally.

The **ASP.Net CheckBoxList control** provides the functionality to display check box controls as List Items in the Web form. You can select the multiple checkbox items of CheckBoxList control by changing their checked state. The selected items of CheckBoxList return the Boolean type result as true or false that determine the checked state of an individual list item or multiple list items based on the C# code logic applied on it. If the list item is in **checked state** then it returns true and if it is in unchecked state then the checkbox control rendered as list item returns false. You can use the ASP.Net CheckBoxList web server control to get the user input as multiple items selected under a specific group. A simple example of CheckBoxList control is displaying a list of site categories to the user among them he can select multiple categories to subscribe for receiving weekly newsletter from the web site.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

The ASP.Net CheckBoxList control consists of following properties that are commonly used to get or set the values or user input. There are two sets of properties below; the first set includes the properties that enable you to get or set the values of list items of CheckBoxList and the second set includes the properties that enable you to customize its appearance:

AppendDataBoundItems: It allows you to add the list items of CheckBoxList statically along with the data-bound list items retrieved from any data source. It is a Boolean type property that accepts value as true or false for allowing or disallowing the append feature of list items.

AutoPostBack: It is also a Boolean type property that enables the control to send the web page to PostBack when user clicks the checkbox list item to check or uncheck it.

DataSource: This property allows you to specify the name of the data source dynamically.

DataSourceID: It accepts the ID of the DataSource control used to retrieve the data from the database.

DataMember: It accepts the name of one of the data table stored in the data source, if there is a collection of data tables in it. For example using SQL queries or stored procedures you can retrieve data in the form of multiple data tables that can be stored into the DataSet class object. This DataSet object can be used to bind the data to different types of Data-bound controls. The DataMember property enables you to specify the name of the table that you want to bind with the data-bound control.

DataTextField: It accepts the name of the Field/Column of the data table stored in the DataSource object that you want to display as the text content of the list item.

DataValueField: It accepts the name of the field of the data table stored in the DataSource object that you want to set as the value attribute of each list item. This value part does not appear as text content.

DataTextFormatString: It allows you to format the text property of the CheckBoxList item. For example you can use {0:C} to format the numeric value as currency. There are different types of formats such as {0:X}, {0:E}, {0:N}, {0:C}.

Layout Properties of ASP.Net CheckBoxList Control:

The **ASP.Net CheckBoxList control** also consists of **CSS styles** based properties and **Layout properties** that allow you to change the appearance and rendering behaviour of the control. You can change the background color of the control, set the border around the control along with its border color. You can specify the style of the border and its width using CSS units to display the bordered edges of CheckBoxList control. It also provides the properties to customize the font style to display the text content of the list items of the control. You can also use the CSS class styles to customize the look and feel as well as layout of the CheckBoxList control.

The Layout properties of CheckBoxList control include the following 3 properties:

RepeatColumns: It accepts the integer value that renders the layout of the CheckBoxList control with specified number of columns. 0 is the default value of RepeatColumns property that renders the list items as a single vertical list of checkbox items.

RepeatDirection: It allows you to set the rendering direction of list items of the CheckBoxList control. RepeatDirection property accepts one of the following predefined values:

- **Horizontal:** this value renders the list items in left to right manner. Each item renders at the right side of the previous until it reaches the specified number of RepeatColumns property value. Then it renders the next item in new line or row of the table based on the RepeatLayout property value.
- **Vertical:** this value renders the list items in top to bottom manner. Each items renders below the previous item in a new row cell each time. It applies some logic to render the vertical layout of the CheckBoxList items.

RepeatLayout: It enables you to specify the layout type of the CheckBoxList control. You can specify the RepeatLayout property using any one of the following predefined keywords:

- **Table:** It renders the list items using HTML table element. It generates the table cells and rows to display the CheckBoxList items using table layout.
- **Flow:** It renders the list items using HTML span element as a container and it uses the HTML
 tag to enter the line break between the checkboxes to generate new rows.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

- **UnorderedList:** It renders the list items using HTML <ul> element as a container and it uses the HTML <li> tag to contain checkbox items.
- **OrderedList:** It renders the list items using HTML <ol> element as a container and it uses the HTML <li> tag to contain checkbox items.

Note: "Table" is the default value of RepeatLayout property.

The UnorderedList and OrderedList layouts only support vertical layout.

The UnorderedList and OrderedList layouts do not support multi-column layouts.

Following is a list of CSS based properties of the ASP.Net CheckBoxList control that can be used to customize its look and feel:

BackColor: allows you to set the background color.

BorderColor: allows you to set the border color.

BorderStyle: allows you to set the predefined style of the border around the CheckBoxList. You can specify the border styles such as solid, dotted, dashed etc.

BorderWidth: allows you to set the width of the border using CSS units such as px, pt, cm etc.

CellPadding: It sets the cell padding for the table cells of the CheckBoxList control when rendered with RepeatLayout property as "Table".

CellSpacing: It sets the cell spacing for the table cells of the CheckBoxList control when rendered with RepeatLayout property as "Table".

CssClass: allows you to specify the name of the CSS class to customize the look and feel of CheckBoxList control.

Font Style Properties: the font style properties of CheckBoxList include font-size, family, bold, italics etc.

ForeColor: Sets the color of the text content of the list items.

TextAlign: It enables you to set the horizontal alignment of the text aligned to left or right direction.

CheckBoxList Control Examples:

- **CheckBoxList Control Sample**
- **Populate CheckBoxList Control Items Dynamically**
- **CheckBoxList Control OnSelectedIndexChanged Event**
- **CheckBoxList Control Selected Items**
- **CheckBoxList Control Layout Properties**

Note: Working with CheckBoxList Control is same as other list bound control like DropDownList & ListBox Control.

CheckBoxList Control Layout Example:

HTML Code:

```
<b>Select Languages:</b>
   <br />
  <asp:CheckBoxList ID="CheckBoxList1" runat="server" RepeatColumns="2" RepeatDirection="Horizontal"  RepeatLayout="Table"
CellPadding="5" CellSpacing="10">
      <asp:ListItem Text="Visual C#" Value="VC#" />
      <asp:ListItem Text="Visual Basic" Value="VB" />
      <asp:ListItem Text="Visual C++" Value="VC++" />
      <asp:ListItem Text="Visual F#" Value="VF#" />
      <asp:ListItem Text="Visual J#" Value="VJ#" />
  </asp:CheckBoxList>
   <asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
   <br /><br />
   <asp:Label ID="lblResult" runat="server" />
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

C# Code:

```
protected void btnSubmit_Click(object sender, EventArgs e)
   {
      string strSelectedItems = string.Empty;
      foreach (ListItem li in CheckBoxList1.Items)
      {
         if (li.Selected)
            strSelectedItems += li.Text + "/" + li.Value + ",";
      }
      if (strSelectedItems.Length > 0)
      {
         strSelectedItems = strSelectedItems.Substring(0, strSelectedItems.Length - 1);
         lblResult.Text = "<b>Selected Languages: </b>" + strSelectedItems;
      }
      else
      {
         lblResult.Text = "<b>No Language Selected</b>";
      }
   }
```

Select Languages:

Visual C# Visual Basic

Visual C++ Visual F#

Visual J#

Submit

Selected Languages: Visual C#/VC#,Visual F#/VF#,Visual J#/VJ#

ASP.Net CheckBoxList Validation using JavaScript:

In **ASP.Net the CheckBoxList control** is generally used to display the number of choices and users have to select the single or multiple features among multiple options available in the form of checkbox list items. By showing **multiple ListItems** as CheckBoxList you can provide the functionality to select one or more checkboxes from the group of options. This functionality requires a **JavaScript validation** to control the number of **selected checkbox list items** that are being checked by the user to specify their selected options. In this document we will learn how to display the multiple checkbox controls using ASP.Net CheckBoxList control and will apply the validation using JavaScript client end script to control the number of selections made by the user.

Following HTML Source code can be used to display the multiple checkbox controls on the web page using ASP.Net CheckBoxList control:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server">
   <asp:ListItem Value="Item 1" onclick="chkCount(this)" >Item 1</asp:ListItem>
   <asp:ListItem Value="Item 2" onclick="chkCount(this)" >Item 2</asp:ListItem>
   <asp:ListItem Value="Item 3" onclick="chkCount(this)" >Item 3</asp:ListItem>
   <asp:ListItem Value="Item 4" onclick="chkCount(this)" >Item 4</asp:ListItem>
   <asp:ListItem Value="Item 5" onclick="chkCount(this)" >Item 5</asp:ListItem>
</asp:CheckBoxList>
<asp:HiddenField ID="hiddenChkCount" runat="server" Value="0" />
```

Next we need a JavaScript function to validate the number of checkbox list items checked by the user to specify his selected options. The JavaScript validation function requires a resource that could store the count of number of checkbox items checked by the user. So we have used the **ASP.Net HiddenField control** to store the count of checked checkbox list items.

Following JavaScript client side validation function can be used to validate the multiple checkbox list items selection:

```
<script type="text/javascript">
      function chkCount(obj) {
         if (obj.checked == true) {
            if (document.getElementById('<%=hiddenChkCount.ClientID %>').value >= 3) {
               alert('You cannot select more than 3 items.');
               obj.checked = false;
            }
            else {
               document.getElementById('<%=hiddenChkCount.ClientID %>').value =
parseInt(document.getElementById('<%=hiddenChkCount.ClientID %>').value) + 1;
            }
         }
         else {
            document.getElementById('<%=hiddenChkCount.ClientID %>').value =
parseInt(document.getElementById('<%=hiddenChkCount.ClientID %>').value) - 1;
         }
      }
   </script>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

In the above JavaScript code we have created a simple logic to store the count of selected checkbox items. When user will click the checkbox list item it will call this client side function. First of all it will verify that whether the user has selected the checkbox or not. For checked action of checkbox item it will look for the count of selected controls from the HiddenField control. In this "if condition" you can specify the limit for the number of list items that can be selected by the user. After this "if condition" test, JavaScript code will increase or decrease the numeric value stored in the HiddenField control based on the user action.

ASP.Net CheckBoxList Select All using C#:

You can use a **single ASP.Net Checkbox control** to raise the server side event that can be handled by C# code for **selecting all** the **checkbox type list items** of the control. You can use the **C# code** to loop over the list items of the **CheckBoxList control** to change the **checked state** of all the list items that will render as checkbox controls. An independent checkbox control can be used to raise the server side event that will change the **selected state** of the list items of the CheckBoxList control.

Requirements for "Select All" Checkbox Control

The decision making Checkbox control that will be used to check all the checkbox list items of CheckBoxList control must be declared with **AutoPostBack property** having value as **"true"** to allow the control to raise the PostBack event of the web page. The other thing is to attach the server side **OnCheckedChanged event handler** to it for handling the click event that will occur at the time when user will click the checkbox control having text property value **"Select All"**.

You can use the following HTML code to display the list of checkbox controls using ASP.Net CheckBoxList control as a single group of options along with a separate Select All checkbox control:

```
<asp:CheckBox ID="CheckBox1"
        runat="server"
        AutoPostBack="True"
        OnCheckedChanged="CheckBox1_CheckedChanged"
        Text="Select All" />
    <br />
    <asp:CheckBoxList ID="CheckBoxList1" runat="server">
        <asp:ListItem Value="Item 1">Item 1</asp:ListItem>
        <asp:ListItem Value="Item 2">Item 2</asp:ListItem>
        <asp:ListItem Value="Item 3">Item 3</asp:ListItem>
        <asp:ListItem Value="Item 4">Item 4</asp:ListItem>
        <asp:ListItem Value="Item 5">Item 5</asp:ListItem>
    </asp:CheckBoxList>
```

The following C# code can be used to handle the **CheckedChanged** server side event handler to select all the list items rendered as checkbox control placed inside the CheckBoxList:

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
  {
    foreach (ListItem listitem in CheckBoxList1.Items)
    {
      listitem.Selected = CheckBox1.Checked;
    }
  }
```

In the above sample code we have used **C# foreach loop** over the List items collection of CheckBoxList control that provides the reference to each list item rendered as checkbox control. Inside the loop, we have used **Selected property** of the **ListItem class object** and **Checked property** of CheckBox control to change the selected state of each list item of CheckBoxList control same as the "Select All" checkbox control.

ASP.Net Select All CheckBoxList Items using JavaScript:

You can also the **JavaScript code to select all** the **list items** of **ASP.Net CheckBoxList control** without reloading the web page. In the previous example:**ASP.Net CheckBoxList Select All using C#** we learnt the use of server side **C# code** to select all the list items of CheckBoxList control that sends the web page to PostBack and executes the code specified for **CheckedChanged event handler** of checkbox. Here we will use client side JavaScript function that will select all the checkbox list items without sending the web page to PostBack. To understand the functionality we have created 5 list items of CheckBoxList control and added one more checkbox control other than CheckBoxList. We have added the separate Checkbox control for decision making that will call the JavaScript function to select all the list items of the CheckBoxList control that will render as multiple checkbox controls.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

You can use the following HTML code to display the group of checkbox list items using CheckBoxList control along with their decision making "Select All" checkbox control:

```
<asp:CheckBox ID="CheckBox1"
        runat="server"
        Text="Select All"
        onclick="checkAll(this);" />
    <br />
    <asp:CheckBoxList ID="CheckBoxList1" runat="server">
        <asp:ListItem Value="Item 1">Item 1</asp:ListItem>
        <asp:ListItem Value="Item 2">Item 2</asp:ListItem>
        <asp:ListItem Value="Item 3">Item 3</asp:ListItem>
        <asp:ListItem Value="Item 4">Item 4</asp:ListItem>
        <asp:ListItem Value="Item 5">Item 5</asp:ListItem>
    </asp:CheckBoxList>
```

In the above sample html markup code we have added the **checkbox onclick event**as an inline attribute that will call the JavaScript client side function. You can also attach the onclick event to the checkbox control programmatically using following C# code:

```
protected void Page_Load(object sender, EventArgs e)
  {
    CheckBox1.Attributes.Add("onclick", "checkAll(this);");
  }
```

Last we need the JavaScript code to execute the logic for selecting all the list items rendered as checkbox control with a single click. You can use the following JavaScript code to check all checkbox type list items:

```
<script type="text/javascript">
    function checkAll(obj1) {
        var checkboxCollection = document.getElementById('<%=CheckBoxList1.ClientID %>').getElementsByTagName('input');

  for (var i = 0; i < checkboxCollection.length; i++) {
    if (checkboxCollection[i].type.toString().toLowerCase() == "checkbox") {
        checkboxCollection[i].checked = obj1.checked;
    }
  }
}
</script>
```

RadioButtonList Control:

This control also works as RadioButton as given above but it has some more properties like, it can have collection of radio buttons, and one great advantage that it has database binding property.
OR
The RadioButtonList control provides the in-built functionality of creating a group of mutually exclusive items. It also allows the selection of a single option at a time.
OR
The RadioButtonList control provides a single-selection checked list. Like other list controls, RadioButtonList has an Items collection with members that correspond to each item in the list. To determine which items are selected, test the selected property of each item.

You can control the rendering of the list with the RepeatLayout and RepeatDirection properties. If the value of RepeatLayout is Table, the list will be rendered in a table. If it is set to Flow, the list will be rendered without any table structure. By default, the value of RepeatDirection is Vertical. Setting this property to Horizontal causes the list to be rendered horizontally.

The **ASP.Net RadioButtonList control** provides the functionality to display radio button controls as ListItems in the Web form. You can select the one radio button item of RadioButtonList control by changing the inactive state of the selected item. The selected item of RadioButtonList returns the Boolean type result as true or false that determines the checked state of an individual list item based on the C# code logic applied on it. If the list item is in **checked state** then it returns true and if it is in unchecked state then the radio button rendered as list item returns false. You can use the ASP.Net RadioButtonList web server control to get the user input as a single selection under a specific group of multiple mutually exclusive options. Some simple examples of RadioButtonList control are displaying a list of titles prefixes for name such as "Mr", "Ms", "Mrs" or "Dr." etc. Users can select only one Title as their Name prefix. Similarly you can

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

display the gender options using RadioButtonList control such as "Male" or "Female" and user can choose one. The ASP.Net RadioButtonList control provides the inbuilt functionality to display the radio button list items as mutually exclusive set of choices that allow the users to select only one item among the multiple options.

The ASP.Net RadioButtonList control consists of following properties that are commonly used to get or set the values or user input. There are two sets of properties below; the first set includes the properties that enable you to get or set the values of list items of RadioButtonList and the second set includes the properties that enable you to customize its appearance:

AppendDataBoundItems: It allows you to add the list items of RadioButtonList statically along with the data-bound list items retrieved from any data source. It is a Boolean type property that accepts value as true or false for allowing or disallowing the append feature of list items.

AutoPostBack: It is also a Boolean type property that enables the control to send the web page to PostBack when user clicks the radio button list item to select it.

DataSource: This property allows you to specify the name of the data source dynamically.

DataSourceID: It accepts the ID of the DataSource control used to retrieve the data from the database.

DataMember: It accepts the name of one of the data table stored in the data source, if there is a collection of data tables in it. For example using SQL queries or stored procedures you can retrieve data in the form of multiple data tables that can be stored into the DataSet class object. This DataSet object can be used to bind the data to different types of Data-bound controls. The DataMember property enables you to specify the name of the table that you want to bind with the data-bound control.

DataTextField: It accepts the name of the Field/Column of the data table stored in the DataSource object that you want to display as the text content of the list item.

DataValueField: It accepts the name of the field of the data table stored in the DataSource object that you want to set as the value attribute of each list item. This value part does not appear as text content.

DataTextFormatString: It allows you to format the text property of the RadioButtonList item. For example you can use {0:C} to format the numeric value as currency. There are different types of formats such as {0:X}, {0:E}, {0:N}, {0:C}.

Layout Properties of ASP.Net RadioButtonList Control:

The **ASP.Net RadioButtonList control** also consists of **CSS styles** based properties and **Layout properties** that allow you to change the appearance and rendering behaviour of the control. You can change the background color of the control, set the border around the control along with its border color. You can specify the style of the border and its width using CSS units to display the bordered edges of RadioButtonList control. It also provides the properties to customize the font style to display the text content of the list items of the control. You can also use the CSS class styles to customize the look and feel as well as layout of the RadioButtonList control.

The Layout properties of RadioButtonList control include the following 3 properties:

RepeatColumns: It accepts the integer value that renders the layout of the RadioButtonList control with specified number of columns. 0 is the default value of RepeatColumns property that renders the list items as a single vertical list of radio button items.

RepeatDirection: It allows you to set the rendering direction of list items of the RadioButtonList control. RepeatDirection property accepts one of the following predefined values:

1. Horizontal: this value renders the list items in left to right manner. Each item renders at the right side of the previous until it reaches the specified number of RepeatColumns property value. Then it renders the next item in new line or row of the table based on the RepeatLayout property value.

2. Vertical: this value renders the list items in top to bottom manner. Each items renders below the previous item in a new row cell each time. It applies some logic to render the vertical layout of the RadioButtonList items.

RepeatLayout: It enables you to specify the layout type of the RadioButtonList control. You can specify the RepeatLayout property using any one of the following predefined keywords:

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

1. Table: It renders the list items using HTML table element. It generates the table cells and rows to display the radiobuttonlist items using table layout.

2. Flow: It renders the list items using HTML span element as a container and it uses the HTML
 tag to enter the line break between the radio button list items to generate new rows.

"Table" is the default value of RepeatLayout property.

Following is a list of CSS based properties of the ASP.Net RadioButtonList control that can be used to customize its look and feel:

BackColor: allows you to set the background color.

BorderColor: allows you to set the border color.

BorderStyle: allows you to set the predefined style of the border around the RadioButtonList. You can specify the border styles such as solid, dotted, dashed etc.

BorderWidth: allows you to set the width of the border using CSS units such as px, pt, cm etc.

CellPadding: It sets the cell padding for the table cells of the RadioButtonList control when rendered with RepeatLayout property as "Table".

CellSpacing: It sets the cell spacing for the table cells of the RadioButtonList control when rendered with RepeatLayout property as "Table".

CssClass: allows you to specify the name of the CSS class to customize the look and feel of RadioButtonList control.

Font Style Properties: the font style properties of RadioButtonList include font-size, family, bold, italics etc.

ForeColor: Sets the color of the text content of the ListItems

TextAlign: It enables you to set the horizontal alignment of the text aligned to left or right direction.

RadioButtonList Control Examples:

- **RadioButtonList Control Sample**
- **Populate RadioButtonList Control Items Dynamically**
- **RadioButtonList Control OnSelectedIndexChanged Event**
- **RadioButtonList Control Layout Properties**

Note: Working with RadioButtonList Control is same as other list bound control like CheckBoxList Control.

RadioButtonList Control Example:

HTML Code:

```
<b>Select Option:</b>
    <asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="true"
OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged">
        <asp:ListItem Text="Option1" Value="Opt1" />
        <asp:ListItem Text="Option2" Value="Opt2" />
        <asp:ListItem Text="Option3" Value="Opt3" />
    </asp:RadioButtonList>
    <asp:Label ID="lblResult" runat="server" />
```

Select Option:
Option1
Option2
Option3
Selected Option: Option2/Opt2

C# Code:

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
  {
    lblResult.Text = "<b>Selected Option: </b>" + RadioButtonList1.SelectedItem.Text + "/" +
RadioButtonList1.SelectedItem.Value;
  }
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

BulletedList Control:

This control is used to display list of items with bullets. We can connect this control to database to display list items. The ASP.Net BulletedList server control provides the functionality to render a list of items similar to HTML ordered list or unordered list tags that show a bullet or numbers in front of each item.
OR
The bulleted list control creates bulleted lists or numbered lists. These controls contain a collection of ListItem objects that could be referred to through the Items property of the control.
OR
The BulletedList control is used to create a list of items formatted with bullets. To specify the individual list items that you want to appear in a BulletedList control, place a ListItem object for each entry between the opening and closing tags of the BulletedList control.

The **ASP.Net BulletedList web server control** provides the functionality to generate ordered or unordered list of items similar to **HTML ol or ul tags**. Using BulletedList control of ASP.Net 2.0 you can display the static list of items by adding the list items manually or you can bind it to any dynamic data source connected to any type of database for example SQL server, MS Access etc. You can customize the look and appearance of BulletedList control using its different types of properties. The ASP.Net BulletedList control is a combination of HTML ul and ol tag that renders with HTML markup based on the value of BulletStyle property. The BulletStyle property enables you to set the bullets, numbers, alphabets, or roman numbers bulleted list item style.

Following are the properties of ASP.Net BulletedList control that can be used to customize its look and behaviour:

1. AppendDataBoundItems: it allows you to add the list items statically along with the data-bound list items retrieved from any data source. It is a Boolean type property that accepts value as true or false for allowing or disallowing the append feature of list items.

2. BulletImageUrl: it accepts the relative or absolute path for the custom image that you want to display in place of bullet symbol. This property works only if the **BulletStyle property** value is set to "**CustomImage**".

3. BulletStyle: it enables you to set the style of bullet type that you want to display in front of each list item. It supports following predefined values to declare the bullet style:

- Circle
- CustomImage
- Disc
- LowerAlpha
- LowerRoman
- NotSet
- Numbered
- Square
- UpperAlpha
- UpperRoman

4. DisplayMode: The BulletedList control provides three types of modes that can be used to set the rendering type of its list items such as:

- Text
- HyperLink
- LinkButton

For hyperlink and LinkButton display modes each list item requires two properties: the **Text** property and the **Value** property.

5. FirstBulletNumber: This property enables you to change the starting point of the ordered list items. It works with ordered type BulletStyle property values such as LowerAlpha, LowerRoman, Numbered, UpperAlpha and UpperRoman. You can specify the numeric value for FirstBulletNumber for example 5, 10 etc. It will start the ordered bulleted list numbering from the specified FirstBulletNumber value.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

BulletedList Examples:

BulletedList Example using Square BulletStyle

```
<b>List of Companies:</b>
  <asp:BulletedList ID="BulletedList1" runat="server" BulletStyle="Square" >
     <asp:ListItem Text="Microsoft" />
     <asp:ListItem Text="Infosys"  />
     <asp:ListItem Text="Google" />
  </asp:BulletedList>
```

List of Companies:

- Microsoft
- Infosys
- Google

BulletedList Example using CustomImage BulletStyle

The "**CustomImage**" value for **BulletStyle property** of ASP.Net BulletedList control works with **BulletImageUrl property**. The "CustomImage" value directs the control to render with the image specified using BulletImageUrl property as a bullet in front of each list item.

```
<b>List of Companies:</b>
  <asp:BulletedList ID="BulletedList1" runat="server" BulletStyle="CustomImage" BulletImageUrl="~/Images/Star.png">
     <asp:ListItem Text="Microsoft" />
     <asp:ListItem Text="Infosys"  />
     <asp:ListItem Text="Google" />
</asp:BulletedList>
```

List of Companies:

- Microsoft
- Infosys
- Google

ASP.Net BulletedList Control with FirstBulletNumber Property:

```
<b>List of Companies:</b>
  <asp:BulletedList ID="BulletedList1" runat="server" BulletStyle="LowerAlpha"
  FirstBulletNumber="1">
     <asp:ListItem Text="Microsoft" />
     <asp:ListItem Text="Infosys"  />
     <asp:ListItem Text="Google" />
  </asp:BulletedList>
```

List of Companies:

a. Microsoft
b. Infosys
c. Google

ASP.Net BulletedList Control with DisplayMode Property

The **DisplayMode property** of **ASP.Net BulletedList control** enables you to change the style and behaviour of the list item. By default display mode of the list items of BulletedList control render as static text. The DisplayMode property provides predefined values that can be used to specify the mode of its list items. Following are the 3 Display Mode values that can be used to render the list items of BulletedList control:

1. Text: It renders the list items as static text. You can set the BulletStyle, BulletImageUrl or FirstBulletNumber property to change the appearance or bullets of the text type list items. The "**Text**" value for DisplayMode property does not support the "**Value**" attribute of ListItem of BulletedList control.

2. HyperLink: It renders the list items as clickable links. The items are generated as HyperLink server controls rendered using HTML <a> tag. Users can the text item to navigate to other pages. The "HyperLink" value for DisplayMode property supports "Value" attribute of ListItem of BulletedList control that enables you to specify the URL for the target web page. The HyperLink based list items of BulletedList control works at client side, no server side event such as click or selection can be attached to it.

3. LinkButton: It also renders the list items as clickable links. The items are generated as LinkButton server controls rendered using HTML <a> tag along with onclick event having javascript:__doPostBack function that sends the page to PostBack and executes the associated events such as Click or SelectedIndexChanged. The "LinkButton" value for DisplayMode property also supports "Value" attribute of ListItem of BulletedList control.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

BulletedList Control Examples using DisplayMode Property

Example 1: Using DisplayMode="Text"

```
<b>List of Companies:</b>
  <asp:BulletedList ID="BulletedList1" runat="server" DisplayMode="Text">
     <asp:ListItem Text="Microsoft" />
     <asp:ListItem Text="Infosys"  />
     <asp:ListItem Text="Google" />
  </asp:BulletedList>
```

Example 2: Using DisplayMode="HyperLink"

```
<b>List of Companies:</b>
   <asp:BulletedList ID="BulletedList1" runat="server" DisplayMode="HyperLink" Target="_blank">
      <asp:ListItem Text="Microsoft" Value="http://www.microsoft.com" />
      <asp:ListItem Text="Infosys" Value="http://www.infosys.com"  />
      <asp:ListItem Text="Google" Value="http://www.google.com" />
   </asp:BulletedList>
```

Example 3: Using DisplayMode="LinkButton" with Click Event

```
<b>List of Companies:</b>
   <asp:BulletedList ID="BulletedList1" runat="server" DisplayMode="LinkButton" OnClick="BulletedList1_Click" >
      <asp:ListItem Text="Microsoft" />
      <asp:ListItem Text="Infosys"  />
      <asp:ListItem Text="Google" />
   </asp:BulletedList>
   <asp:Label ID="lblResult" runat="server" />
```

C# Code:

```
protected void BulletedList1_Click(object sender, BulletedListEventArgs e)
   {
      lblResult.Text = "<b>Selected Company: </b>" + BulletedList1.Items[e.Index].Text;
   }
```

Image Control:

The image control is used for displaying images on the web page, or some alternative text, if the image is not available.
OR
The Image control displays the image on a web page defined by its ImageUrl property.
OR
This control is used to display the image on the form. Like if you open any community website and have look your image means they have used image control. Using ImageUrl property we can change the image.
OR
The Image server control displays the specified image on the web page. It is also similar to the HTML <img /> tag but contains advanced properties that cannot be applied to HTML img tag without using CSS properties.

The **ASP.Net Image control** provides the functionality to display images on ASP.Net web page. The image server control renders using **HTML img tag** in the web browser. As a web server control it can be accessed at server side that enables you to set the images dynamically using C# code or by specifying the image URL retrieved from SQL database table. It provides properties similar to HTML img tag that allow you to set the path of image file that you want to display on web page along with its alignment and alternate text properties that render as attributes of HTML img tag in the web browser. You can get or set these properties at design time as well as using C# code programmatically. The ASP.Net image control allows you to display the images only for design purposes or as a part of the page content but does not support any server events to execute the server side code if a user clicks on it or moves the mouse over it.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Microsoft .NET Solutions
Robust, Scalable & Secure Applications
Developed for Diverse Business Needs
Develop your Microsoft skills and knowledge through great training
NARESH i technologies
Microsoft .net

Properties of ASP.Net Image Control

Following are the commonly used properties of ASP.Net image control that enable you to set the image and various types of text based information about the associated image for better accessibility on different types of web browsers:

1. AlternateText: You can specify the text value for this property that appears as tooltip for the displayed image in the web browsers. All web browsers do not support this property to display the text as tooltip. The web browsers display this alternate text value if the specified image does not load for any reason like unreachable or unavailable at that time. This property renders as alt attribute of img tag.

2. DescriptionUrl: You can specify the URL of the web page containing the full description regarding the image.

3. GenerateEmptyAlternateText: It is a Boolean type property that allows or disallows to generate the empty alt attribute while rendering the image control.

4. ImageAlign: It enables you to specify the alignment of image as a part of inline content or web page elements.

5. ImageUrl: You can specify the URL of the image that you want to display on the web page.

6. ToolTip: It enables you to specify the tooltip text for the image control. The tooltip property generates the title attribute of img tag that appears as tooltip in some browsers.

Apart from these properties there some other CSS based properties that enable you to set the border, background color, height and width of the image. The CSS based properties include: BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Height and Width that allow you to customize the appearance of image.

Image Control Example:

```
<asp:Image ID="Image1" runat="server" ImageUrl="~/Images/Ads1.jpg" AlternateText="Image" ToolTip="This is an image"
Width="300" Height="300" BorderStyle="Dashed" BorderWidth="5" BorderColor="Red"/>
```

Hyperlink Control:

This control is used to link any other page. Like when user clicks on Home page it navigates to home.aspx page, this is because of hyperlink feature. The HyperLink control is like the HTML <a> element.

Now you can add the server side Hyperlinks on the ASP.Net web page using HyperLink server control that renders the HTML anchor <a> tag. It supports few more properties than HTML anchor tag.

The **ASP.Net HyperLink control** renders as a link on the page that enables the users to navigate from one page to the other by clicking it. It has different set of properties that provides the functionality to specify the text for HyperLink control and navigation URL to which you want to navigate the user. The HyperLink control is the server control of ASP.Net that renders as a client side link using **HTML anchor a tag**. When it renders in the web page accessed via any web browser it works similar to **HTML a tag** and possesses same attributes such as href, target etc. according to the properties defined for it. The HyperLink control of ASP.Net has one extra feature that allows you to specify the path to the image which you want to display as link instead of text link.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

HyperLink Control Properties

The HyperLink control also contains two types of properties that enable you to set the CSS style of link to customize its appearance and the other properties set the behaviour of hyperlink. Following are the properties of HyperLink control that allow you to set the behaviour of link:

1. NavigateUrl: You can specify the destination URL of the page to which you want to move the user.

2. ImageUrl: It accepts the URL of the image that you want to display as image link.

3. Target: It allows you to specify the target frame in which you want to open the hyperlinked web page URL.

4. Text: It accepts the text string to display as text link.

5. ToolTip: It allows you to display the tooltip for the text link when user hovers over the link text.

The CSS based properties of ASP.Net HyperLink control includes: BackColor, BorderColor, BorderStyle, BorderWidth, CssClass, Font-Bold, Font-Italic, Font-Names, Font-Overline, Font-Size, Font-StrikeOut, Font-Underline, ForeColor, Height and Width. These properties render as inline CSS styles that enable you to customize the appearance of Hyperlinked text.

Hyperlink Control Example:

1. **Hyperlink Control as text link**

```
<asp:HyperLink ID="HyperLink1" runat="server" Text="ASP.NET AJAX" ToolTip="Go to ASP.NET AJAX"
NavigateUrl="http://www.asp.net/ajax" Target="_blank"/>
```

2. **Hyperlink Control as image link**

```
<asp:HyperLink ID="HyperLink1" runat="server" ImageUrl="~/Images/AJAXLogo.jpg" ToolTip="Go to ASP.NET AJAX"
NavigateUrl="http://www.asp.net/ajax" Target="_blank" ImageWidth="150" ImageHeight="100" />
```

FileUpload Control:

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.
OR
This control is used to upload any type of file. After writing some code behind it only selects specified format of files, like we can set it to select only jpg and gif files. It provides the user a way to send the file to any other computer using internet/intranet.

The **ASP.Net FileUpload control** is new web server control in ASP.Net 2.0 framework that provides the advanced functionality to the users to send a file from their computer system to the online web server. It renders using HTML input type file control that appears as combination of two controls such as TextBox control and a Button control. The TextBox control sends the path of the file selected from the user's computer to the server whereas button control provides the functionality to open a **file-navigation dialog box**. Users can navigate between the directories and files available on their system and select a file that they want to upload to the web server. The security feature of FileUpload control does not all you to specify the file path dynamically and users are also not allowed to type-in the file path in the textbox field available with FileUpload control. Users have to browse the file using file-navigation dialog box to select it for uploading.

The ASP.Net FileUpload control provides different types of properties that enable you to examine the characteristics of the uploaded file and save it to a specific location on the web server.

Following are the properties that you can use in your **C# server code** to read the binary stream of the uploaded file and save it with new or same filename at some specific location on the web server:

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

FileBytes: It provides the access to the bytes array generated from the stream of file selected by the user to upload.
FileContent: It provides the System.IO.Stream object that points to the file specified for uploading. You access all the methods and properties Stream class that allows you to read or write the bytes array stream of the file that is being uploaded by the FileUpload control.
FileName: It returns the name of the File selected by the user to upload using FielUpload control.
SaveAs: The SaveAs method of fileupload control provides the functionality to save the uploaded file at the location specified in the code to write it on the server disk.
PostedFile: The PostedFile provides as object of HttpPostedFile class that points to the single file that have been uploaded using FileUpload control. It provides the access to the properties such as ContentLength, ContentType, FileName, InputStream and SaveAs method.
The FileUpload control of ASP.Net standard web server controls renders using following HTML markup code on the web page:

ASP.Net Markup

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

HTML Code Rendered in Web Browser

```
<input type="file" name="FileUpload1" id="FileUpload1" />
```

Note: The ASP.Net FileUpload control caches the entire contents of the selected file from the client's computer system to the server memory after finishing the caching process it move forward to execute the code specified in the web page.

ASP.Net FileUpload PostedFile Property:

The **PostedFile property** of **ASP.Net FileUpload control** provides an object of **HttpPostedFile class** that points to the file uploaded by the control. The FileUpload control does not starts the file caching on the server automatically it need a specific code to upload the file and save it on the web server hard disk. The PostedFile property serves the methods and properties to send the file to the server and save it there at the specified location. You can provide an ASP.Net Button control to enable the user for uploading the selected file when he clicks on the button. You can write the click event handler code in the code behind file where you can handle the click event of the button control and add the code for uploading the file.
You can sue the following properties of the PostedFile property of ASP.Net **FileUpload control** that provides the HttpPostedFile class object and its member properties and methods to send the file from user's machine to the online web server:
ContentLength: It returns the integer type value that gets the value of size of uploaded file in bytes.
ContentType: It returns the string value as MIME type of the uploaded file content e.g.: image/gif, image/jpg, audio/mpeg etc.
FileName: It returns the name of the uploaded file along with its file extension such as "filename.gif", "filename.jpg" etc.
InputStream: The InputStream property of FileUpload control provides the object of **System.IO.Stream class** that points to the uploaded file and prepares it for reading its content in the form of **bytes array**.
SaveAs: The SaveAs method of PostedFile property enables you to save the uploaded file at a specific location on the web server.
Before implementing the above discussed member properties of PostedFile property of ASP.Net FileUpload control you must use its **HasFile property** that returns the **Boolean type** result as true/false. The HasFile property ensures that whether the user has selected any file for uploading or not. If the HasFile property returns true then you can execute the **SaveAs method of PostedFile property** to save the file on web server. If it returns false then you can display the warning the message to the user that the FileUpload control has no file specified for uploading.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net File Upload Example using C# Code:

HTML Code:

```
<b>Upload File:</b>
<asp:FileUpload ID="FileUpload1" runat="server" />
<br />
<asp:Button ID="btnUpload" runat="server" Text="Upload" OnClick="btnUpload_Click" />
<br />
<asp:Label ID="lblStatus" runat="server" />
```

C# Code:

```
using System.IO;
protected void btnUpload_Click(object sender, EventArgs e)
  {
    if (FileUpload1.HasFile)
    {
      try
      {
        string fileName = Path.GetFileName(FileUpload1.PostedFile.FileName);

        string serverFolder = Server.MapPath("Uploaded\\");
        if (!Directory.Exists(serverFolder))
          Directory.CreateDirectory(serverFolder);

        string serverFilePath = serverFolder + fileName;

        FileUpload1.SaveAs(serverFilePath);
        lblStatus.Text = "<b style='color:green'>File has been uploaded successfully!!!</b>";
      }
      catch (Exception)
      {
        lblStatus.Text = "<b style='color:red'>Some problem occurred while uploading the file. Please try after some time.</b>";
      }
    }
    else
    {
      lblStatus.Text = "<b style='color:red'>Please select a file to upload!!!</b>";
    }
  }
```

ASP.Net FileUpload Type & Size Limit Example:

Create an example of FileUpload Control to upload a file of any image type and size must be less than 2MB only.

HTML Code:

```
<b>Upload Image File:</b>
<asp:FileUpload ID="FileUpload1" runat="server" />
<br />
<asp:Button ID="btnUpload" runat="server" Text="Upload" OnClick="btnUpload_Click" />
<br />
<asp:Label ID="lblStatus" runat="server" />
```

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

C# Code:

```
using System.IO;
protected void btnUpload_Click(object sender, EventArgs e)
  {
    if (FileUpload1.HasFile)
    {
      try
      {
        string fileType = FileUpload1.PostedFile.ContentType;

        if (fileType == "image/jpeg" ||
          fileType == "image/jpg" ||
          fileType == "image/gif" ||
          fileType == "image/png" ||
          fileType == "image/bmp")
        {
          int fileSize = FileUpload1.PostedFile.ContentLength;

          if (fileSize <= 2097152) //Max Size: 2MB
          {
            string fileName = Path.GetFileName(FileUpload1.PostedFile.FileName);

            string serverFolder = Server.MapPath("Uploaded\\");
            if (!Directory.Exists(serverFolder))
              Directory.CreateDirectory(serverFolder);

            string serverFilePath = serverFolder + fileName;

            FileUpload1.SaveAs(serverFilePath);

            lblStatus.Text = "<b style='color:green'>File has been uploaded successfully!!!</b>";
          }
          else
          {
            lblStatus.Text = "<b style='color:red'>Please select a file of max size 2MB only.</b>";
          }
        }
        else
        {
          lblStatus.Text = "<b style='color:red'>Please select an image file only (Ex:*.jpg/*.jpeg/*.gif/*.png/*.bmp)</b>";
        }
      }
      catch (Exception)
      {
        lblStatus.Text = "<b style='color:red'>Some problem occurred while uploading the file. Please try after some time.</b>";
      }
    }
    else
    {
      lblStatus.Text = "<b style='color:red'>Please select a file to upload!!!</b>";
    }
  }
```

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Note: By Default, ASP.NET supports max size of 4MB only file request to be uploaded on server but it can be increase or decrease the size limit by providing the configuration settings using maxRequestLength attribute of httpRuntime section in an application's configuration file i.e. Web.config as following:

Web.config: Set The Max Size Limit: 8MB (8388608 Bytes)

```
<configuration>
<system.web>
<httpRuntime maxRequestLength="8388608" targetFramework="4.5" />
</system.web>
</configuration>
```

ASP.NET FileUpload Control with Multiple Files Upload Example:

Since HTML 5 provides support for multiple files upload from browsers, ASP.NET 4.5 has provided new methods and properties to support multiple files upload.

Following are the new properties of FileUpload control that support multiple files upload.

AllowMultiple: Specifies that FileUpload control should allow uploading multiple files. Of course this is possible only when Browser supports that feature.

HasFiles: Returns true if any files have been uploaded.

PostedFiles: Represents all files uploaded from client.

Example:

HTML Code:

```
<b>Upload Files:</b>
  <asp:FileUpload ID="FileUpload1" runat="server" AllowMultiple="true" />
  <br />
  <asp:Button ID="btnUpload" runat="server" Text="Upload" OnClick="btnUpload_Click" />
  <br />
  <asp:Label ID="lblStatus" runat="server" />
```

C# Code:

```
using System.IO;
protected void btnUpload_Click(object sender, EventArgs e)
  {
    if (FileUpload1.HasFiles)
    {
      try
      {
        string serverFolder = Server.MapPath("Uploaded\\");
        if (!Directory.Exists(serverFolder))
          Directory.CreateDirectory(serverFolder);

        foreach (HttpPostedFile file in FileUpload1.PostedFiles)
        {
          string serverFilePath = serverFolder + Path.GetFileName(file.FileName);

          FileUpload1.SaveAs(serverFilePath);
        }
        lblStatus.Text = "<b style='color:green'>" + FileUpload1.PostedFiles.Count + " Files has been uploaded successfully!!!</b>";
      }
      catch (Exception)
      {
        lblStatus.Text = "<b style='color:red'>Some problem occurred while uploading the file. Please try after some time.</b>";
      }
    }
    else
    {
      lblStatus.Text = "<b style='color:red'>Please select files to be uploaded!!!</b>";
    }
  }
```

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Literal Control:

A Literal control is used to display text. The developer cannot apply a style to a literal control.
OR
This control is used to display text directly to client browsers. It works as container and we can't apply any style to it.
OR
The Literal server controls provides the functionality to set the inner text or HTML without adding any HTML control on the ASP.Net web page. You can dynamically change the text of Literal control using server code.

The **ASP.Net Literal Control** provides the functionality to reserve the space on the web page to display the static text. It is similar to the **ASP.Net Label control** that enables you to display the text on the page. You can display the text by specifying it in the HTML source code of Literal control or programmatically by assigning the text value to its **Text property** using C# or VB code at server side. Unlike Label control the Literal control does not provide the inbuilt CSS based style properties to customize the look and appearance of the text displayed by it. Instead it allows you to display the plain text as well as HTML formatted text and other client side scripts. You can also display the user input on the web page by assigning that value to the Literal control programmatically. As the Literal control can render scripts also it is recommended to validate the user input before rendering it dynamically using Literal control as the user input may contain malicious scripts that could be vulnerable.

Properties of Literal Control

Following are the two main properties of the ASP.Net Literal control that enable you to control its output behaviour while displaying the text on the web page dynamically:

1. Mode: It accepts the predefined enumeration values as Literal modes that enable you to control the rendering behaviour of the content specified to be displayed on the web page.

2. Text: You can specify the content to the Text property of the Literal control that you want to display on the web page. The Text property can be declared either in the HTML markup code block of the Literal control or programmatically in the code behind.

Sample Code for ASP.Net Literal Control

```
<asp:Literal ID="Literal1"
   runat="server"
   Text="This is a literal Control.">
</asp:Literal>
```

In the above sample we have specified the text to display in the HTML code of the Literal control.

C# Sample Code

```
Literal1.Text = "Text specified dynamically.";
```

You can also assign the text value programmatically at page load or any event to the Text property of the Literal control as shown in the above sample code.

ASP.Net Literal Mode Property:

The **Mode property** of **ASP.Net Literal Control** enables you to direct the control for customizing the rendering behaviour of the specified text that you want to display on the web page. The Mode property accepts the predefined enumeration value that renders the output text according to the specified value. It enables you to display the HTML formatted text as the plain text containing encoded markup tags. You can also display the HTML formatted text as it is. The Literal control supports the dynamic rendering of text value assigned at the server side that may be retrieved from the user input. Using the Mode property you can encode the scripts before rendering them on the web page to prevent the security vulnerabilities.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

ASP.Net Literal Mode Property Values

The Mode property supports the predefined set of values stored as enumeration of LiteralMode. Each value controls the Literal output in a different way to render the specified text on the web page. Following are the three types of Literal Modes:

1. Encode: It converts the specified text content to the HTML encoded string. It converts the & to & < to < and > to >.

2. PassThrough: It does not modify the specified content and renders it by applying the HTML formatting or scripts specified in it.

3. Transform: It removes the markup tags of the unsupported markup languages when the Literal control renders on the HTML or XHTML compliance web browsers. If the specified content contains the HTML formatting then it renders the text by applying the HTML formats without modifying the content.

Sample Code for Literal Control with Mode Property

```
<asp:Literal ID="Literal1" runat="server"
Text="This is a literal Control.<br /><string>This is a second text line.</string>"> </asp:Literal>
<br /><br />
<asp:Literal ID="Literal2" runat="server" Mode="Encode"
Text="This is a literal Control.<br /><b>This is a second text line.<b>">
</asp:Literal>
<br /><br />
<asp:Literal ID="Literal3" runat="server" Mode="PassThrough"
Text="This is a literal Control.<br /><b>This is a second text line.</b>">
</asp:Literal>
<br /><br />
<asp:Literal ID="Literal4" runat="server" Mode="Transform"
Text="<transform>This is a literal Control.<br /><b>This is a second text line.</b><transform>">
</asp:Literal>
```

In the above sample code the second Literal control we have specified the Mode as **"Encode"** that will convert the content to the HTML encoded content. It will not render the specified HTML formatting while rendering the content on the web page.

For the third Literal control we have specified the **"PassThrough"** Mode that will render the specified content by applying the HTML formats coded in the text.

In the last Literal control we have specified the Mode property value as **"Transform"**. It will remove the unsupported markup tags from the specified text such as <transform> and </transform>. The transform mode will render the HTML formatting specified in the text without modifying its content on the web page. The "Transform" mode of the Literal control is the default behaviour that will render the first Literal control also be removing the unsupported markup tag such as <string> and </string> but it will render the HTML
 tag while generating the output.

HiddenField Control:

This control enables a developer to store a non-displayed value.
OR
This control gives the developer a way to store information without displaying it on the page or say to user. Mostly it is used to store the states.
OR
ASP.Net server controls also include the HiddenField control that provides the server side functionality similar to the HTML input type hidden control. This control does not render its physical appearance on the web page but holds the values inside the HTML markup.
The **ASP.Net HiddenField Control** provides the functionality to store the values without displaying them on the web page. It enables you to store the information that needs to be persisted across the post backs to the server without losing the data. The HiddenField control in ASP.Net rendered as an **HTML input field** as **hidden type** that does not display its physical appearance on the web page. In ASP.Net usually view state, session state or cookies are used to store the persistent information to maintain the state of the web page. If these methods are not available then you can use the HiddenField control to maintain the data persistency. The functionality of hidden field is very much similar to view state management feature of ASP.Net.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

HiddenField Control Properties and Events

It has following properties that can be used to utilize its functionality on ASP.Net web page:

1. Value: It allows you to get or set the data stored in the control.

2. Visible: It provides the functionality to enable or disable the client side rendering of hidden field control. If you will set the value of "Visible" property to "false" then the view state of the associated web page will take the responsibility to maintain the data stored in the hidden field control.

You can use the following event of hidden field control for detecting changed data:

OnValueChanged Event: It is raised when the value of hidden field control changes between the postbacks to the server. It allows you to provide a custom routine event handler to detect the changed data.

Security and HiddenField Control

The HiddenField control renders with **value property** that holds the information in it at the client side. Users can view the hidden fields and the value stored in them by viewing the page's HTML source code. That is why the HiddenField is not recommended to store the sensitive information like passwords or credit card information.

HiddenField Control Examples:

Sample Code for ASP.Net HiddenField Control

HTML Code

```
<asp:Label ID="Label1" runat="server">
      </asp:Label>

      <asp:Button ID="Button1"
         runat="server"
         Text="Click It"
         OnClick="Button1_Click" />

      <asp:HiddenField ID="HiddenField1"
         runat="server"
         Value="0" />
```

C# Code

```
protected void Page_Load(object sender, EventArgs e)
   {
      if (!IsPostBack)
         Label1.Text = string.Format("Clicked {0} times", HiddenField1.Value);
   }
   protected void Button1_Click(object sender, EventArgs e)
   {
      HiddenField1.Value = (Convert.ToInt32(HiddenField1.Value) + 1).ToString();

      Label1.Text = string.Format("Clicked {0} times", HiddenField1.Value);
   }
```

In the above sample code we have used the HiddenField control to store the click count of button control. When user will hit the button it will execute the server click handler that will increment the value stored in the ASP.Net HiddenField control.

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

PlaceHolder Control:

This control is used to place new controls dynamically or say at run time. It is simply a place of control which only results at run time.
OR
The PlaceHolder control adds an empty container on the ASP.Net page that enables you to add the user controls at the run-time dynamically.
OR
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 **ASP.Net PlaceHolder control** provides the functionality to reserve a location on ASP.Net web page that can be further used for adding child controls dynamically. You can change the control hierarchy of web page using this amazing **PlaceHolder control** that enables you to add and render the ASP.Net server controls dynamically at the runtime. The ASP.Net PlaceHolder control does not appear as a physical element on the web page when it loads. Instead it holds a particular location on the page that expands to adjust the new child control added dynamically into it. You can add a single or multiple controls as a collection of controls to the PlaceHolder control. The **Controls collection** property of PlaceHolder control provides access to the methods such as **Add**, **AddAt**, **Remove**, **RemoveAt**, **Clear** etc. These methods allow you to add or remove the controls dynamically.

Sample Code for ASP.Net PlaceHolder Control

```
<asp:PlaceHolder ID="PlaceHolder1" runat="server">
</asp:PlaceHolder>
```

If you will add the above HTML markup code of PlaceHolder control in the web page it will not display any output control in the browser window. Even if you will try to view the page source code, there will be no HTML element associated to it.

C# Code:

```
using System.Web.UI.WebControls;
public partial class Default : System.Web.UI.Page
{
   Label Label1;
   TextBox TextBox1;
   Button Button1;
   Label Label2;
   protected void Page_Load(object sender, EventArgs e)
   {
      this.Label1 = new Label();
      this.Label1.Text = "Enter Text Here: ";
      this.Label1.Font.Bold = true;
      this.TextBox1 = new TextBox();
      this.TextBox1.BackColor = System.Drawing.Color.Yellow;
      this.Button1 = new Button();
      this.Button1.Text = "Submit";
      this.Button1.Click += new System.EventHandler(Button1_Click);
      this.Label2 = new Label();

      this.PlaceHolder1.Controls.Add(this.Label1);
      this.PlaceHolder1.Controls.Add(this.TextBox1);
      this.PlaceHolder1.Controls.Add(this.Button1);
      this.PlaceHolder1.Controls.Add(this.Label2);
   }

   protected void Button1_Click(object sender, EventArgs e)
   {
      Label2.Text = "You Enetred: " +TextBox1.Text.Trim();
   }
}
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Panel Control:

The Panel control is a container for other controls. It is especially useful when you want to generate controls programmatically or hide or show a group of controls.
OR
The Panel control works as a container for other controls on the page. It controls the appearance and visibility of the controls it contains. It also allows generating controls programmatically.
OR
This control is very useful because it works like container of every control. Assume you have very long content page and wishing to display it in very short area on web page, surely you can use it by enabling ScrollBars property.
OR
You can create group of controls wrapped inside the Panel control that further allows you to dynamically add the child controls into it. Panel control also enables you to show or hide the whole group of controls placed inside a single panel.

The **ASP.Net Panel control** is a web server control that renders as **div element of HTML**. The **div tag** of HTML stands for logical division block that can hold the other controls. It serves as a container for other web server or HTML controls to provide the managed layout to the web page. The Panel control is a server control that renders on web page using the basic HTML markup code of div element. It has different types of properties that can be used to customize its appearance on web page. These properties render as attributes of div tag that renders the panel control.

Properties of the Panel Control:

Following are the properties of ASP.Net Panel Control:

BackColor: It allows you to set the background color of the panel control.

BackImageUrl: It sets the background image of the control.

BorderColor: It sets the color of the border around the panel control.

BorderStyle: You can set the style of border using this property. It supports the following predefined types of border styles:

- Dashed
- Dotted
- Double
- Groove
- Inset
- None
- NotSet - It is the default value of BorderStyle property that does not render the border around the panel control.
- Outset
- Ridge
- Solid

BorderWidth: It allows you to set the width of the border using CSS units such as px, pt, cm etc.

CssClass: You can specify the CSS class name for this property to apply the CSS styles to Panel control.

DefaultButton: It accepts the ID of the Button control that you want to set as default button. The default button works as the submit button for a specific group of controls whether the controls are placed inside the Panel control or web form. Specifying the ID of the Button control to the **DefaultButton** property of Panel control executes the associated server side event of that button when user hits the enter key from the keyboard.

Direction: It accepts one of the predefined keyword values that sets the direction of the text placed inside the Panel control. It supports the following predefined values:

- NotSet - default value.
- LeftToRight
- RightToLeft

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Font-properties: The font property group contains following set of properties which allow you to change appearance of text placed inside the Panel.

- Font-Bold
- Font-Italics
- Font-Names
- Font-Overline
- Font-Size
- Font-Strikeout
- Font-Underline

ForeColor: It allows you to set the font color of the text.

GroupingText: It renders the group box with a text at top left corner inside the Panel control. The group box is being rendered by fieldset and legend HTML elements that display the bordered group box with a caption text.

HorizontalAlign: It sets the horizontal alignment of the text placed inside the Panel control.

Height: It sets height of the control.

Scrollbars: It enables you to display the scrollbars for the Panel control. You can control the appearance of scrollbars by specifying one of the predefined values for this property:

- None
- Horizontal
- Vertical
- Both
- Auto

This property sets the **CSS overflow property** value for the div tag that generates the panel control on the web page.

Width: It sets the width of the control.

Wrap: It allows to enable or disable the text wrapping.

You can also customize the look of ASP.Net Panel control using inline style attribute by specifying the CSS properties in the HTML markup code block.

Panel Control Examples:

Sample Code for ASP.Net Panel Control

```
<asp:Panel ID="Panel1" runat="server" BackImageUrl="Images/Ads1.jpg" Width="265" Height="190" DefaultButton="btnSubmit"
Wrap="false" Direction="LeftToRight" GroupingText="Panel Control Example" ScrollBars="Auto" HorizontalAlign="Center">
      <b>First Name:</b><asp:TextBox ID="txtFirstName" runat="server" />
      <br /><br />
      <b>Last Name:</b><asp:TextBox ID="txtLastName" runat="server" />
      <br /><br />
      <asp:Button ID="btnSubmit" runat="server" Text="Submit" OnClick="btnSubmit_Click" />
       
      <asp:Button ID="btnCancel" runat="server" Text="Cancel" OnClick="btnCancel_Click" />
      <br /><br />
      <asp:Label ID="lblResult" runat="server" />
 </asp:Panel>
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

Microsoft .NET Solutions
Robust, Scalable & Secure Applications
Developed for Diverse Business Needs
Develop your Microsoft skills and knowledge through great training
NARESH i technologies
Microsoft .net

C# Code:

```
protected void Page_Load(object sender, EventArgs e)
   {
      txtFirstName.Focus();
   }

protected void btnSubmit_Click(object sender, EventArgs e)
   {
      lblResult.Text = "<b>First Name: </b>" + txtFirstName.Text.Trim() + "<br />" + "<b>Last Name: </b>" +
txtLastName.Text.Trim();
   }

   protected void btnCancel_Click(object sender, EventArgs e)
   {
      Response.Redirect("~/Default.aspx");
   }
```

Output:

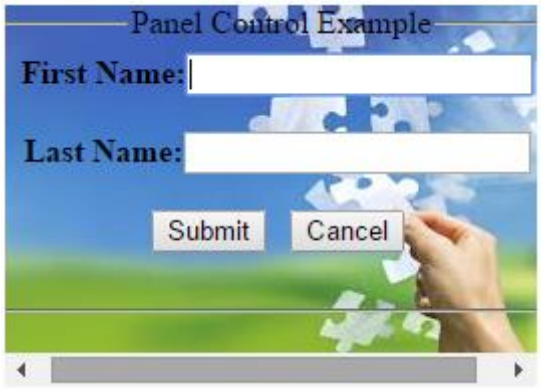


Table Control:

This control enables us to create table on web page. We can create rows and columns in table.
OR
The Table control builds up a table programmatically by adding TableRows to the Rows collection of the table, and TableCells to the Cells collection of the row. You can add content to a table cell programmatically by adding controls to the Controls collection of the cell.
OR
The ASP.Net Table server control provides the dynamic way to generate the complex HTML table structure using server code. You can create Table Cells and add them into the Table Rows to generate <tr> and <td> tags.

Properties of Table Control:

The following are the Properties of the **ASP.NET Table Control**.

Property	Description
BackImageUrl	To define the URL for the Table control's background image.
Caption	To specify the caption of the Table Control.

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

CaptionAlign	To specify Caption text's alignment.
CellPadding	To specify cell walls and contents space in pixels (px).
CellSpacing	To specify cell space in pixels (px).
GridLines	To define the Table Control's Gridline format.
HorizontalAlign	To define the ASP.NET Table Controls Horizontal Alignment.
Rows	A set of Rows which make up the ASP.NET Table Control

Example1: Working with the ASP.NET Table Control Design Time to represent data in a tabular format as following:

HTML Code:

```
<asp:Table ID="Table1" runat="server" Caption="Customer Data" Width="100%" CellPadding="5" CellSpacing="5"
GridLines="Both">
      <asp:TableHeaderRow>
         <asp:TableHeaderCell>Customer Id</asp:TableHeaderCell>
         <asp:TableHeaderCell>Customer Name</asp:TableHeaderCell>
         <asp:TableHeaderCell>Address</asp:TableHeaderCell>
      </asp:TableHeaderRow>
      <asp:TableRow>
         <asp:TableCell Text="101" />
          <asp:TableCell Text="ABC" />
          <asp:TableCell Text="Hyderabad" />
      </asp:TableRow>
      <asp:TableFooterRow>
         <asp:TableCell ColumnSpan="3" HorizontalAlign="Center">End of Data</asp:TableCell>
      </asp:TableFooterRow>
   </asp:Table>
```

Output:

Customer Data

Customer Id	Customer Name	Address
101	ABC	Hyderabad
End of Data		

Example2: Working with the ASP.NET Table Control Run Time (Programmatically) to represent data in a tabular format as following:

C# Code:

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

public partial class Default : System.Web.UI.Page
{
   Table Table1;
   protected void Page_Load(object sender, EventArgs e)
   {
      this.Table1 = new Table();
```

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.

```
    this.Table1.Caption = "Customer Data";
    this.Table1.Width = Unit.Percentage(100);
    this.Table1.CellPadding = 5;
    this.Table1.CellSpacing = 5;
    this.Table1.GridLines = GridLines.Both;

    TableHeaderRow thr = new TableHeaderRow();

    TableHeaderCell[] thc = new TableHeaderCell[3]
    {
       new TableHeaderCell(),
       new TableHeaderCell(),
       new TableHeaderCell()
    };

    thc[0].Text = "Customer Id";
    thc[1].Text = "Customer Name";
    thc[2].Text = "Address";

    thr.Cells.AddRange(thc);

    TableRow tr1 = new TableRow();

    TableCell[] tr1Tc = new TableCell[3]
    {
       new TableCell(),
       new TableCell(),
       new TableCell()
    };
    tr1Tc[0].Text = "101";
    tr1Tc[1].Text = "ABC";
    tr1Tc[2].Text = "Hyderabad";

    tr1.Cells.AddRange(tr1Tc);

    TableFooterRow tfr = new TableFooterRow();

    TableCell tfrTc = new TableCell();
    tfrTc.ColumnSpan = 3;
    tfrTc.Text = "End Of Data";
    tfrTc.HorizontalAlign = HorizontalAlign.Center;
    tfr.Cells.Add(tfrTc);

    this.Table1.Rows.Add(thr);
    this.Table1.Rows.Add(tr1);
    this.Table1.Rows.Add(tfr);

    this.form1.Controls.Add(this.Table1);
  }
}
```

Output:

Customer Data

Customer Id	Customer Name	Address
101	ABC	Hyderabad
End of Data		

RAKESH.NET

Copyright © 2015 https://www.facebook.com/rakeshdotnet All Rights Reserved.