WINDOWS FORMS

What is Windows Forms?

Windows Forms is a smart client technology for the .NET Framework, a set of managed libraries that simplify common application tasks such as reading and writing to the file system. When you use a development environment like Visual Studio, you can create Windows Forms smart-client applications that display information, request input from users, and communicate with remote computers over a network.

Lessons Overview

Lesson 1:- Working with Windows Forms

In this lesson will cover the following topics

- Introduction to Windows Forms Application
- Console Application vs Windows Application
- Basic Controls in Windows forms
- Event Driven Programming
- Dialog Boxes
- SDI(Single Document Interface) vs MDI(Multiple Document Interface)
- Menus in Windows Forms Application

Introduction to Windows Form Application

Windows Forms is a rich programming framework for building client applications that provides improved ease-of-use, tool support, and lower deployment costs.

Traditional Windows desktop applications are often referred to as rich-client applications to differentiate them from browser-based Web applications that are downloaded from a central location.

Classes used to create rich-client applications are found in the System.Windows.Forms namespace and are collectively known as the Windows Forms classes.

This namespace includes the Form class, which is used as a base class for all dialog boxes and top-level windows in a .NET desktop application. In addition, the System.Windows.Forms namespace includes classes that manage controls, interaction with the clipboard, menus, printing, and more.

Console Application vs Windows Application

Console Application: - C# Console Application has the following characteristics:

No visual component

Only text input and output

Run under Command Prompt or DOS Prompt

Windows Application: - Windows Application has the following characteristics

It forms with many different input and output types.

It contains Graphical User Interfaces (GUI).

GUIs make the input and output more user friendly!

Controls:-

Controls are objects that are contained within form objects.

Each type of control has its own set of properties, methods, and events that make it suitable for a particular purpose.

You can manipulate controls in the designer and write code to add controls dynamically at run time.

As you design and modify the user interface of your Windows Forms applications, you will need to add, align, and position controls.

Following are some of the commonly used controls in Windows Form

- Text Box
- Label
- Button
- Combo Box
- List Box, etc.

Control 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.

Some of the commonly used properties are as follows

- Name
- Text
- ForeColor
- BackColor, etc.

Control 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. Every control in windows application has a number of events associated with them. We can program these events as per the application requirements.

Some of the commonly used events are as follows

- Click
- DoubleClick
- TextChanged
- Selected Index Changed, etc.

Anchor and Dock Property

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. Many programs solve the problem by simply disallowing the window from being resized, which is clearly the easiest way around the problem, but not the best. The Anchor and Dock properties that have been introduced with .NET lets you solve this problem without writing a single line of code.

The 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.

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.

Event Driven Programming:-

An event is an action which you can respond to, or "handle," in code. Events can be generated by a user action, such as clicking the mouse or pressing a key; by program code; or by the system.

Event-driven applications execute code in response to an event. Each form and control exposes a predefined set of events that you can program against. If one of these events occurs and there is code in the associated event handler, that code is invoked.

The types of events raised by an object vary, but many types are common to most controls. For example, most objects will handle a Click event. If a user clicks a form, code in the form's Click event handler is executed.

Delegates and Events:-

Delegates are classes commonly used within the .NET Framework to build event-handling mechanisms.

Delegates roughly equate to function pointers, commonly used in Visual C++ and other object-oriented languages.

Unlike function pointers however, delegates are object-oriented, type-safe, and secure.

In addition, where a function pointer contains only a reference to a particular function, a delegate consists of a reference to an object, and references to one or more methods within the object.

Events uses delegates to bind events to the methods that are used to handle them.

The delegate enables other classes to register for event notification by specifying a handler method. When the event occurs, the delegate calls the bound method.

When an event is recorded by the application, the control raises the event by invoking the delegate for that event. The delegate in turn calls the bound method.

Dialog Boxes:-

Dialog boxes are used to interact with the user and retrieve information.

Dialog boxes can range from basic forms that contain a simple text box control to elaborate forms with large numbers of controls.

Some types of dialog boxes are used so often they're included as part of the operating system. These dialog boxes, known collectively as the common dialog boxes, include prebuilt dialog boxes that handle tasks such as selecting files, specifying colors, choosing fonts, and configuring printer settings.

Dialog Box provided by .Net Frameworks are as follows

- Font Dialog
- Color Dialog
- OpenFileDialog
- SaveFileDialog
- FolderBrowserDialog, etc.

Types of Dialog Boxes

Modal Dialog Box:-

A Modal Dialog box is one that forces the user to interact with it before they can go back to using the parent application. Open File dialog is an example of Modal Dialog Box

Modaless Dialog Box:-

Modaless dialog box is one that allows us to interact with other windows even when the modaless dialog is still on screen. Find and replace is an example of Modaless Dialog Box.

SDI vs MDI

SDI applications allow only one open document frame window at a time.

MDI applications allow multiple document frame windows to be open in the same instance of an application.

An MDI application has a window within which multiple MDI child windows, which are frame windows themselves, can be opened, each containing a separate document.

In some applications, the child windows can be of different types, such as chart windows and spreadsheet windows. In that case, the menu bar can change as MDI child windows of different types are activated.

Menus in Windows Application:-

Menus provide a convenient way to group similar or related commands in one place.

Most users are familiar with the menu bar concept and expect standard menus such as File, Edit, and Help to appear in their applications.

Following are the Menus controls supported in Windows Form Application

- MenuStrip
- ToolStrip
- ContextMenu