Textbox Label and button

Textbox and Label

Text box controls are typically used to accept input from the user. A text box control can accept one or more lines of text depending upon the settings of the TextMode attribute.

Label controls provide an easy way to display text which can be changed from one execution of a page to the next. If you want to display text that does not change, you use the literal text.

Basic syntax of text control:

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

<asp:Label Id=”Label1” runat=”server”></asp:Label>
```

Common Properties of the Text Box and Labels:

Property	Description
TextMode	Specifies the type of text box. SingleLine creates a standard text box, MultiLIne creates a text box that accepts more than one line of text and the Password causes the characters that are entered to be masked. The default is SingleLine.
Text	The text content of the text box.
MaxLength	The maximum number of characters that can be entered into the text box.
Wrap	It determines whether or not text wraps automatically for multi-line text box; default is true.
ReadOnly	Determines whether the user can change the text in the box; default is false, i.e., the user can not change the text.
Columns	The width of the text box in characters. The actual width is

	determined based on the font that is used for the text entry.
Rows	The height of a multi-line text box in lines. The default value is 0, means a single line text box.

The mostly used attribute for a label control is 'Text', which implies the text displayed on the label.

Button Controls

ASP.NET provides three types of button control:

- **Button** : It displays text within a rectangular area.
- **Link Button** : It displays text that looks like a hyperlink.
- **Image Button** : It displays an image.

When a user clicks a button, two events are raised: Click and Command.

Basic syntax of button control:

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

Common properties of the button control:

Property	Description
Text	The text displayed on the button. This is for button and link button controls only.
ImageUrl	For image button control only. The image to be displayed for the button.
AlternateText	For image button control only. The text to be displayed if the browser cannot display the image.
CausesValidation	Determines whether page validation occurs when a user clicks the

	button. The default is true.
CommandName	A string value that is passed to the command event when a user clicks the button.
CommandArgument	A string value that is passed to the command event when a user clicks the button.
PostBackUrl	The URL of the page that is requested when the user clicks the button.

Check Boxes and Radio Buttons

A check box displays a single option that the user can either check or uncheck and radio buttons present a group of options from which the user can select just one option.

To create a group of radio buttons, you specify the same name for the GroupName attribute of each radio button in the group. If more than one group is required in a single form, then specify a different group name for each group.

If you want check box or radio button to be selected when the form is initially displayed, set its checked attribute to true. If the Checked attribute is set to true for multiple radio buttons in a group, then only the last one is considered as true.

Basic syntax of check box:

```
<asp:CheckBox ID= "chkoption" runat= "Server">

</asp:CheckBox>
```

Basic syntax of radio button:

```
<asp:RadioButton ID= "rdboption" runat= "Server">

</asp: RadioButton>
```

Common properties of check boxes and radio buttons:

Property	Description
Text	The text displayed next to the check box or radio button.
Checked	Specifies whether it is selected or not, default is false.
GroupName	Name of the group the control belongs to.

List Controls

ASP.NET provides the following controls

- Drop-down list,
- List box,
- Radio button list,
- Check box list,
- Bulleted list.

Listbox and Dropdownlist

These control let a user choose from one or more items from the list. List boxes and drop-down lists contain one or more list items. These lists can be loaded either by code or by the ListItemCollection editor.

Basic syntax of list box control:

```
<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged">

</asp:ListBox>
```

Basic syntax of drop-down list control:

```
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">

</asp:DropDownList>
```

Common properties of list box and drop-down Lists:

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

Common properties of each list item objects:

Property	Description
Text	The text displayed for the item.
Selected	Indicates whether the item is selected.
Value	A string value associated with the item.

It is important to notes that:

- To work with the items in a drop-down list or list box, you use the Items property of the control. This property returns a ListItemCollection object which contains all the items of the list.
- The SelectedIndexChanged event is raised when the user selects a different item from a drop-down list or list box.

Radio Button list and Check Box list

A radio button list presents a list of mutually exclusive options. A check box list presents a list of independent options. These controls contain a collection of ListItem objects that could be referred to through the Items property of the control.

Basic syntax of radio button list:

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

  OnSelectedIndexChanged="RadioButtonList1_SelectedIndexChanged">

</asp:RadioButtonList>
```

Basic syntax of check box list:

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" AutoPostBack="True"

  OnSelectedIndexChanged="CheckBoxList1_SelectedIndexChanged">

</asp:CheckBoxList>
```

Common properties of check box and radio button lists:

Property	Description
RepeatLayout	This attribute specifies whether the table tags or the normal html flow to use while formatting the list when it is rendered. The default is Table.
RepeatDirection	It specifies the direction in which the controls to be repeated. The values available are Horizontal and Vertical. Default is Vertical.
RepeatColumns	It specifies the number of columns to use when repeating the controls; default is 0.

Bulleted lists or numbered list

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.

Basic syntax of a bulleted list:

```
<asp:BulletedList ID="BulletedList1" runat="server">

</asp:BulletedList>
```

Common properties of the bulleted list:

Property	Description
BulletStyle	This property specifies the style and looks of the bullets, or numbers.
RepeatDirection	It specifies the direction in which the controls to be repeated. The values available are Horizontal and Vertical. Default is Vertical.
RepeatColumns	It specifies the number of columns to use when repeating the controls; default is 0.

Image control and file upload

Image Control

The image control is used for displaying images on the web page, or some alternative text, if the image is not available.

Basic syntax for an image control:

```
<asp:Image ID="Image1" runat="server">
```

It has the following important properties:

Property	Description
AlternateText	Alternate text to be displayed in absence of the image.
ImageAlign	Alignment options for the control.
ImageUrl	Path of the image to be displayed by the control.

File upload

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

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

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

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

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

The basic syntax of FileUpload is:

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

The FileUpload class is derived from the WebControl class, and inherits all its members. Apart from those, the FileUpload class has the following read-only properties:

Properties	Description
FileBytes	Returns an array of the bytes in a file to be uploaded.
FileContent	Returns the stream object pointing to the file to be uploaded.
FileName	Returns the name of the file to be uploaded.
HasFile	Specifies whether the control has a file to upload.
PostedFile	Returns a reference to the uploaded file.

The posted file is encapsulated in an object of type HttpPostedFile, which could be accessed through the PostedFile property of the FileUpload class.

The HttpPostedFile class has the following frequently used properties:

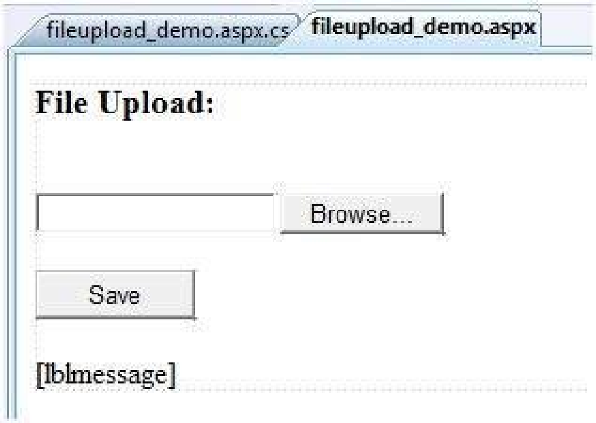
Properties	Description
ContentLength	Returns the size of the uploaded file in bytes.
ContentType	Returns the MIME type of the uploaded file.
FileName	Returns the full filename.

InputStream	Returns a stream object pointing to the uploaded file.

Example

The following example demonstrates the FileUpload control and its properties. The form has a FileUpload control along with a save button and a label control for displaying the file name, file type, and file length.

In the design view, the form looks as follows:



The content file code is as given:

```
<body>
  <form id="form1" runat="server">
    <div>
      <h3> File Upload:</h3>
      <br />
      <asp:FileUpload ID="FileUpload1" runat="server" />
      <br /><br />
      <asp:Button ID="btnsave" runat="server" onclick="btnsave_Click"  Text="Save"
style="width:85px" />
      <br /><br />
      <asp:Label ID="lblmessage" runat="server" />
    </div>

  </form>
</body>
```

Hyperlink, table, panel and wizard

Hyperlink

The HyperLink control is like the HTML <a> element.

Basic syntax for a hyperlink control:

```
<asp:HyperLink ID="HyperLink1" runat="server">

  HyperLink

</asp:HyperLink>
```

It has the following important properties:

Property	Description
ImageUrl	Path of the image to be displayed by the control.
NavigateUrl	Target link URL.
Text	The text to be displayed as the link.
Target	The window or frame which loads the linked page.

Table

The System.Web.UI.Controls namespace defines the Table class, along with the other Web controls. You can create tables in .NET using a Table control and its helper controls TableRow and TableCell. As with all Web controls, you can create a Table control at run-time as well as at design-time using the VS .NET IDE. Table 7-9 describes the Table control and its helper controls.

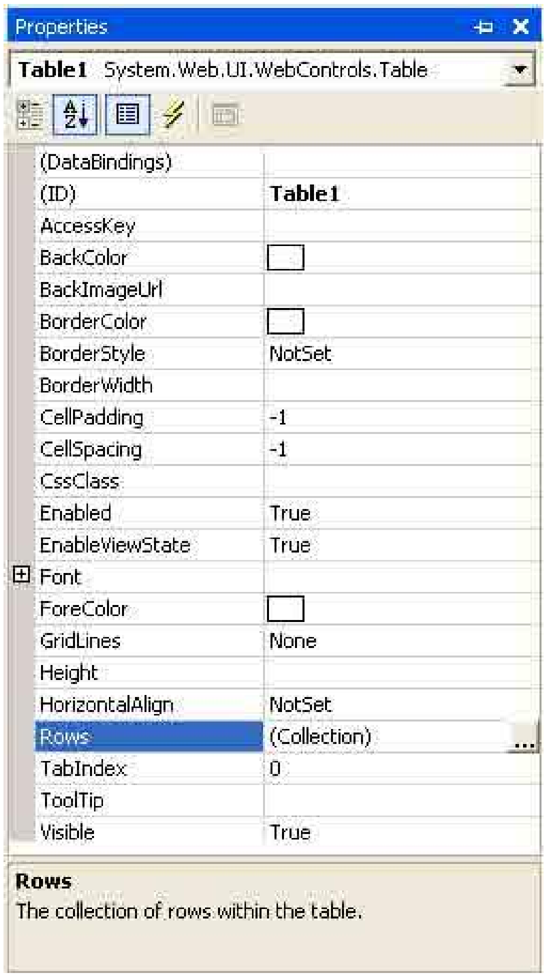
Table 7-9 ASP.NET table and its Helper Control Classes

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

Creating a table at design-time

You can create a table at design as well as run-time. To add a table control to your form at design-time, just drag and drop a table control from the Web Control toolbox to a Web form. Then use the Properties windows to add the rows and cells to the table.

The Rows property of a Table control adds rows to a table (see Figure 7-55).



Properties

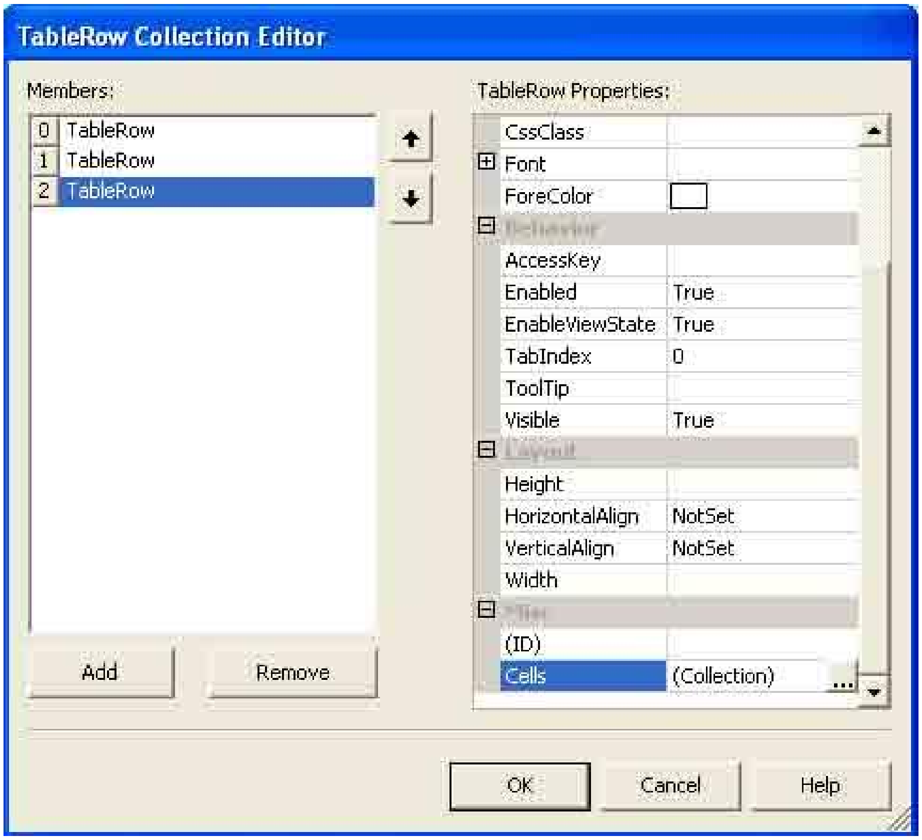
Table1 System.Web.UI.WebControls.Table

(DataBindings)	
(ID)	**Table1**
AccessKey	
BackColor	□
BackImageUrl	
BorderColor	□
BorderStyle	NotSet
BorderWidth	
CellPadding	-1
CellSpacing	-1
CssClass	
Enabled	True
EnableViewState	True
⊞ Font	
ForeColor	□
GridLines	None
Height	
HorizontalAlign	NotSet
Rows	(Collection) ...
TabIndex	0
ToolTip	
Visible	True

Rows
The collection of rows within the table.

Figure 7-55. Table control properties

In this example, I added three rows to the table by using the Add button of the TableRow Collectionn Editor. (Clicking on the Collection of rows properties launches the TableRow Collection Editor; See Figure 7-56).



TableRow Collection Editor

Members:

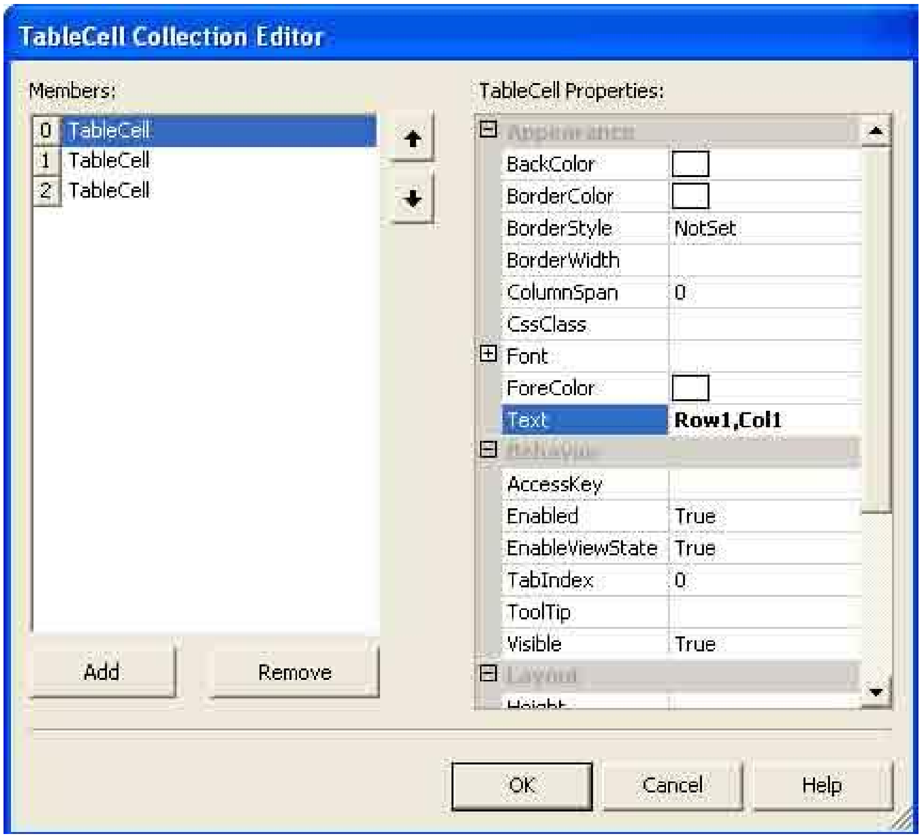
0 TableRow
1 TableRow
2 TableRow

TableRow Properties:

CssClass	
⊞ Font	
ForeColor	□
⊟ Behavior	
AccessKey	
Enabled	True
EnableViewState	True
TabIndex	0
ToolTip	
Visible	True
⊟ Layout	
Height	
HorizontalAlign	NotSet
VerticalAlign	NotSet
Width	
⊟ Misc	
(ID)	
Cells	(Collection)

Figure 7-56. Adding rows with the TableRow Collection Editor

Now simply add cells to the rows by using the Cells property of a row. Use the Text property of a cell to add a value to the cell, and use the Add and Remove buttons of the TableCell Collection Editor to add and remove cells from a row. (You can launch the TableCell Collection Editor by clicking on the Collection of Cells property; see Figure 7-57)



Members:

0 TableCell
1 TableCell
2 TableCell

TableCell Properties:

⊟ Appearance	
BackColor	□
BorderColor	□
BorderStyle	NotSet
BorderWidth	
ColumnSpan	0
CssClass	
⊞ Font	
ForeColor	□
Text	Row1,Col1
⊟ Behavior	
AccessKey	
Enabled	True
EnableViewState	True
TabIndex	0
ToolTip	
Visible	True
⊟ Layout	
Height	

Figure 7-57. TableCell Collection Editor

After taking this action, the IDE writes the ASP code for you, which looks like Listing 7-26.

Listing 7-26. ASP.NET code for a table control

```
<asp:Table ID="Table1" runat="server" Height="123px" Width="567px">
  <asp:TableRow runat="server">
    <asp:TableCell runat="server"></asp:TableCell>
    <asp:TableCell runat="server"></asp:TableCell>
    <asp:TableCell runat="server"></asp:TableCell>
  </asp:TableRow>
  <asp:TableRow runat="server">
    <asp:TableCell runat="server"></asp:TableCell>
    <asp:TableCell runat="server"></asp:TableCell>
    <asp:TableCell runat="server"></asp:TableCell>
  </asp:TableRow>
  <asp:TableRow runat="server">
    <asp:TableCell runat="server"></asp:TableCell>
    <asp:TableCell runat="server"></asp:TableCell>
```

```
        <asp:TableCell runat="server"></asp:TableCell>
    </asp:TableRow>
</asp:Table>
```

Panel control

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.

The basic syntax of panel control is as follows:

```
<asp:Panel ID= "Panel1"  runat = "server">

</asp:Panel>
```

The Panel control is derived from the WebControl class. Hence it inherits all the properties, methods and events of the same. It does not have any method or event of its own. However it has the following properties of its own:

Properties	Description
BackImageUrl	URL of the background image of the panel.
DefaultButton	Gets or sets the identifier for the default button that is contained in the Panel control.
Direction	Text direction in the panel.
GroupingText	Allows grouping of text as a field.
HorizontalAlign	Horizontal alignment of the content in the panel.
ScrollBars	Specifies visibility and location of scrollbars within the panel.
Wrap	Allows text wrapping.

Working with the Panel Control

Let us start with a simple scrollable panel of specific height and width and a border style. The ScrollBars property is set to both the scrollbars, hence both the scrollbars are rendered.

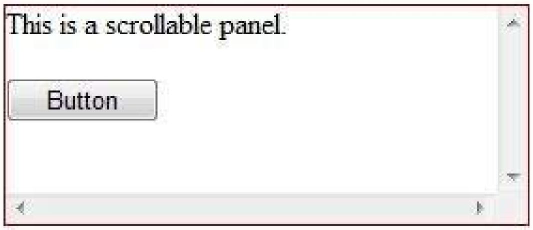
The source file has the following code for the panel tag:

```
<asp:Panel ID="Panel1" runat="server" BorderColor="#990000" BorderStyle="Solid"
  Borderstyle="width:1px" Height="116px" ScrollBars="Both" style="width:278px">

  This is a scrollable panel.
  <br />
  <br />

  <asp:Button ID="btnpanel" runat="server" Text="Button" style="width:82px" />
</asp:Panel>
```

The panel is rendered as follows:

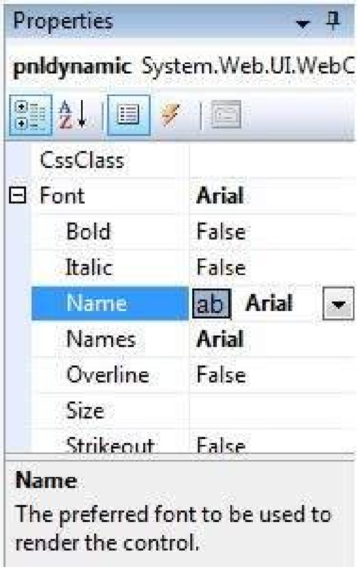


This is a scrollable panel.

Example

The following example demonstrates dynamic content generation. The user provides the number of label controls and textboxes to be generated on the panel. The controls are generated programmatically.

Change the properties of the panel using the properties window. When you select a control on the design view, the properties window displays the properties of that particular control and allows you to make changes without typing.



pnldynamic System.Web.UI.WebC

CssClass	
⊟ Font	**Arial**
Bold	False
Italic	False
Name	ab **Arial**
Names	**Arial**
Overline	False
Size	
Strikeout	False

Name
The preferred font to be used to render the control.

Wizard control

Wizard control eliminates the need to design forms to execute a step by step process in the actual business flow. This simplifies the work of developers to design and write the code.

The control provides a mechanism that allows you to easily build the desired wizard as a collection of steps, add a new step, or reorder the steps. You don't have to write any infrastructure whatsoever for navigation or to persist user data between steps.

The asp:wizard is a control name and wizardsteps is the parent element for the asp:wizardstep element.

```
<asp:wizard  id="wizard1" runat="server>
<wizardsteps>
  <asp:WizardStep runat="server" Title="Personal Information" StepType="Auto">
//Place the control here.
</asp:wizardstep>
</wizardsteps>
</asp:wizard>
```

There can be multiple wizard step which iterates from by form. The control can be placed in the wizard step tag.

The step is the enum type which has the attributes Auto, Complete, Finish, Start and Step,

The first form will have the next button and all the subsequent forms will have the previous and next button.
The last wizard step tag should be defined as the complete step type.

There are events which are used to write the events on the controls in the wizard control.

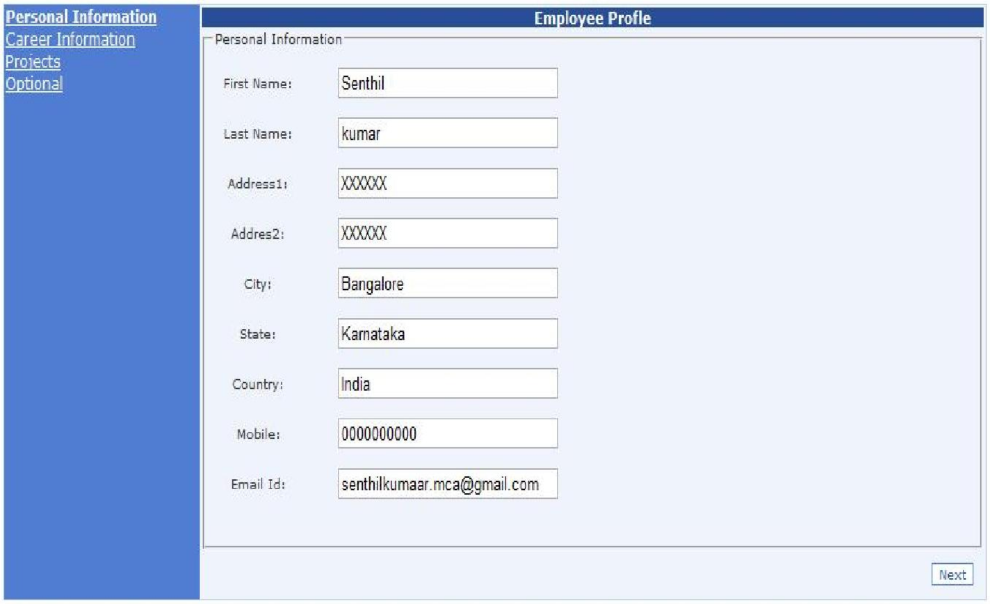
OnActiveStepChanged
OnCancelButtonClick
OnPreviousButtonClick
OnNextButtonClick
OnFinishButtonClick
OnSideBarButtonClick

Wizard Control Properties:

Property	Description
ActiveStep	Returns the current wizard step object; the object is an instance of the WizardStep class
ActiveStepIndex	Gets and sets the zero-based index of current wizard step
DisplaySideBar	Toggles the visibility of the sidebar; the default value is True
FinishStepButtonText	Gets and sets the text for the Finish button
HeaderText	Gets and sets the title of the wizard
NextStepButtonText	Gets and sets the text for the Next button
PreviousStepButtonText	Gets and sets the text for the Previous button

I have designed an example application for the Wizard control.

The wizard ActiveStep property can be set by the ActiveStepIndex="0" of the wizard control. The first form will have the next navigation control.



First Name: Senthil

Last Name: kumar

Address1: XXXXXX

Addres2: XXXXXX

City: Bangalore

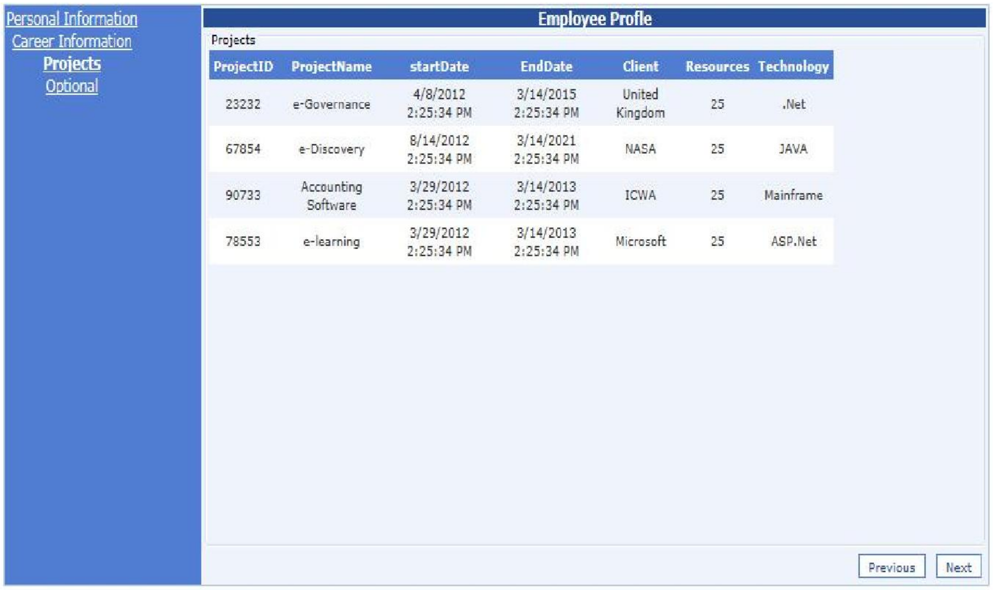
State: Karnataka

Country: India

Mobile: 0000000000

Email Id: senthilkumaar.mca@gmail.com

The next form can be navigated through the "Next" button or user can jump into any wizard by clicking side bar navigation control.

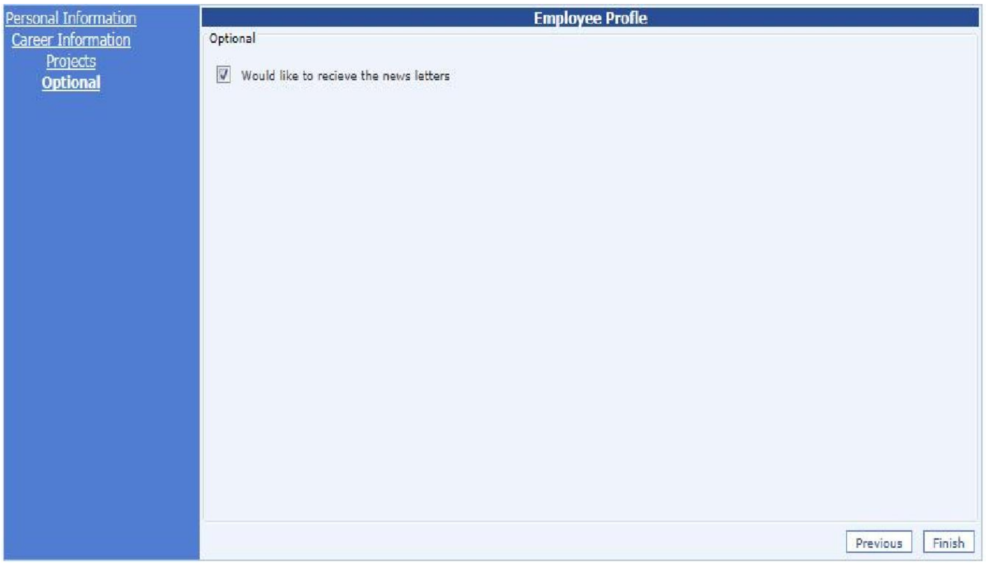


Employee Profle

Projects

ProjectID	ProjectName	startDate	EndDate	Client	Resources	Technology
23232	e-Governance	4/8/2012 2:25:34 PM	3/14/2015 2:25:34 PM	United Kingdom	25	.Net
67854	e-Discovery	8/14/2012 2:25:34 PM	3/14/2021 2:25:34 PM	NASA	25	JAVA
90733	Accounting Software	3/29/2012 2:25:34 PM	3/14/2013 2:25:34 PM	ICWA	25	Mainframe
78553	e-learning	3/29/2012 2:25:34 PM	3/14/2013 2:25:34 PM	Microsoft	25	ASP.Net

The Last form in the wizard will have the finish button. The user can identify the step by ActiveStepType in the finish button.



Employee Profle

Optional

☑ Would like to recieve the news letters