**C# Label Control**

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");**

The following C# source code shows how to set some properties of the Label through coding.

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

label1.Text = "This is my first Lable";

label1.BorderStyle = BorderStyle.FixedSingle;

label1.TextAlign = ContentAlignment.MiddleCenter;

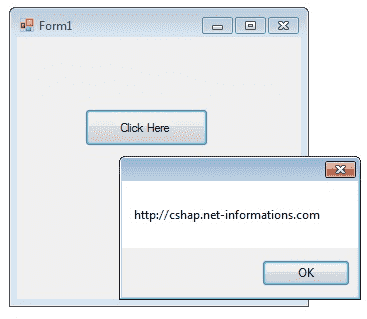
}

}

}

**C# Button Control**

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");**

The following C# source code shows how to change the button Text property while Form loading event and to display a message box when pressing a Button Control.

private void Form1\_Load(object sender, EventArgs e)

{

button1.Text = "Click Here";

}

private void button1\_Click(object sender, EventArgs e)

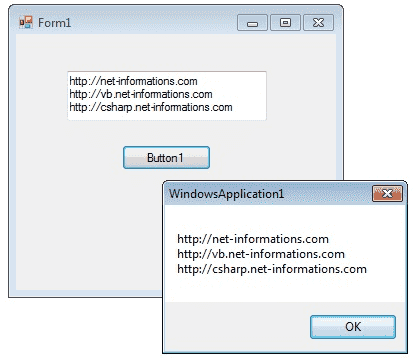
{

MessageBox.Show("http://cshap.net-informations.com");

}

**C# TextBox Control**

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.

For displaying a text in a TextBox control , you can code like this

**textBox1.Text = "http://csharp.net-informations.com";**

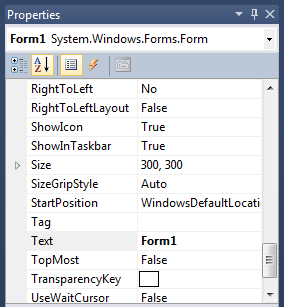
You can also collect the input value from a TextBox control to a variable like this way

**string var;**

**var = textBox1.Text;**

**C# TextBox Properties**

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.

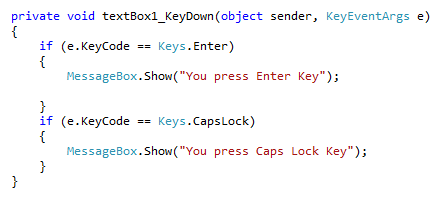
textBox1.BorderStyle = BorderStyle.Fixed3D;

**TextBox Events**

Keydown event

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

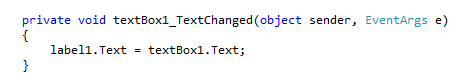
e.g.



TextChanged Event

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

e.g.



**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 passowrd 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 Numeric only textbox ?**

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);

From the following C# source code you can see some important property settings to a TextBox control.

private void Form1\_Load(object sender, EventArgs e)

{

textBox1.Width = 250;

textBox1.Height = 50;

textBox1.Multiline = true;

textBox1.BackColor = Color.Blue;

textBox1.ForeColor = Color.White;

textBox1.BorderStyle = BorderStyle.Fixed3D;

}

private void button1\_Click(object sender, EventArgs e)

{

string var;

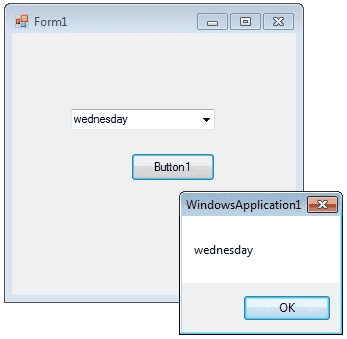
var = textBox1.Text;

MessageBox.Show(var);

}

**C# ComboBox Control**

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.

**Add item to combo box**

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

**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);**

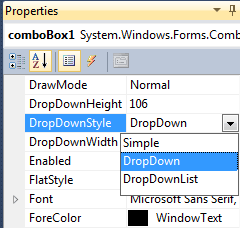
The above code will remove the second item from the combobox.

**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;**

**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");

**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";

**Combobox SelectedIndexChanged event**

The SelectedIndexChanged event of a combobox fire when you change the slected item in a combobox. If you want to do something when you change the selection, you can write the program on SelectedIndexChanged event. From the following code you can understand how to set values in the SelectedIndexChanged event of a combobox. Drag and drop two combobox on the Form and copy and paste the following source code.

private void Form1\_Load(object sender, EventArgs e)

{

comboBox1.Items.Add("weekdays");

comboBox1.Items.Add("year");

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

comboBox2.Items.Clear();

if (comboBox1.SelectedItem == "weekdays")

{

comboBox2.Items.Add("Sunday");

comboBox2.Items.Add("Monday");

comboBox2.Items.Add("Tuesday");

}

else if (comboBox1.SelectedItem == "year")

{

comboBox2.Items.Add("2012");

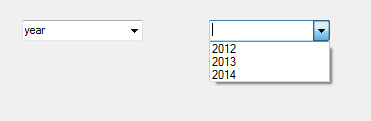
comboBox2.Items.Add("2013");

comboBox2.Items.Add("2014");

}

}

Output



The following C# source code add seven days in a week to a combo box while load event of a Windows Form and int Button click event it displays the selected text in the Combo Box.

private void Form1\_Load(object sender, EventArgs e)

{

comboBox1.Items.Add("Sunday");

comboBox1.Items.Add("Monday");

comboBox1.Items.Add("Tuesday");

comboBox1.Items.Add("wednesday");

comboBox1.Items.Add("Thursday");

comboBox1.Items.Add("Friday");

comboBox1.Items.Add("Saturday");

comboBox1.SelectedIndex = comboBox1.FindStringExact("Sunday");

}

private void button1\_Click(object sender, EventArgs e)

{

string var;

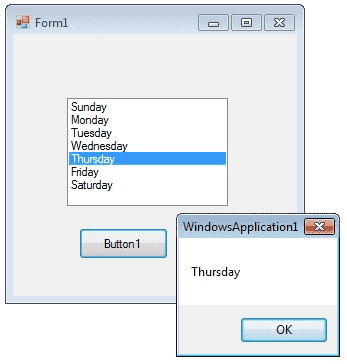
var = comboBox1.Text;

MessageBox.Show(var);

}

**C# ListBox Control**

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



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 you want to retrieve a single selected item to a variable , you can code like this

**string var;**

**var = listBox1.Text;**

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 following C# program initially fill seven days in a week while in the form load event and set the selection mode property to MultiSimple. At the Button click event it will display the selected items.

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

listBox1.Items.Add("Sunday");

listBox1.Items.Add("Monday");

listBox1.Items.Add("Tuesday");

listBox1.Items.Add("Wednesday");

listBox1.Items.Add("Thursday");

listBox1.Items.Add("Friday");

listBox1.Items.Add("Saturday");

listBox1.SelectionMode = SelectionMode.MultiSimple;

}

private void button1\_Click(object sender, EventArgs e)

{

foreach (Object obj in listBox1.SelectedItems )

{

MessageBox.Show(obj.ToString ());

}

}

}

}

**How to bind a ListBox to a List ?**

First you should create a fresh List Object and add items to the List.

List nList = new List();

nList.Add("January");

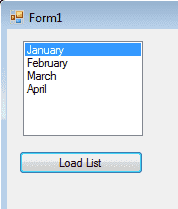
nList.Add("February");

nList.Add("March");

nList.Add("April");

The next step is to bind this List to the Listbox. In order to do that you should set datasource of the Listbox.

**listBox1.DataSource = nList;**



Full Source code

private void button1\_Click(object sender, EventArgs e)

{

List nList = new List();

nList.Add("January");

nList.Add("February");

nList.Add("March");

nList.Add("April");

listBox1.DataSource = nList;

}

**How to bind a listbox to database values ?**

First you should create a connection string and fetch data from database to a Dataset.

connetionString = "Data Source=ServerName;Initial Catalog=databasename;User ID=userid;Password=yourpassword";

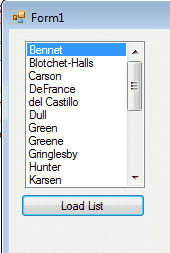
sql = "select au\_id,au\_lname from authors";

After that you should set Listbox datasoure as Dataset.

listBox1.DataSource = ds.Tables[0];

listBox1.ValueMember = "au\_id";

listBox1.DisplayMember = "au\_lname";



using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString = null;

SqlConnection connection;

SqlCommand command;

SqlDataAdapter adapter = new SqlDataAdapter();

DataSet ds = new DataSet();

int i = 0;

string sql = null;

//connetionString = "Data Source=ServerName;Initial Catalog=databasename;User ID=userid;Password=yourpassword";

//sql = "select au\_id,au\_lname from authors";

connection = new SqlConnection(connetionString);

try

{

connection.Open();

command = new SqlCommand(sql, connection);

adapter.SelectCommand = command;

adapter.Fill(ds);

adapter.Dispose();

command.Dispose();

connection.Close();

listBox1.DataSource = ds.Tables[0];

listBox1.ValueMember = "au\_id";

listBox1.DisplayMember = "au\_lname";

}

catch (Exception ex)

{

MessageBox.Show("Cannot open connection ! ");

}

}

}

}

**How to refresh DataSource of a ListBox ?**

**How to clear the Listbox if its already binded with datasource ?**

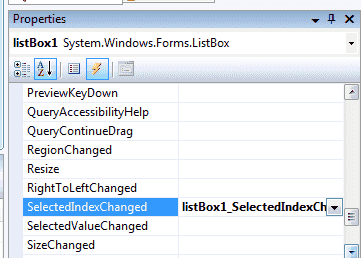
When you want to clear the Listbox, if the ListBox already binded with Datasource, you have to set the Datasource of Listbox as null.

listBox1.DataSource = null;

**How to SelectedIndexChanged event in ListBox ?**

This event is fired when the item selection is changed in a ListBox. You can use this event in a situation that you want select an item from your listbox and accodring to this selection you can perform other programming needs.

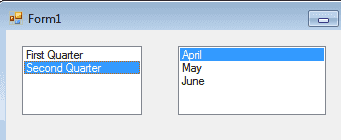
You can add the event handler using the Properties Window and selecting the Event icon and double-clicking on SelectedIndexChanged as you can see in following image.



The event will fire again when you select a new item. You can write your code within SelectedIndexChanged event . When you double click on ListBox the code will automatically come in you code editor like the following image.

SelectedIndexChanged

From the following example you can understand how to fire the SelectedIndexChanged event



First you should drag two listboxes on your Form. First listbox you should set the List as Datasource, the List contents follows:

List nList = new List();

nList.Add("First Quarter");

nList.Add("Second Quarter");

When you load this form you can see the listbox is populated with List and displayed first quarter and second quarter. When you click the "Fist Quarter" the next listbox is populated with first quarter months and when you click "Second Quarter" you can see the second listbox is changed to second quarter months. From the following program you can understand how this happened.

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using System.Collections.Generic;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

List < string > fQ = new List < string > ();

List < string > sQ = new List < string > ();

private void Form1\_Load(object sender, EventArgs e)

{

fQ.Add("January");

fQ.Add("February");

fQ.Add("March");

sQ.Add("April");

sQ.Add("May");

sQ.Add("June");

List < string > nList = new List < string > ();

nList.Add("First Quarter");

nList.Add("Second Quarter");

listBox1.DataSource = nList;

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBox1.SelectedIndex == 0)

{

listBox2.DataSource = null;

listBox2.DataSource = fQ;

}

else if (listBox1.SelectedIndex == 1)

{

listBox2.DataSource = null;

listBox2.DataSource = sQ;

}

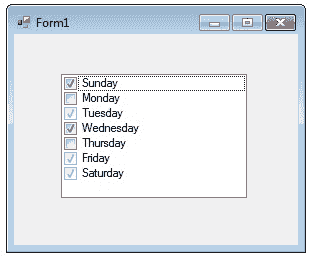
}

}

}

**C# Checked ListBox Control**

The CheckedListBox control gives you all the capability of a list box and also allows you to display a check mark next to the items in the list box.



The user can place a check mark by one or more items and the checked items can be navigated with the CheckedListBox.CheckedItemCollection and CheckedListBox.CheckedIndexCollection. To add objects to the list at run time, assign an array of object references with the AddRange method. The list then displays the default string value for each object.

**Dim days As String() = {"Sunday", "Monday", "Tuesday"};**

**checkedListBox1.Items.AddRange(days);**

You can add individual items to the list with the Add method. The CheckedListBox object supports three states through the CheckState enumeration: Checked, Indeterminate, and Unchecked.

**checkedListBox1.Items.Add("Sunday", CheckState.Checked);**

**checkedListBox1.Items.Add("Monday", CheckState.Unchecked);**

**checkedListBox1.Items.Add("Tuesday", CheckState.Indeterminate);**

private void Form1\_Load(object sender, EventArgs e)

{

checkedListBox1.Items.Add("Sunday", CheckState.Checked);

checkedListBox1.Items.Add("Monday", CheckState.Unchecked);

checkedListBox1.Items.Add("Tuesday", CheckState.Indeterminate);

checkedListBox1.Items.Add("Wednesday", CheckState.Checked);

checkedListBox1.Items.Add("Thursday", CheckState.Unchecked);

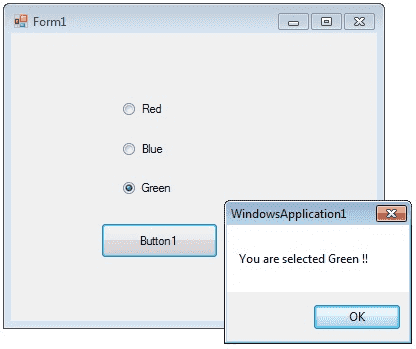
checkedListBox1.Items.Add("Friday", CheckState.Indeterminate);

checkedListBox1.Items.Add("Saturday", CheckState.Indeterminate);

}

**C# RadioButton Control**

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.

private void Form1\_Load(object sender, EventArgs e)

{

radioButton1.Checked = true;

}

private void button1\_Click(object sender, EventArgs e)

{

if (radioButton1.Checked == true)

{

MessageBox.Show ("You are selected Red !! ");

return;

}

else if (radioButton2.Checked == true)

{

MessageBox.Show("You are selected Blue !! ");

return;

}

else

{

MessageBox.Show("You are selected Green !! ");

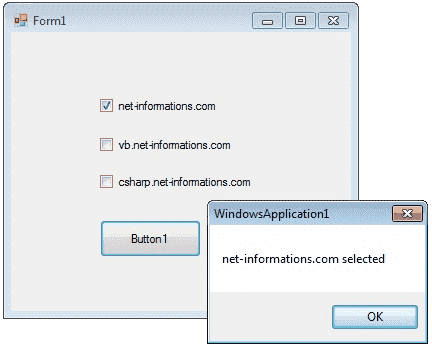
return;

}

}

**C# CheckBox Control**

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.

private void button1\_Click(object sender, EventArgs e)

{

string msg = "";

if (checkBox1.Checked == true)

{

msg = "net-informations.com";

}

if (checkBox2.Checked == true)

{

msg = msg + " vb.net-informations.com";

}

if (checkBox3.Checked == true)

{

msg = msg + " csharp.net-informations.com";

}

if (msg.Length > 0)

{

MessageBox.Show (msg + " selected ");

}

else

{

MessageBox.Show ("No checkbox selected");

}

checkBox1.ThreeState = true;

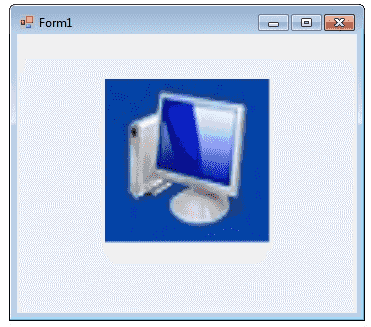
}

}

}

**C# PictureBox Control**

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.

private void Form1\_Load(object sender, EventArgs e)

{

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

pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

}

in the dialog box.

**Exercise 1:**

You are requested to design an ordering form for an e-commerce website. You should include ALL the information below in your design.

* Logo for the company
* Form title
* Product Code, name and unit price
* Quantity ordered
* Total price of the ordered goods
* Picture of the products
* Order button
* Cancel button
* Confirmation Message

You should include the following C#Controls in your design:

Label, Button, Textbox, Combo box/ Listbox, Radio button, Check box, picture box, Group box.

Actions:

1. When the Order button is clicked, the Confirmation Message will display like this:

“You have ordered 5 units of ProductA with total cost equal to HK$500.”

1. When the Cancel button is clicked, all the inputs are reset.
2. Use Combo box for Product Code. When one of the item is selected, respective name and unit price are displayed.
3. Total price is calculated.