TRAINING & REFERENCE

murach's C# 2015

(Chapter 2)

Thanks for downloading this chapter from ***Murach's C# 2015***. We hope it will show you how easy it is to learn from any Murach book, with its paired-pages presentation, its "how-to" headings, its practical coding examples, and its clear, concise style.

To view the full table of contents for this book, you can go to our **website**. From there, you can read more about this book, you can find out about any additional downloads that are available, and you can review our other books on related topics.

Thanks for your interest in our books!

MIKE MURACH & ASSOCIATES, INC.
1-800-221-5528 • (559) 440-9071 • Fax: (559) 440-0963
murachbooks@murach.com • **www.murach.com**

What C# developers have said about previous editions

"I personally think that Microsoft should just hand over all documentation for their technologies to Murach! You make it fun, intuitive, and interesting to learn."

Joanne Wood, C# Web Developer, Rhode Island

"I am actually flying through the C# book! And a lot of the topics I had problems with in the past are now making perfect sense in this book."

Jim Bonner, Test Lab Engineer, Seattle, Washington

"In 20+ years of software development, I have never read another book so well organized. Topics flow in a well- thought-out, logical order and are easy to follow and understand. When I need new technical books in the future, I will always search for a Murach book first."

Jeff Rumage, Lower Alabama .NET User Group

"You guys rock and I will refer you to anyone who is interested in C#/Visual Studio. Your book is very well written and enjoyable to read. Great left/right layout idea; the book is also very easy to reference later on. Rare find."

Matt Brandau, Project Developer, California

"Murach has yet again hit a home run! This book gets you from zero to Jr Dev knowledge in no time."

Brian Knight, Founder, Pragmatic Works, Florida

"I have to tell you that your C# book is far and away the best resource I have seen to date. It is simple, straightforward, presents logical examples, and the two-page format is the best."

Timothy Layton, Developer, St. Louis, Missouri

2

How to design a Windows Forms application

In the last chapter, you learned the basic skills for working with Visual Studio, you toured a Windows Forms application, and you tested an application with three Windows forms. Now, in this chapter, you'll learn how to use Visual Studio to design the user interface for a Windows Forms application.

How to set options and create a new project

Before you start your first Windows Forms application with Visual Studio 2015, you probably should change a few of the Visual Studio options. That will make it easier for you to create a new project. You may also want to change the import and export settings.

How to set the Visual Studio options

To set the options for Visual Studio, you use the Options dialog box shown in figure 2-1. Once this dialog box is open, you can expand the Projects and Solutions group by clicking on the ▷ symbol to the left of that group, and you can click on the General group to display the options shown in this figure.

You can set the default project location by typing a path directly into the text box, or you can click the button to the right of the text box to display a dialog box that lets you navigate to the folder you want to use. This will set the default location for new projects, but you can always override the default when you create a new project.

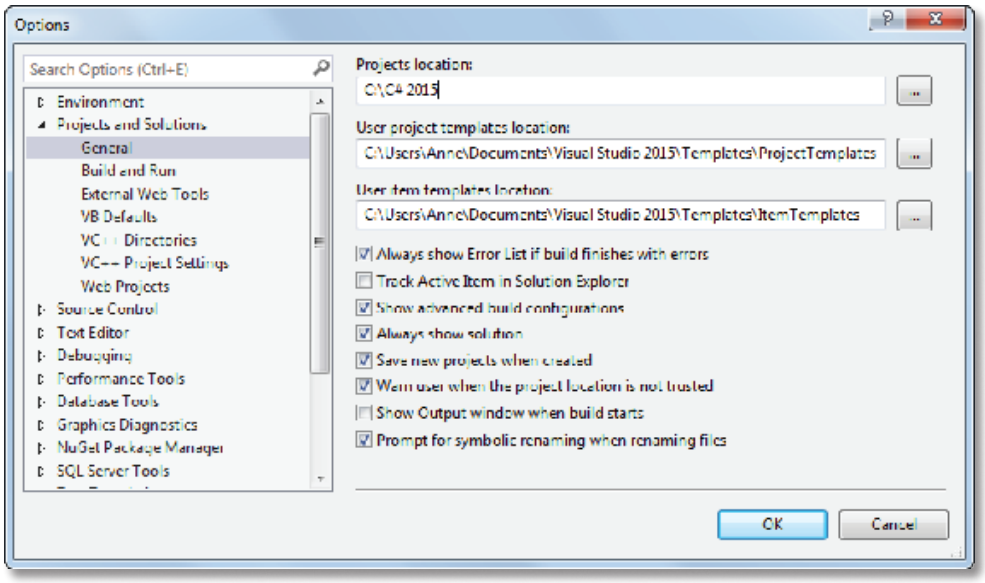
By default, most of the options available from this dialog box are set the way you want. However, it's worth taking a few minutes to review the options that are available. Then, you can change them if Visual Studio isn't working the way you want it to. For instance, you may want to use the Startup group within the Environment group to change what's displayed when Visual Studio starts.

How to change the import and export settings

The first time you start Visual Studio 2015, you're asked what Default Environment Settings you want to use. You can choose from several options including Visual Basic, Visual C#, Visual C++, and Web Development. Among other things, your choice affects what items are available from some menus and what buttons are available from the Standard toolbar. If, for example, you choose the Visual C# settings, you open a project with the File→Open→Project/Solution command. But if you choose the Visual Basic settings, you open a project with the File→Open Project command.

To change these settings, you use the Import and Export Settings Wizard as described in this figure. In the first step of the wizard, choose the Reset All Settings option. In the second step, choose the Yes, Save My Current Settings option. And in the last step, if you want your menus to work as described in this book, choose the Visual C# option. Later, if you switch to Visual Basic, C++, or web development, you can change the settings again.

The Options dialog box for setting the project options



How to use the Options dialog box

- To display the Options dialog box, select the Tools→Options command.
- To expand and collapse a group of options, you can use the ▷ and ◢ symbols to the left of each group. To display the options for a group, click on the group.
- To set the default location for all projects that you start from Visual Studio, you can change the Projects Location as shown above.
- If you want Visual Studio to work as described in this book, be sure that the Always Show Solution and Save New Projects When Created options are checked.
- To change the color theme for Visual Studio, select the General category in the Environment group and then select an option from the Color Theme drop-down list.
- Although most of the options should be set the way you want them, you may want to familiarize yourself with the options in each category so you know what's available.

How to set the Import and Export Settings

- The first time you start Visual Studio 2015, you are asked to choose the default environment settings. These settings affect how the menus work and what buttons are displayed on the Standard toolbar.
- To change the settings to the ones used for this book, use the Tools→Import and Export Settings command to start the Settings Wizard. Then, choose the Reset All Settings option, the Save My Current Settings option, and the Visual C# Development Settings option as you step through the wizard.

Figure 2-1 How to set the Visual Studio options

How to create a new project

To create a new project, you use the New Project dialog box shown in figure 2-2. This dialog box lets you select the type of project you want to create by choosing one of several *templates*. To create a Windows Forms application, for example, you select the Windows Forms Application template. Among other things, this template includes references to all of the assemblies that contain the namespaces you're most likely to use as you develop a Windows application.

Note that you can select a template from the Installed category as shown here. You can select a template from the Recent and Online categories. Or you can search for an installed template using the search text box in the upper right corner of the dialog box.

The New Project dialog box also lets you specify the name for the project, and it lets you identify the folder in which it will be stored. By default, projects are stored in the Visual Studio 2015\Projects folder under the My Documents folder, but you can change that as shown in the previous figure.

If you want to change the location that's shown in the New Project dialog box, you can click the Browse button to select a different location; display the Location drop-down list to select a location you've used recently; or type a path directly. If you specify a path that doesn't exist, Visual Studio will create the necessary folders for you.

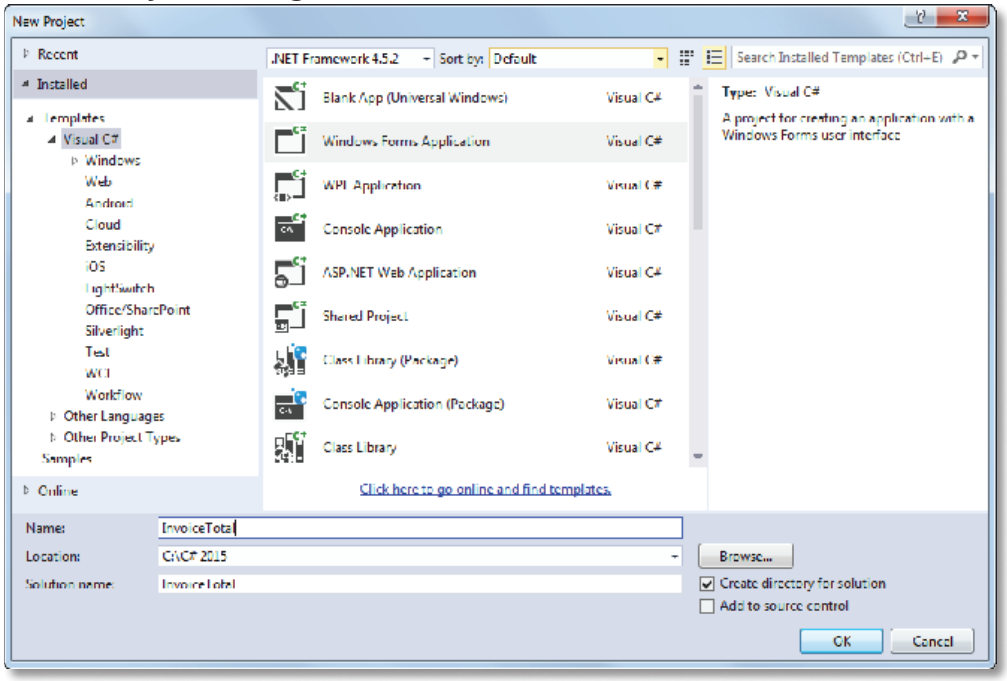
When you click the OK button, Visual Studio automatically creates a new folder for the project using the project name you specify. In the dialog box in this figure, for example, InvoiceTotal is the project name and C:\C# 2015 is the location. By default, Visual Studio also creates a new folder for the solution using the same name as the project. As a result, Visual Studio will create one folder for the solution, and a subfolder for the project. Then, you can add another project to the same solution by selecting Add To Solution from the Solution drop-down list in the New Project dialog box. (This list is displayed only if a solution is open.)

If a solution will contain a single project, though, you may want to store the project and solution in the same folder. To do that, you can deselect the Create Directory For Solution check box. Then, the solution is given the same name as the project. In the applications for this book, the project and solution files are typically stored in separate folders even though most solutions contain a single project.

Incidentally, the terms *folder* and *directory* are used as synonyms throughout this book. Although Microsoft started referring to directories as folders some time ago, most of the Visual Studio documentation still uses the term *directory*. That's why this book uses whichever term seems more appropriate at the time.

By default, the new projects you create target .NET Framework 4.5.2, which is the version that came with one of the updates to Visual Studio 2013. If you use any of the features that come with this framework, you should know that any computer that you want to run the application on must also have .NET Framework 4.5.2. If that's not the case, you can change the target framework using the drop-down list at the top of the New Project dialog box. Then, only the features of the framework you choose will be available from Visual Studio.

The New Project dialog box



How to create a new C# project

1. Use the File→New→Project command to open the New Project dialog box.
2. Choose Visual C# from the Installed→Templates category, and choose the Windows Forms Application template for a Windows Forms application.
3. Enter a name for the project, which will enter the same name for the solution. Then, enter the location (folder) for the project (and solution).
4. Click the OK button to create the new project.

Description

- The project *template* that you select determines the initial files, assembly references, code, and property settings that are added to the project.
- If the Create Directory For Solution box is checked, Visual Studio creates a folder for the solution and a subfolder for the project. Otherwise, these files are stored in the same folder.
- If the Save New Projects When Created option is on as shown in the previous figure, the project is saved right after it's created. Otherwise, the New Project dialog box asks only that you select a template and enter a name for the project. Then, when you save the project, the Save Project dialog box asks for the other information shown above.
- If you want to target a version of the .NET Framework other than version 4.5.2, you can select the version from the drop-down list at the top of the dialog box. .NET Framework 4.6 is the most recent version, and it's the version that comes with Visual Studio 2015. You won't typically need to use the features of this framework, though.

Figure 2-2 How to create a new project

How to design a form

When you create a new project, the project begins with a single, blank form. You can then add controls to this form and set the properties of the form and controls so they look and work the way you want.

The design of the Invoice Total form

Before I show you how to add controls to a form and set the properties of the form and controls, I want to describe the Invoice Total form that I'll use as an example throughout this chapter and the next chapter. This form is presented in figure 2-3. As you can see, the form consists of ten controls: four text boxes, four labels, and two buttons.

The Invoice Total form lets the user enter a subtotal into the first text box, and then calculates the discount percent, discount amount, and total for that order when the user clicks the Calculate button. For this simple application, the discount percent is based upon the amount of the subtotal, and the results of the calculation are displayed in read-only text box controls.

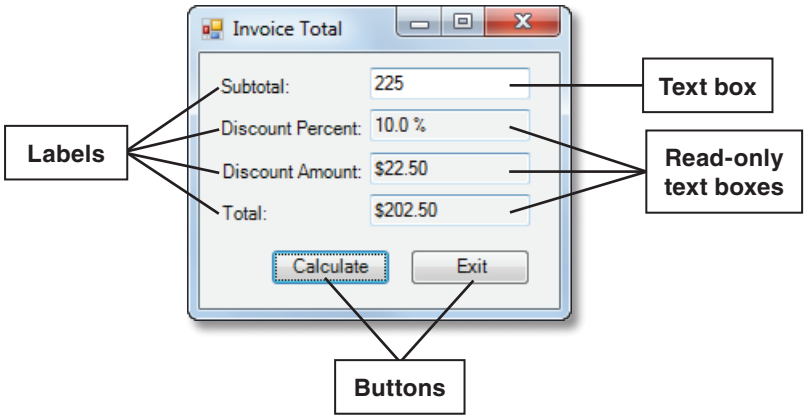
After the results of the calculation are displayed, the user can enter a different subtotal and click the Calculate button again to perform another calculation. This cycle continues until the user clicks the Close button in the upper right corner of the form or clicks the Exit button. Then, the form is closed and the application ends.

This application also provides keystroke options for users who prefer using the keyboard to the mouse. In particular, the user can activate the Calculate button by pressing the Enter key and the Exit button by pressing the Esc key. The user can also activate the Calculate button by pressing Alt+C and the Exit button by pressing Alt+X.

In the early days of computing, it was a common practice to sketch the user interface for an application on paper before developing the application. That's because a programmer had to enter the code that defined the user interface, and the task of writing this code would have been error prone if the interface wasn't planned out first. As you'll see in this chapter, however, the Form Designer makes it easy to design a form at the same time that you implement it. Because of that, you usually don't need to sketch the layout of a form before you design it in Visual Studio.

As you use the Form Designer, Visual Studio automatically generates the C# code that's needed to define the form and its controls. In other words, the form that you see in the Form Designer is just a visual representation of the form that the C# code is going to display later on. Then, all you have to do is write the C# code that gives the form its functionality, and you'll learn how to do that in the next chapter.

The Invoice Total form



Description

- A text box is used to get the subtotal from the user. Read-only text boxes are used to display the discount percent, discount amount, and total. And label controls are used to identify the values that are in the text boxes on the form.
- After entering a subtotal, the user can click the Calculate button to calculate the discount percent, discount amount, and total. Alternatively, the user can press the Enter key to perform the calculation.
- To calculate another invoice total, the user can enter another subtotal and then click the Calculate button or press the Enter key again.
- To close the form and end the application, the user can click the Close button in the upper right corner of the form or click the Exit button. Alternatively, the user can press the Esc key to exit from the form.
- The user can press Alt+C to activate the Calculate button or Alt+X to activate the Exit button. On most systems, the letters that activate these buttons aren't underlined until the user presses the Alt key.

Three types of controls

- A *label* displays text on a form.
- A *text box* lets the user enter text on a form.
- A *button* initiates form processing when clicked.

Figure 2-3 The design of the Invoice Total form

How to add controls to a form

Figure 2-4 shows how you can use the Toolbox to add controls to a form. The easiest way to do that is to click on the control in the Toolbox, then click the form at the location where you want to add the control. In this figure, for example, the button control is selected in the Toolbox, and the mouse pointer is positioned over the form.

Once you add a control to a form, you can resize the control by selecting it and dragging one of its handles, and you can move the control by dragging the control to a new location on the form. If you prefer, you can place and size the control in a single operation by clicking the control in the Toolbox, then clicking and dragging in the form.

A second way to add a control is to drag the control from the Toolbox to the form. The control is placed wherever you drop it. You can then resize the control.

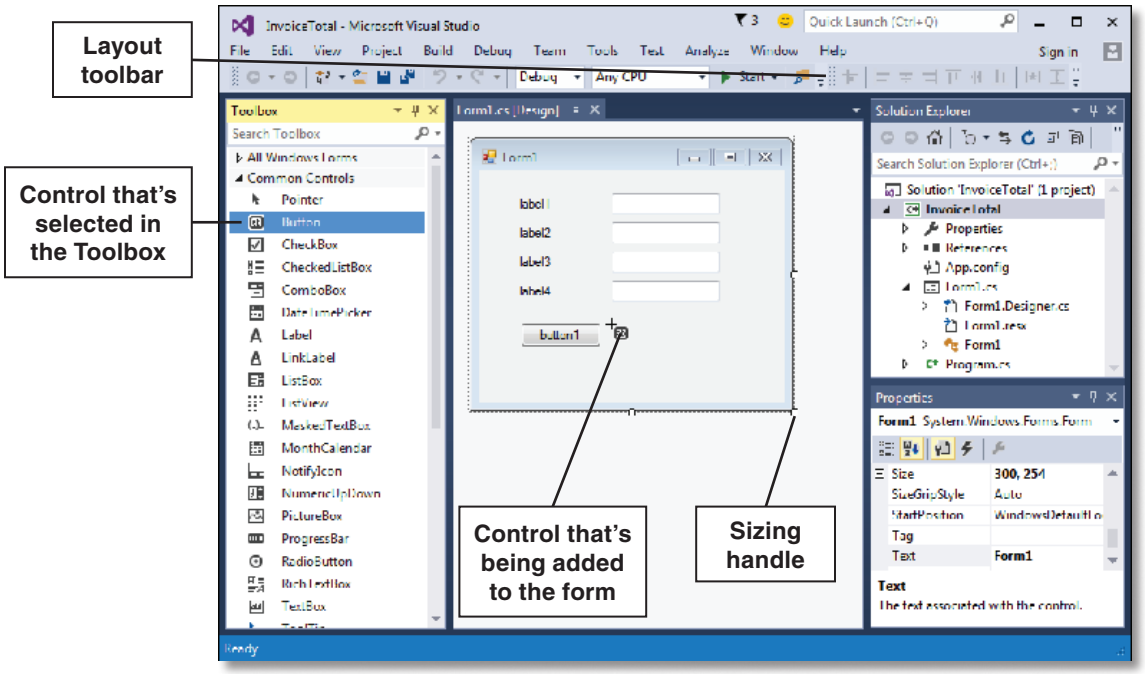
A third method for adding controls is to double-click the control you want to add in the Toolbox. This places the control in the upper left corner of the form. You can then move and resize the control.

If the AutoHide feature is activated for the Toolbox and you click on the Toolbox tab to display it, the display frequently obscures some or all of the form. This makes it difficult to add controls. As a result, it's a good idea to turn off the AutoHide feature when you're adding controls. To do that, just click the pushpin button in the upper right corner of the Toolbox.

After you have added controls to the form, you can work with several controls at once. For example, let's say that you have four text box controls on your form and you want to make them all the same size with the same alignment. To do that, first select all four controls by holding down the Ctrl or Shift key as you click on them or by using the mouse pointer to drag a dotted rectangular line around the controls. Then, use the commands in the Format menu or the buttons in the Layout toolbar to move, size, and align the controls relative to the *primary control*. If you select the controls one at a time, the primary control will be the first control you select. If you select the controls by dragging around them, the primary control will be the last control in the group. To change the primary control, just click on it. (The primary control will have different color handles so you can identify it.)

Although these techniques may be hard to visualize as you read about them, you'll find that they're relatively easy to use. All you need is a little practice, which you'll get in the exercise for this chapter.

A form after some controls have been added to it



Three ways to add a control to a form

- Select the control in the Toolbox. Then, click in the form where you want to place the control. Or, drag the pointer on the form to place the control and size it at the same time.
- Double-click the control in the Toolbox. Then, the control is placed in the upper left corner of the form.
- Drag the control from the Toolbox and drop it on the form. Then, the control is placed wherever you drop it.

How to select and work with controls

- To select a control on the form, click it. To move a control, drag it.
- To size a selected control, drag one of its handles. Note, however, that a label is sized automatically based on the amount of text that it contains. As a result, you can't size a label by dragging its handles unless you change its AutoSize property to False.
- To select more than one control, hold down the Shift or Ctrl key as you click on each control. You can also select a group of controls by clicking on a blank spot in the form and then dragging around the controls.
- To align, size, or space a group of selected controls, click on a control to make it the *primary control*. Then, use the commands in the Format menu or the buttons on the Layout toolbar to align, size, or space the controls relative to the primary control.
- You can also size all of the controls in a group by sizing the primary control in the group. And you can drag any of the selected controls to move all the controls.
- To change the size of a form, click the form and drag one of its sizing handles.

Figure 2-4 How to add controls to a form

How to set properties

After you have placed controls on a form, you need to set each control's *properties*. These are the values that determine how the controls will look and work when the form is displayed. In addition, you need to set some of the properties for the form itself.

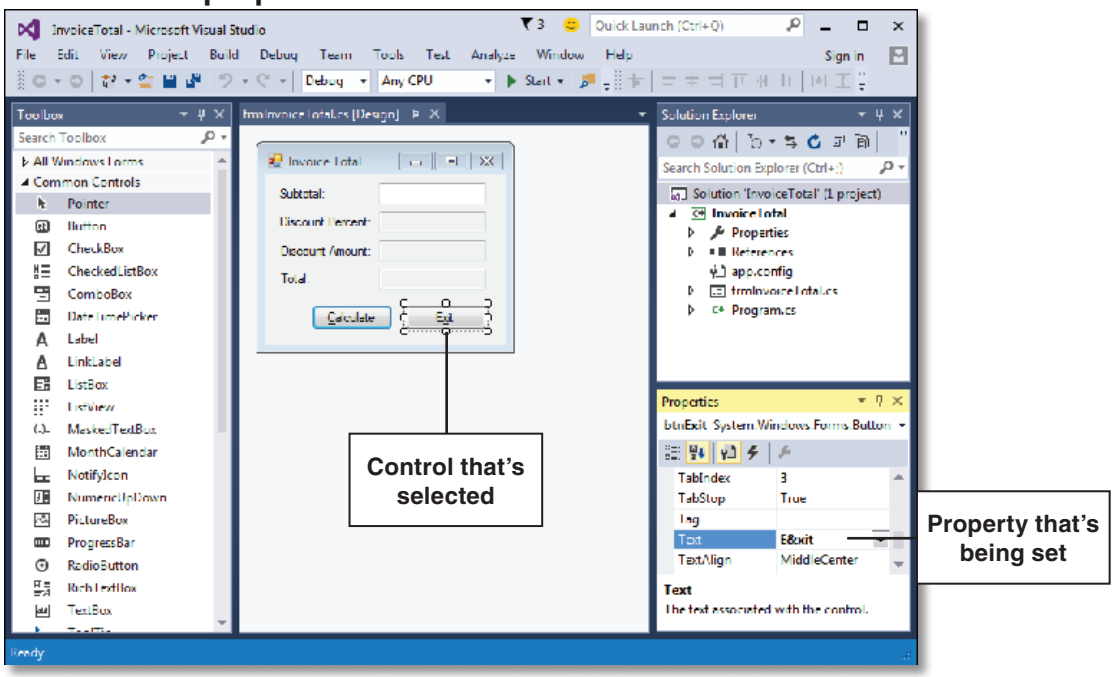
To set the properties of a form or control, you work with the Properties window as shown in figure 2-5. To display the properties for a specific control, click on it in the Form Designer window to select the control. To display the properties for the form, click the form's title bar or any blank area of the form.

In the Properties window, you can select a property by clicking it. When you do, a brief description of that property is given at the bottom of the Properties window. (If you can't see this description, you can drag the bottom line of the window upward.) Then, to change a property setting, you change the entry to the right of the property name by typing a new value or choosing a new value from a drop-down list.

To display properties alphabetically or by category, you can click the appropriate button at the top of the Properties window. At first, you may want to display the properties by category so you have an idea of what the different properties do. Once you become more familiar with the properties, though, you may be able to find the ones you're looking for faster if you display them alphabetically.

As you work with properties, you'll find that most are set the way you want them by default. In addition, some properties such as Height and Width are set interactively as you size and position the form and its controls in the Form Designer window. As a result, you usually only need to change a few properties for each object.

A form after the properties have been set



Description

- The Properties window displays the *properties* for the object that's currently selected in the Form Designer window. To display the properties for another object, click on that object or select the object from the drop-down list at the top of the Properties window.
- To change a property, enter a value into the text box or select a value from its drop-down list if it has one. If a button with an ellipsis (…) appears at the right side of a property's text box, you can click on the ellipsis to display a dialog box that lets you set options for the property.
- To change the properties for two or more controls at the same time, select the controls. Then, the common properties of the controls are displayed in the Properties window.
- When you click on a property in the Properties window, a brief explanation of the property appears in a pane at the bottom of the window. For more information, press F1 to display the help information for the property.
- You can use the first two buttons at the top of the Properties window to sort the properties by category or alphabetically.
- You can use the plus (+) and minus (-) signs displayed to the left of some of the properties and categories in the Properties window to expand and collapse the list of properties.

Note

- If a description isn't displayed when you click on a property in the Properties window, right-click on the window and select Description from the shortcut menu.

Figure 2-5 How to set properties

Common properties for forms and controls

Figure 2-6 shows some common properties for forms and controls. The first two properties apply to both forms and controls. The other properties are presented in two groups: properties that apply to forms and properties that apply to controls. Note that some of the control properties only apply to certain types of controls. That's because different types of controls have different properties.

Since all forms and controls must have a Name property, Visual Studio creates generic names for all forms and controls, such as Form1 or button1. Often, though, you should change these generic names to something more meaningful, especially if you're going to refer to them in your C# code.

To make your program's code easier to read and understand, you can begin each name with a two- or three-letter prefix in lowercase letters to identify the control's type. Then, you can complete the name by describing the function of the control. For instance, you can use a name like btnExit for the Exit button and txtSubtotal for the Subtotal text box.

For Label controls, you can leave the generic names unchanged unless you plan on modifying the properties of the labels in your code. For example, if you want to use a label control to display a message to the user, you can give that label a meaningful name such as lblMessage. But there's no reason to change the names for label controls that display text that won't be changed by the program.

Forms and most controls also have a Text property that is visible when the form is displayed. A form's Text property is displayed in the form's title bar. For a control, the Text property is usually displayed somewhere within the control. The Text property of a button, for example, is displayed on the button, and the Text property of a text box is displayed in the text box.

As you work with properties, you'll find that you can set some of them by selecting a value from a drop-down list. For example, you can select a True or False value for the TabStop property of a control. For other properties, you have to enter a number or text value. And for some properties, a button with an ellipsis (...) is displayed. Then, when you click this button, a dialog box appears that lets you set the property.

The Name property

- Sets the name you use to identify a control in your C# code.
- Can be changed to provide a more descriptive and memorable name for forms and controls that you will refer to when you write your code (such as text boxes and buttons).
- Doesn't need to be changed for controls that you won't refer to when you write your code (such as most labels).
- Can use a three-letter prefix to indicate whether the name refers to a form (frm), button (btn), label (lbl), or text box (txt).

The Text property

- Sets the text that's displayed on the form or control. Some controls such as forms and labels display the generic form or control name that's generated by Visual Studio, which you'll almost always want to change.
- For a form, the Text value is displayed in the title bar. For controls, the Text value is displayed directly on the control.
- For a text box, the Text value changes when the user types text into the control, and you can write code that uses the Text property to get the text that was entered by the user.

Other properties for forms

Property	Description
`AcceptButton`	Identifies the button that will be activated when the user presses the Enter key.
`CancelButton`	Identifies the button that will be activated when the user presses the Esc key.
`StartPosition`	Sets the position at which the form is displayed. To center the form, set this property to CenterScreen.

Other properties for controls

Property	Description
`Enabled`	Determines whether the control will be enabled or disabled.
`ReadOnly`	Determines whether the text in some controls like text boxes can be edited.
`TabIndex`	Indicates the control's position in the tab order, which determines the order in which the controls will receive the focus when the user presses the Tab key.
`TabStop`	Determines whether the control will accept the focus when the user presses the Tab key to move from one control to another. Some controls, like labels, don't have the TabStop property because they can't receive the focus.
`TextAlign`	Sets the alignment for the text displayed on a control.

Figure 2-6 Common properties for forms and controls

How to add navigation features

Windows forms have features that make it easier for users to move around in the forms without using the mouse. These navigation features are described in figure 2-7.

The *tab order* is the order in which the controls on a form receive the *focus* when the user presses the Tab key. The tab order should usually be set so the focus moves left-to-right and top-to-bottom, beginning at the top left of the form and ending at the bottom right. However, in some cases you'll want to deviate from that order. For example, if you have controls arranged in columns, you may want the tab order to move down each column.

The tab order is initially set based on the order in which you add controls to the form. So if you add the controls in the right order, you won't need to alter the tab order. But if you do need to change the tab order, you can do so by adjusting the TabIndex property settings. The TabIndex property is simply a number that represents the control's position in the tab order, beginning with zero. So, the first control in the tab order has a TabIndex of 0, the second control's TabIndex is 1, and so on.

Incidentally, chapter 10 will show you another way to set the tab order of the controls for a form. You can do that by using Tab Order view. When a form consists of more than a few controls, it is easier to use this view than to set the tab order for one control at a time.

Access keys are shortcut keys that let the user move the focus directly to a control. You set a control's access key by using the Text property. Just precede the letter in the Text property value that you want to use as the access key with an ampersand (&). Then, the user can move the focus to the control by pressing Alt plus the access key.

If you assign an access key to a control that can't receive the focus, such as a label control, pressing the access key causes the focus to move to the next control in the tab order that can receive the focus. As a result, you can use an access key with a label control to create a shortcut for a text box control, which can't have an access key.

If you assign an access key to a button control, you should know that pressing Alt plus the access key doesn't simply move the focus to the control. Instead, it activates the control just as if it was clicked. Without an access key, a user would have to tab to the button control and then press the Enter key to activate it using the keyboard. The exception is if the button is selected for the AcceptButton or CancelButton property for the form.

The AcceptButton and CancelButton properties specify the buttons that are activated when the user presses the Enter and Esc keys. That can make it easier for a user to work with a form. If, for example, the AcceptButton property of the Invoice Total form in figure 2-3 is set to the Calculate button, the user can press the Enter key after entering a subtotal instead of using the mouse to click the Calculate button or using the access key to activate it.

How to adjust the tab order

- *Tab order* refers to the sequence in which the controls receive the *focus* when the user presses the Tab key. You should adjust the tab order so the Tab key moves the focus from one control to the next in a logical sequence.
- Each control has a TabIndex property that indicates the control's position in the tab order. You can change this property to change a control's tab order position.
- If you don't want a control to receive the focus when the user presses the Tab key, change that control's TabStop property to False.
- Label controls don't have a TabStop property so they can't receive the focus.

How to set access keys

- *Access keys* are shortcut keys that the user can use in combination with the Alt key to quickly move to individual controls on the form.
- You use the Text property to set the access key for a control by placing an ampersand immediately before the letter you want to use for the access key. For example, &Invoice sets the access key to *I*, but I&nvoice sets the access key to *n*.
- Since the access keys aren't case sensitive, &N and &n set the same access key.
- When you set access keys, make sure to use a unique letter for each control. If you don't, the user may have to press the access key two or more times to select a control.
- You can't set the access key for a text box. However, if you set an access key for a label that immediately precedes the text box in the tab order, the access key will take the user to the text box.
- If you assign an access key to a button, the button is activated when you press Alt plus the access key.

How to set the Enter and Esc keys

- The AcceptButton property of the form sets the button that will be activated if the user presses the Enter key.
- The CancelButton property of the form sets the button that will be activated if the user presses the Esc key. This property should usually be set to the Exit button.
- You set the AcceptButton or CancelButton values by choosing the button from a drop-down list that shows all of the buttons on the form. So be sure to create and name the buttons you want to use before you attempt to set these values.

Another way to set the tab order

- In chapter 10, you'll learn how to use Tab Order view to set the tab order of the controls on the form. If the form consists of more than a few controls, that is the best way to set that order.

Figure 2-7 How to add navigation features

The property settings for the Invoice Total form

Figure 2-8 shows the property settings for the Invoice Total form. As you can see, you don't need to change many properties to finish the design of this form. You only need to set four properties for the form, and you only use six of the properties (Name, Text, TextAlign, ReadOnly, TabStop, and TabIndex) for the controls. Depending on the order in which you create the controls, though, you may not need to change the TabIndex settings.

Notice that the three text boxes that display the form's calculation have their ReadOnly property set to True. This setting gives the text boxes a shaded appearance, as you saw in figure 2-3, and it prevents the user from entering text into these controls. In addition, the TabStop property for these text boxes has been set to False so the user can't use the Tab key to move the focus to these controls.

Finally, the settings for the TabIndex properties of the text box and the two buttons are 1, 2, and 3. Since the label controls can't receive the focus, and since the TabStop property for the three read-only text boxes has been set to False, the user can press the Tab key to move the focus from the Subtotal text box to the Calculate button to the Exit button.

In addition, the Subtotal label has a TabIndex property of 0 and a Text property that includes an access key of S. As a result, the user can press Alt+S to move the focus to the control that has the next available tab index. In this case, that control is the Subtotal text box, which has a TabIndex property of 1.

Of course, this is just one way that the TabIndex properties could be set. If, for example, the TabIndex properties for the 10 controls were set from 0 through 9, from top to bottom in this summary, the tab order would work the same.

How to use Document Outline view

Document Outline view is a feature that first became available with Visual Studio 2005. To open the window for this view, you use the View→Other Windows→Document Outline command. This opens a window to the left of the Form Designer that lists the names of all of the controls that have been added to the current form.

This view makes it easy to check whether you've named all of the controls that you're going to refer to in your C# code. You can also select a control in the Form Designer by clicking on the name of the control in Document Outline view. Although these are minor benefits, it's worth experimenting with this view to see whether you're going to want to use it.

The property settings for the form

Default name	Property	Setting
Form1	Text	Invoice Total
	AcceptButton	btnCalculate
	CancelButton	btnExit
	StartPosition	CenterScreen

The property settings for the controls

Default name	Property	Setting
label1	Text	&Subtotal:
	TextAlign	MiddleLeft
	TabIndex	0
label2	Text	Discount percent:
	TextAlign	MiddleLeft
label3	Text	Discount amount:
	TextAlign	MiddleLeft
label4	Text	Total:
	TextAlign	MiddleLeft
textBox1	Name	txtSubtotal
	TabIndex	1
textBox2	Name	txtDiscountPercent
	ReadOnly	True
	TabStop	False
textBox3	Name	txtDiscountAmount
	ReadOnly	True
	TabStop	False
textBox4	Name	txtTotal
	ReadOnly	True
	TabStop	False
button1	Name	btnCalculate
	Text	&Calculate
	TabIndex	2
button2	Name	btnExit
	Text	E&xit
	TabIndex	3

Note

- To provide an access key for the Subtotal text box, you can set the TabIndex and Text properties for the Subtotal label as shown above.

Figure 2-8 The property settings for the Invoice Total form

How to name and save the files of a project

When you're working on a project, you may want to change the names of some of the files from their defaults. Then, you'll want to save the files with their new names.

How to name the files of a project

You may have noticed throughout this chapter that I didn't change the default name of the form (Form1.cs) that was added to the Invoice Total project when the project was created. In practice, though, you usually change the name of this form so it's more descriptive. For example, figure 2-9 shows how to use the Solution Explorer to change the name of the form file to frmInvoiceTotal.cs. When you do that, Visual Studio will also change the File Name property for the form from Form1 to frmInvoiceTotal, and it will ask you if you want modify any references to the form.

You may also want to change the name of the project or solution. For example, if you accepted the default project name when you started the project (WindowsApplication1.proj), you may want to change it to something more meaningful. Or, you may want to change the name of the solution so it's different from the project name. If so, you can use the technique presented in this figure to do that too.

By default, the name of the assembly that's created for a project and the name of the namespace that contains the project are the same as the project name. Unfortunately, if you change the project name, the assembly and namespace names aren't automatically changed. To change these names, you can use the Application tab of the Project Properties window as described in this figure.

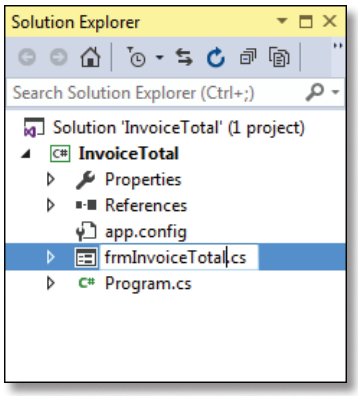
How to save the files of a project

Figure 2-9 also describes how to save the files of a project. Because Visual Studio saves any changes you make to the files in a project when you build the project, you won't usually need to save them explicitly. However, it's easy to do if you need to.

Notice in this figure that two factors determine which files are saved: what's selected in the Solution Explorer and the command you use to perform the save operation. If, for example, a single file is selected, you can use the Save command to save just that file, and you can use the Save All command to save the file along with the project and solution that contain the file. In contrast, if a project is selected in the Solution Explorer, the Save command causes the entire project to be saved, and the Save All command causes the entire solution to be saved.

If you haven't saved all of your recent changes when you close a project, Visual Studio will ask whether you want to save them. As a result, you don't need to worry that your changes will be lost.

The Solution Explorer as a form file is being renamed



How to rename a file, project, or solution

- You can rename a file, project, or solution by right-clicking on it in the Solution Explorer window and selecting the Rename command from the shortcut menu. Or, you can select the file, project, or solution in the Solution Explorer and press F2. Then, you can enter the new name for the file, project, or solution.
- Be sure not to change or omit the file extension when you rename a file. Remember too that using a three-letter prefix to indicate the contents of the file (like *frm* for a form file) makes it easier to tell what each file represents.
- When you change the name of a file, Visual Studio also changes the File Name property for the form. It also asks you if you want to change all references to the file, which is usually what you want.
- If you change the name of a project, you may also want to change the name of the assembly that's created for the project and the name of the namespace that contains the project. To do that, you can use the Project→ProjectName Properties command to display the Project Properties window. Then, you can change the names in the Assembly Name and Default Namespace text boxes on the Application tab.

How to save a file, project, or solution

- You can use the Save All button in the Standard toolbar or the Save All command in the File menu to save all files and projects in the solution.
- You can use the Save button in the Standard toolbar or the Save command in the File menu to save a file, project, or solution. The files that are saved depend on what's selected in the Solution Explorer window. If a single file is selected, just that file is saved. If a project is selected, the entire project and its solution are saved. And if a solution is selected, the entire solution and all its projects are saved.
- If you try to close a solution that contains modified files, a dialog box is displayed that asks you if you want to save those files.

Figure 2-9 How to name and save the files of a project

Perspective

If you can design the Invoice Total form that's presented in this chapter, you've taken a critical first step toward learning how to develop Windows Forms applications with Visual Studio 2015. The next step is to add the code that makes the form work the way you want it to, and that's what you'll learn to do in the next chapter.

Terms

template
label
text box
button
primary control
property
tab order
focus
access key

Exercise 2-1 Design the Invoice Total form

This exercise will guide you through the process of starting a new project and developing the user interface for the Invoice Total form shown in this chapter.

Set the default path and start a new project

1. Start Visual Studio. If you want to change the import and export settings to make sure your menus are the same as the ones in this book, use the Tools→Import and Export Settings command described in figure 2-1 to specify the default Visual C# development settings.
2. Use the Tools→Options command to display the Options dialog box as shown in figure 2-1. Then, expand the Projects and Solutions group, select the General category, and change the projects location setting to C:\C# 2015.
3. If the Save New Projects When Created box and the Always Show Solution box aren't checked, check them.
4. If you want to stop the Start Page from being displayed each time you start Visual Studio, click on the Startup category within the Environment group. Then, select another option from the At Startup drop-down list.
5. If you're interested, take a few minutes to review the other options that are available in this dialog box. Then, close the dialog box.
6. Start a new project as shown in figure 2-2. The project should be named InvoiceTotal, it should be stored in the C:\C# 2015\Chapter 02 folder, and the solution should be stored in its own folder.

Add controls to the new form and set the properties

7. Use the techniques in figure 2-4 to add controls to the form so they have approximately the same sizes and locations as in figure 2-5. But don't worry about the size of the labels, just their locations.

8. Select groups of controls and use the buttons in the Layout toolbar to size and align the controls. But here again, let the labels automatically size themselves. Then, size the form so it looks like the one in figure 2-4.
9. Use the Properties window to set the properties for the form and its controls so it looks like the form in figure 2-3. These properties are summarized in figure 2-8.
10. Use the View→Other Windows→Document Outline command to open the window for Document Outline view. Next, use this window to check that you've named all of the controls that have Name properties in figure 2-8. Then, click on the controls in this view to see what happens, and close this window when you're done.

Test the user interface

11. Press F5 to build and run the project. That should display the form in the center of the screen, and it should look like the one in figure 2-3.
12. Experiment with the form to see what it can do. When you press the Tab key, notice how the focus moves from one control to another. When you click a button, notice how it indents and then pops back out just like any other Windows button control. Nothing else happens in response to these button clicks, though, because you haven't written the code for them yet.

    Notice that the Calculate button has a dark outline around it to indicate that its function will be executed if you press the Enter key. (If it doesn't have a dark outline, you haven't set the AcceptButton property of the form to the button.)

    When you press the Alt key, notice that an underline appears under the s in Subtotal, the first c in Calculate, and the x in Exit to indicate that you can use an access key to work with these controls. (If the underlines don't show, you haven't entered the Text properties correctly.)
13. If you notice that some of the properties are set incorrectly, click the Close button in the upper right corner of the form to close the form. Then, make the necessary changes and run the project again. When you're satisfied that the form is working right, close the form to return to the Form Designer.

Experiment with the properties for the form and its controls

14. In the Form Designer, click on the form so it is selected. Then, if necessary, adjust the Properties window so you can see the description for each property. To do that, drag the bottom boundary of the window up.
15. Click on the Categorized button at the top of the Properties window to display the properties by category. Then, review the properties in the Appearance, Behavior, Layout, and Window Style categories. Although you won't understand all of the descriptions, you should understand some of them.
16. In the Window Style category, change the settings for the MaximizeBox and MinimizeBox to false to see how that changes the form. Then, to undo those changes, click twice on the Undo button in the Standard toolbar or press Ctrl+Z twice.

17. Click on the first text box and review the Appearance, Behavior, and Layout properties for that control. Then, repeat this process for one of the labels and one of the buttons. Here again, you won't understand all of the descriptions, but you should understand some of them.
18. Select all four of the labels, click on the plus sign before the Font property in the Appearance group, and change the Bold setting to True to see how that changes the form. Then, undo that change.

Change the name of the form files

19. Use one of the techniques presented in figure 2-9 to change the name of the form file from Form1.cs to frmInvoiceTotal.cs. When you do that, a dialog box is displayed that asks whether you want to change all of the references to the form name too. Click on Yes to close that dialog box, and Visual Studio changes all references to Form1 in the C# code that has been generated to frmInvoiceTotal.
20. Note in the Solution Explorer that this also changes the name of the two subordinate files to frmInvoiceTotal.Designer.cs and frmInvoiceTotal.resx.

Close the project and exit from Visual Studio

21. Use the File→Close Solution command to close the project. If you've made any changes to the project since the last time you tested it, a dialog box is displayed that asks whether you want to save the changes that you made. If you want to save those changes, click Yes.
22. Use the File→Exit command to exit from Visual Studio.

How to build your C# programming skills

Mike Murach, Publisher

The easiest way is to let **Murach's C# 2015** be your guide! So if you've enjoyed this chapter, I hope you'll get your own copy of the book today. You can use it to:

- Teach yourself how to code Windows Forms applications in C#
- Take advantage of all the time- and work-saving features of Visual Studio 2015 as you develop C# applications
- Build database applications using RAD features like data sources and the DataGridView control...and start using features like ADO.NET code, LINQ, and the Entity Framework as well
- Use object-oriented programming techniques the way the pros do
- Pick up a new skill whenever you want or need to by focusing on material that's new to you
- Look up coding details or refresh your memory on forgotten details when you're in the middle of developing a C# application
- Loan to your colleagues who will be asking you more and more questions about C# programming

To get your copy, you can order online at **www.murach.com** or call us at 1-800-221-5528 (toll-free in the U.S. and Canada). And remember, when you order directly from us, this book comes with my personal guarantee:

100% Guarantee

You must be satisfied. Each book you buy directly from us must outperform any competing book or course you've ever tried, or send it back within 60 days for a full refund…no questions asked.

Thanks for your interest in Murach books!