So far we have seen how to work with C# to create console based applications. But in a real-life scenario team normally use Visual Studio and C# to create either Windows Forms or Web-based applications.

A windows form application is an application, which is designed to run on a computer. It will not run on web browser because then it becomes a web application.

This Tutorial will focus on how we can create Windows-based applications. We will also learn some basics on how to work with the various elements of Windows applications.

In this tutorial, you will learn-

- Windows Forms Basics
- Hello World in Windows Forms
- Adding Controls to a form
- Event Handling for Controls
- Tree and PictureBox Control

Windows Forms Basics

A Windows forms application is one that runs on the desktop computer. A Windows forms application will normally have a collection of controls such as labels, textboxes, list boxes, etc.

Below is an example of a simple Windows form application. It shows a simple Login screen, which is accessible by the user. The user will enter the required credentials and then will click the Login button to proceed.
So an example of the controls available in the above application

1. This is a collection of label controls which are normally used to describe adjacent controls. So in our case, we have 2 textboxes, and the labels are used to tell the user that one textbox is for entering the user name and the other for the password.
2. The 2 textboxes are used to hold the username and password which will be entered by the user.
3. Finally, we have the button control. The button control will normally have some code attached to perform a certain set of actions. So for example in the above case, we could have the button perform an action of validating the user name and password which is entered by the user.

**C# Hello World**

Now let's look at an example of how we can implement a simple 'hello world' application in Visual Studio. For this, we would need to implement the below-mentioned steps

**Step 1)** The first step involves the creation of a new project in Visual Studio. After launching Visual Studio, you need to choose the menu option New->Project.
Step 2) The next step is to choose the project type as a Windows Forms application. Here we also need to mention the name and location of our project.

1. In the project dialog box, we can see various options for creating different types of projects in Visual Studio. Click the Windows option on the left-hand side.
2. When we click the Windows options in the previous step, we will be able to see an option for Windows Forms Application. Click this option.
3. We then give a name for the application which in our case is DemoApplication. We also need to provide a location to store our application.
4. Finally, we click the 'OK' button to let Visual Studio create our project.

If the above steps are followed, you will get the below
You will see a Form Designer displayed in Visual Studio. It's in this Form Designer that you will start building your Windows Forms application.

In the Solution Explorer, you will also be able to see the DemoApplication Solution. This solution will contain the below 2 project files

1. A Form application called Forms1.cs. This file will contain all of the code for the Windows Form application.
2. The Main program called Program.cs is default code file which is created when a new application is created in Visual Studio. This code will contain the startup code for the application as a whole.
On the left-hand side of Visual Studio, you will also see a ToolBox. The toolbox contains all the controls which can be added to a Windows Forms. Controls like a text box or a label are just some of the controls which can be added to a Windows Forms.

Below is a screenshot of how the Toolbox looks like.

Step 3) In this step, we will now add a label to the Form which will display "Hello World." From the toolbox, you will need to choose the Label control and simply drag it onto the Form.
Once you drag the label to the form, you can see the label embedded on the form as shown below.

**Step 4)** The next step is to go to the properties of the control and change the text to 'Hello World'.

To go to the properties of a control, you need to right-click the control and choose the Properties menu option.
- The properties panel also shows up in Visual Studio. So for the label control, in the properties control, go to the Text section and enter "Hello World".
- Each Control has a set of properties which describe the control.

If you follow all of the above steps and run your program in Visual Studio, you will get the following output

Output: -
In the output, you can see that the Windows Form is displayed. You can also see 'Hello World' is displayed on the form.

**Adding Controls to a form**

We had already seen how to add a control to a form when we added the label control in the earlier section to display "Hello World."

Let's look at the other controls available for Windows forms and see some of their common properties.

In our example, we will create one form which will have the following functionality.

1. The ability for the user to enter name and address.
2. An option to choose the city in which the user resides in
3. The ability for the user to enter an option for the gender.
4. An option to choose a course which the user wants to learn. There will make choices for both C# and ASP.Net

So let's look at each control in detail and add them to build the form with the above-mentioned functionality.

**Group Box**

A group box is used for logical grouping controls into a section. Let's take an example if you had a
collection of controls for entering details such as name and address of a person. Ideally, these are details of a person, so you would want to have these details in a separate section on the Form. For this purpose, you can have a group box. Let's see how we can implement this with an example shown below

**Step 1)** The first step is to drag the Groupbox control onto the Windows Form from the toolbox as shown below

![Groupbox Control](image)

**Step 2)** Once the groupbox has been added, go to the properties window by clicking on the groupbox control. In the properties window, go to the Text property and change it to "User Details".
Once you make the above changes, you will see the following output:

Output:

In the output, you can clearly see that the Groupbox was added to the form. You can also see that the text of the groupbox was changed to "User Details."

**Label Control**

Next comes the Label Control. The label control is used to display a text or a message to the user on the form. The label control is normally used along with other controls. Common examples are wherein a label is added along with the textbox control.

The label indicates to the user on what is expected to
fill up in the textbox. Let's see how we can implement this with an example shown below. We will add 2 labels, one which will be called 'name' and the other called 'address.' They will be used in conjunction with the textbox controls which will be added in the later section.

**Step 1)** The first step is to drag the label control onto the Windows Form from the toolbox as shown below. Make sure you drag the label control 2 times so that you can have one for the 'name' and the other for the 'address'.

![Use the label control](image)

**Step 2)** Once the label has been added, go to the properties window by clicking on the label control. In the properties window, go to the Text property of each label control.

![Change the text for both labels](image)
Once you make the above changes, you will see the following output

**Output:-**

![Image showing a form with labels for name and address]

You can see the label controls added to the form.

**Textbox**

A textbox is used for allowing a user to enter some text on the forms application. Let's see how we can implement this with an example shown below. We will add 2 textboxes to the form, one for the Name and the other for the address to be entered for the user.

**Step 1)** The first step is to drag the textbox control onto the Windows Form from the toolbox as shown below.
Step 2) Once the text boxes have been added, go to the properties window by clicking on the textbox control. In the properties window, go to the Name property and add a meaningful name to each textbox. For example, name the textbox for the user as txtUser and that for the address as txtAddress. A naming convention and standard should be made for controls because it becomes easier to add extra functionality to these controls, which we will see later on.
Once you make the above changes, you will see the following output

Output:-

In the output, you can clearly see that the Textboxes was added to the form.

**List box**

A Listbox is used to showcase a list of items on the Windows form. Let's see how we can implement this with an example shown below. We will add a list box to the form to store some city locations.

**Step 1)** The first step is to drag the list box control onto the Windows Form from the toolbox as shown below.

**Step 2)** Once the list box has been added, go to the properties window by clicking on the list box control.
1. First, change the property of the Listbox box control, in our case, we have changed this to lstCity
2. Click on the Items property. This will allow you to add different items which can show up in the list box. In our case, we have selected items "collection".
3. In the String Collection Editor, which pops up, enter the city names. In our case, we have entered "Mumbai", "Bangalore" and "Hyderabad".
4. Finally, click on the 'OK' button.

Once you make the above changes, you will see the following output

Output:-

In the output, you can see that the Listbox was added to the form. You can also see that the list box has been populated with the city values.
**RadioButton**

A Radiobutton is used to showcase a list of items out of which the user can choose one. Let's see how we can implement this with an example shown below. We will add a radio button for a male/female option.

**Step 1)** The first step is to drag the 'radiobutton' control onto the Windows Form from the toolbox as shown below.

**Step 2)** Once the Radiobutton has been added, go to the properties window by clicking on the Radiobutton control.

1. First, you need to change the text property of both Radio controls. Go the properties windows and
change the text to a male of one radiobutton and the text of the other to female.

2. Similarly, change the name property of both Radio controls. Go the properties windows and change the name to 'rdMale' of one radiobutton and to 'rdfemale' for the other one.

One you make the above changes, you will see the following output

**Output:**

![Radio buttons added](image)

You will see the Radio buttons added to the Windows form.

**Checkbox**

A checkbox is used to provide a list of options in which the user can choose multiple choices. Let's see how we can implement this with an example shown below. We will add 2 checkboxes to our Windows forms. These checkboxes will provide an option to the user on whether they want to learn C# or ASP.Net.

**Step 1)** The first step is to drag the checkbox control onto the Windows Form from the toolbox as shown below
Step 2) Once the checkbox has been added, go to the properties window by clicking on the Checkbox control.

In the properties window,

1. First, you need to change the text property of both checkbox controls. Go the properties windows and change the text to C# and ASP.Net.
2. Similarly, change the name property of both Radio controls. Go the properties windows and change the name to chkC of one checkbox and to chkASP for the other one.

Once you make the above changes, you will see the following output

Output:-
A button is used to allow the user to click on a button which would then start the processing of the form. Let's see how we can implement this with an example shown below. We will add a simple button called 'Submit' which will be used to submit all the information on the form.

**Step 1)** The first step is to drag the button control onto the Windows Form from the toolbox as shown below.

**Step 2)** Once the Button has been added, go to the properties window by clicking on the Button control.
1. First, you need to change the text property of the button control. Go the properties windows and change the text to 'submit'.

2. Similarly, change the name property of the control. Go the properties windows and change the name to 'btnSubmit'.

Once you make the above changes, you will see the following output

Output:-

Congrats, you now have your first basic Windows Form in place. Let's now go to the next topic to see
how we can do Event handling for Controls.

C# Event Handling for Controls

When working with windows form, you can add events to controls. An event is something that happens when an action is performed. Probably the most common action is the clicking of a button on a form. In Windows Forms, you can add code which can be used to perform certain actions when a button is pressed on the form.

Normally when a button is pressed on a form, it means that some processing should take place.

Let's take a look at one of the event and how it can be handled before we go to the button event scenario.

The below example will showcase an event for the Listbox control. So whenever an item is selected in the listbox control, a message box should pop up which shows the item selected. Let's perform the following steps to achieve this.

Step 1) Double click on the Listbox in the form designer. By doing this, Visual Studio will automatically open up the code file for the form. And it will automatically add an event method to the code. This event method will be triggered, whenever any item in the listbox is selected.

Above is the snippet of code which is automatically added by Visual Studio, when you double-click the Listbox control on the form. Now let's add the below section of code to this snippet of code, to add the required functionality to the listbox event.
1. This is the event handler method which is automatically created by Visual Studio when you double-click the List box control. You don't need to worry about the complexity of the method name or the parameters passed to the method.

2. Here we are getting the SelectedItem through the lstCity.SelectedItem property. Remember that lstCity is the name of our Listbox control. We then use the GetItemText method to get the actual value of the selected item. We then assign this value to the text variable.

3. Finally, we use the MessageBox method to display the text variable value to the user.

One you make the above changes, and run the program in Visual Studio you will see the following output

**Output:-**

From the output, you can see that when any item from the list box is selected, a message box will pops up. This will show the selected item from the listbox.

Now let's look at the final control which is the button click Method. Again this follows the same philosophy. Just double click the button in the Forms Designer and it will automatically add the method for the button event handler. Then you just need to add the below
1. This is the event handler method which is automatically created by Visual Studio when you double click the button control. You don't need to worry on the complexity of the method name or the parameters passed to the method.

2. Here we are getting values entered in the name and address textbox. The values can be taken from the text property of the textbox. We then assign the values to 2 variables, name, and address accordingly.

3. Finally, we use the MessageBox method to display the name and address values to the user.

One you make the above changes, and run the program in Visual Studio you will see the following output

**Output:-**

1. First, enter a value in the name and address field.
2. Then click on the Submit button

Once you click the Submit button, a message box will pop, and it will correctly show you what you entered in the user details section.

**Summary**

- A Windows forms application is one that runs on the desktop of a computer. Visual Studio along
with C# can be used to create a Windows Forms application.

- Controls can be added to the Windows forms via the Toolbox in Visual Studio. Controls such as labels, checkboxes, radio buttons, etc. can be added to the form via the toolbox.
- One can also use advanced controls like the tree view control and the PictureBox control.
- Event handlers are used to respond to events generated from controls. The most common one is the one added for the button clicked event.