WINDOWS FORM PROGRAMMING WITH C#

- Working with Forms

The most basic Windows Applications that can be created in .NET is the blank form. A sample of it is shown in Figure 1.

- Creating a Windows Application

To create a new Windows Application, from the **File** menu, select **New** and then select **Solution...**.

- This opens the **New Project** dialog box.
- From this dialog you can choose from many different project templates that are grouped into different categories.
- **Source**

In this part, you will see the source code of your windows application.

- **Design**

An overview of your Windows Application.

- **Building the Application**
- From the **Build** menu select either **Build Solution** or **Build <Your Project Name>**. (in our example, Build CreatingForm)
- **Running the Application**

From the **Debug** menu select **Run** or **Run without debugger**.

- Your application will then be started and its main form will be displayed.
- **Adding a New Form**

SharpDevelop provides many different types of file templates which you can add to your project:

- Configuration classes
- Class
- Form
- Interface
- Module
- MSBuild File
- Setup Dialog (WiX)
- Setup Document (WiX)
- Singleton Class

- Struct

To add a new form to your Windows Application, open the **Project Explorer**, if it is not already open, by selecting **Projects** from the **View** menu

- In the **Projects Explorer** select the name of your project, right click, select **Add** and then **New Item...**.
- This opens the **New File** dialog box
- The Toolbox
- Let's have a closer look at the toolbox. If you haven't already, move your mouse pointer over the toolbox on the left of the screen, and pin it to the foreground by clicking the pin at the top right of the panel that unfolds:
- The toolbox contains a selection of all the controls available to you as a .NET developer. In particular, it provides the selection that is of importance to you as a Windows Application developer. If you had chosen to create a Web Forms project, rather than a Windows Application, you would have been given a different toolbox to use.
- You are not limited to use this selection. You can customize the toolbox to fit your needs, but in this chapter, we'll be focusing on the controls found in the selection that is shown in the picture above – in fact, we'll look at most of the controls that are shown here.
- CONTROLS
- Controls

Windows Forms **controls** are reusable components that encapsulate user interface functionality and are used in client side Windows applications.

- Properties

All controls have a number of properties that are used to manipulate the behavior of the control. The base class of most controls, Control, has a number of properties that other controls either inherit directly or override to provide some kind of custom behavior.

- Commonly used properties
- Commonly used properties
- Anchor and Dock Properties
- These two properties are especially useful when you are designing your form.
- Ensuring that a window doesn't become a mess to look at if the user decides to resize the window is far from trivial, and numerous lines of code have been written to achieve this.
- Anchor Property
- Anchor Property is used to specify how the control behaves when a user resizes the window. You can specify if the control should resize itself,

anchoring itself in proportion to its own edges, or stay the same size, anchoring its position relative to the window's edges.

- Dock Property
- The Dock property is related to the Anchor property.
- You can use it to specify that a control should dock to an edge of its container. If a user resizes the window, the control will continue to be docked to the edge of the window.
- If, for instance, you specify that a control should dock with the bottom of its container, the control will resize itself to always occupy the bottom part of the screen, no matter how the window is resized. The control will not be resized in the process; it simply stays docked to the edge of the window.
- EVENTS
- EVENTS
- When a user clicks a button or presses a button, you as the programmer of the application, want to be told that this has happened. To do so, controls use events. The Control class defines a number of events that are common to the controls we'll use in this chapter. The table below describes a number of these events.
- EVENTS
- EVENTS
- EVENTS
- Basic Form Controls
- Pointer

Label - A **Label control** is used as a display medium for text on Forms. Label control does not participate in user input or capture mouse or keyboard events.

- Properties of a Label Control

➢ **Name** - represents a unique name of a Label control.

Ex. label1.Name = "label1";

➢ **Location, Height, Width, and Size -** The Location property takes a Point that specifies the starting position of the Label on a Form. The Size property specifies the size of the control. You can also use Width and Height property instead of Size property.

Ex. label1.Location = new Point(20, 150);

label1.Height = 40;

label1.Width = 300;

- **Background, Foreground, BorderStyle** - BackColor and ForeColor properties are used to set background and foreground color of a Label respectively.

Ex. label1.BackColor = Color.Red;

label1.ForeColor = Color.Blue;

- You can also set borders style of a Label by using the BorderStyle property. The BorderStyle property is represented by a BorderStyle enumeration that has three values – FixedSingle, Fixed3D, and None. The default value of border style is Fixed3D.

Ex. label1.BorderStyle = BorderStyle.FixedSingle;

- **Font -** Font property represents the font of text of a Label control.

Ex. label1.Font = new Font("Arial", 16);

- **Text and TextAlign, and TextLength -** Text property of a Label represents the current text of a Label control. The TextAlign property represents text alignment that can be Left, Center, or Right. The TextLength property returns the length of a Label contents.

Ex. label1.Text = "This is a Label Control";

label1.TextAlign = HorizontalAlignment.Left;

- TEXT CONTROL
- **Textbox Controls**

A **TextBox** is used to get input from users or it can also be used to display some values to the user.

- **Properties**

- **Text** - textBox1.Text = "This is a TextBox";
- **Size** - you can change the width: textBox1.Width = 250; or height: textBox1.Height = 50;
- **Background/Foreground Color** - textBox1.BackColor = Color.Blue; or textBox1.ForeColor = Color.White;
- **Maximum Length** - textBox1.MaxLength = 40;

- Properties

- **Read-Only** - textBox1.ReadOnly = true;
- **Multiple lines** - textBox1.Multiline = true;
- **Password Type** - textBox1.PasswordChar = '*'; //you can use any character rather than *.

- Button Controls
- Button Controls

The **button** is very important part of every software. Because we deal every action and event with buttons in any software.

- **Text** - button1.Text = "Example Button";
- **Image** –

button1.Image = Image.FromFile("C:\\Users\\Jade\\Pictures\\brownImage.jpg");

- **BackColor** - button1.BackColor = Color.Aqua;
- **ForeColor** - button1.ForeColor = Color.White;
- **Font** - button1.Font = new Font(button1.Font.FontFamily, 33);

- **Event**

- **Click** - This event will occur when end user will click on the button once.
- **Text Changed** - this event occurs when the text of the button is changed. This is the built-in method with the name _**TextChanged**.
- **MouseHover** - this event occurs when user will hover the mouse cursor on the button. This is a built-in event with the name _**MouseHover** after the default name of the button or your desired button named.
- **MouseLeave** - this event is occurring when user will leave the button or move the cursor from the button boundaries.

- Picturebox Control
- Picturebox Control

- it displays graphical files such as bitmaps and icons in a frame.

- Checkbox
- **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.
- Radio Button
- The **RadioButton** Provides a User Interface for an Exclusive Selection. **RadioButton** or option Buttons enables the user to select a single option from a group of choices when Paired with other **Radio Buttons**
- ComboBox
- 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**.
- ChecklistBox

- 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**.
- DateTimePicker
- The **DateTimePicker** control allows you to display and collect date and time from the user with a specified **format**. The **DateTimePicker** control has two parts, a label that displays the selected date and a popup calendar that allows users to select a new date.
- GroupBox
- A **GroupBox** control is a container control that is used to place **Windows Forms** child controls in a group. The purpose of a **GroupBox** is to define user interfaces where we can categories related controls in a group.
  A **GroupBox** control is a container control that is used to place **Windows Forms** child controls in a group
- ListBox
- In **Windows Forms**, **ListBox** control is used to show multiple elements in a list, from which a user can select one or more elements and the elements are generally displayed in multiple columns. The **ListBox** class is used to represent the **windows list box** and also provide different types of properties, **methods**, and events.
- Timer
- **Timer in C#** executes a block of code repeatedly at a given interval of time. The execution occurs via a **timer** event. For example, backing up a folder every 10 minutes, or writing to a log file every second. The **method** that needs to be executed is placed inside the event of the **timer**.
- Advance Controls
- Binding Navigator
- You can use the **BindingNavigator** control to create a standardized means for users to search and change data on a **Windows Form**. You frequently use **BindingNavigator** with the **BindingSource** component to enable users to move through data records on a **form** and interact with the records.
- Binding Source
- The **BindingSource** component serves many purposes. First, it simplifies **binding** controls on a **form** to data by providing currency management, change notification, and other services between **Windows Forms** controls and data sources.
- DataGrid
- It Provides a user interface to ADO.NET datasets,displays ADO.NET data in scrollable grid and allows updates to data source.

- DataGrid View
- The **DataGridView** control is highly configurable and extensible, and it provides many properties, methods, and events to customize its appearance and behavior. The **DataGridView** control makes it easy to define the basic appearance of cells and the display formatting of cell values
- DataSet and DataTable
- A **DataSet** is an in-memory representation of a database-like structure which has collection of **DataTables**
- A **DataTable** is an in-memory representation of a single database table which has collection of rows and columns.
- DataView
- A **DataView** enables you to create different **views** of the **data** stored in a **DataTable**, a capability that is often used in **data**-binding applications. Using a **DataView**, you can expose the **data** in a table with different sort orders, and you can filter the **data** by row state or based on a filter expression