Multiple Forms

A Visual C# project can have multiple forms. Each form has its own class that can be instantiated and displayed on the screen.

Every form in a Visual C# project has a class. For example, if a project has a form named Form1, then the project has a class named Form1, which is stored in a file named Form1.cs. When you add additional forms to a project, you add additional classes, which are stored in their own files.

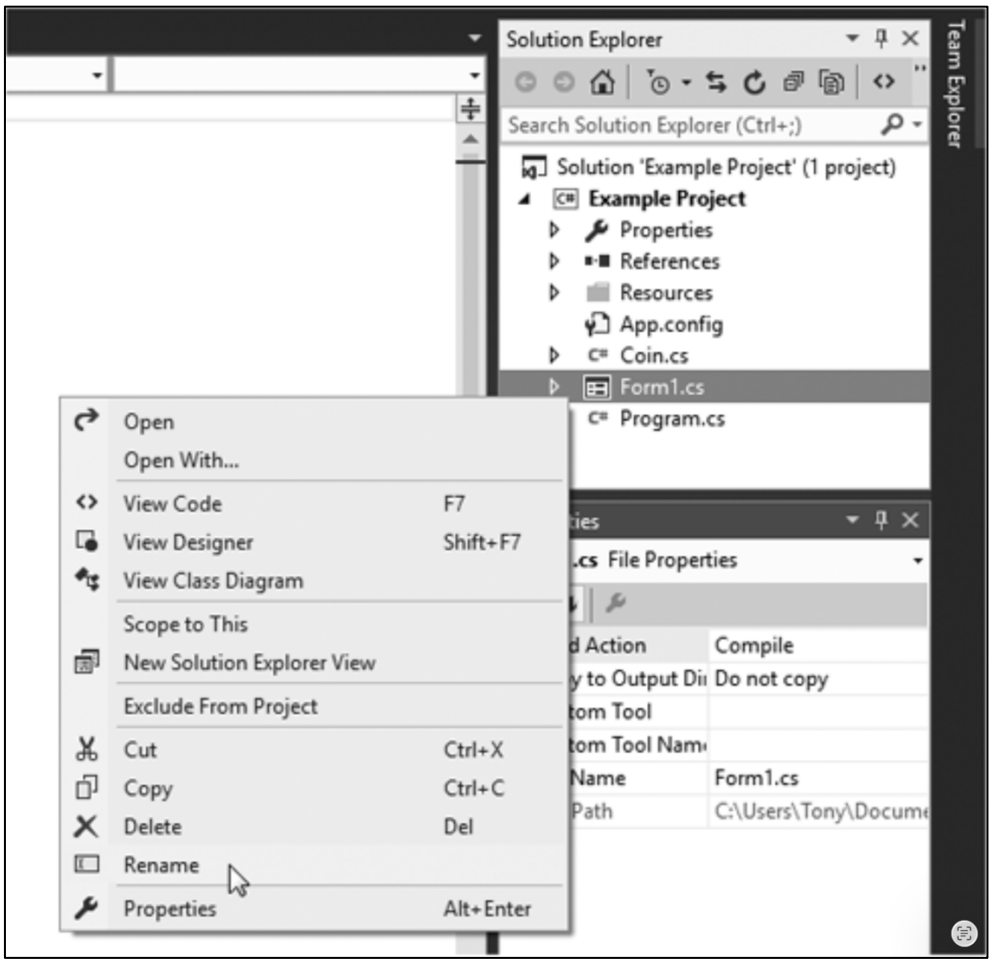
Renaming the Form1 Form

When you add forms to a Visual C# project, they are given default names such as Form1, Form2, and so forth.

when you have multiple forms in a project, you should give each form a meaningful name that describes its purpose.

To change the form's name, you use the Solution Explorer to change the name of the Form1.cs file to MainForm.cs. When you do this, Visual Studio automatically changes the name of the Form1 form to MainForm. The pop-up menu shown in Figure 6-1 should appear.

Figure 6-1 Right-clicking the form file in the Solution Explorer

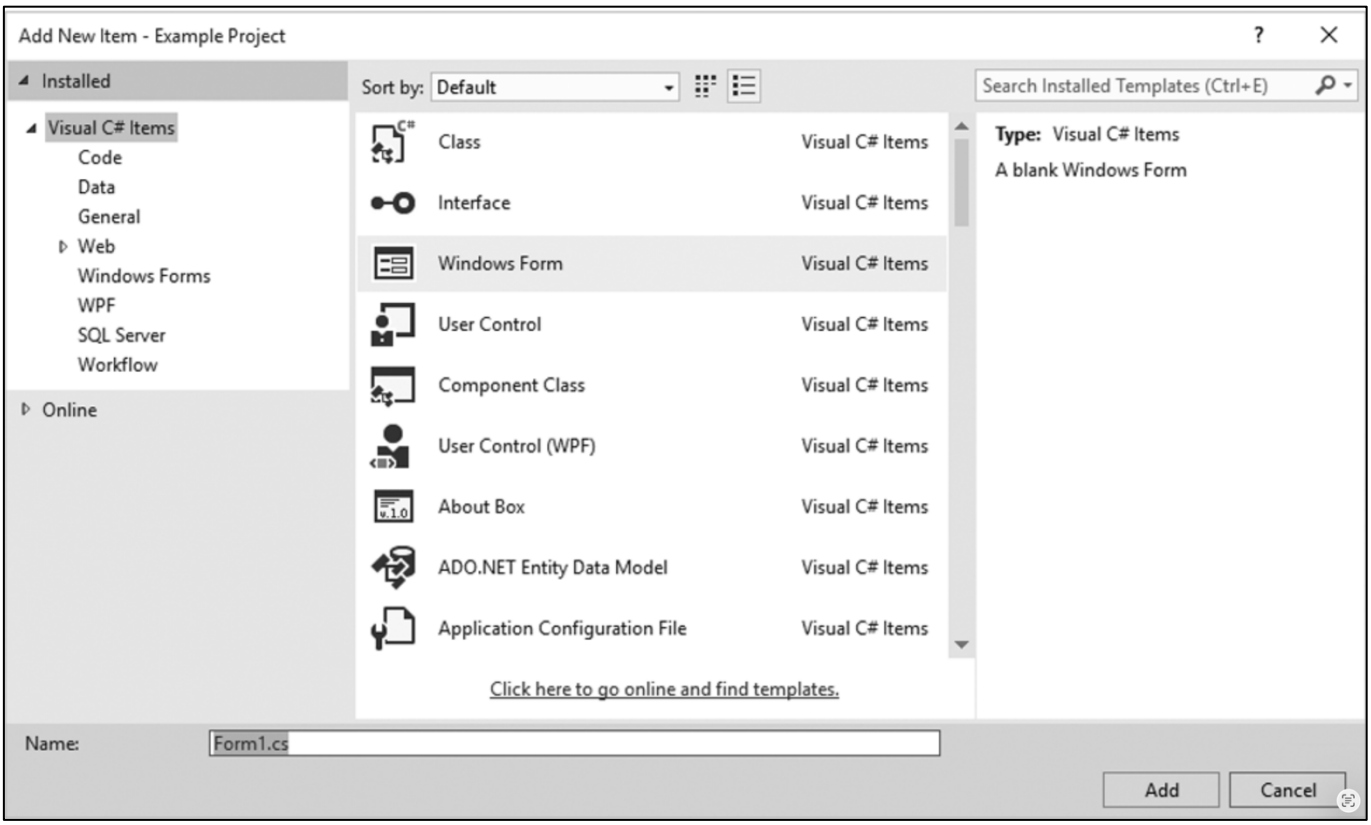


Adding a New Form to a Project

To add a new form to a project:

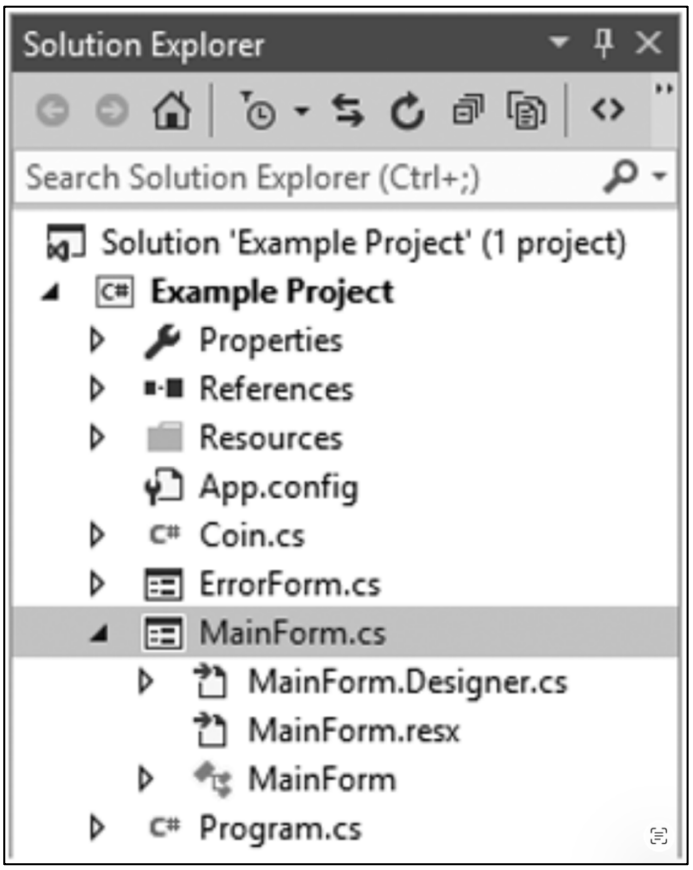
Click Project on the Visual Studio menu bar, and then select Add Windows Form. . . from the Project menu. The Add New Item window, shown in Figure 6-2, should appear.

Figure 6-2 Add New Item window



After completing these steps, a new blank form is added to your project. The new form is displayed in the Designer, and an entry for the new form's file appears in the Solution Explorer. Figure 6-3 shows an example of the Solution Explorer with two form files: ErrorForm.cs and MainForm.cs.

Figure 6-3 Solution Explorer window showing two forms



Displaying a Form

In your application's code, the first step in displaying a form is to create an instance of the form's class. For example, suppose a project has a form named Form2. The following statement creates an instance of the Form2 class:

Form2 myForm2 = new Form2();

This statement declares a reference variable named **myForm2**. After this statement executes, you will be able to use the **myForm2** variable to perform operations with the form.

Creating an instance of a form's class does not display the form on the screen. The next step is to call the form's Show method. Here is an example:

```
myForm2.Show();
```

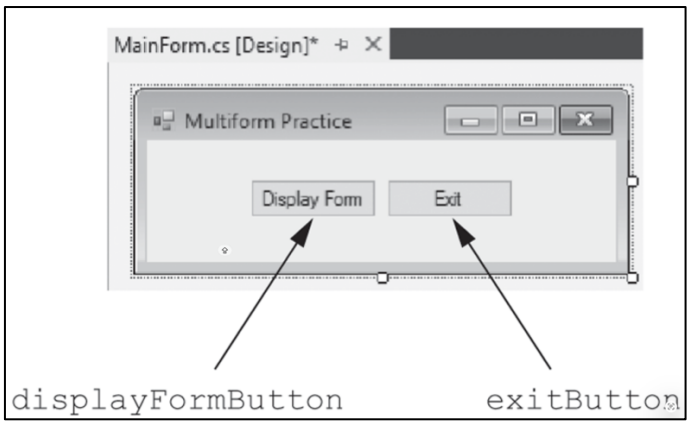
The **Show** method displays a form on the screen, and it gives that form the **focus**.

Creating an Application with Two Forms

1- Rename the Form1.cs file to MainForm.cs.

2- In the Designer, set up the MainForm form as shown in Figure 6-4.

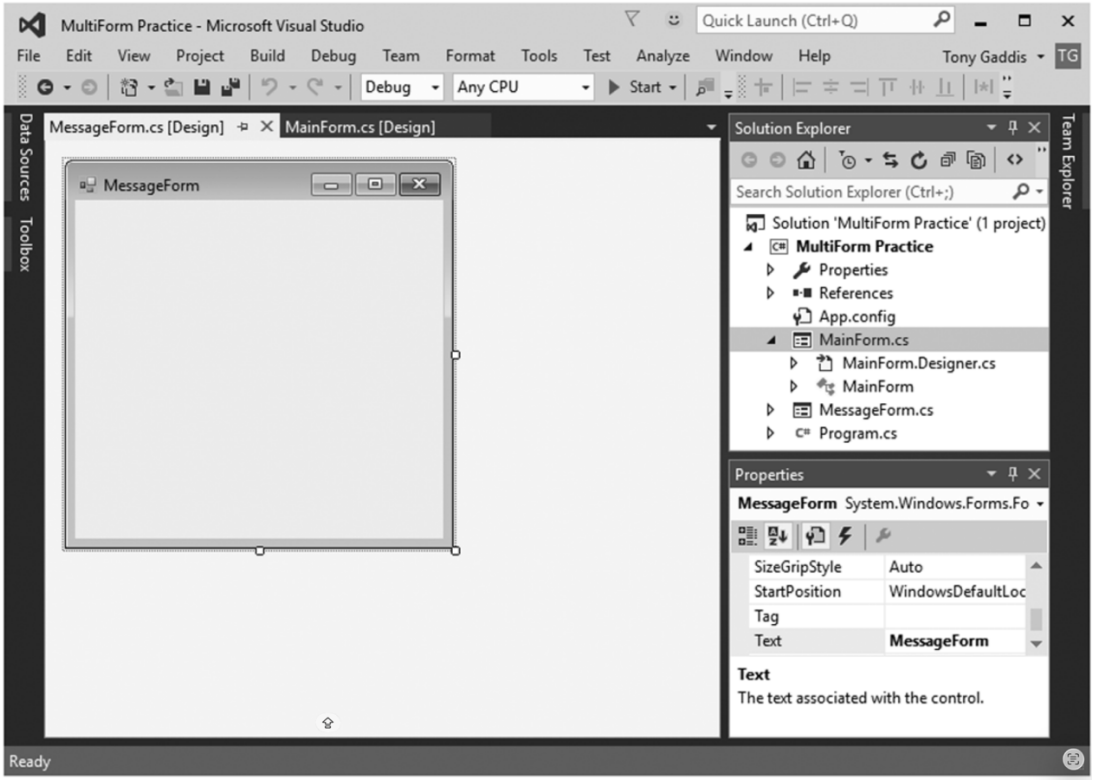
Figure 6-4 The MainForm form



3- Create another form named MessageForm in the project.

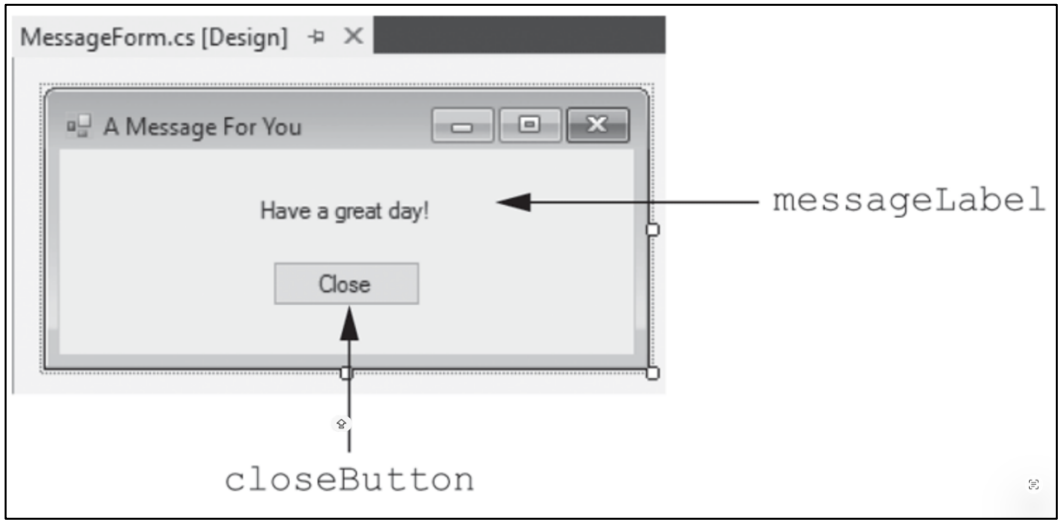
As shown in Figure 6-5, a new form named MessageForm will appear in the Designer.

Figure 6-5 MessageForm added to the project



4- In the Designer, set up the MessageForm as shown in Figure 6-6.

Figure 6-6 MessageForm



5- Next, you create the Click event handler for the closeButton control.
6- Now you create the Click event handler for the displayFormButton control.
7- Finally, the message is displayed upon clicking the Display Form button, as illustrated in Figure 6-7.

Figure 6-7 The MainForm and the MessageForm forms displayed

CODE IN Display Form button:

```
MessageForm    myMessageForm    = new MessageForm();

myMessageForm.Show();
```

CODE IN Exit button:

```
This.Close();
```