Chapter 5: Windows Application in C#.NET

Introduction to GUI Programming:

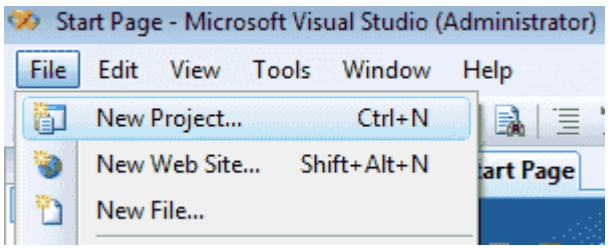
C# has all the features of any powerful, modern language. In C#, the most rapid and convenient way to create your user interface is to do so visually, using the **Windows Forms** Designer and **Toolbox**. Windows Forms controls are reusable components that encapsulate user interface functionality and are used in client side Windows based applications.

A control is a component on a form used to display information or accept user input. The Control class provides the base functionality for all controls that are displayed on a Form.

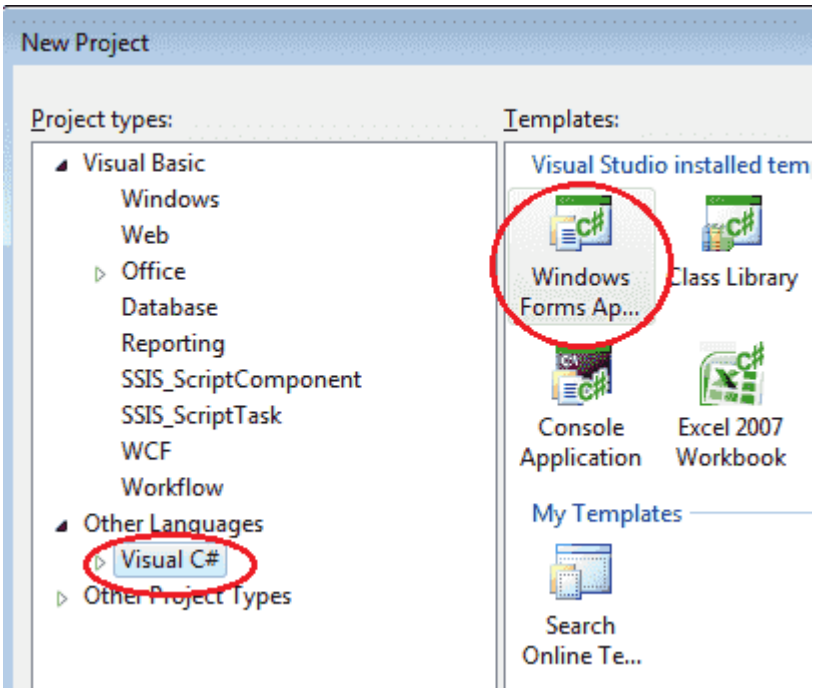
From the following steps you can understand how to place a new control on windows Form.

How to create a new project in C# ?

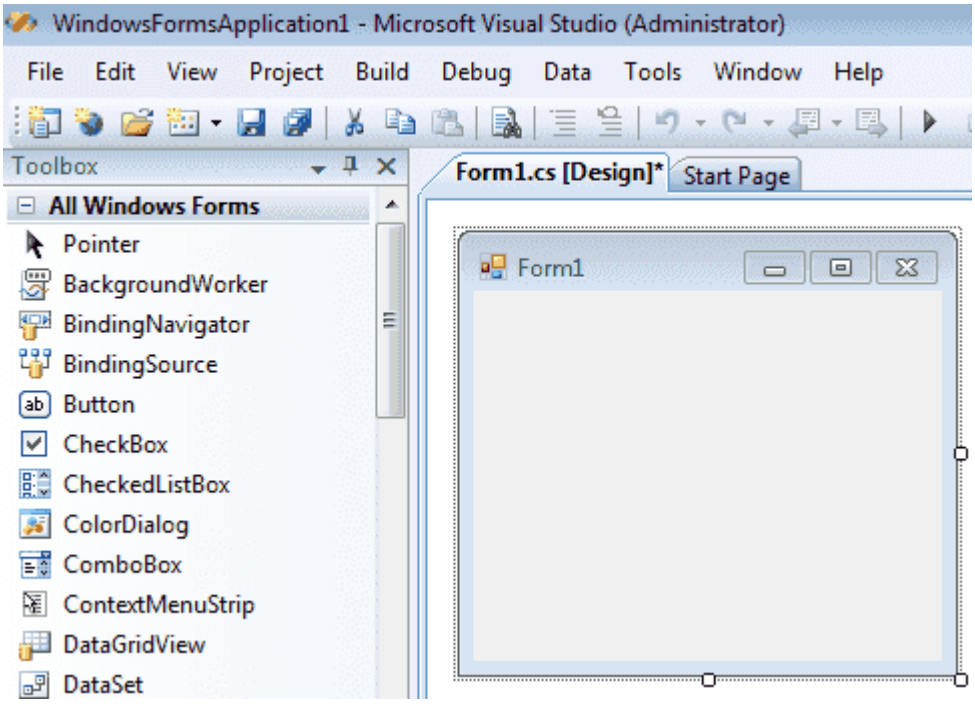
Open your Visual Studio Environment and Click **File->New Project**



Then you will get a **New Project Dialogue Box** asking in which language you want to create a new project.



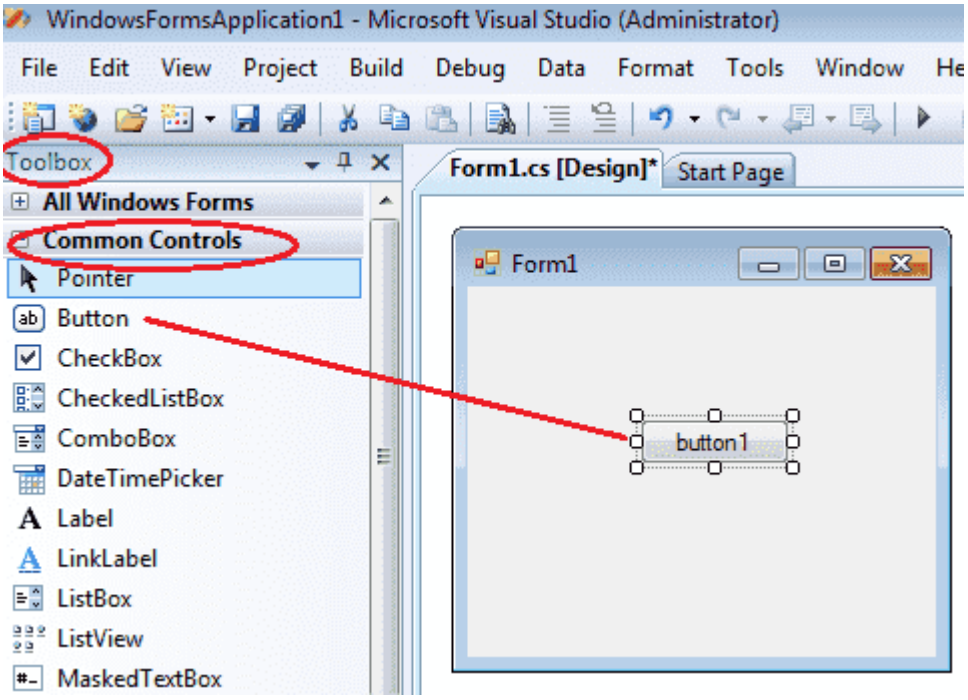
Select Visual C# from the list, then you will get the following screen.



Now you can add controls in your Form Control.

How to add controls to Form ?

In the left side of the Visual Studio Environment you can see the **Tool Box**. There are lots of controls grouping there in the Tool Box according to their functionalities. Just click the + sign before each group then you can see the controls inside the group. You can select basic controls from **Common Controls** group. You can place the control in your Form by drag and drop the control from your toolbox to Form control.



How to drag and drop controls ?

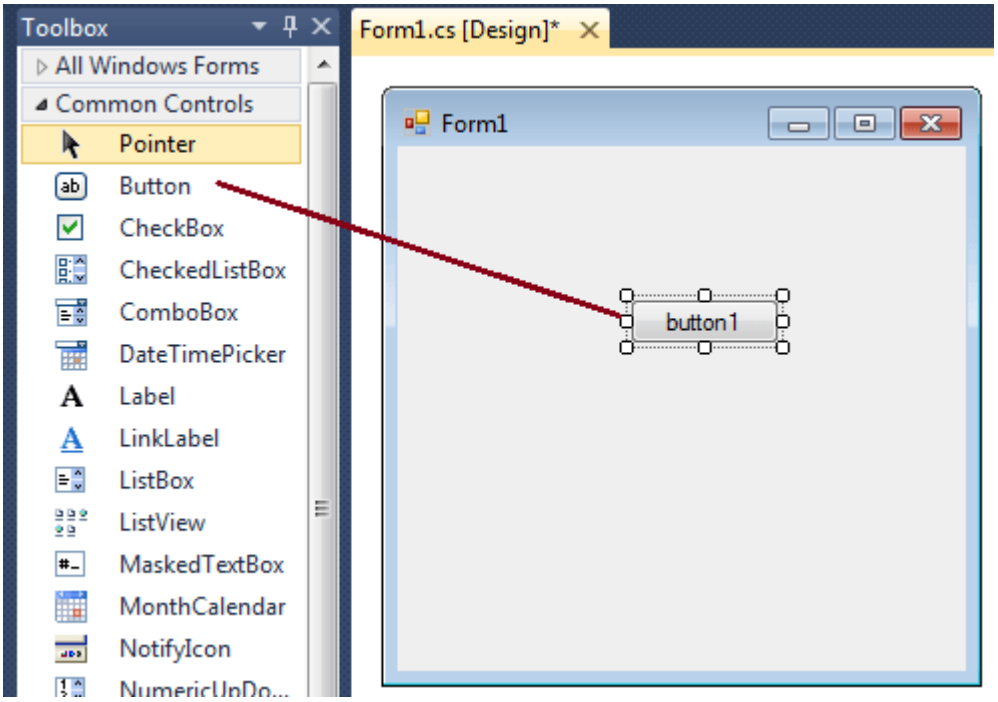
In the above picture we drag and drop the Button control from Toolbox - Common control to Form control.

Now you can start write codes on each control to create your programs.

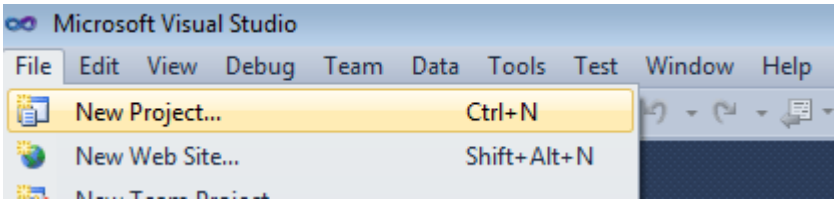
GUI Components/ Controls:

Windows Forms

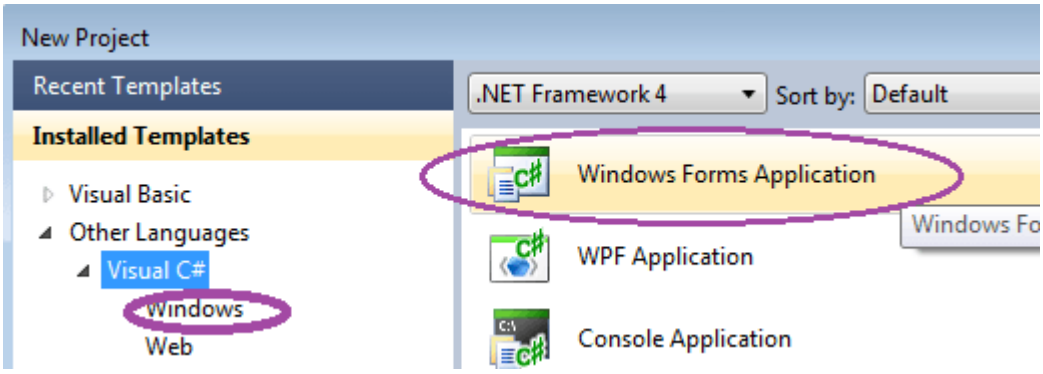
C# programmers have made extensive use of forms to build user interfaces. Each time you create a Windows application, Visual Studio will display a default blank form, onto which you can drag the controls onto your applications main form and adjust their size and position.



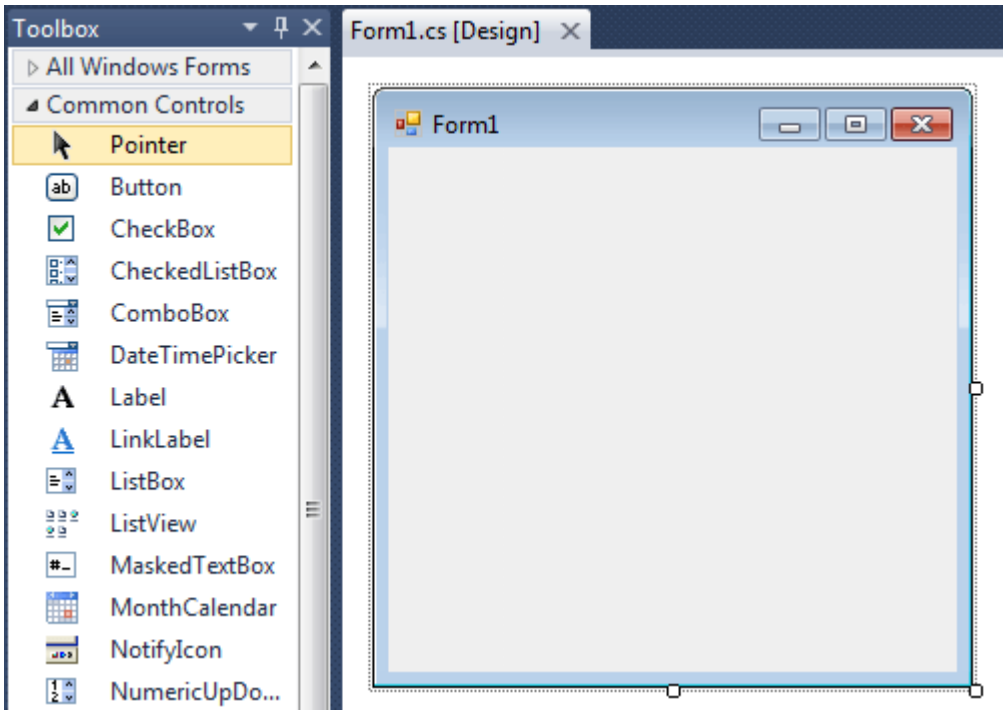
The first step is to start a new project and build a form. Open your Visual Studio and select File->New Project and from the new project dialog box select Other Languages->Visual C# and select Windows Forms Application. Enter a project name at the bottom of the dialouge box and click OK button. The following picture shows how to create a new Form in Visual Studio.



Select Windows Forms Application from New Project dialog box.

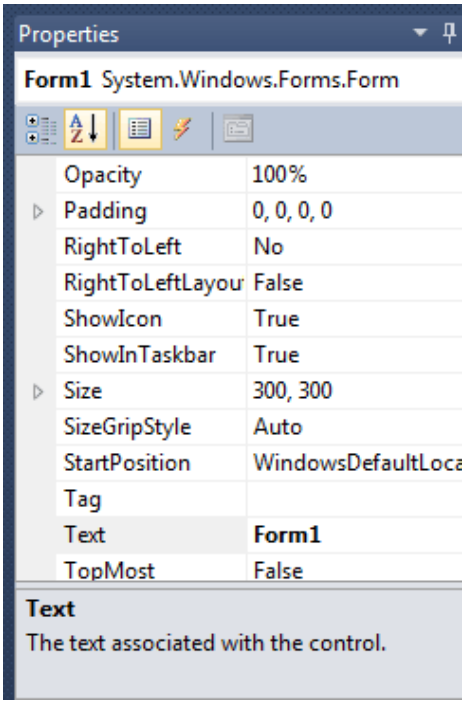


After selecting Windows Forms Application , you can see a default Form (Form1) in your new C# project. The Windows Form you see in Designer view is a visual representation of the window that will open when your application is opened. You can switch between this view and Code view at any time by right-clicking the design surface or code window and then clicking View Code or View Designer. The following picture shows how is the default Form (Form1) looks like.



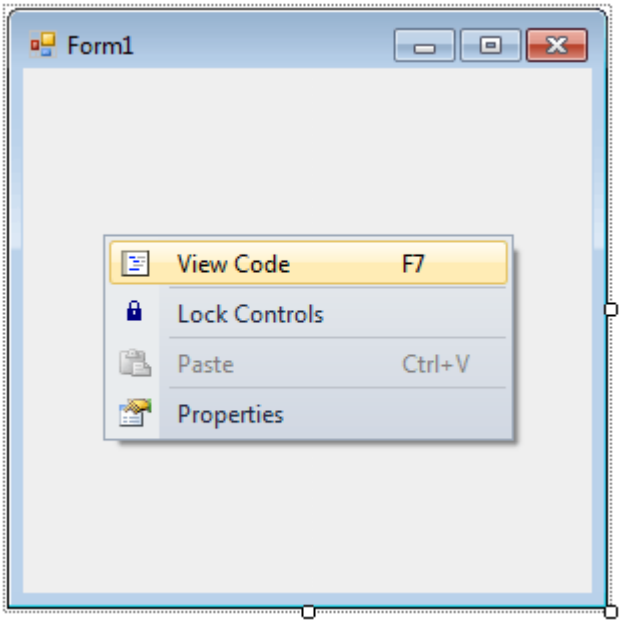
At the top of the form there is a title bar which displays the forms title. Form1 is the default name, and you can change the name to your convenience . The title bar also includes the control box, which holds the minimize, maximize, and close buttons.

If you want to set any properties of the Form, you can use Visual Studio Property window to change it. If you do not see the Properties window, on the View menu, click Properties window. This window lists the properties of the currently selected Windows Form or control, and its here that you can change the existing values.



For example , to change the forms title from Form1 to MyForm, click on Form1 and move to the right side down Properties window, set Text property to MyForm. Then you can see the Title of the form is changed. Likewise you can set any properties of Form through Properties window.

You can also set the properties of the Form1 through coding. For coding, you should right-click the design surface or code window and then clicking View Code.



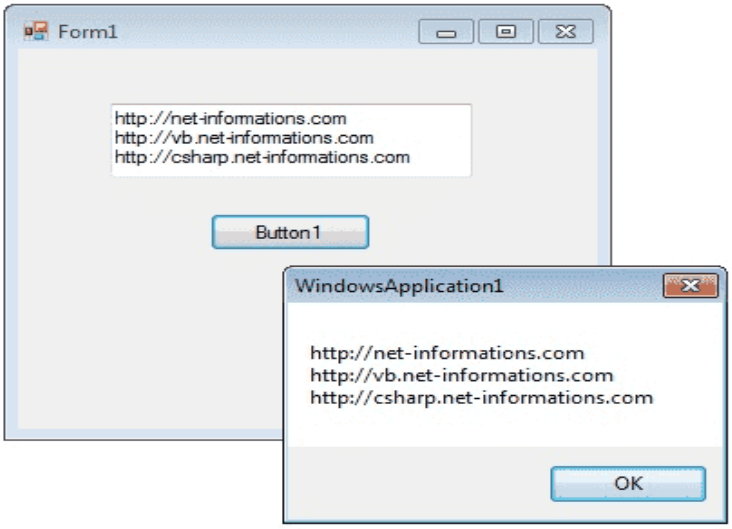
When you right click on Form then you will get code behind window, there you can write your code

For example , if you want to change the back color of the form to Brown , you can code in the Form1_Load event like the following.

```
private void Form1_Load(object sender, EventArgs e)

{

    this.BackColor = Color.Brown;

}
```

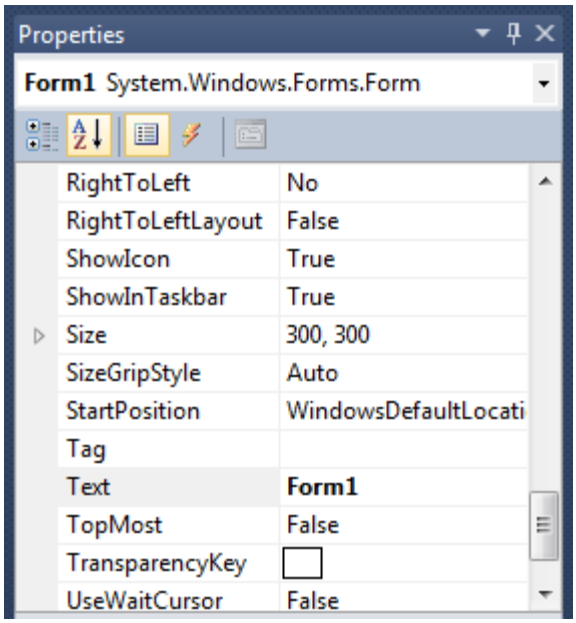
Text Boxes:

A TextBox control is used to display, or accept as input, a single line of text. This control has additional functionality that is not found in the standard Windows text box control, including multiline editing and password character masking.



A text box object is used to display text on a form or to get user input while a C# program is running. In a text box, a user can type data or paste it into the control from the clipboard.

You can set TextBox properties through Property window or through program. You can open Properties window by pressing F4 or right click on a control and select Properties menu item.



The below code set a textbox width as 250 and height as 50 through source code.

```
textBox1.Width = 250;

textBox1.Height = 50;
```

Background Color and Foreground Color

You can set background color and foreground color through property window and programmatically.

```
textBox1.BackColor = Color.Blue;

textBox1.ForeColor = Color.White;
```

Textbox BorderStyle

You can set 3 different types of border style for textbox, they are None, FixedSingle and fixed3d.

textBox1.BorderStyle = BorderStyle.Fixed3D;

TextBox Events

Keydown event

You can capture which key is pressed by the user using KeyDown event

e.g.

```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        MessageBox.Show("You press Enter Key");

    }
    if (e.KeyCode == Keys.CapsLock)
    {
        MessageBox.Show("You press Caps Lock Key");
    }
}
```

TextChanged Event

When user input or setting the Text property to a new value raises the TextChanged event

e.g.

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label1.Text = textBox1.Text;
}
```

Textbox Maximum Length

Sets the maximum number of characters or words the user can input into the text box control.

textBox1.MaxLength = 40;

Textbox ReadOnly

When a program wants to prevent a user from changing the text that appears in a text box, the program can set the controls Read-only property is to True.

textBox1.ReadOnly = true;

Multiline TextBox

You can use the Multiline and ScrollBars properties to enable multiple lines of text to be displayed or entered.

```
textBox1.Multiline = true;
```

Textbox password character

TextBox controls can also be used to accept passwords and other sensitive information. You can use the PasswordChar property to mask characters entered in a single line version of the control

```
textBox1.PasswordChar = '*';
```

The above code set the PasswordChar to * , so when the user enter password then it display only * instead of typed characters.

How to Newline in a TextBox

You can add new line in a textbox using many ways.

```
textBox1.Text += "your text" + "\r\n";
```

OR

```
textBox1.Text += "your text" + Environment.NewLine;
```

How to retrieve integer values from textbox ?

```
int i;

i = int.Parse (textBox1.Text);
```

Parse method Converts the string representation of a number to its integer equivalent.

String to Float conversion

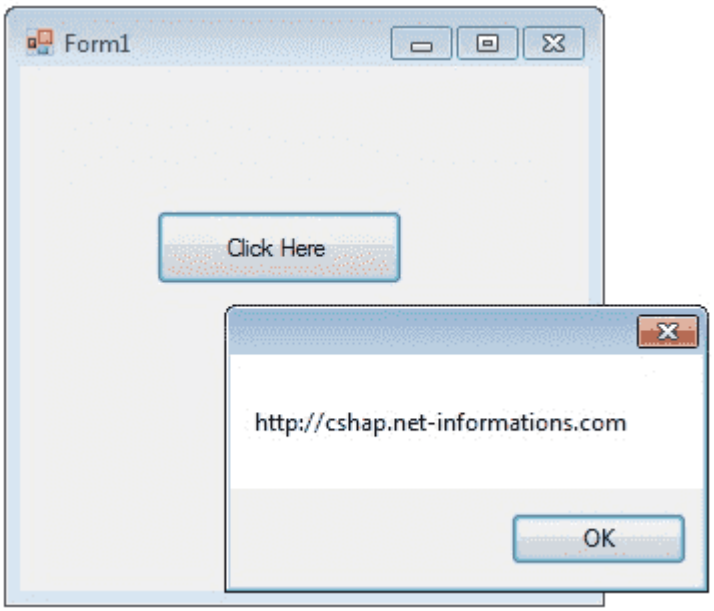
```
float  i;

i = float.Parse (textBox1.Text);
```

String to Double conversion

```
double   i;

i = float.Parse (textBox1.Text);
```

Buttons:

Windows Forms controls are reusable components that encapsulate user interface functionality and are used in client side Windows applications. A button is a control, which is an interactive component that enables users to communicate with an application. The Button class inherits directly from the ButtonBase class. A Button can be clicked by using the mouse, ENTER key, or SPACEBAR if the button has focus.



When you want to change display text of the Button , you can change the Text property of the button.

```
button1.Text = "Click Here";
```

Similarly if you want to load an Image to a Button control , you can code like this.

```
button1.Image = Image.FromFile("C:\\testimage.jpg");
```

Labels:

Labels are one of the most frequently used C# control. We can use the Label control to display text in a set location on the page. Label controls can also be used to add descriptive text to a Form to provide the user with helpful information. The Label class is defined in the System.Windows.Forms namespace.

Add a Label control to the form - Click Label in the Toolbox and drag it over the forms Designer and drop it in the desired location.

If you want to change the display text of the Label, you have to set a new text to the Text property of Label.

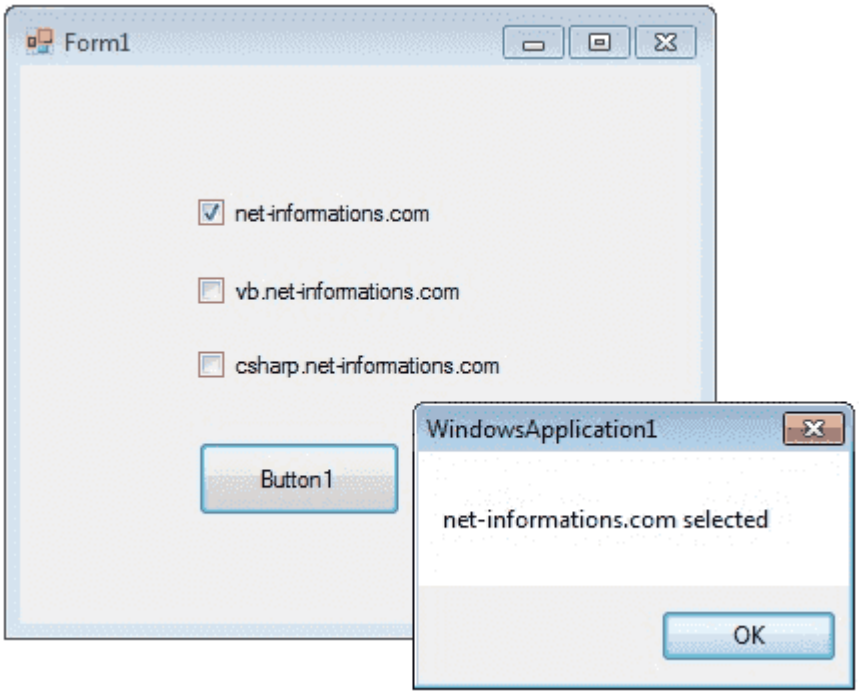
```
label1.Text = "This is my first Label";
```

In addition to displaying text, the Label control can also display an image using the Image property, or a combination of the ImageIndex and ImageList properties.

```
label1.Image = Image.FromFile("C:\\testimage.jpg");
```

Check Boxes:

CheckBoxes allow the user to make multiple selections from a number of options. CheckBox to give the user an option, such as true/false or yes/no. You can click a check box to select it and click it again to deselect it.



The CheckBox control can display an image or text or both. Usually CheckBox comes with a caption, which you can set in the Text property.

```
checkBox1.Text = "Net-informations.com";
```

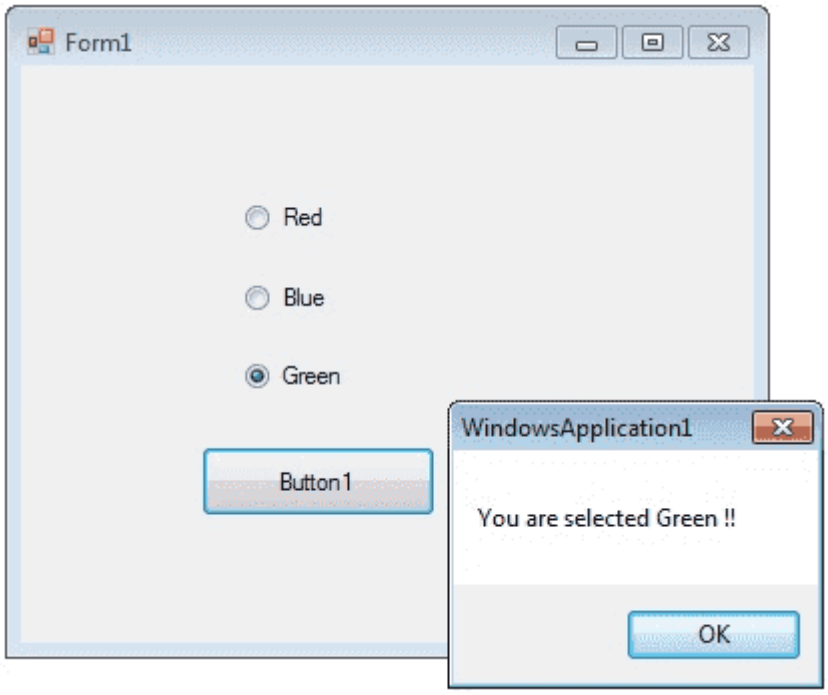
You can use the CheckBox control ThreeState property to direct the control to return the Checked, Unchecked, and Indeterminate values. You need to set the check boxs ThreeState property to True to indicate that you want it to support three states.

```
checkBox1.ThreeState = true;
```

The radio button and the check box are used for different functions. Use a radio button when you want the user to choose only one option.When you want the user to choose all appropriate options, use a check box. The following C# program shows how to find a checkbox is selected or not.

Radio Buttons:

A radio button or option button enables the user to select a single option from a group of choices when paired with other RadioButton controls. When a user clicks on a radio button, it becomes checked, and all other radio buttons with same group become unchecked.



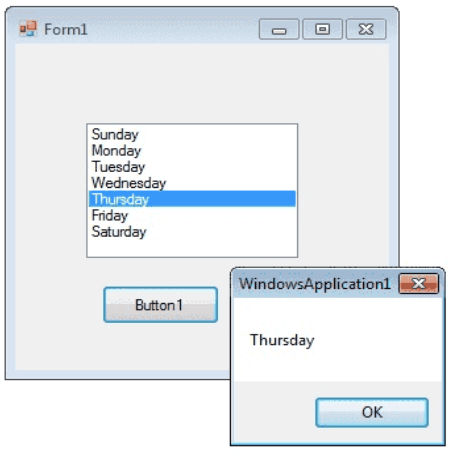
The RadioButton control can display text, an Image, or both. Use the Checked property to get or set the state of a RadioButton.

radioButton1.Checked = true;

The radio button and the check box are used for different functions. Use a radio button when you want the user to choose only one option. When you want the user to choose all appropriate options, use a check box. Like check boxes, radio buttons support a Checked property that indicates whether the radio button is selected.

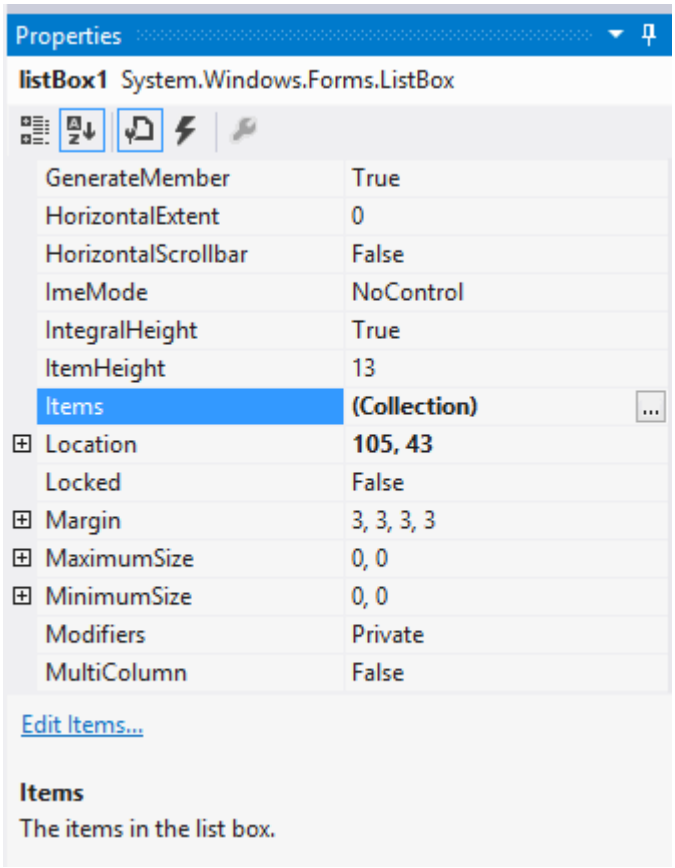
List Boxes:

The **ListBox control** enables you to display a list of items to the user that the user can select by clicking.



Setting ListBox Properties

You can set **ListBox properties** by using Properties Window. In order to get Properties window you can Press F4 or by right-clicking on a control to get the "Properties" menu item.



You can set property values in the right side column of the property window.

Add Items in a Listbox

Syntax

public int Add (object item);

In addition to display and selection functionality, the **ListBox** also provides features that enable you to efficiently add items to the ListBox and to find text within the items of the list. You can use the Add or Insert method to add items to a list box. The **Add method** adds new items at the end of an unsorted list box.

listBox1.Items.Add("Sunday");

If the **Sorted property** of the C# ListBox is set to true, the item is inserted into the list alphabetically. Otherwise, the item is inserted at the end of the ListBox.

Insert Items in a Listbox

Syntax

public void Insert (int index, object item);

You can **inserts** an item into the list box at the specified index.

```
listBox1.Items.Insert(0, "First");
listBox1.Items.Insert(1, "Second");
listBox1.Items.Insert(2, "Third");
listBox1.Items.Insert(3, "Forth");
```

Listbox Selected Item

If you want to retrieve a **single selected** item to a variable , use the following code.

```
string item = listBox1.GetItemText(listBox1.SelectedItem);
```

Or you can use

```
string item = listBox1.SelectedItem.ToString();
```

Or

```
string item= listBox1.Text;
```

Selecting Multiple Items from Listbox

The **SelectionMode property** determines how many items in the list can be selected at a time. A ListBox control can provide single or **multiple selections** using the SelectionMode property . If you change the selection mode property to multiple select , then you will retrieve a collection of items from ListBox1.SelectedItems property.

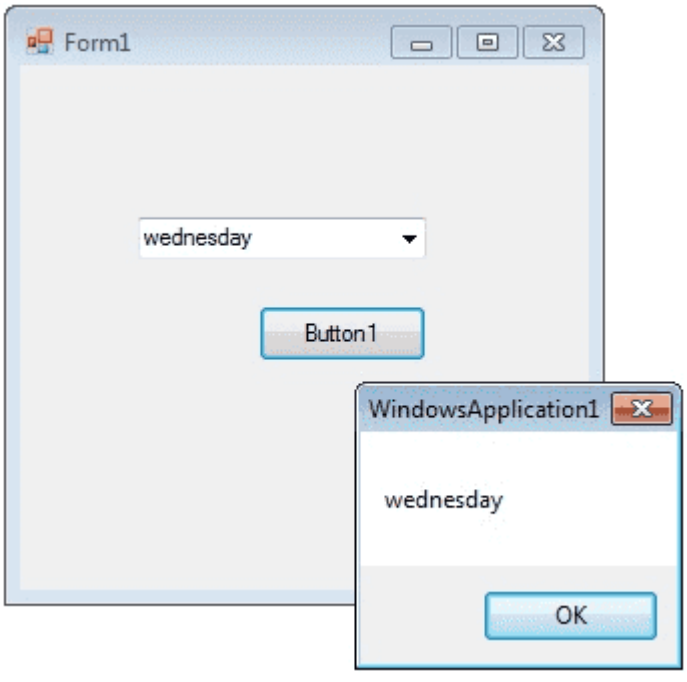
```
listBox1.SelectionMode = SelectionMode.MultiSimple;
```

The ListBox class has two SelectionMode. **Multiple** or **Extended** .

In **Multiple mode** , you can select or deselect any item in a ListBox by clicking it. In Extended mode, you need to hold down the Ctrl key to select additional items or the Shift key to select a range of items.

Combo Boxes:

C# controls are located in the Toolbox of the development environment, and you use them to create objects on a form with a simple series of mouse clicks and dragging motions. A ComboBox displays a text box combined with a ListBox, which enables the user to select items from the list or enter a new value.



The user can type a value in the text field or click the button to display a drop down list. You can add individual objects with the Add method. You can delete items with the Remove method or clear the entire list with the Clear method.

How add a item to combobox

```
comboBox1.Items.Add("Sunday");

comboBox1.Items.Add("Monday");

comboBox1.Items.Add("Tuesday");
```

ComboBox SelectedItem

How to retrieve value from ComboBox

If you want to retrieve the displayed item to a string variable , you can code like this

```
string var;

var = comboBox1.Text;

Or

var item = this.comboBox1.GetItemText(this.comboBox1.SelectedItem);

MessageBox.Show(item);
```

How to remove an item from ComboBox

You can remove items from a combobox in two ways. You can remove item at a the specified index or giving a specified item by name.

```
comboBox1.Items.RemoveAt(1);
```

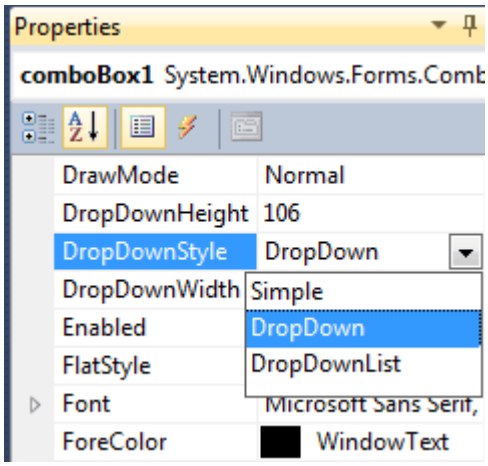
The above code will remove the second item from the combobox.

```
comboBox1.Items.Remove("Friday");
```

The above code will remove the item "Friday" from the combobox.

DropDownStyle

The DropDownStyle property specifies whether the list is always displayed or whether the list is displayed in a drop-down. The DropDownStyle property also specifies whether the text portion can be edited.




```
comboBox1.DropDownStyle = ComboBoxStyle.DropDown;
```

ComboBox Selected Value

How to set the selected item in a comboBox

You can display selected item in a combobox in two ways.

```
comboBox1.Items.Add("test1");

comboBox1.Items.Add("test2");

comboBox1.Items.Add("test3");

comboBox1.SelectedItem = "test3";
```

or

```
comboBox1.SelectedIndex = comboBox1.FindStringExact("test3");
```

ComboBox DataSource Property

How to populate a combo box with a DataSet ?

You can Programmatically Binding DataSource to ComboBox in a simple way..

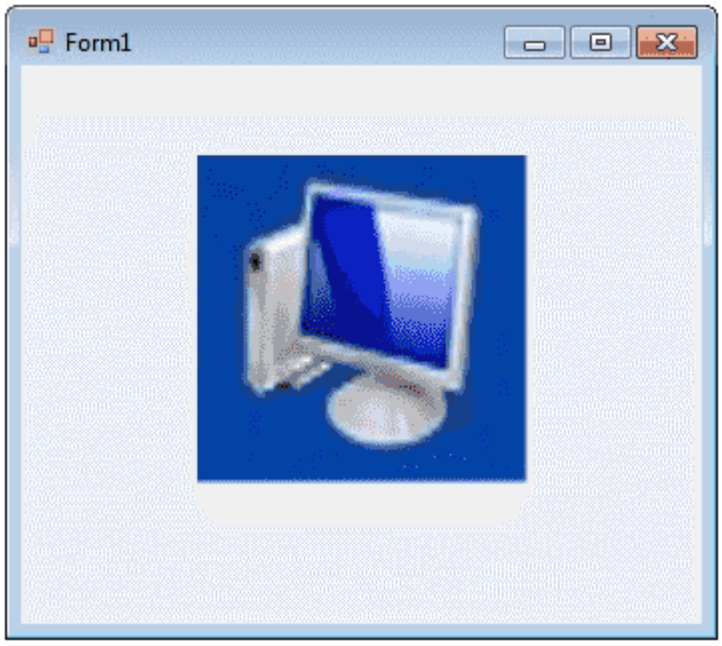
Consider an sql string like...."select au_id,au_lname from authors";

Make a datasource and bind it like the following...

```
comboBox1.DataSource = ds.Tables[0];

comboBox1.ValueMember = "au_id";

comboBox1.DisplayMember = "au_lname";
```

Piture Boxes:

The Windows Forms PictureBox control is used to display images in bitmap, GIF , icon , or JPEG formats.



You can set the Image property to the Image you want to display, either at design time or at run time. You can programmatically change the image displayed in a picture box, which is particularly useful when you use a single form to display different pieces of information.

```
pictureBox1.Image = Image.FromFile("c:\\testImage.jpg");
```

The SizeMode property, which is set to values in the PictureBoxSizeMode enumeration, controls the clipping and positioning of the image in the display area.

```
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
```

There are five different PictureBoxSizeMode is available to PictureBox control.

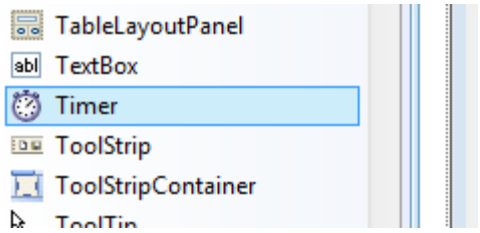
- AutoSize - Sizes the picture box to the image.
- CenterImage - Centers the image in the picture box.
- Normal - Places the upper-left corner of the image at upper left in the picture box
- StretchImage - Allows you to stretch the image in code

The PictureBox is not a selectable control, which means that it cannot receive input focus. The following C# program shows how to load a picture from a file and display it in streach mode.

Timer:

What is C# Timer Control ?

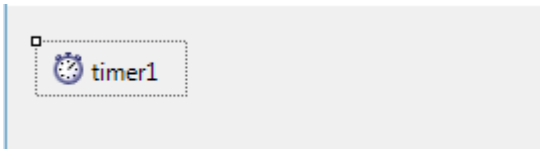
The **Timer Control** plays an important role in the development of programs both Client side and Server side development as well as in Windows Services. With the **Timer Control** we can raise events at a specific interval of time without the interaction of another thread.



Use of Timer Control

We require **Timer Object** in many situations on our development environment. We have to use Timer Object when we want to set an interval between events, periodic checking, to start a process at a fixed time schedule, to increase or decrease the speed in an animation graphics with **time schedule** etc.

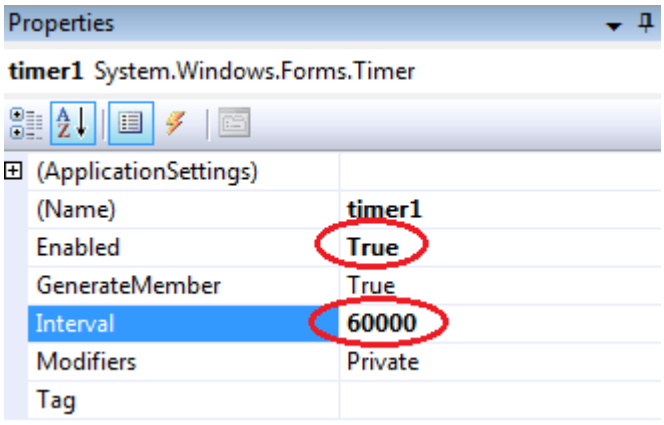
A **Timer control** does not have a visual representation and works as a component in the background.



How to use C# Timer Control ?

We can control programs with Timer Control in **millisecond** , seconds, minutes and even in hours. The Timer Control allows us to set **Interval property** in milliseconds. That is, one second is equal to 1000 milliseconds. For example, if we want to set an interval of 1 minute we set the value at Interval property as 60000, means 60x1000 .

By default the **Enabled property** of Timer Control is False. So before running the program we have to set the Enabled property is True , then only the Timer Control starts its function.



Timer example

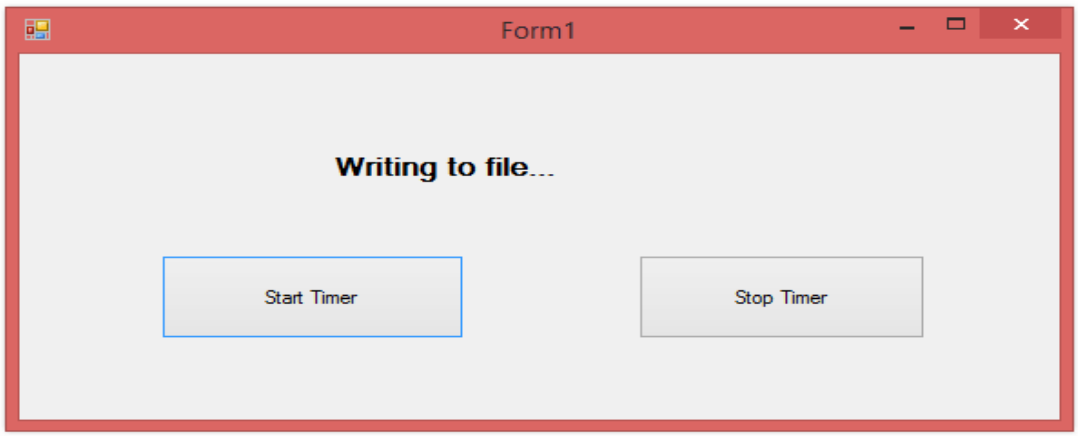
In the following program we display the **current time** in a Label Control. In order to develop this program, we need a **Timer Control** and a Label Control. Here we set the timer interval as 1000 milliseconds, that means one second, for displaying current system time in Label control for the interval of one second.

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
```

```
public partial class Form1 : Form
{
   public Form1()
   {
      InitializeComponent();
   }
   private void timer1_Tick(object sender, EventArgs e)
   {
      label1.Text = DateTime.Now.ToString();
   }   } }
```

Start and Stop Timer Control

The Timer control have included the Start and Stop methods for start and stop the Timer control functions. The following **C# program** shows how to use a timer to write some text to a text file each seconds. The program has two buttons, Start and Stop. The application will write a line to a text file every 1 second once the **Start button** is clicked. The application stops writing to the text file once the **Stop button** is clicked.



Timer Tick Event

Timer **Tick event** occurs when the specified timer interval has elapsed and the timer is enabled.

```
myTimer.Tick += new EventHandler(TimerEventProcessor);
```

Timer Elapsed Event

Timer **Elapsed event** occurs when the interval elapses. The Elapsed event is raised if the Enabled property is true and the **time interval** (in milliseconds) defined by the Interval property elapses.

```
MyTimer.Elapsed += OnTimedEvent;
```

Timer Interval Property

Timer **Interval property** gets or sets the time, in milliseconds, before the Tick event is raised relative to the last occurrence of the **Tick event** .

```
// Sets the timer interval to 1 seconds.
```

myTimer.Interval = 1000;

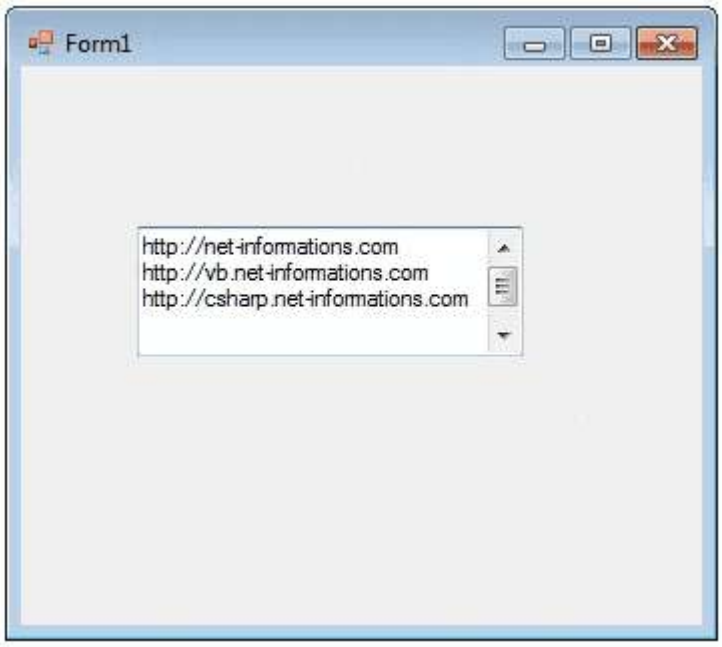
Timer Reset Property

Timer **AutoReset property** gets or sets a Boolean indicating whether the Timer should raise the **Elapsed event** only once (false) or repeatedly (true).

MyTimer.AutoReset = false;

Scrollbars:

A ScrollBar allows you to view content that is outside of the current viewing area by sliding the Thumb to make the content visible.

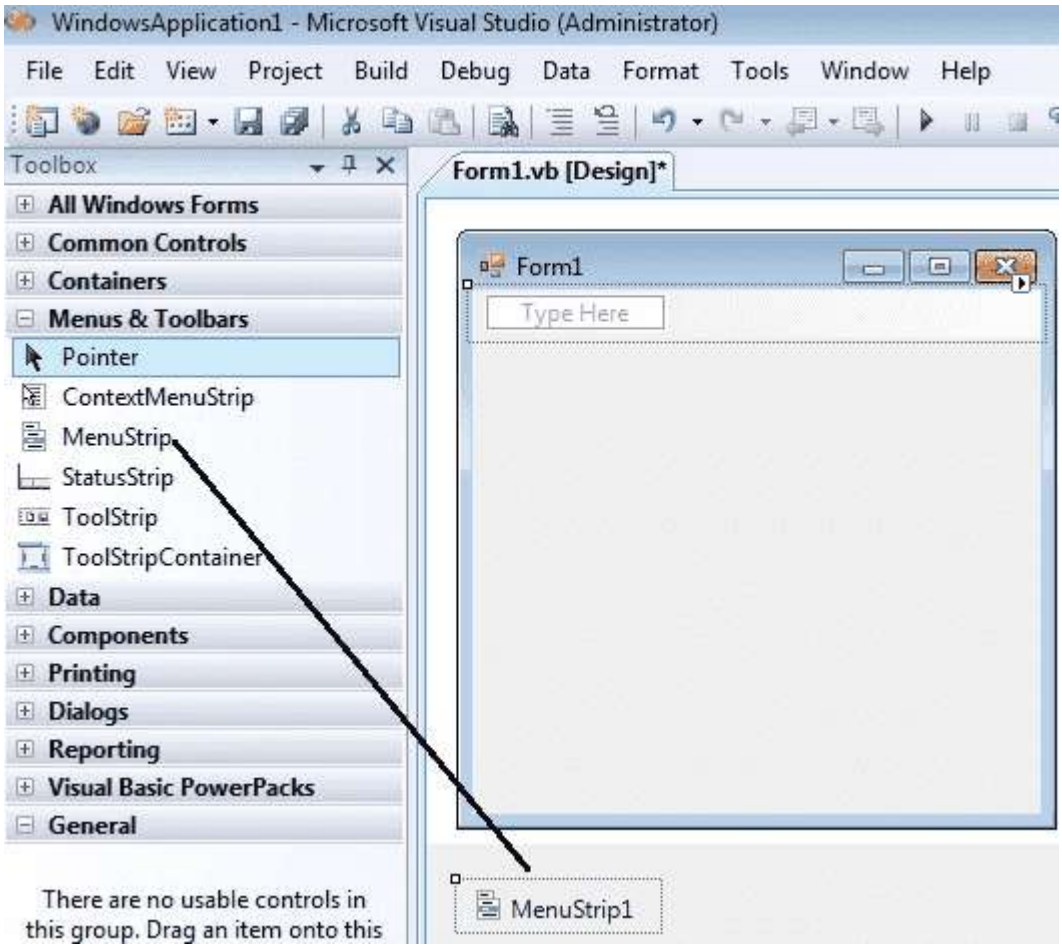


The ScrollBar control contains a Track control. The Track control consists of a Thumb control and two RepeatButton controls. You can increase and decrease the Value property of the ScrollBar control by pressing the RepeatButton controls or by moving the Thumb. You can set the Value property yourself in code, which moves the scroll box to match. The Minimum and Maximum properties determine the range of values that the control can display. The default range of values for the Value property is from 0 to 1.

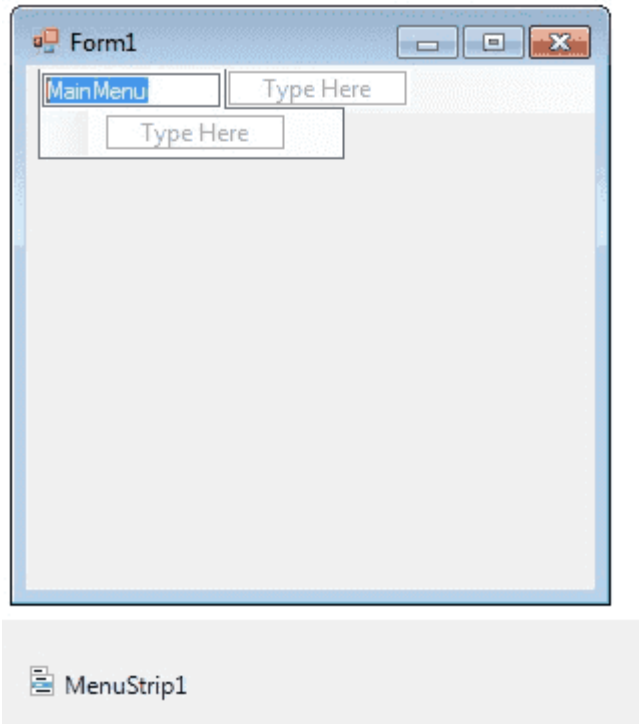
Menus:

A Menu on a Windows Form is created with a MainMenu object, which is a collection of MenuItem objects. MainMenu is the container for the Menu structure of the form and menus are made of MenuItem objects that represent individual parts of a menu.

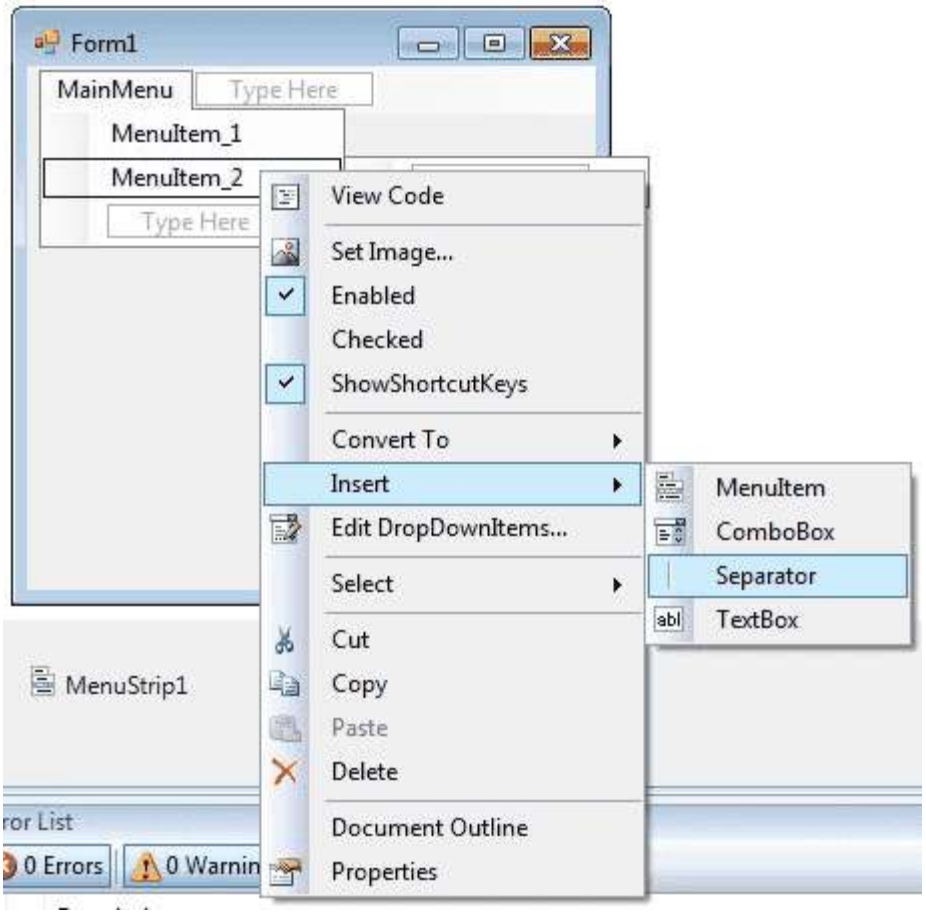
You can add menus to Windows Forms at design time by adding the MainMenu component and then appending menu items to it using the Menu Designer.



After drag the Menustrip on your form you can directly create the menu items by type a value into the "Type Here" box on the menubar part of your form. From the following picture you can understand how to create each menu items on mainmenu Object.



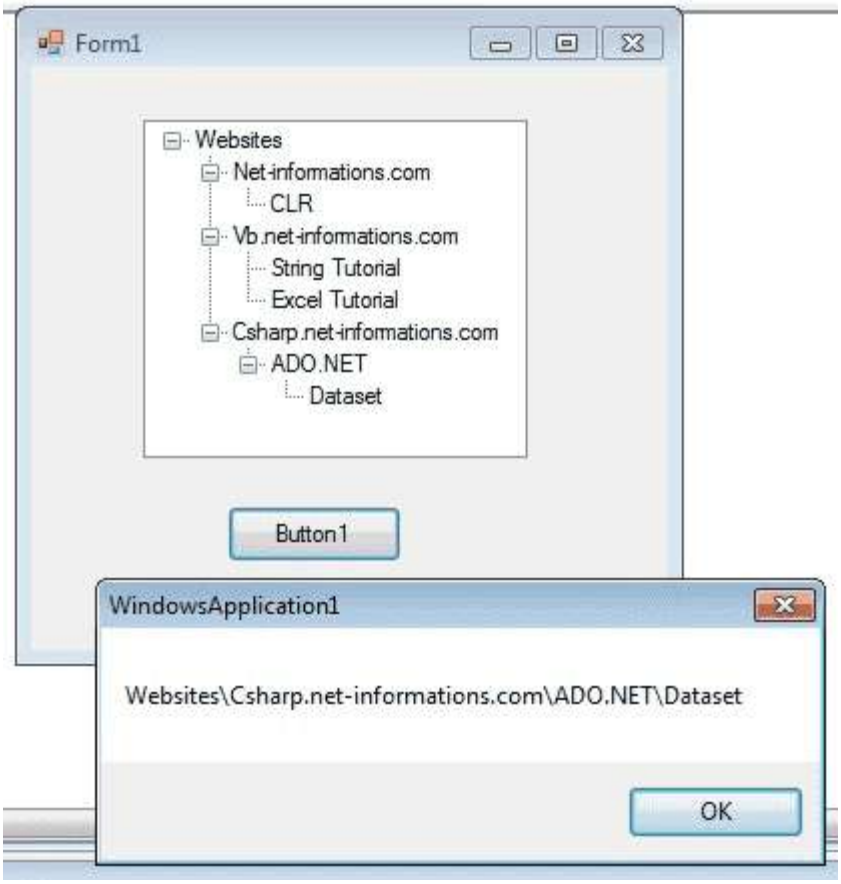
If you need a seperator bar , right click on your menu then go to insert->Seperator.



After creating the Menu on the form , you have to double click on each menu item and write the programs there depends on your requirements. The following C# program shows how to show a messagebox when clicking a Menu item.

Tree Views:

The TreeView control contains a hierarchy of TreeViewItem controls. It provides a way to display information in a hierarchical structure by using collapsible nodes . The top level in a tree view are root nodes that can be expanded or collapsed if the nodes have child nodes.



You can explicitly define the TreeView content or a data source can provide the content. The user can expand the TreeNode by clicking the plus sign (+) button, if one is displayed next to the TreeNode, or you can expand the TreeNode by calling the TreeNode.Expand method. You can also navigate through tree views with various properties: FirstNode, LastNode, NextNode, PrevNode, NextVisibleNode, PrevVisibleNode.

The fullpath method of treeview control provides the path from root node to the selected node.

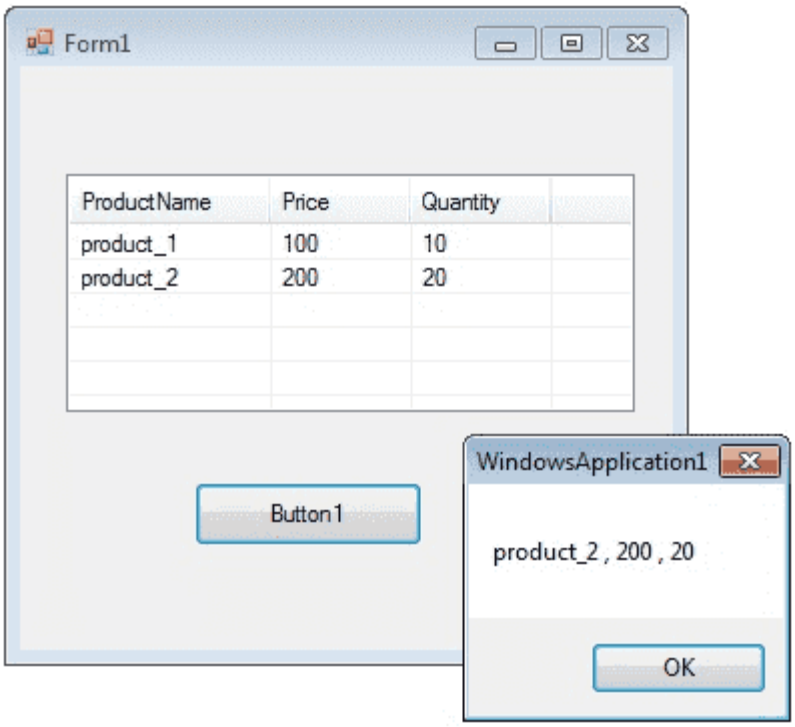
```
treeView1.SelectedNode.FullPath.ToString ();
```

Tree nodes can optionally display check boxes. To display the check boxes, set the CheckBoxes property of the TreeView to true.

```
treeView1.CheckBoxes = true;
```

List Views:

The ListView control is an ItemsControl that is derived from ListBox.



Add Columns in ListView

You can add columns in Listview by using Columns.Add() method. This method takes two arguments, first one is the Column heading and second one the column width.

```
listView1.Columns.Add("ProductName", 100);
```

In the above code, "ProductName" is column heading and 100 is column width.

Add Item in Listview

You can add items in listbox using ListViewItem which represents an item in a ListView control.

```
string[] arr = new string[4];
ListViewItem itm;
//add items to ListView
arr[0] = "product_1";
arr[1] = "100";
arr[2] = "10";
itm = new ListViewItem(arr);
listView1.Items.Add(itm);
```

Get selected item from ListView

```
productName = listView1.SelectedItems[0].SubItems[0].Text;
```

Above code will return the itme from first column of first row.

Sorting Listview Items

If the Sorted property of Listview is set to true, then the ListView items are sorted. The following code sorts the ListView items: ListView1.Sorted = true;