VB.Net - Dialog Boxes

There are many built-in dialog boxes to be used in Windows forms for various tasks like opening and saving files, printing a page, providing choices for colors, fonts, page setup, etc., to the user of an application. These built-in dialog boxes reduce the developer's time and workload.

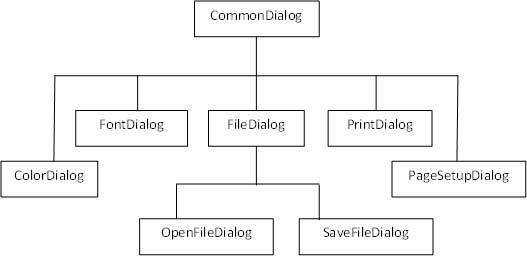
All of these dialog box control classes inherit from the **CommonDialog** class and override the *RunDialog()* function of the base class to create the specific dialog box.

The RunDialog() function is automatically invoked when a user of a dialog box calls its *ShowDialog()* function.

The **ShowDialog** method is used to display all the dialog box controls at run-time. It returns a value of the type of **DialogResult** enumeration. The values of DialogResult enumeration are:

* **Abort** - returns DialogResult.Abort value, when user clicks an Abort button.
* **Cancel**- returns DialogResult.Cancel, when user clicks a Cancel button.
* **Ignore** - returns DialogResult.Ignore, when user clicks an Ignore button.
* **No** - returns DialogResult.No, when user clicks a No button.
* **None** - returns nothing and the dialog box continues running.
* **OK** - returns DialogResult.OK, when user clicks an OK button
* **Retry** - returns DialogResult.Retry , when user clicks an Retry button
* **Yes** - returns DialogResult.Yes, when user clicks an Yes button

The following diagram shows the common dialog class inheritance:



All these above-mentioned classes have corresponding controls that could be added from the Toolbox during design time. You can include relevant functionality of these classes to your application, either by instantiating the class programmatically or by using relevant controls.

When you double click any of the dialog controls in the toolbox or drag the control onto the form, it appears in the Component tray at the bottom of the Windows Forms Designer, they do not directly show up on the form.

The following table lists the commonly used dialog box controls. Click the following links to check their detail:

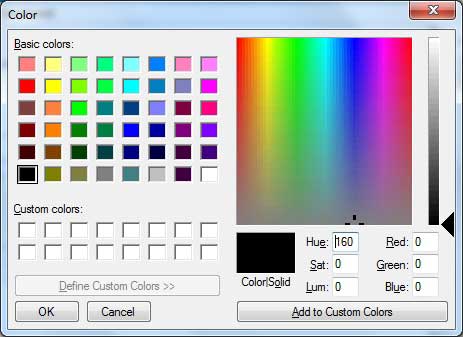
|  |  |
| --- | --- |
| **S.N.** | **Control & Description** |
| 1 | [**ColorDialog**](http://www.tutorialspoint.com/vb.net/vb.net_color_dialog.htm)  It represents a common dialog box that displays available colors along with controls that enable the user to define custom colors. |
| 2 | [**FontDialog**](http://www.tutorialspoint.com/vb.net/vb.net_font_dialog.htm)  It prompts the user to choose a font from among those installed on the local computer and lets the user select the font, font size, and color. |
| 3 | [**OpenFileDialog**](http://www.tutorialspoint.com/vb.net/vb.net_openfile_dialog.htm)  It prompts the user to open a file and allows the user to select a file to open. |
| 4 | [**SaveFileDialog**](http://www.tutorialspoint.com/vb.net/vb.net_savefile_dialog.htm)  It prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data. |
| 5 | [**PrintDialog**](http://www.tutorialspoint.com/vb.net/vb.net_print_dialog.htm)  It lets the user to print documents by selecting a printer and choosing which sections of the document to print from a Windows Forms application. |

# 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:



## Properties of the ColorDialog Control

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

|  |  |  |
| --- | --- | --- |
| **S.N** | **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:

|  |  |
| --- | --- |
| **S.N** | **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:

|  |  |  |
| --- | --- | --- |
| **S.N** | **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

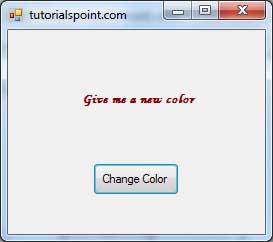
If ColorDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then

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:

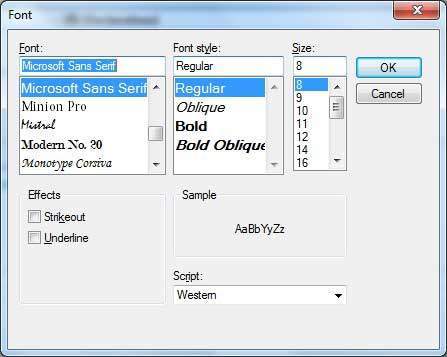


Clicking on the Change Color button, the color dialog appears, select a color and click the OK button. The selected color will be applied as the forecolor of the text of the label.

# VB.Net - FontDialog Control

It prompts the user to choose a font from among those installed on the local computer and lets the user select the font, font size, and color. It returns the Font and Color objects.

Following is the Font dialog box:



By default, the Color ComboBox is not shown on the Font dialog box. You should set the **ShowColor** property of the FontDialog control to be **True**.

## Properties of the FontDialog Control

The following are some of the commonly used properties of the FontDialog control:

|  |  |  |
| --- | --- | --- |
| **S.N** | **Property** | **Description** |
| 1 | **AllowSimulations** | Gets or sets a value indicating whether the dialog box allows graphics device interface (GDI) font simulations. |
| 2 | **AllowVectorFonts** | Gets or sets a value indicating whether the dialog box allows vector font selections. |
| 3 | **AllowVerticalFonts** | Gets or sets a value indicating whether the dialog box displays both vertical and horizontal fonts, or only horizontal fonts. |
| 4 | **Color** | Gets or sets the selected font color. |
| 5 | **FixedPitchOnly** | Gets or sets a value indicating whether the dialog box allows only the selection of fixed-pitch fonts. |
| 6 | **Font** | Gets or sets the selected font. |
| 7 | **FontMustExist** | Gets or sets a value indicating whether the dialog box specifies an error condition if the user attempts to select a font or style that does not exist. |
| 8 | **MaxSize** | Gets or sets the maximum point size a user can select. |
| 9 | **MinSize** | Gets or sets the minimum point size a user can select. |
| 10 | **ScriptsOnly** | Gets or sets a value indicating whether the dialog box allows selection of fonts for all non-OEM and Symbol character sets, as well as the ANSI character set. |
| 11 | **ShowApply** | Gets or sets a value indicating whether the dialog box contains an **Apply** button. |
| 12 | **ShowColor** | Gets or sets a value indicating whether the dialog box displays the color choice. |
| 13 | **ShowEffects** | Gets or sets a value indicating whether the dialog box contains controls that allow the user to specify strikethrough, underline, and text color options. |
| 14 | **ShowHelp** | Gets or sets a value indicating whether the dialog box displays a Help button. |

## Methods of the FontDialog Control

The following are some of the commonly used methods of the FontDialog control:

|  |  |
| --- | --- |
| **S.N** | **Method Name & Description** |
| 1 | **Reset**  Resets all options 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 FontDialog Control

The following are some of the commonly used events of the FontDialog control:

|  |  |  |
| --- | --- | --- |
| **S.N** | **Event** | **Description** |
| 1 | **Apply** | Occurs when the Apply button on the font dialog box is clicked. |

## Example

In this example, let's change the font and color of the text from a rich text control using the Font dialog box. Take the following steps:

* Drag and drop a RichTextBox control, a Button control and a FontDialog control on the form.
* Set the Text property of the button control to 'Change Font'.
* Set the ShowColor property of the FontDialog control to True.
* 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 FontDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then

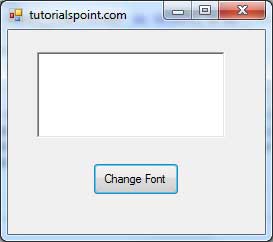
RichTextBox1.ForeColor = FontDialog1.Color

RichTextBox1.Font = FontDialog1.Font

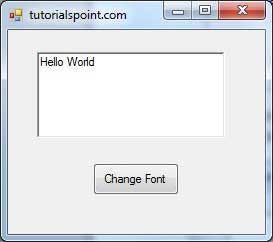
End If

End Sub

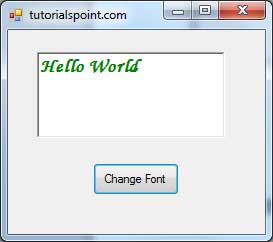
When the application is compiled and run using **Start** button available at the Microsoft Visual Studio tool bar, it will show the following window:



Enter some text and Click on the Change Font button.



The Font dialog appears, select a font and a color and click the OK button. The selected font and color will be applied as the font and fore color of the text of the rich text box.

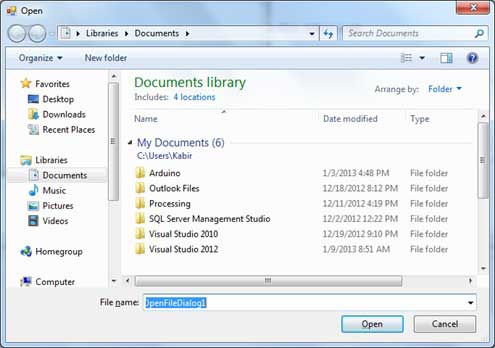


# VB.Net - OpenFileDialog Control

The **OpenFileDialog** control prompts the user to open a file and allows the user to select a file to open. The user can check if the file exists and then open it. The OpenFileDialog control class inherits from the abstract class **FileDialog**.

If the ShowReadOnly property is set to True, then a read-only check box appears in the dialog box. You can also set the ReadOnlyChecked property to True, so that the read-only check box appears checked.

Following is the Open File dialog box:



## Properties of the OpenFileDialog Control

The following are some of the commonly used properties of the OpenFileDialog control:

|  |  |  |
| --- | --- | --- |
| **S.N** | **Property** | **Description** |
| 1 | **AddExtension** | Gets or sets a value indicating whether the dialog box automatically adds an extension to a file name if the user omits the extension. |
| 2 | **AutoUpgradeEnabled** | Gets or sets a value indicating whether this FileDialog instance should automatically upgrade appearance and behavior when running on Windows Vista. |
| 3 | **CheckFileExists** | Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a file name that does not exist. |
| 4 | **CheckPathExists** | Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a path that does not exist. |
| 5 | **CustomPlaces** | Gets the custom places collection for this FileDialog instance. |
| 6 | **DefaultExt** | Gets or sets the default file name extension. |
| 7 | **DereferenceLinks** | Gets or sets a value indicating whether the dialog box returns the location of the file referenced by the shortcut or whether it returns the location of the shortcut (.lnk). |
| 8 | **FileName** | Gets or sets a string containing the file name selected in the file dialog box. |
| 9 | **FileNames** | Gets the file names of all selected files in the dialog box. |
| 10 | **Filter** | Gets or sets the current file name filter string, which determines the choices that appear in the "Save as file type" or "Files of type" box in the dialog box. |
| 11 | **FilterIndex** | Gets or sets the index of the filter currently selected in the file dialog box. |
| 12 | **InitialDirectory** | Gets or sets the initial directory displayed by the file dialog box. |
| 13 | **Multiselect** | Gets or sets a value indicating whether the dialog box allows multiple files to be selected. |
| 14 | **ReadOnlyChecked** | Gets or sets a value indicating whether the read-only check box is selected. |
| 15 | **RestoreDirectory** | Gets or sets a value indicating whether the dialog box restores the current directory before closing. |
| 16 | **SafeFileName** | Gets the file name and extension for the file selected in the dialog box. The file name does not include the path. |
| 17 | **SafeFileNames** | Gets an array of file names and extensions for all the selected files in the dialog box. The file names do not include the path. |
| 18 | **ShowHelp** | Gets or sets a value indicating whether the Help button is displayed in the file dialog box. |
| 19 | **ShowReadOnly** | Gets or sets a value indicating whether the dialog box contains a read-only check box. |
| 20 | **SupportMultiDottedExtensions** | Gets or sets whether the dialog box supports displaying and saving files that have multiple file name extensions. |
| 21 | **Title** | Gets or sets the file dialog box title. |
| 22 | **ValidateNames** | Gets or sets a value indicating whether the dialog box accepts only valid Win32 file names. |

## Methods of the OpenFileDialog Control

The following are some of the commonly used methods of the OpenFileDialog control:

|  |  |
| --- | --- |
| **S.N** | **Method Name & Description** |
| 1 | **OpenFile**  Opens the file selected by the user, with read-only permission. The file is specified by the FileName property. |
| 2 | **Reset**  Resets all options to their default value. |

## Example

In this example, let's load an image file in a picture box, using the open file dialog box. Take the following steps:

* Drag and drop a PictureBox control, a Button control and a OpenFileDialog control on the form.
* Set the Text property of the button control to 'Load Image File'.
* Double-click the Load Image File button and modify the code of the Click event:

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

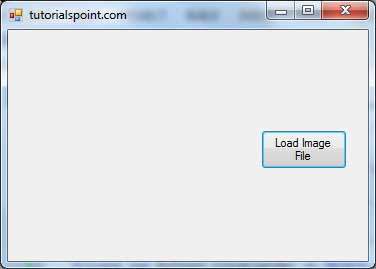
If OpenFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then

PictureBox1.Image = Image.FromFile(OpenFileDialog1.FileName)

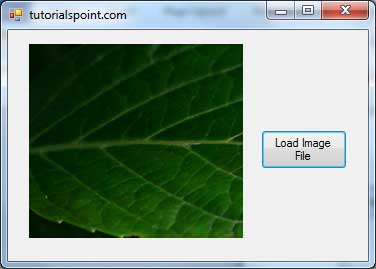
End If

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:



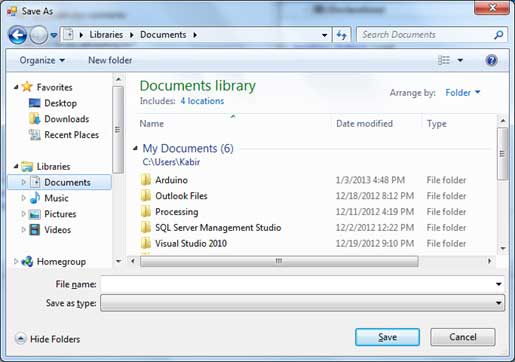
Click on the Load Image File button to load an image stored in your computer.



# VB.Net - SaveFileDialog Control

The **SaveFileDialog** control prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data. The SaveFileDialog control class inherits from the abstract class FileDialog.

Following is the Save File dialog box:



## Properties of the SaveFileDialog Control

The following are some of the commonly used properties of the SaveFileDialog control:

|  |  |  |
| --- | --- | --- |
| **S.N** | **Property** | **Description** |
| 1 | **AddExtension** | Gets or sets a value indicating whether the dialog box automatically adds an extension to a file name if the user omits the extension. |
| 2 | **CheckFileExists** | Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a file name that does not exist. |
| 3 | **CheckPathExists** | Gets or sets a value indicating whether the dialog box displays a warning if the user specifies a path that does not exist. |
| 4 | **CreatePrompt** | Gets or sets a value indicating whether the dialog box prompts the user for permission to create a file if the user specifies a file that does not exist. |
| 5 | **DefaultExt** | Gets or sets the default file name extension. |
| 6 | **DereferenceLinks** | Gets or sets a value indicating whether the dialog box returns the location of the file referenced by the shortcut or whether it returns the location of the shortcut (.lnk). |
| 7 | **FileName** | Gets or sets a string containing the file name selected in the file dialog box. |
| 8 | **FileNames** | Gets the file names of all selected files in the dialog box. |
| 9 | **Filter** | Gets or sets the current file name filter string, which determines the choices that appear in the "Save as file type" or "Files of type" box in the dialog box. |
| 10 | **FilterIndex** | Gets or sets the index of the filter currently selected in the file dialog box. |
| 11 | **InitialDirectory** | Gets or sets the initial directory displayed by the file dialog box. |
| 12 | **OverwritePrompt** | Gets or sets a value indicating whether the Save As dialog box displays a warning if the user specifies a file name that already exists. |
| 13 | **RestoreDirectory** | Gets or sets a value indicating whether the dialog box restores the current directory before closing. |
| 14 | **ShowHelp** | Gets or sets a value indicating whether the Help button is displayed in the file dialog box. |
| 15 | **SupportMultiDottedExtensions** | Gets or sets whether the dialog box supports displaying and saving files that have multiple file name extensions. |
| 16 | **Title** | Gets or sets the file dialog box title. |
| 17 | **ValidateNames** | Gets or sets a value indicating whether the dialog box accepts only valid Win32 file names. |

## Methods of the SaveFileDialog Control

The following are some of the commonly used methods of the SaveFileDialog control:

|  |  |
| --- | --- |
| **S.N** | **Method Name & Description** |
| 1 | **OpenFile**  Opens the file with read/write permission. |
| 2 | **Reset**  Resets all dialog box options to their default values. |

## Example

In this example, let's save the text entered into a rich text box by the user using the save file dialog box. Take the following steps:

* Drag and drop a Label control, a RichTextBox control, a Button control and a SaveFileDialog control on the form.
* Set the Text property of the label and the button control to 'We appreciate your comments' and 'Save Comments', respectively.
* Double-click the Save Comments button and modify the code of the Click event as shown:

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

SaveFileDialog1.Filter = "TXT Files (\*.txt\*)|\*.txt"

If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK \_

Then

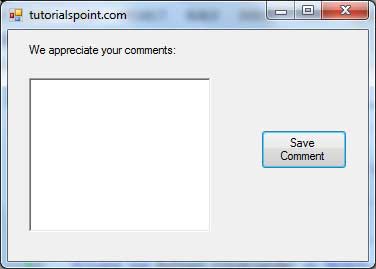
My.Computer.FileSystem.WriteAllText \_

(SaveFileDialog1.FileName, RichTextBox1.Text, True)

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:



We have set the Filter property of the SaveFileDialog control to display text file types with .txt extensions only.

Write some text in the text box and click on the Save Comment button to save the text as a text file in your computer.

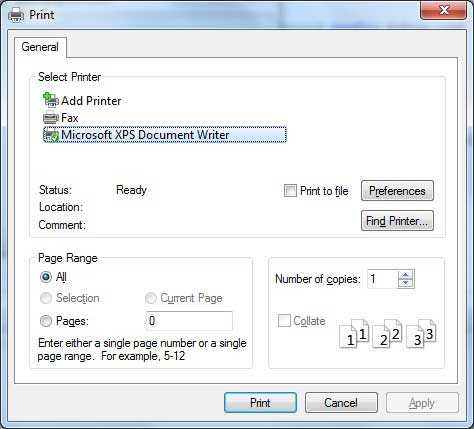
# VB.Net - PrintDialog Control

The PrintDialog control lets the user to print documents by selecting a printer and choosing which sections of the document to print from a Windows Forms application.

There are various other controls related to printing of documents. Let us have a brief look at these controls and their purpose. These other controls are:

* The **PrintDocument** control - it provides support for actual events and operations of printing in Visual Basic and sets the properties for printing.
* The **PrinterSettings** control - it is used to configure how a document is printed by specifying the printer.
* The **PageSetUpDialog** control - it allows the user to specify page-related print settings including page orientation, paper size and margin size.
* The **PrintPreviewControl** control - it represents the raw preview part of print previewing from a Windows Forms application, without any dialog boxes or buttons.
* The **PrintPreviewDialog** control - it represents a dialog box form that contains a PrintPreviewControl for printing from a Windows Forms application.

Following is the Print dialog box:



## Properties of the PrintDialog Control

The following are some of the commonly used properties of the PrintDialog control:

|  |  |  |
| --- | --- | --- |
| **S.N** | **Property** | **Description** |
| 1 | **AllowCurrentPage** | Gets or sets a value indicating whether the **Current Page** option button is displayed. |
| 2 | **AllowPrintToFile** | Gets or sets a value indicating whether the **Print to file** check box is enabled. |
| 3 | **AllowSelection** | Gets or sets a value indicating whether the **Selection**option button is enabled. |
| 4 | **AllowSomePages** | Gets or sets a value indicating whether the **Pages**option button is enabled. |
| 5 | **Document** | Gets or sets a value indicating the PrintDocument used to obtain PrinterSettings. |
| 6 | **PrinterSettings** | Gets or sets the printer settings the dialog box modifies. |
| 7 | **PrintToFile** | Gets or sets a value indicating whether the **Print to file** check box is selected. |
| 8 | **ShowHelp** | Gets or sets a value indicating whether the **Help**button is displayed. |
| 9 | **ShowNetwork** | Gets or sets a value indicating whether the **Network**button is displayed. |

## Methods of the PrintDialog Control

The following are some of the commonly used methods of the PrintDialog control:

|  |  |
| --- | --- |
| **S.N** | **Method Name & Description** |
| 1 | **Reset**  Resets all options 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. |

## Example

In this example, let us see how to show a Print dialog box in a form. Take the following steps:

* Add a PrintDocument control, a PrintDialog control and a Button control on the form. The PrintDocument and the PrintDialog controls are found on the Print category of the controls toolbox.
* Change the text of the button to 'Print'.
* Double-click the Print button and modify the code of the Click event as shown:

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

PrintDialog1.Document = PrintDocument1

PrintDialog1.PrinterSettings = PrintDocument1.PrinterSettings

PrintDialog1.AllowSomePages = True

If PrintDialog1.ShowDialog = DialogResult.OK Then

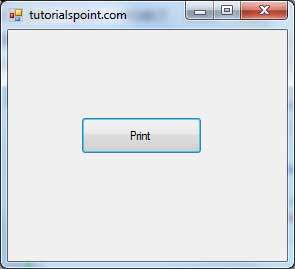
PrintDocument1.PrinterSettings = PrintDialog1.PrinterSettings

PrintDocument1.Print()

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:



Click the Print button to make the Print dialog box appear.

Curtsy: http://www.tutorialspoint.com/vb.net/vb.net\_dialog\_boxes.htm