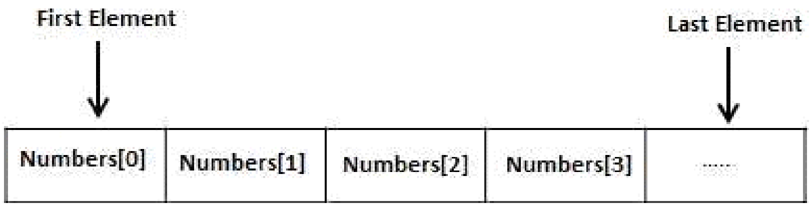
VB.Net - Arrays

An array stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Creating Arrays in VB.Net

To declare an array in VB.Net, you use the Dim statement. For example,

```
Dim intData(30)   ' an array of 31 elements
Dim strData(20) As String   ' an array of 21 strings
Dim twoDarray(10, 20) As Integer       'a two
dimensional array of integers
Dim ranges(10, 100)   'a two dimensional array
```

You can also initialize the array elements while declaring the array. For example,

```
Dim intData() As Integer = {12, 16, 20, 24, 28, 32}
Dim names() As String = {"Karthik", "Sandhya", _
"Shivangi", "Ashwitha", "Somnath"}
Dim miscData() As Object = {"Hello World", 12d, 16ui,
"A"c}
```

The elements in an array can be stored and accessed by using the index of the array. The following program demonstrates this –

```
Module arrayApl
   Sub Main()
      Dim n(10) As Integer ' n is an array of 11 integers '
      Dim i, j As Integer
      ' initialize elements of array n '

      For i = 0 To 10
         n(i) = i + 100 ' set element at location i to i + 100
      Next i
      ' output each array element's value '

      For j = 0 To 10
         Console.WriteLine("Element({0}) = {1}", j, n(j))
      Next j
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result –

```
Element(0) = 100
Element(1) = 101
Element(2) = 102
Element(3) = 103
Element(4) = 104
Element(5) = 105
Element(6) = 106
Element(7) = 107
Element(8) = 108
Element(9) = 109
Element(10) = 110
```

Dynamic Arrays

Dynamic arrays are arrays that can be dimensioned and re-dimensioned as par the need of the program. You can declare a dynamic array using the **ReDim** statement.

Syntax for ReDim statement –

```
ReDim [Preserve] arrayname(subscripts)
```

Where,

- The **Preserve** keyword helps to preserve the data in an existing array, when you resize it.
- **arrayname** is the name of the array to re-dimension.
- **subscripts** specifies the new dimension.

```
Module arrayApl
   Sub Main()
      Dim marks() As Integer
      ReDim marks(2)
      marks(0) = 85
      marks(1) = 75
      marks(2) = 90

      ReDim Preserve marks(10)
      marks(3) = 80
      marks(4) = 76
      marks(5) = 92
      marks(6) = 99
      marks(7) = 79
      marks(8) = 75
      For i = 0 To 10
         Console.WriteLine(i & vbTab & marks(i))
      Next i
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result –

```
0      85
1      75
2      90
3      80
4      76
5      92
6      99
7      79
8      75
9      0
10     0
```

Multi-Dimensional Arrays

VB.Net allows multidimensional arrays. Multidimensional arrays are also called rectangular arrays.

You can declare a 2-dimensional array of strings as –

```
Dim twoDStringArray(10, 20) As String
```

or, a 3-dimensional array of Integer variables –

```
Dim threeDIntArray(10, 10, 10) As Integer
```

The following program demonstrates creating and using a 2-dimensional array –

```
Module arrayApl
   Sub Main()
      ' an array with 5 rows and 2 columns
      Dim a(,) As Integer = {{0, 0}, {1, 2}, {2, 4}, {3,
6}, {4, 8}}
      Dim i, j As Integer
      ' output each array element's value '

      For i = 0 To 4
         For j = 0 To 1
            Console.WriteLine("a[{0},{1}] = {2}", i, j, a(i,
j))
         Next j
      Next i
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result –

```
a[0,0]: 0
a[0,1]: 0
a[1,0]: 1
a[1,1]: 2
a[2,0]: 2
a[2,1]: 4
a[3,0]: 3
a[3,1]: 6
a[4,0]: 4
a[4,1]: 8
```

Jagged Array

A Jagged array is an array of arrays. The following code shows declaring a jagged array named *scores* of Integers −

```
Dim scores As Integer()() = New Integer(5)(){}
```

The following example illustrates using a jagged array −

```
Module arrayApl
   Sub Main()
      'a jagged array of 5 array of integers
      Dim a As Integer()() = New Integer(4)() {}
      a(0) = New Integer() {0, 0}
      a(1) = New Integer() {1, 2}
      a(2) = New Integer() {2, 4}
      a(3) = New Integer() {3, 6}
      a(4) = New Integer() {4, 8}
      Dim i, j As Integer
      ' output each array element's value

      For i = 0 To 4
         For j = 0 To 1
            Console.WriteLine("a[{0},{1}] = {2}", i, j, 
a(i)(j))
         Next j
      Next i
      Console.ReadKey()
   End Sub
End Module
```

When the above code is compiled and executed, it produces the following result −

```
a[0][0]: 0
a[0][1]: 0
```

```
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

The Array Class

The Array class is the base class for all the arrays in VB.Net. It is defined in the System namespace. The Array class provides various properties and methods to work with arrays.

Properties of the Array Class

The following table provides some of the most commonly used **properties** of the **Array** class –

Sr.No	Property Name & Description
1	**IsFixedSize** Gets a value indicating whether the Array has a fixed size.
2	**IsReadOnly** Gets a value indicating whether the Array is read-only.
3	**Length** Gets a 32-bit integer that represents the total number of elements in all the dimensions of the Array.

4	**LongLength** Gets a 64-bit integer that represents the total number of elements in all the dimensions of the Array.
5	**Rank** Gets the rank (number of dimensions) of the Array.

Methods of the Array Class

The following table provides some of the most commonly used **methods** of the **Array** class −

Sr.No	Method Name & Description
1	**Public Shared Sub Clear (array As Array, index As Integer, length As Integer)** Sets a range of elements in the Array to zero, to false, or to null, depending on the element type.
2	**Public Shared Sub Copy (sourceArray As Array, destinationArray As Array, length As Integer)** Copies a range of elements from an Array starting at the first element and pastes them into another Array starting at the first element. The length is specified as a 32-bit integer.
3	**Public Sub CopyTo (array As Array, index As Integer)** Copies all the elements of the current one-dimensional Array to the specified one-

	dimensional Array starting at the specified destination Array index. The index is specified as a 32-bit integer.
4	**Public Function GetLength (dimension As Integer) As Integer** Gets a 32-bit integer that represents the number of elements in the specified dimension of the Array.
5	**Public Function GetLongLength (dimension As Integer) As Long** Gets a 64-bit integer that represents the number of elements in the specified dimension of the Array.
6	**Public Function GetLowerBound (dimension As Integer) As Integer** Gets the lower bound of the specified dimension in the Array.
7	**Public Function GetType As Type** Gets the Type of the current instance (Inherited from Object).
8	**Public Function GetUpperBound (dimension As Integer) As Integer** Gets the upper bound of the specified dimension in the Array.
9	**Public Function GetValue (index As Integer) As Object**

	Gets the value at the specified position in the one-dimensional Array. The index is specified as a 32-bit integer.
10	**Public Shared Function IndexOf (array As Array,value As Object) As Integer** Searches for the specified object and returns the index of the first occurrence within the entire one-dimensional Array.
11	**Public Shared Sub Reverse (array As Array)** Reverses the sequence of the elements in the entire one-dimensional Array.
12	**Public Sub SetValue (value As Object, index As Integer)** Sets a value to the element at the specified position in the one-dimensional Array. The index is specified as a 32-bit integer.
13	**Public Shared Sub Sort (array As Array)** Sorts the elements in an entire one-dimensional Array using the IComparable implementation of each element of the Array.
14	**Public Overridable Function ToString As String** Returns a string that represents the current object (Inherited from Object).

For complete list of Array class properties and methods, please consult Microsoft documentation.

Example

The following program demonstrates use of some of the methods of the Array class:

```
Module arrayApl
   Sub Main()
      Dim list As Integer() = {34, 72, 13, 44, 25, 30, 10}
      Dim temp As Integer() = list
      Dim i As Integer
      Console.Write("Original Array: ")

      For Each i In list
         Console.Write("{0} ", i)
      Next i
      Console.WriteLine()
      ' reverse the array
      Array.Reverse(temp)
      Console.Write("Reversed Array: ")

      For Each i In temp
         Console.Write("{0} ", i)
      Next i
      Console.WriteLine()
      'sort the array
      Array.Sort(list)
      Console.Write("Sorted Array: ")

      For Each i In list
         Console.Write("{0} ", i)
      Next i
      Console.WriteLine()
      Console.ReadKey()
   End Sub
End Module
```

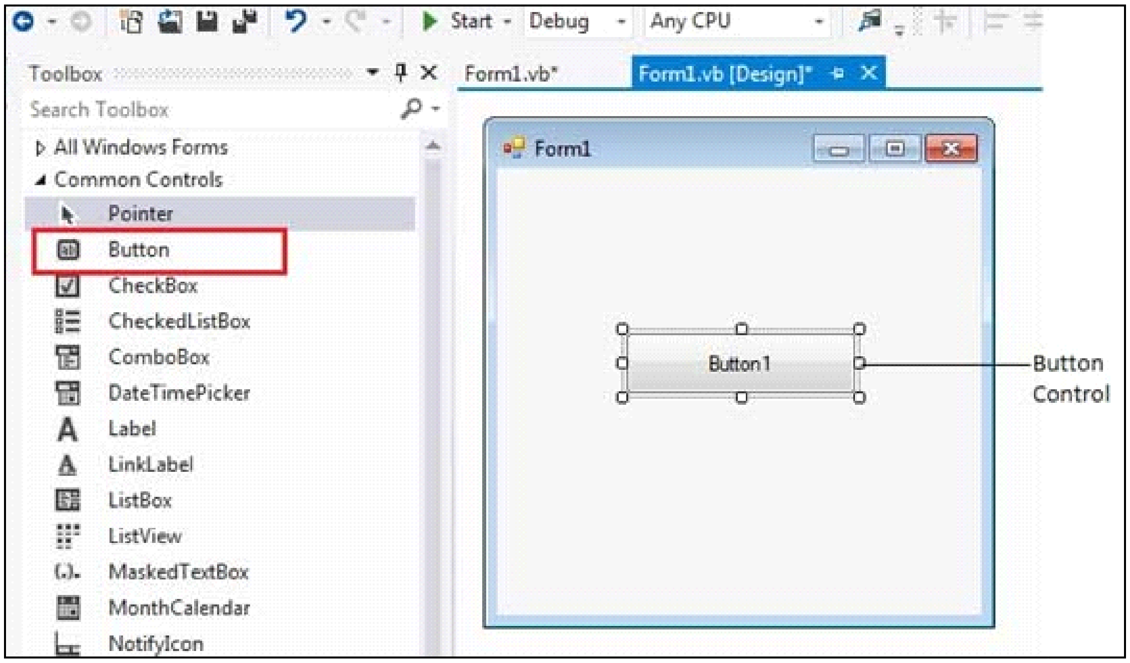
When the above code is compiled and executed, it produces the following result –

```
Original Array: 34 72 13 44 25 30 10
Reversed Array: 10 30 25 44 13 72 34
Sorted Array: 10 13 25 30 34 44 72
```

VB.Net - Button Control

The Button control represents a standard Windows button. It is generally used to generate a Click event by providing a handler for the Click event.

Let's create a label by dragging a Button control from the Toolbox ad dropping it on the form.



Properties of the Button Control

The following are some of the commonly used properties of the Button control –

Sr.No.	Property & Description
1	**AutoSizeMode** Gets or sets the mode by which the Button automatically resizes itself.
2	**BackColor** Gets or sets the background color of the control.
3	**BackgroundImage** Gets or sets the background image displayed in the control.
4	**DialogResult**

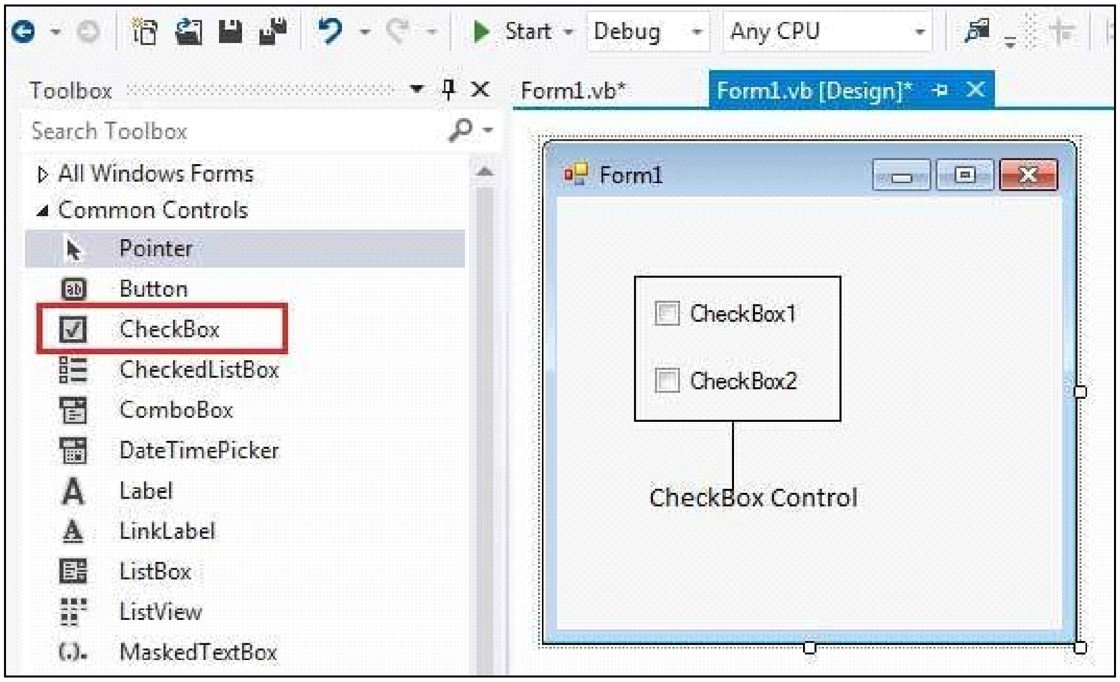
	Gets or sets a value that is returned to the parent form when the button is clicked. This is used while creating dialog boxes.
5	**ForeColor** Gets or sets the foreground color of the control.
6	**Image** Gets or sets the image that is displayed on a button control.
7	**Location** Gets or sets the coordinates of the upper-left corner of the control relative to the upper-left corner of its container.
8	**TabIndex** Gets or sets the tab order of the control within its container.
9	**Text** Gets or sets the text associated with this control

VB.Net - CheckBox Control

The CheckBox control allows the user to set true/false or yes/no type options. The user can select or deselect it.

When a check box is selected it has the value True, and when it is cleared, it holds the value False.

Let's create two check boxes by dragging CheckBox controls from the Toolbox and dropping on the form.



The CheckBox control has three states, **checked**, **unchecked** and **indeterminate**. In the indeterminate state, the check box is grayed out. To enable the indeterminate state, the *ThreeState* property of the check box is set to be **True**.

Properties of the CheckBox Control

The following are some of the commonly used properties of the CheckBox control −

Sr.No.	Property & Description
1	**Appearance** Gets or sets a value determining the appearance of the check box.

2	**AutoCheck** Gets or sets a value indicating whether the Checked or CheckedState value and the appearance of the control automatically change when the check box is selected.
3	**CheckAlign** Gets or sets the horizontal and vertical alignment of the check mark on the check box.
4	**Checked** Gets or sets a value indicating whether the check box is selected.
5	**CheckState** Gets or sets the state of a check box.
6	**Text** Gets or sets the caption of a check box.
7	**ThreeState** Gets or sets a value indicating whether or not a check box should allow three check states rather than two.

Methods of the CheckBox Control

The following are some of the commonly used methods of the CheckBox control −

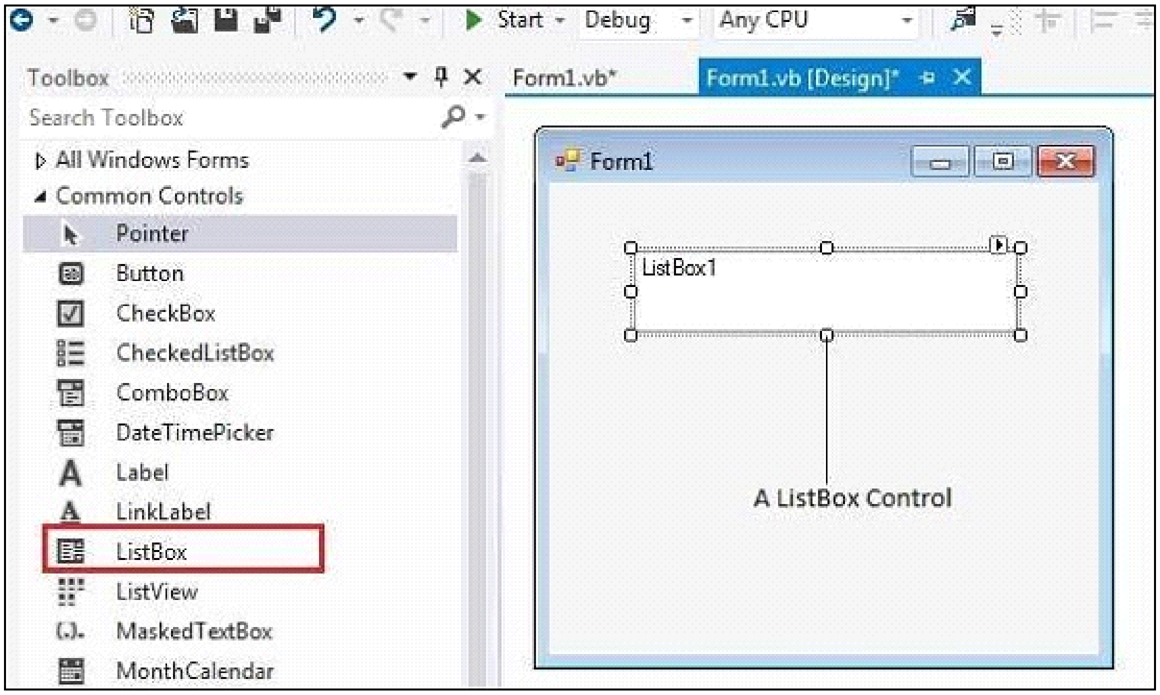
Sr.No.	Method Name & Description
1	**OnCheckedChanged**

	Raises the CheckedChanged event.
2	**OnCheckStateChanged** Raises the CheckStateChanged event.
3	**OnClick** Raises the OnClick event.

VB.Net - ListBox Control

The ListBox represents a Windows control to display a list of items to a user. A user can select an item from the list. It allows the programmer to add items at design time by using the properties window or at the runtime.

Let's create a list box by dragging a ListBox control from the Toolbox and dropping it on the form.



You can populate the list box items either from the properties window or at runtime. To add items to a ListBox, select the ListBox control and get to the properties window, for the properties of this control. Click the ellipses (...) button next to the Items property. This opens the String Collection Editor dialog box, where you can enter the values one at a line.

Properties of the ListBox Control

The following are some of the commonly used properties of the ListBox control −

Sr.No.	Property & Description
1	**AllowSelection** Gets a value indicating whether the ListBox currently enables selection of list items.
2	**BorderStyle** Gets or sets the type of border drawn around

	the list box.
3	**ColumnWidth** Gets of sets the width of columns in a multicolumn list box.
4	**HorizontalExtent** Gets or sets the horizontal scrolling area of a list box.
5	**HorizontalScrollBar** Gets or sets the value indicating whether a horizontal scrollbar is displayed in the list box.
6	**ItemHeight** Gets or sets the height of an item in the list box.
7	**Items** Gets the items of the list box.
8	**MultiColumn** Gets or sets a value indicating whether the list box supports multiple columns.
9	**ScrollAlwaysVisible** Gets or sets a value indicating whether the vertical scroll bar is shown at all times.
10	**SelectedIndex** Gets or sets the zero-based index of the currently selected item in a list box.

11	**SelectedIndices** Gets a collection that contains the zero-based indexes of all currently selected items in the list box.
12	**SelectedItem** Gets or sets the currently selected item in the list box.
13	**SelectedItems** Gets a collection containing the currently selected items in the list box.
14	**SelectedValue** Gets or sets the value of the member property specified by the ValueMember property.
15	**SelectionMode** Gets or sets the method in which items are selected in the list box. This property has values – • None • One • MultiSimple • MultiExtended
16	**Sorted** Gets or sets a value indicating whether the items in the list box are sorted alphabetically.
17	**Text**

	Gets or searches for the text of the currently selected item in the list box.
18	**TopIndex** Gets or sets the index of the first visible item of a list box.

Methods of the ListBox Control

The following are some of the commonly used methods of the ListBox control −

Sr.No.	Method Name & Description
1	**BeginUpdate** Prevents the control from drawing until the EndUpdate method is called, while items are added to the ListBox one at a time.
2	**ClearSelected** Unselects all items in the ListBox.
3	**EndUpdate** Resumes drawing of a list box after it was turned off by the BeginUpdate method.
4	**FindString** Finds the first item in the ListBox that starts with the string specified as an argument.
5	**FindStringExact** Finds the first item in the ListBox that exactly matches the specified string.

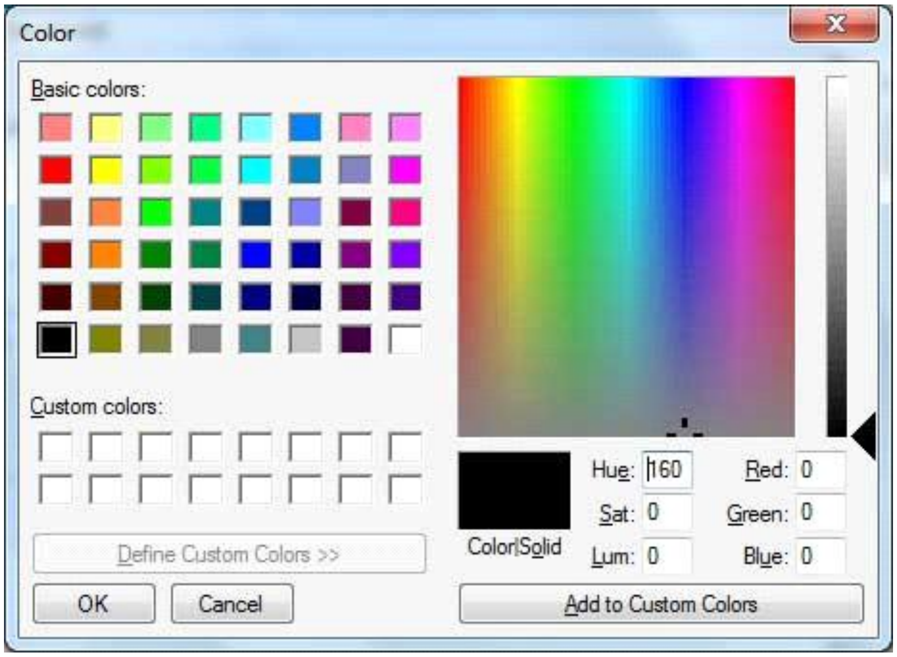
6	**GetSelected** Returns a value indicating whether the specified item is selected.
7	**SetSelected** Selects or clears the selection for the specified item in a ListBox.
8	**OnSelectedIndexChanged** Raises the SelectedIndexChanged event.
8	**OnSelectedValueChanged** Raises the SelectedValueChanged event

VB.Net - ColorDialog Control

The ColorDialog control class represents a common dialog box that displays available colors along with controls that enable the user to define custom colors. It lets the user select a color.

The main property of the ColorDialog control is *Color*, which returns a **Color** object.

Following is the Color dialog box –



Hue: 160
Sat: 0
Lum: 0
Red: 0
Green: 0
Blue: 0

Properties of the ColorDialog Control

The following are some of the commonly used properties of the ColorDialog control −

Sr.No.	Property & Description
1	**AllowFullOpen** Gets or sets a value indicating whether the user can use the dialog box to define custom colors.
2	**AnyColor** Gets or sets a value indicating whether the dialog box displays all available colors in the set of basic colors.
3	**CanRaiseEvents** Gets a value indicating whether the component can raise an event.
4	**Color**

	Gets or sets the color selected by the user.
5	**CustomColors** Gets or sets the set of custom colors shown in the dialog box.
6	**FullOpen** Gets or sets a value indicating whether the controls used to create custom colors are visible when the dialog box is opened.
7	**ShowHelp** Gets or sets a value indicating whether a Help button appears in the color dialog box.
8	**SolidColorOnly** Gets or sets a value indicating whether the dialog box will restrict users to selecting solid colors only.

Methods of the ColorDialog Control

The following are some of the commonly used methods of the ColorDialog control −

Sr.No.	Method Name & Description
1	**Reset** Resets all options to their default values, the last selected color to black, and the custom colors to their default values.
2	**RunDialog**

	When overridden in a derived class, specifies a common dialog box.
3	**ShowDialog** Runs a common dialog box with a default owner.

Events of the ColorDialog Control

The following are some of the commonly used events of the ColorDialog control −

Sr.No.	Event & Description
1	**HelpRequest** Occurs when the user clicks the Help button on a common dialog box.

Example

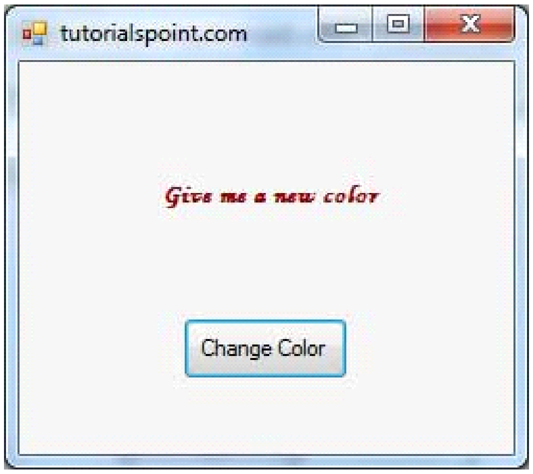
In this example, let's change the forecolor of a label control using the color dialog box. Take the following steps −

- Drag and drop a label control, a button control and a ColorDialog control on the form.
- Set the Text property of the label and the button control to 'Give me a new Color' and 'Change Color', respectively.
- Change the font of the label as per your likings.
- Double-click the Change Color button and modify the code of the Click event.

```
Private Sub Button1_Click(sender As Object, e As
EventArgs) Handles Button1.Click
```

```
  If ColorDialog1.ShowDialog <> 
Windows.Forms.DialogResult.Cancel Then
    Label1.ForeColor = ColorDialog1.Color
  End If
End Sub
```

When the application is compiled and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window –

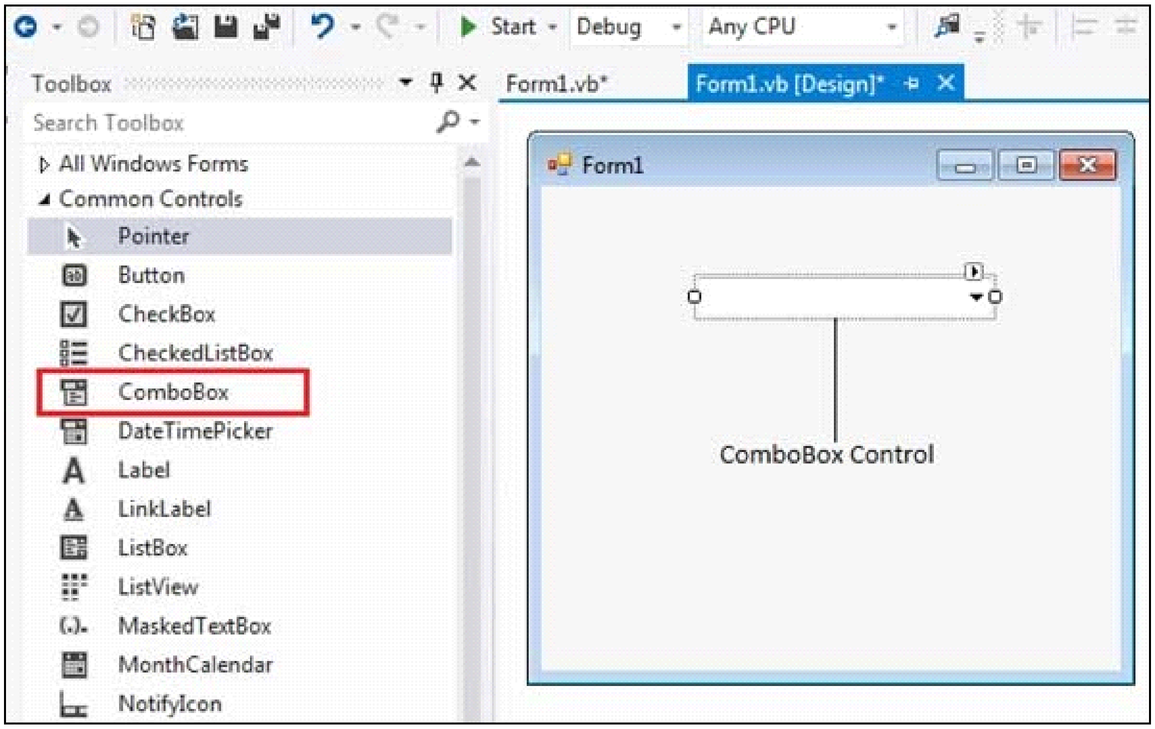


Clicking on the Change Color button, the color dialog appears, select a color and click the OK button. The selected color will be applied as the forecolor of the text of the label.

VB.Net - ComboBox Control

The ComboBox control is used to display a drop-down list of various items. It is a combination of a text box in which the user enters an item and a drop-down list from which the user selects an item.

Let's create a combo box by dragging a ComboBox control from the Toolbox and dropping it on the form.



You can populate the list box items either from the properties window or at runtime. To add items to a ComboBox, select the ComboBox control and go to the properties window for the properties of this control. Click the ellipses (...) button next to the Items property. This opens the String Collection Editor dialog box, where you can enter the values one at a line.

Properties of the ComboBox Control

The following are some of the commonly used properties of the ComboBox control −

Sr.No.	Property & Description
1	**AllowSelection** Gets a value indicating whether the list enables selection of list items.
2	**AutoCompleteCustomSource**

	Gets or sets a custom System.Collections .Specialized.StringCollection to use when the AutoCompleteSourceproperty is set to CustomSource.
3	**AutoCompleteMode** Gets or sets an option that controls how automatic completion works for the ComboBox.
4	**AutoCompleteSource** Gets or sets a value specifying the source of complete strings used for automatic completion.
5	**DataBindings** Gets the data bindings for the control.
6	**DataManager** Gets the CurrencyManager associated with this control.
7	**DataSource** Gets or sets the data source for this ComboBox.
8	**DropDownHeight** Gets or sets the height in pixels of the drop-down portion of the ComboBox.
9	**DropDownStyle** Gets or sets a value specifying the style of the combo box.
10	**DropDownWidth**

	Gets or sets the width of the of the drop-down portion of a combo box.
11	**DroppedDown** Gets or sets a value indicating whether the combo box is displaying its drop-down portion.
12	**FlatStyle** Gets or sets the appearance of the ComboBox.
13	**ItemHeight** Gets or sets the height of an item in the combo box.
14	**Items** Gets an object representing the collection of the items contained in this ComboBox.
15	**MaxDropDownItems** Gets or sets the maximum number of items to be displayed in the drop-down part of the combo box.
16	**MaxLength** Gets or sets the maximum number of characters a user can enter in the editable area of the combo box.
17	**SelectedIndex** Gets or sets the index specifying the currently selected item.

18	**SelectedItem** Gets or sets currently selected item in the ComboBox.
19	**SelectedText** Gets or sets the text that is selected in the editable portion of a ComboBox.
20	**SelectedValue** Gets or sets the value of the member property specified by the ValueMember property.
21	**SelectionLength** Gets or sets the number of characters selected in the editable portion of the combo box.
22	**SelectionStart** Gets or sets the starting index of text selected in the combo box.
23	**Sorted** Gets or sets a value indicating whether the items in the combo box are sorted.
24	**Text** Gets or sets the text associated with this control.

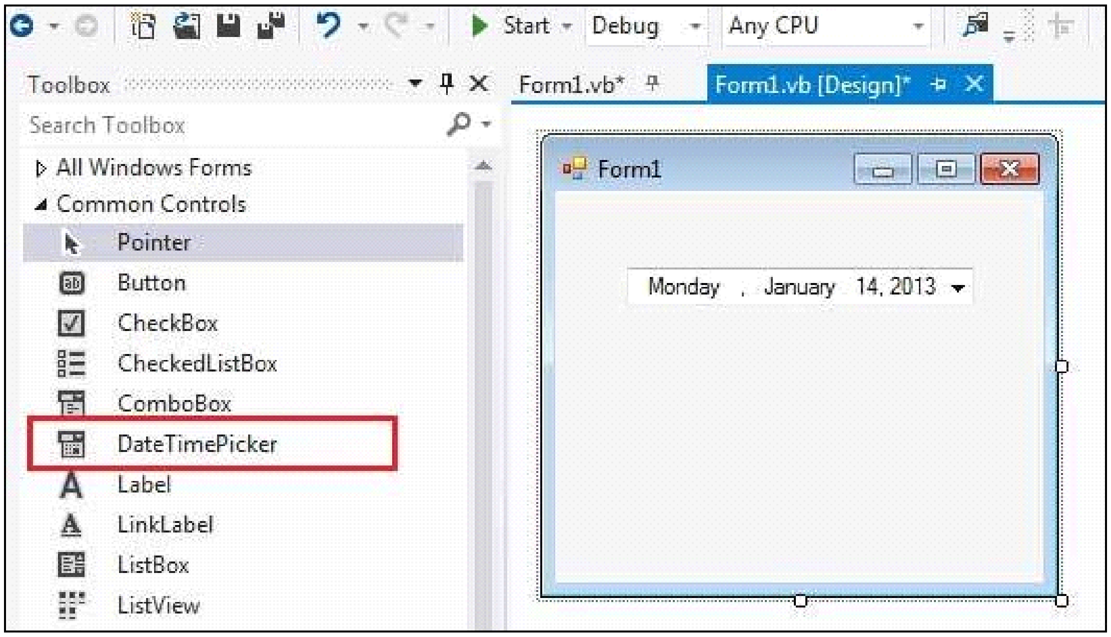
Methods of the ComboBox Control

The following are some of the commonly used methods of the ComboBox control –

Sr.No.	Method Name & Description
1	**BeginUpdate** Prevents the control from drawing until the EndUpdate method is called, while items are added to the combo box one at a time.
2	**EndUpdate** Resumes drawing of a combo box, after it was turned off by the BeginUpdate method.
3	**FindString** Finds the first item in the combo box that starts with the string specified as an argument.
4	**FindStringExact** Finds the first item in the combo box that exactly matches the specified string.
5	**SelectAll** Selects all the text in the editable area of the combo box

VB.Net - DateTimePicker Control

The DateTimePicker control allows selecting a date and time by editing the displayed values in the control. If you click the arrow in the DateTimePicker control, it displays a month calendar, like a combo box control. The user can make selection by clicking the required date. The new selected value appears in the text box part of the control.



The **MinDate** and the **MaxDate** properties allow you to put limits on the date range.

Properties of the DateTimePicker Control

The following are some of the commonly used properties of the DateTimePicker control –

Sr.No.	Property & Description
1	**BackColor** Gets or sets a value indicating the background color of the DateTimePicker control.

2	**BackgroundImage** Gets or sets the background image for the control.
3	**BackgroundImageLayout** Gets or sets the layout of the background image of the DateTimePicker control.
4	**CalendarFont** Gets or sets the font style applied to the calendar.
5	**CalendarForeColor** Gets or sets the foreground color of the calendar.
6	**CalendarMonthBackground** Gets or sets the background color of the calendar month.
7	**CalendarTitleBackColor** Gets or sets the background color of the calendar title.
8	**CalendarTitleForeColor** Gets or sets the foreground color of the calendar title.
9	**CalendarTrailingForeColor** Gets or sets the foreground color of the calendar trailing dates.

10	**Checked** Gets or sets a value indicating whether the Value property has been set with a valid date/time value and the displayed value is able to be updated.
11	**CustomFormat** Gets or sets the custom date/time format string.
12	**DropDownAlign** Gets or sets the alignment of the drop-down calendar on the DateTimePicker control.
13	**ForeColor** Gets or sets the foreground color of the DateTimePicker control.
14	**Format** Gets or sets the format of the date and time displayed in the control.
15	**MaxDate** Gets or sets the maximum date and time that can be selected in the control.
16	**MaximumDateTime** Gets the maximum date value allowed for the DateTimePicker control.
17	**MinDate**

	Gets or sets the minimum date and time that can be selected in the control.
18	**MinimumDateTime** Gets the minimum date value allowed for the DateTimePicker control.
19	**PreferredHeight** Gets the preferred height of the DateTimePicker control.
20	**RightToLeftLayout** Gets or sets whether the contents of the DateTimePicker are laid out from right to left.
21	**ShowCheckBox** Gets or sets a value indicating whether a check box is displayed to the left of the selected date.
22	**ShowUpDown** Gets or sets a value indicating whether a spin button control (also known as an up-down control) is used to adjust the date/time value.
23	**Text** Gets or sets the text associated with this control.
24	**Value** Gets or sets the date/time value assigned to the control.

Methods of the DateTimePicker Control

The following are some of the commonly used methods of the DateTimePicker control −

Sr.No.	Method Name & Description
1	**ToString** Returns the string representing the control.

Events of the DateTimePicker Control

The following are some of the commonly used events of the DateTimePicker control −

Sr.No.	Event & Description
1	**BackColorChanged** Occurs when the value of the BackColor property changes.
2	**BackgroundImageChanged** Occurs when the value of the BackgroundImage property changes.
3	**BackgroundImageLayoutChanged** Occurs when the value of the BackgroundImageLayout property changes.
4	**Click** Occurs when the control is clicked.
5	**CloseUp** Occurs when the drop-down calendar is dismissed and disappears.

6	**DoubleClick** Occurs when the control is double-clicked.
7	**DragDrop** Occurs when a drag-and-drop operation is completed.
8	**ForeColorChanged** Occurs when the value of the ForeColor property changes.
9	**FormatChanged** Occurs when the Format property value has changed.
10	**MouseClick** Occurs when the control is clicked with the mouse.
11	**MouseDoubleClick** Occurs when the control is double-clicked with the mouse.
12	**PaddingChanged** Occurs when the value of the Padding property changes.
13	**Paint** Occurs when the control is redrawn.
14	**RightToLeftLayoutChanged**

	Occurs when the RightToLeftLayout property changes.
15	**TextChanged** Occurs when the value of the Text property changes.
16	**ValueChanged** Occurs when the Value property changes.

VB.Net - Label Control

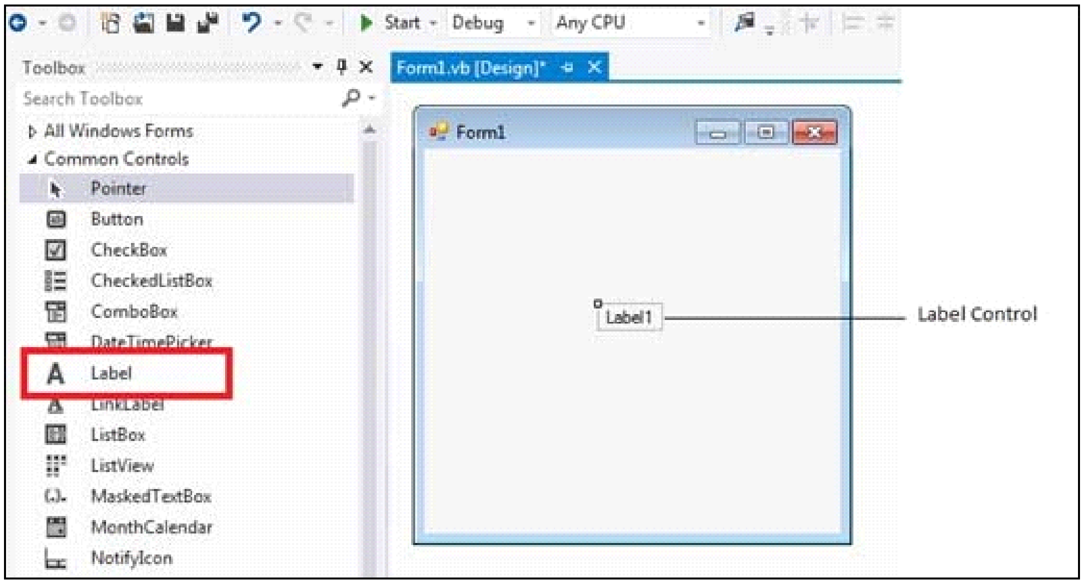
Advertisements

Previous Page

Next Page

The Label control represents a standard Windows label. It is generally used to display some informative text on the GUI which is not changed during runtime.

Let's create a label by dragging a Label control from the Toolbox and dropping it on the form.



Properties of the Label Control

The following are some of the commonly used properties of the Label control –

Sr.No.	Property & Description
1	**Autosize** Gets or sets a value specifying if the control should be automatically resized to display all its contents.
2	**BorderStyle** Gets or sets the border style for the control.
3	**FlatStyle** Gets or sets the flat style appearance of the Label control
4	**Font**

	Gets or sets the font of the text displayed by the control.
5	**FontHeight** Gets or sets the height of the font of the control.
6	**ForeColor** Gets or sets the foreground color of the control.
7	**PreferredHeight** Gets the preferred height of the control.
8	**PreferredWidth** Gets the preferred width of the control.
9	**TabStop** Gets or sets a value indicating whether the user can tab to the Label. This property is not used by this class.
10	**Text** Gets or sets the text associated with this control.
11	**TextAlign** Gets or sets the alignment of text in the label.

Methods of the Label Control

The following are some of the commonly used methods of the Label control −

Sr.No.	Method Name & Description
1	**GetPreferredSize** Retrieves the size of a rectangular area into which a control can be fitted.
2	**Refresh** Forces the control to invalidate its client area and immediately redraw itself and any child controls.
3	**Select** Activates the control.
4	**Show** Displays the control to the user.
5	**ToString** Returns a String that contains the name of the control.

Events of the Label Control

The following are some of the commonly used events of the Label control −

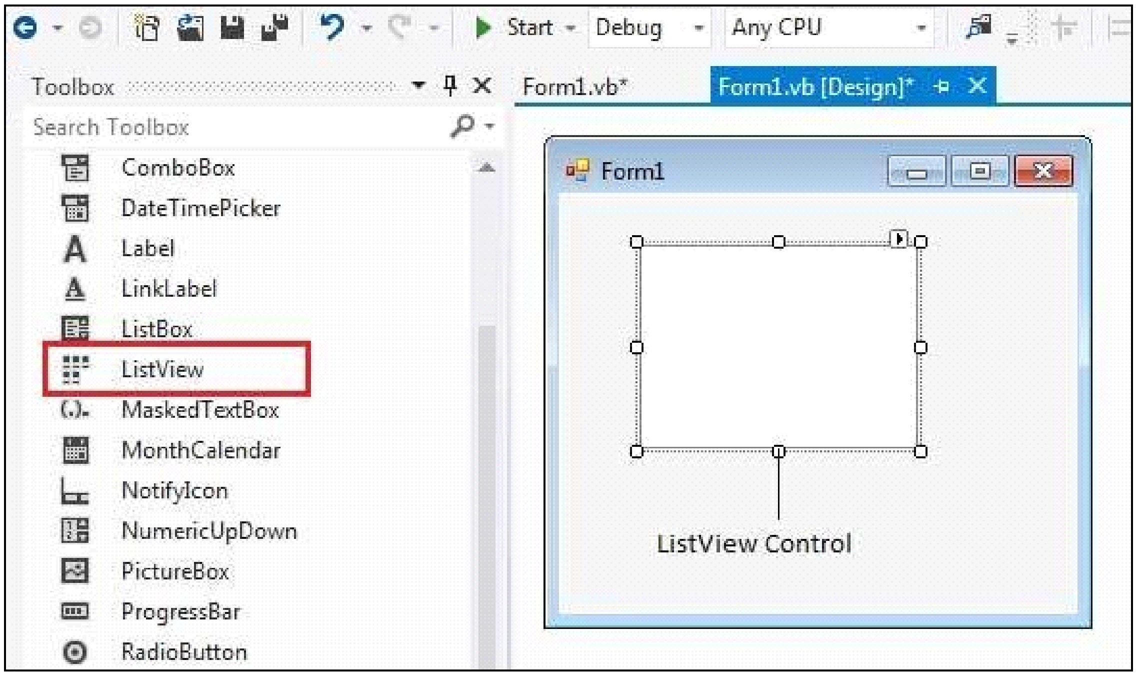
Sr.No.	Event & Description

1	**AutoSizeChanged** Occurs when the value of the AutoSize property changes.
2	**Click** Occurs when the control is clicked.
3	**DoubleClick** Occurs when the control is double-clicked.
4	**GotFocus** Occurs when the control receives focus.
5	**Leave** Occurs when the input focus leaves the control.
6	**LostFocus** Occurs when the control loses focus.
7	**TabIndexChanged** Occurs when the TabIndex property value changes.
8	**TabStopChanged** Occurs when the TabStop property changes.
9	**TextChanged** Occurs when the Text property valu

VB.Net - ListView Control

The ListView control is used to display a list of items. Along with the TreeView control, it allows you to create a Windows Explorer like interface.

Let's click on a ListView control from the Toolbox and place it on the form.



The *ListView* control displays a list of items along with icons. The Item property of the ListView control allows you to add and remove items from it. The *SelectedItem* property contains a collection of the selected items. The *MultiSelect* property allows you to set select more than one item in the list view. The *CheckBoxes* property allows you to set check boxes next to the items.

Properties of the ListView Control

The following are some of the commonly used properties of the ListView control –

Sr.No	Property & Description
1	**Alignment** Gets or sets the alignment of items in the control.
2	**AutoArrange** Gets or sets whether icons are automatically kept arranged.
3	**BackColor** Gets or sets the background color.
4	**CheckBoxes** Gets or sets a value indicating whether a check box appears next to each item in the control.
5	**CheckedIndices** Gets the indexes of the currently checked items in the control.
6	**CheckedItems** Gets the currently checked items in the control.
7	**Columns** Gets the collection of all column headers that appear in the control.
8	**GridLines**

	Gets or sets a value indicating whether grid lines appear between the rows and columns containing the items and subitems in the control.
9	**HeaderStyle** Gets or sets the column header style.
10	**HideSelection** Gets or sets a value indicating whether the selected item in the control remains highlighted when the control loses focus.
11	**HotTracking** Gets or sets a value indicating whether the text of an item or subitem has the appearance of a hyperlink when the mouse pointer passes over it.
12	**HoverSelection** Gets or sets a value indicating whether an item is automatically selected when the mouse pointer remains over the item for a few seconds.
13	**InsertionMark** Gets an object used to indicate the expected drop location when an item is dragged within a ListView control.
14	**Items** Gets a collection containing all items in the control.

15	**LabelWrap** Gets or sets a value indicating whether item labels wrap when items are displayed in the control as icons.
16	**LargeImageList** Gets or sets the ImageList to use when displaying items as large icons in the control.
17	**MultiSelect** Gets or sets a value indicating whether multiple items can be selected.
18	**RightToLeftLayout** Gets or sets a value indicating whether the control is laid out from right to left.
19	**Scrollable** Gets or sets a value indicating whether a scroll bar is added to the control when there is not enough room to display all items.
20	**SelectedIndices** Gets the indexes of the selected items in the control.
21	**SelectedItems** Gets the items that are selected in the control.
22	**ShowGroups** Gets or sets a value indicating whether items

	are displayed in groups.
23	**ShowItemToolTips** Gets or sets a value indicating whether ToolTips are shown for the ListViewItem objects contained in theListView.
24	**SmallImageList** Gets or sets the ImageList to use when displaying items as small icons in the control.
25	**Sorting** Gets or sets the sort order for items in the control.
26	**StateImageList** Gets or sets the ImageList associated with application-defined states in the control.
27	**TopItem** Gets or sets the first visible item in the control.
28	**View** Gets or sets how items are displayed in the control. This property has the following values: • LargeIcon – displays large items with a large 32 x 32 pixels icon. • SmallIcon – displays items with a small 16 x 16 pixels icon • List – displays small icons always in one column

	• Details – displays items in multiple columns with column headers and fields • Tile – displays items as full-size icons with item labels and sub-item information.
29	**VirtualListSize** Gets or sets the number of ListViewItem objects contained in the list when in virtual mode.
30	**VirtualMode** Gets or sets a value indicating whether you have provided your own data-management operations for the ListView control.

Methods of the ListView Control

The following are some of the commonly used methods of the ListView control –

Sr.No.	Method Name & Description
1	**Clear** Removes all items from the ListView control.
1	**ToString** Returns a string containing the string representation of the control.

Events of the ListView Control

The following are some of the commonly used events of the ListView control –

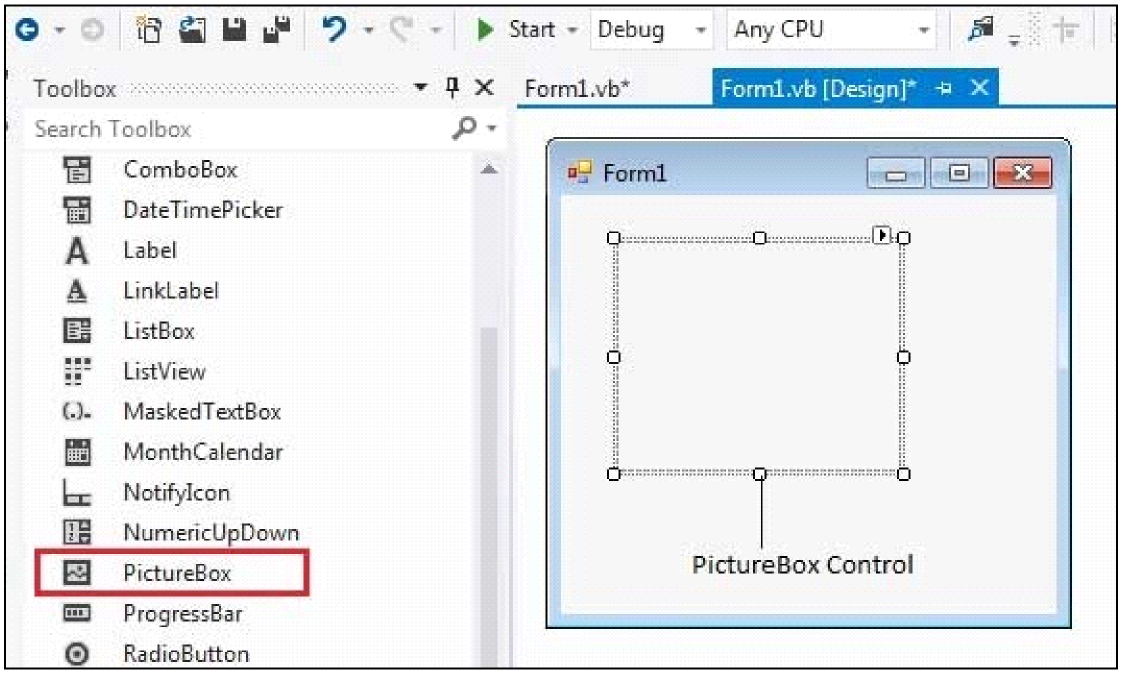
Sr.No.	Event & Description

1	**ColumnClick** Occurs when a column header is clicked.
2	**ItemCheck** Occurs when an item in the control is checked or unchecked.
3	**SelectedIndexChanged** Occurs when the selected index is changed.
4	**TextChanged** Occurs when the Text property is changed.

VB.Net - PictureBox Control

The PictureBox control is used for displaying images on the form. The Image property of the control allows you to set an image both at design time or at run time.

Let's create a picture box by dragging a PictureBox control from the Toolbox and dropping it on the form.



Properties of the PictureBox Control

The following are some of the commonly used properties of the PictureBox control –

Sr.No.	Property & Description
1	**AllowDrop** Specifies whether the picture box accepts data that a user drags on it.
2	**ErrorImage** Gets or specifies an image to be displayed when an error occurs during the image-loading process or if the image load is cancelled.
3	**Image** Gets or sets the image that is displayed in the control.

4	**ImageLocation** Gets or sets the path or the URL for the image displayed in the control.
5	**InitialImage** Gets or sets the image displayed in the control when the main image is loaded.
6	**SizeMode** Determines the size of the image to be displayed in the control. This property takes its value from the PictureBoxSizeMode enumeration, which has values – • **Normal** – the upper left corner of the image is placed at upper left part of the picture box • **StrechImage** – allows stretching of the image • **AutoSize** – allows resizing the picture box to the size of the image • **CenterImage** – allows centering the image in the picture box • **Zoom** – allows increasing or decreasing the image size to maintain the size ratio.
7	**TabIndex** Gets or sets the tab index value.
8	**TabStop** Specifies whether the user will be able to focus

	on the picture box by using the TAB key.
9	**Text** Gets or sets the text for the picture box.
10	**WaitOnLoad** Specifies whether or not an image is loaded synchronously.

Methods of the PictureBox Control

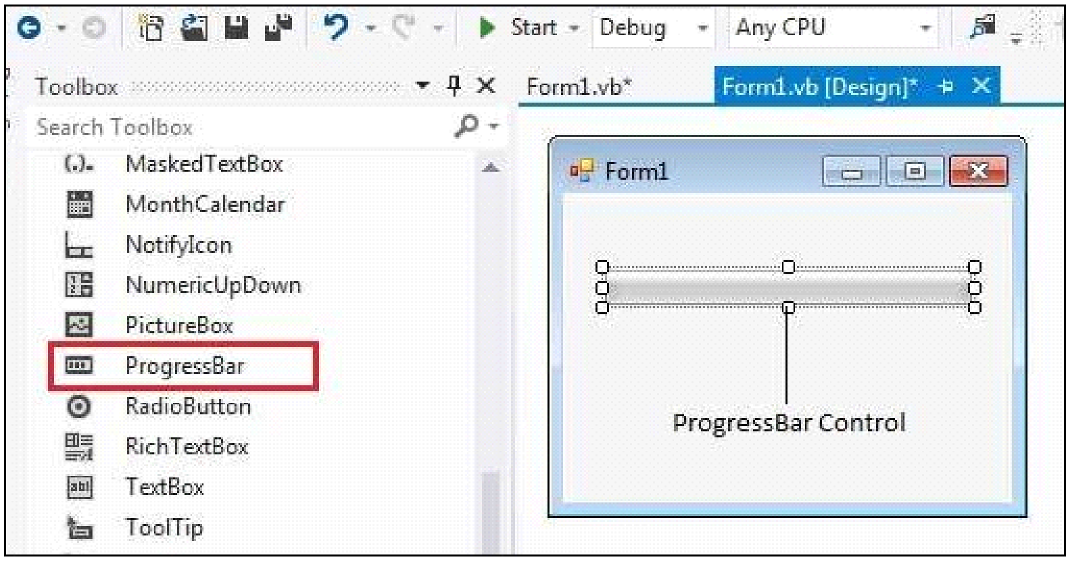
The following are some of the commonly used methods of the PictureBox control −

Sr.No.	Method Name & Description
1	**CancelAsync** Cancels an asynchronous image load.
2	**Load** Displays an image in the picture box
3	**LoadAsync** Loads image asynchronously.
4	**ToString** Returns the string that represents the current picture box.

VB.Net - ProgressBar Control

It represents a Windows progress bar control. It is used to provide visual feedback to your users about the status of some task. It shows a bar that fills in from left to right as the operation progresses.

Let's click on a ProgressBar control from the Toolbox and place it on the form.



The main properties of a progress bar are *Value*, *Maximum* and *Minimum*. The Minimum and Maximum properties are used to set the minimum and maximum values that the progress bar can display. The Value property specifies the current position of the progress bar.

The ProgressBar control is typically used when an application performs tasks such as copying files or printing documents. To a user the application might look unresponsive if there is no visual cue. In such cases, using the ProgressBar allows the programmer to provide a visual status of progress.

Properties of the ProgressBar Control

The following are some of the commonly used properties of the ProgressBar control −

Sr.No.	Property & Description
1	**AllowDrop** Overrides Control.AllowDrop.
2	**BackgroundImage** Gets or sets the background image for the ProgressBar control.
3	**BackgroundImageLayout** Gets or sets the layout of the background image of the progress bar.
4	**CausesValidation** Gets or sets a value indicating whether the control, when it receives focus, causes validation to be performed on any controls that require validation.
5	**Font** Gets or sets the font of text in the ProgressBar.
6	**ImeMode** Gets or sets the input method editor (IME) for the ProgressBar.
7	**ImeModeBase** Gets or sets the IME mode of a control.

8	**MarqueeAnimationSpeed** Gets or sets the time period, in milliseconds, that it takes the progress block to scroll across the progress bar.
9	**Maximum** Gets or sets the maximum value of the range of the control.v
10	**Minimum** Gets or sets the minimum value of the range of the control.
11	**Padding** Gets or sets the space between the edges of a ProgressBar control and its contents.
12	**RightToLeftLayout** Gets or sets a value indicating whether the ProgressBar and any text it contains is displayed from right to left.
13	**Step** Gets or sets the amount by which a call to the PerformStep method increases the current position of the progress bar.
14	**Style** Gets or sets the manner in which progress should be indicated on the progress bar.

15	**Value** Gets or sets the current position of the progress bar.v

Methods of the ProgressBar Control

The following are some of the commonly used methods of the ProgressBar control –

Sr.No.	Method Name & Description
1	**Increment** Increments the current position of the ProgressBar control by specified amount.
2	**PerformStep** Increments the value by the specified step.
3	**ResetText** Resets the Text property to its default value.
4	**ToString** Returns a string that represents the progress bar control.