

**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING
SCIENCES**



C-Sharp Manual

CL2005- Database Systems Lab

Instructor: Samia Aziz

Email: samia.aziz@nu.edu.pk

C# Windows Form Application Manual

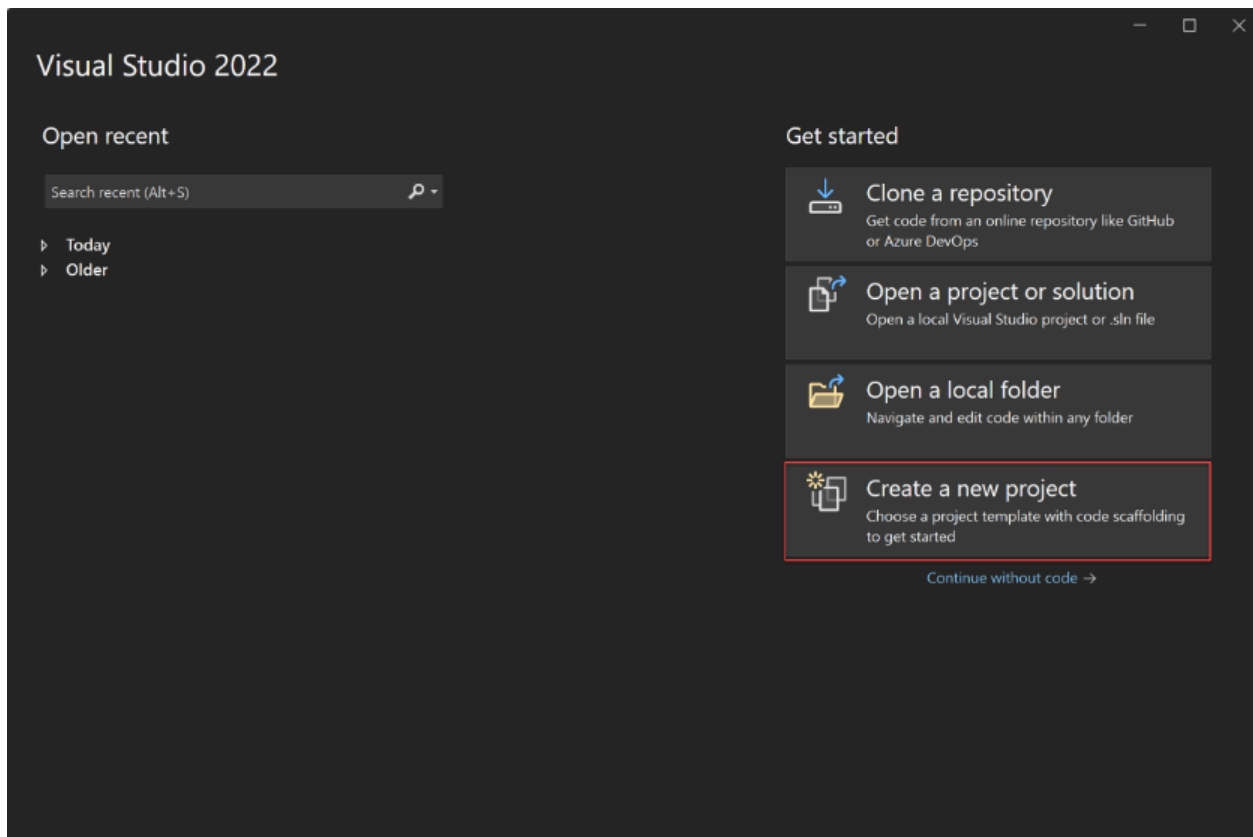
In this tutorial, you'll create a simple C# application that has a Windows-based user interface (UI).

If you haven't already installed Visual Studio, go to the [Visual Studio 2022 downloads page](#) to install it for free.

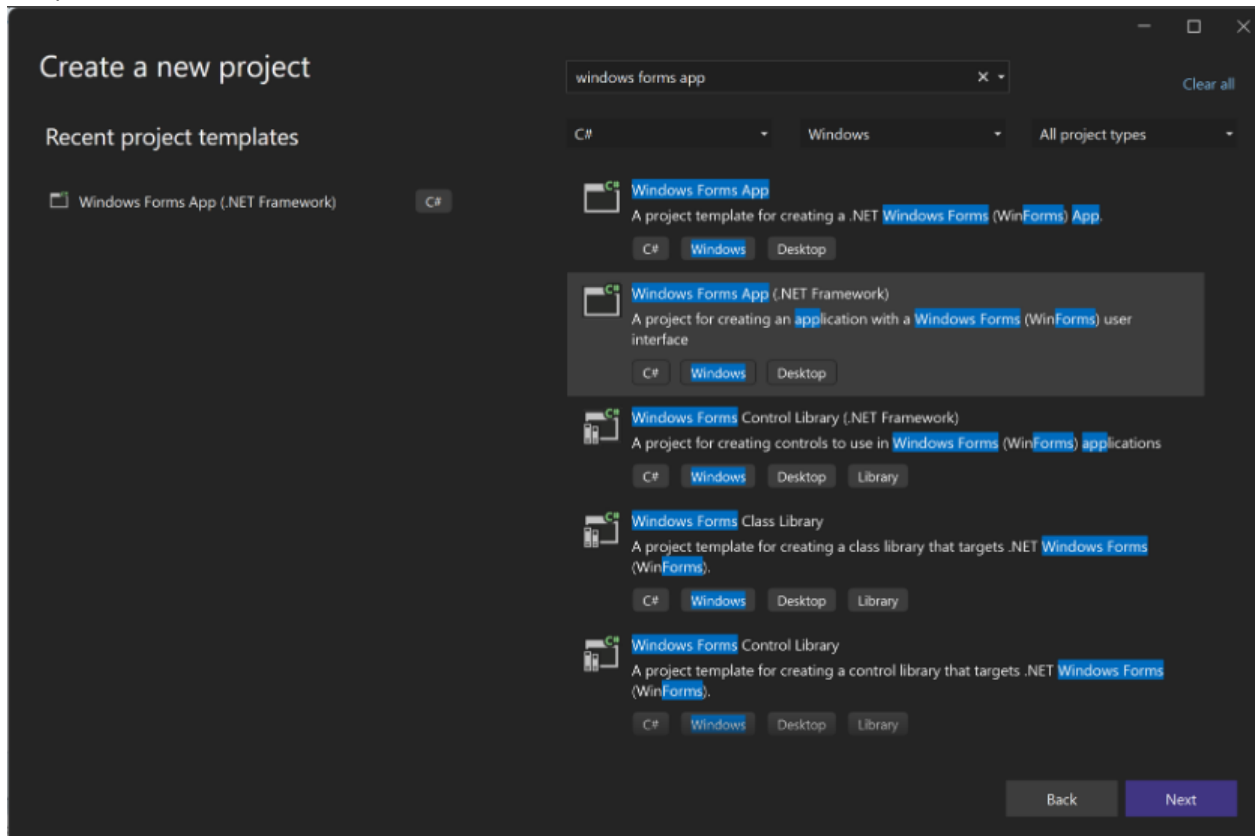
Create a project

First, you'll create a C# application project. The project type comes with all the template files you'll need, before you've even added anything.

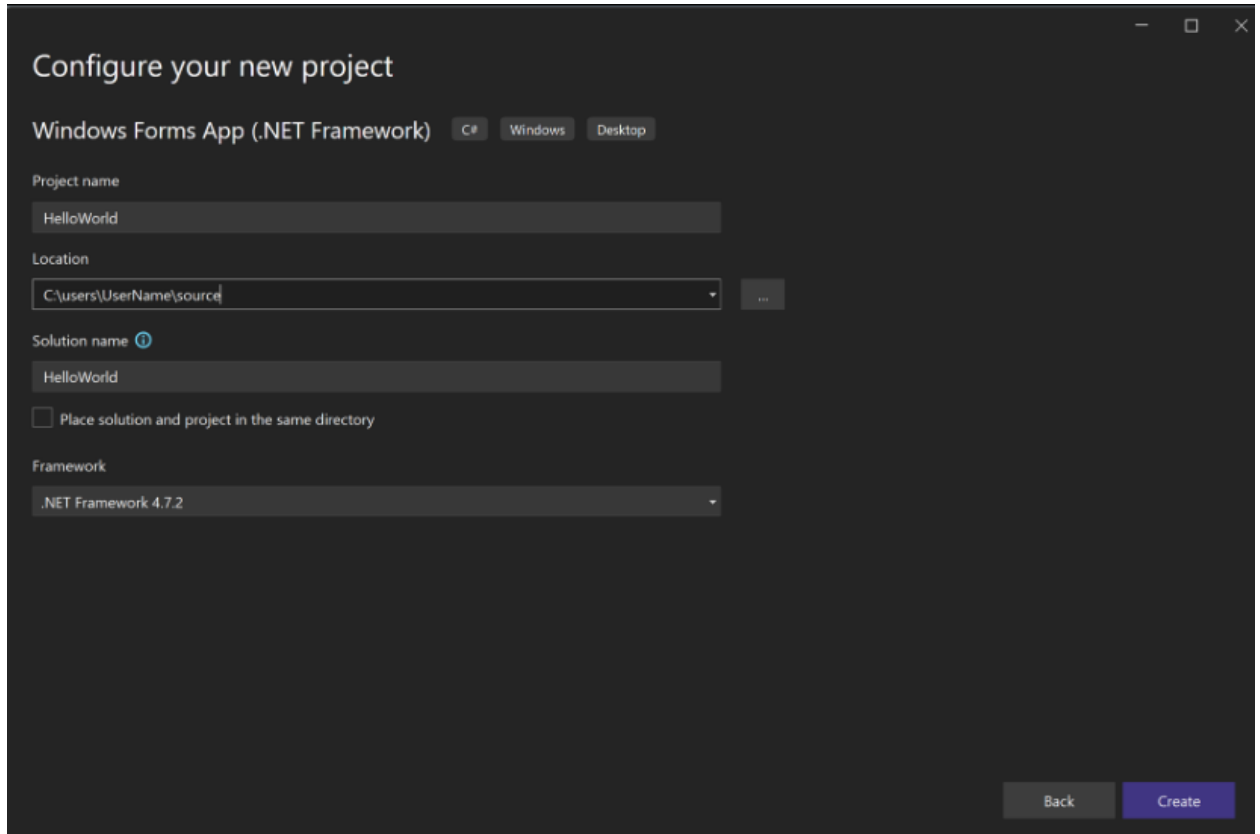
1. Open Visual Studio.
2. On the start window, select Create a new project.



3. On the Create a new project window, select the Windows Forms App (.NET Framework) template for C#.



4. In the Configure your new project window, type or enter Project Name in the Project name box. Then, select Create.



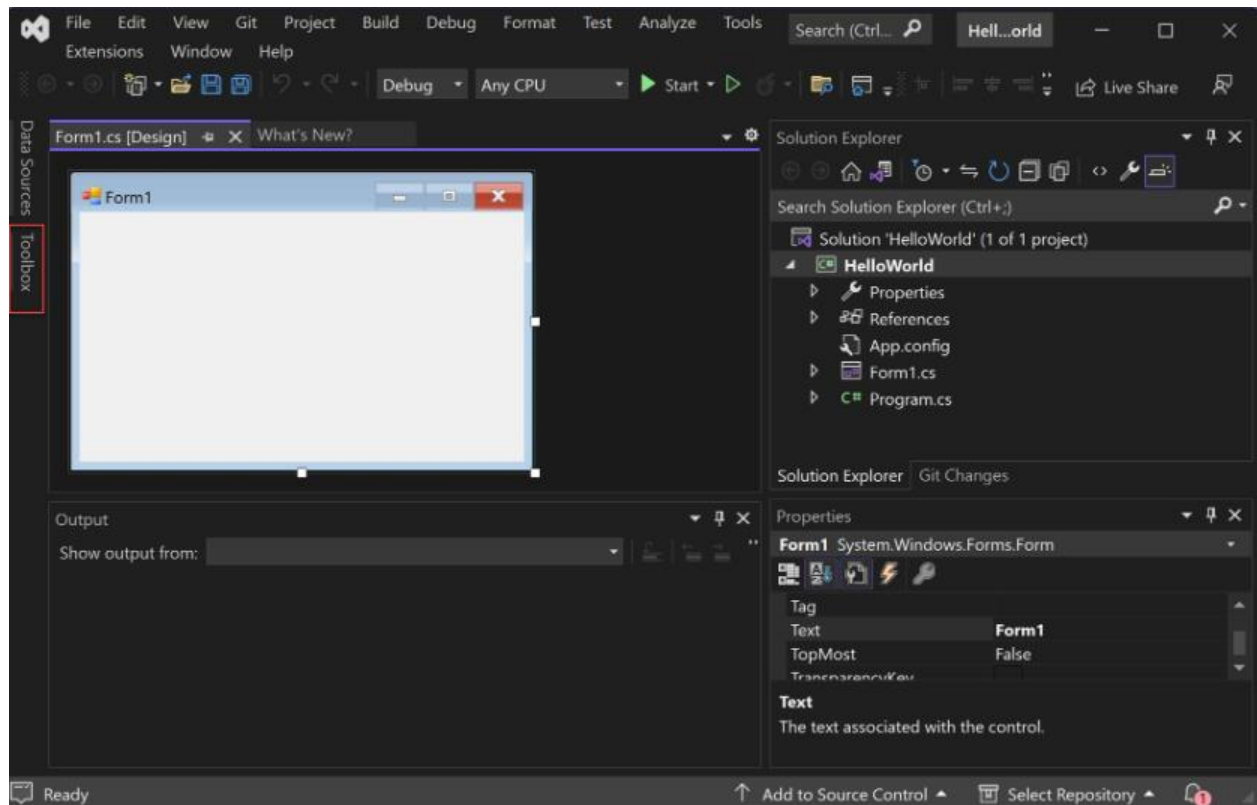
The screenshot shows the 'Configure your new project' window in Visual Studio. The title bar includes standard window controls (minimize, maximize, close). The main title is 'Configure your new project'. Below it, the project type is 'Windows Forms App (.NET Framework)', with tabs for 'C#' (selected), 'Windows', and 'Desktop'. The 'Project name' field contains 'HelloWorld'. The 'Location' field shows a file explorer icon and the path 'C:\users\UserName\source'. The 'Solution name' field also contains 'HelloWorld', with a help icon to its left. Below this is a checkbox labeled 'Place solution and project in the same directory', which is currently unchecked. The 'Framework' dropdown menu is set to '.NET Framework 4.7.2'. At the bottom right, there are 'Back' and 'Create' buttons.

Create the application

After you select your C# project template and name your file, Visual Studio opens a form for you. A form is a Windows user interface. We'll create a "Hello World" application by adding controls to the form, and then we'll run the app.

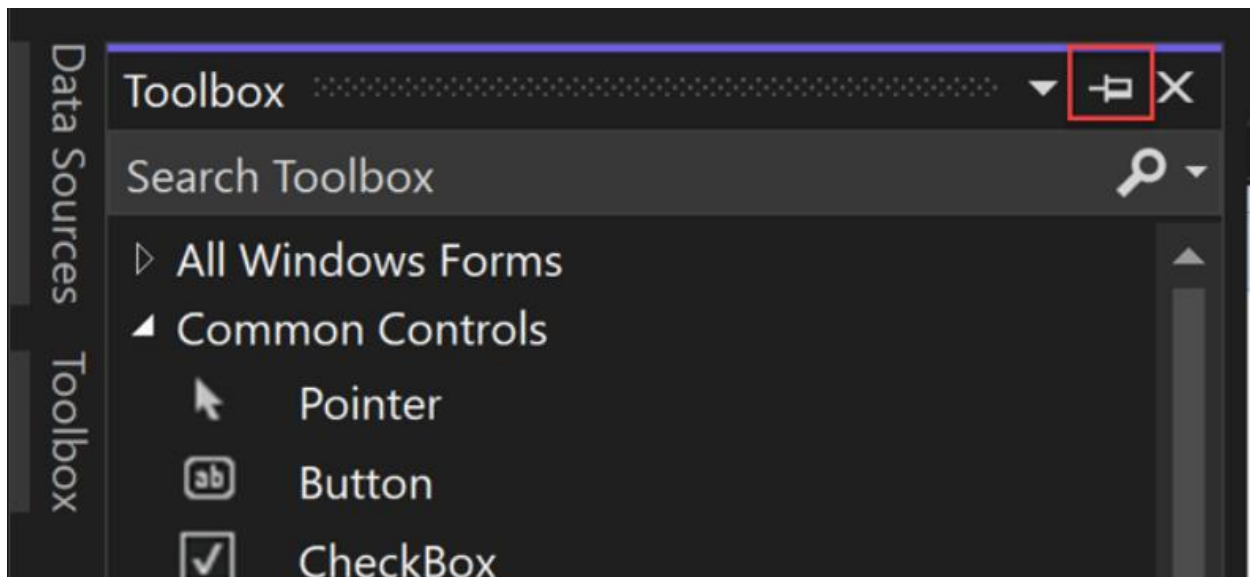
Add a button to the form

1. Select Toolbox to open the Toolbox fly-out window.

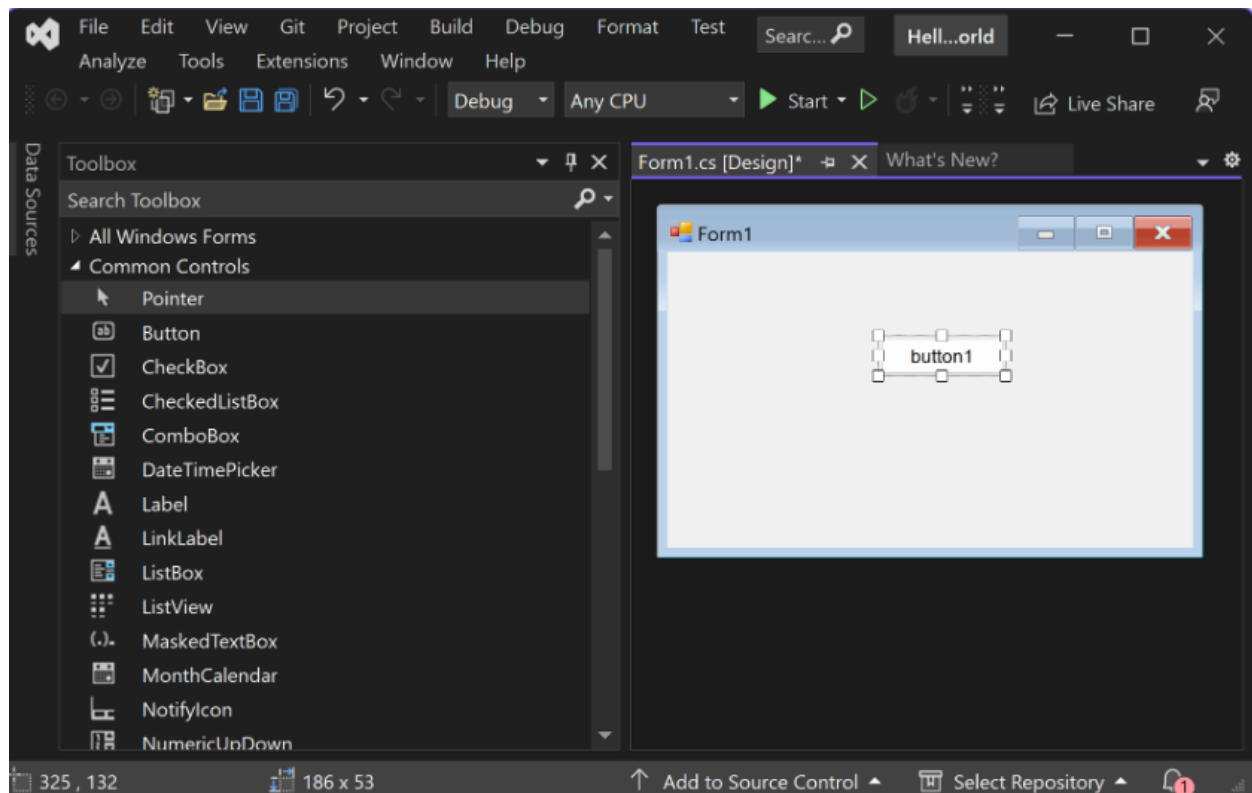


(If you don't see the Toolbox fly-out option, you can open it from the menu bar. To do so, View > Toolbox. Or, press Ctrl+Alt+X.)

2. Expand Common Controls and select the Pin icon to dock the Toolbox window.

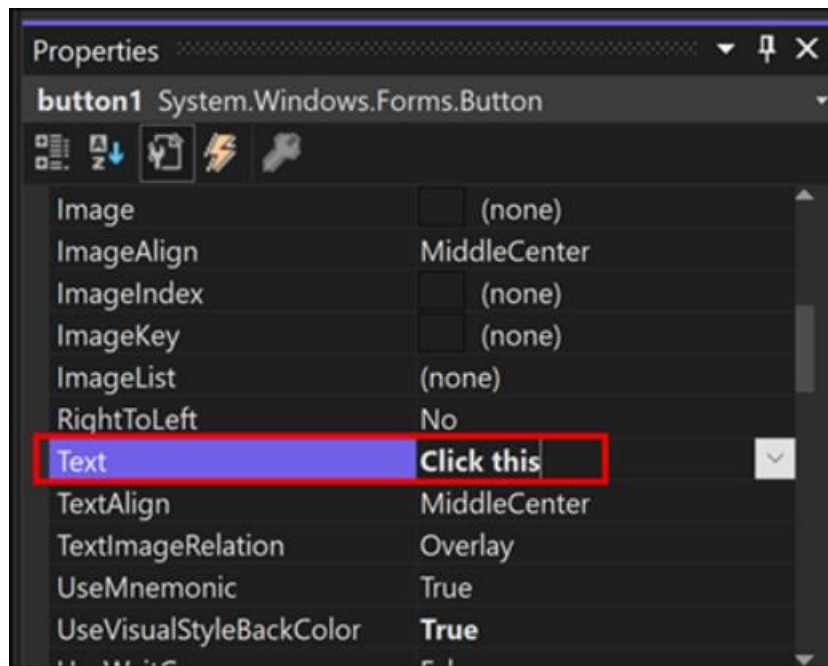


3. Select the Button control and then drag it onto the form.



For More Tools information visit <https://www.guru99.com/c-sharp-windows-forms-application.html>

4. In the Properties window, locate Text, change the name from button1 to Click this, and then press Enter.



1. (If you don't see the Properties window, you can open it from the menu bar. To do so, select View > Properties Window. Or, press F4.). This window is used to change the different properties of the selected item in the Solution Explorer. Also, you can change the properties of components or controls that you will add to the forms. You can add or remove borders, increase and decrease size and can add function on click also . Change background and text color etc .

Message Box:

In C#, a MessageBox is a graphical user interface (GUI) dialog box that displays a message to the user. It is a part of the System.Windows.Forms namespace and provides a simple way to communicate with the user by presenting messages or asking for input. MessageBox is commonly used in Windows Forms applications.

Here is a basic example of how you can use MessageBox in C#:

```
using System;
```

```
using System.Windows.Forms;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
{  
    // Display a simple message box with an OK button  
    MessageBox.Show("Hello, this is a MessageBox!", "Information",  
        MessageBoxButtons.OK, MessageBoxIcon.Information);  
  
    // Display a message box with Yes and No buttons and handle the result  
    DialogResult result = MessageBox.Show("Do you want to continue?",  
        "Question", MessageBoxButtons.YesNo, MessageBoxIcon.Question);  
  
    if (result == DialogResult.Yes)  
    {  
        Console.WriteLine("User clicked Yes.");  
    }  
    else  
    {  
        Console.WriteLine("User clicked No.");  
    }  
}
```

In this example:

- `MessageBox.Show` is used to display a message box with a specified message, title, buttons, and icon.
- The first argument is the message to be displayed.
- The second argument is the title of the message box.
- The third argument specifies the buttons to be displayed (e.g., OK, YesNo).
- The fourth argument specifies the icon to be displayed (e.g., Information, Warning).

The Show method returns a DialogResult enum value, which can be used to determine which button the user clicked.










Common MessageBoxButtons:

- MessageBoxButtons.OK
- MessageBoxButtons.OKCancel
- MessageBoxButtons.YesNo
- MessageBoxButtons.YesNoCancel

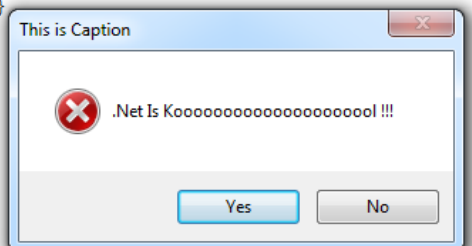
Common MessageBoxIcon values:

- MessageBoxIcon.None
- MessageBoxIcon.Information
- MessageBoxIcon.Warning
- MessageBoxIcon.Error
- MessageBoxIcon.Question

MessageBox provides a convenient way to interact with users for simple dialogs and messages within a Windows Forms application. You can icon in a message box by following way :

MessageBoxIcon	Description	
None		
Asterisk		
Error		
Exclamation		
Hand		
Information		
Question		
Stop		
Warning		

```
private void Form1_Load(object sender, EventArgs e)
{
    MessageBox.Show(".Net Is Koooooooooooooooooooool !!! ", "This is Caption", MessageBoxButtons.YesNo, MessageBoxIcon.Error);
}
```



For message Box detailed information visit this site

<https://www.c-sharpcorner.com/UploadFile/mahesh/understanding-message-box-in-windows-forms-using-C-Sharp/>