Dive Into® Visual Studio Express 2012 for Windows Desktop

Seeing is believing.

—Proverb

Form ever follows function.

—Louis Henri Sullivan

OBJECTIVES

In this chapter you'll:

- Learn the basics of the Visual Studio Express 2012 for Windows Desktop Integrated Development Environment (IDE) for writing, running and debugging your apps.
- Use Visual Studio's help features.
- Learn key commands contained in the IDE's menus and toolbars.
- Understand the purpose of the various kinds of windows in the Visual Studio Express 2012 for Windows Desktop IDE.
- Understand what visual app development is and how it simplifies and speeds app development.
- Use visual app development to create, compile and execute a simple Visual C# app that displays text and an image.

2.1 Introduction

- Visual Studio 2012 is Microsoft's Integrated Development Environment (IDE) for creating, running and debugging apps (also called applications) written in various .NET programming languages.
- This chapter provides an overview of the Visual Studio 2012 IDE and shows how to create a simple Visual C# app by dragging and dropping predefined building blocks into place—a technique known as visual app development.

2.2 Overview of the Visual Studio Express 2012 IDE

- Most of this book's examples are based on the Visual Studio Express 2012 for Windows Desktop, which supports only the Visual C# programming language.
- See the Before You Begin section that follows the Preface for information on installing the software.
- Our screen captures and discussions focus on Visual Studio Express 2012 for Windows Desktop.
- The examples will work on full versions of Visual Studio as well—though some options, menus and instructions might differ.
- We'll refer to the Visual Studio Express 2012 for Windows Desktop IDE simply as "Visual Studio" or "the IDE."

2.2 Overview of the Visual Studio 2012 IDE

- ***Introduction to Microsoft Visual Studio Express 2012 for Windows Desktop***
  - We use the > character to indicate the selection of a *menu item* from a *menu*.
  - For example, we use the notation FILE > Open File… to indicate that you should select the Open File… menu item from the FILE menu.
  - To start the IDE, select Start > All Programs > Microsoft Visual Studio 2012 Express > VS Express for Desktop (on Windows 8, click the VS for Desktop tile on the Start screen).
  - Once the Express Edition begins execution, the Start Page displays.
  - The Start Page contains a list of links to Visual Studio resources and web-based resources.
  - At any time, you can return to the Start Page by selecting VIEW > Start Page.

2.2 Overview of the Visual Studio 2012 IDE

Links on the Start Page

- The Start Page links are organized into two columns.
  - The left column's Start section contains options that enable you to start building new apps or to continue working on existing ones.
  - The Recent section contains links to projects you've recently created or modified.
- You can also create new projects or open existing ones by clicking the links in the Start section.

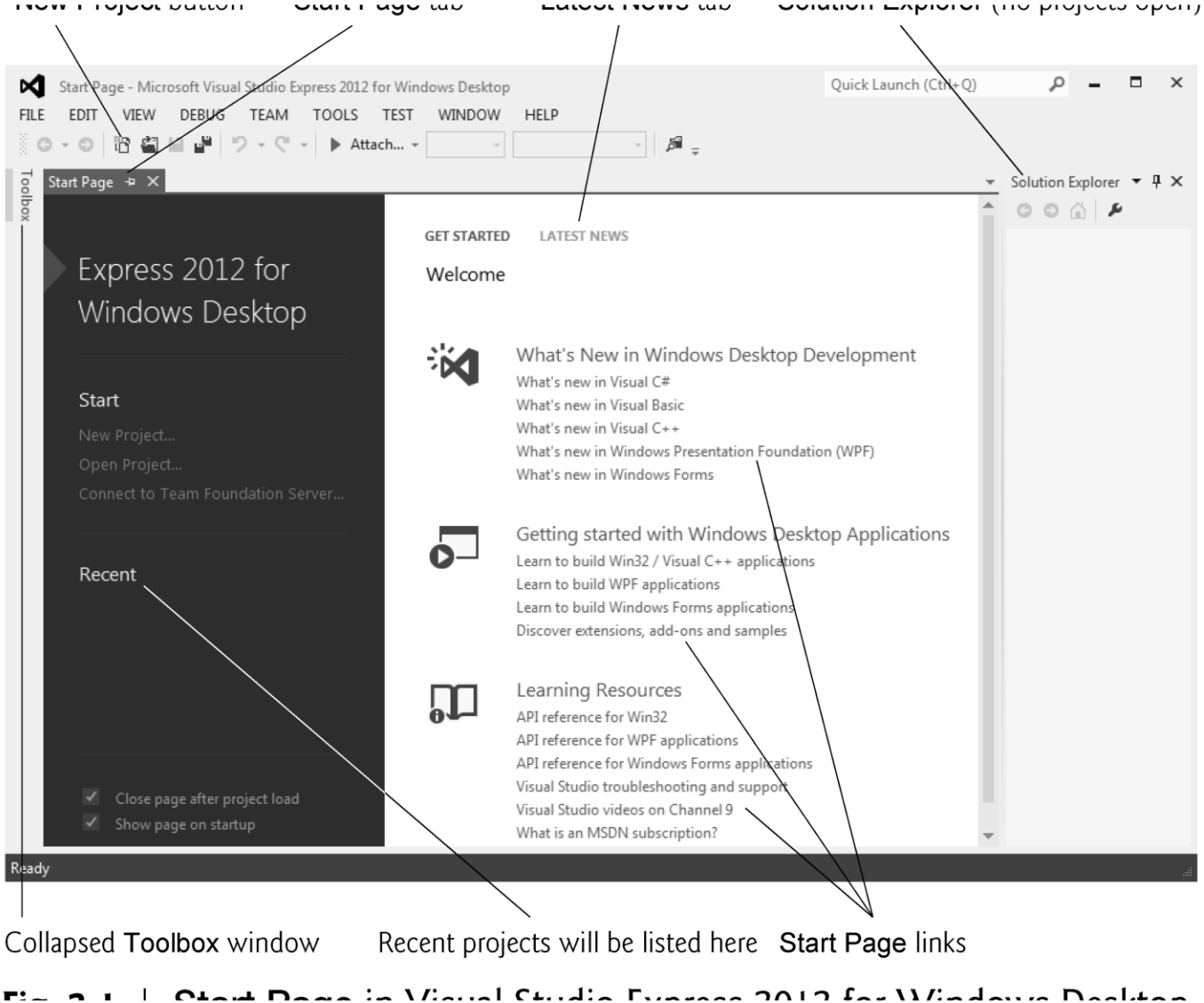


Fig. 3.1 | Start Page in Visual Studio Express 2012 for Windows Desktop

2.2 Overview of the Visual Studio 2012 IDE

- The links in the GET STARTED tab provide information about the programming languages supported by Visual Studio and various learning resources.
- An Internet connection is required for the IDE to access most of this information.
- The LATEST NEWS tab includes an Enable RSS Feed button.
  - Once you click this button, the IDE will display links to the latest Visual Studio developments (such as updates and bug fixes) and to information on advanced app-development topics.

2.2 Overview of the Visual Studio 2012 IDE

- The MSDN site contains articles, downloads and tutorials on technologies of interest to Visual Studio developers.
- You can also browse the web from the IDE by selecting VIEW > Other Windows > Web Browser.
- To request a web page, type its URL into the location bar and press the *Enter* key—your computer, of course, must be connected to the Internet.
- The web page that you wish to view appears as another tab in the IDE.

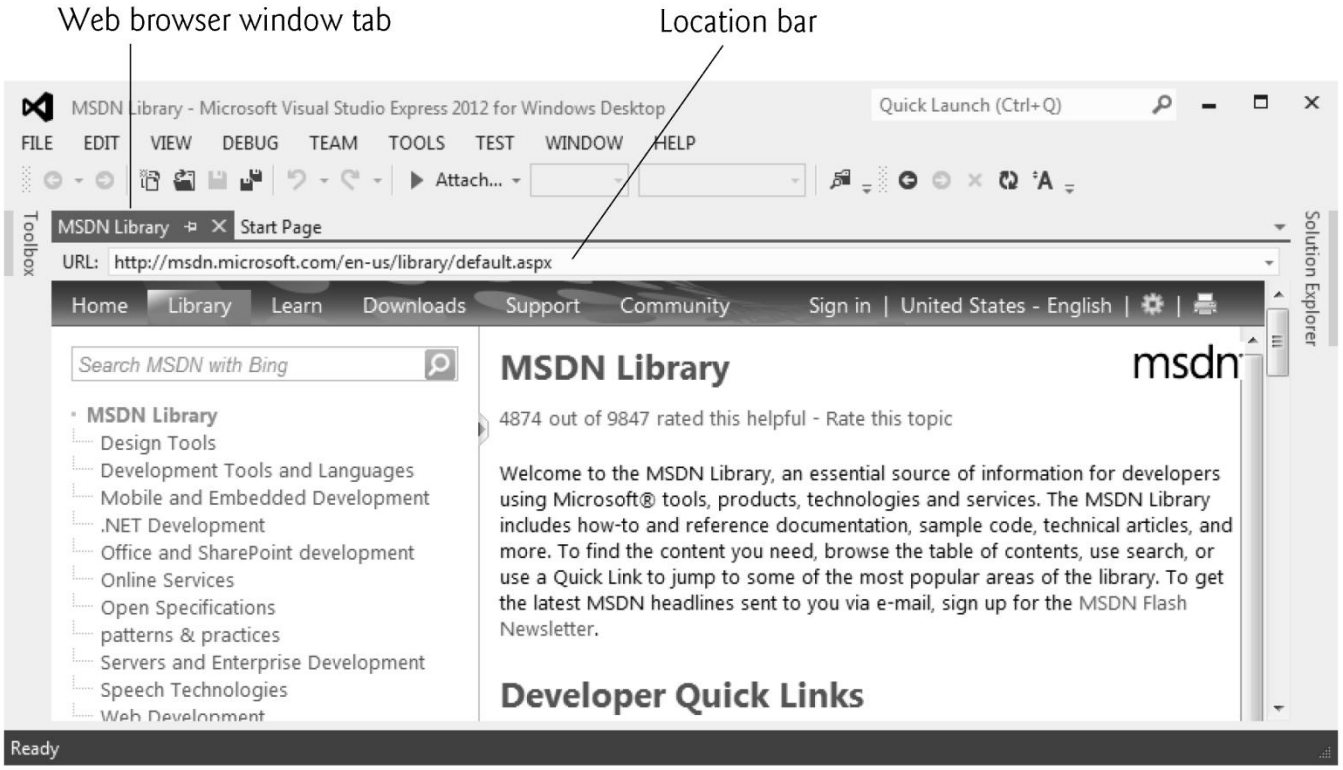


Fig. 2.2 | **MSDN Library** web page in Visual Studio.

2.2 Overview of the Visual Studio 2012 IDE

- ***Creating a New Project***
  - To begin app development in Visual C#, you must create a new project or open an existing one.
  - You select FILE > New Project… to create a new project or FILE > Open Project… to open an existing one.
  - From the Start Page's Start section, you can also click the links New Project… or Open Project….
  - A project is a group of related files, such as the Visual C# code and any images that might make up an app.
  - Visual Studio organizes apps into projects and solutions, which contain one or more projects.

2.2 Overview of the Visual Studio 2012 IDE

- Multiple-project solutions are for large-scale apps.
- Most apps we create consist of a solution containing a single project.

2.2 Overview of the Visual Studio 2012 IDE

New Project Dialog and Project Templates

- When you select FILE > New Project… or click the New Project… link on the Start Page, the New Project dialog displays.
- Dialogs are windows that facilitate user–computer communication.
- Visual Studio provides several templates—the *project types* users can create in Visual C# and other languages.
- The templates include Windows Forms apps, WPF apps and others—full versions of Visual Studio provide many additional templates.

2.2 Overview of the Visual Studio 2012 IDE

- In this chapter, we build a Windows Forms Application.
- A Windows Forms app executes within a Windows operating system (such as Windows 7 or Windows 8) and typically has a graphical user interface (GUI)—users interact with this *visual* part of the app.
- Windows apps include Microsoft software products like Microsoft Word, Internet Explorer and Visual Studio; software products created by other vendors; and customized software that you and other app developers create.

2.2 Overview of the Visual Studio 2012 IDE

- By default, Visual Studio assigns the name WindowsFormsApplication1 to a new Windows Forms Application project and solution.
- Select Windows Forms Application, then click OK to display the IDE in Design view, which contains the features that enable you to create an app's GUI.

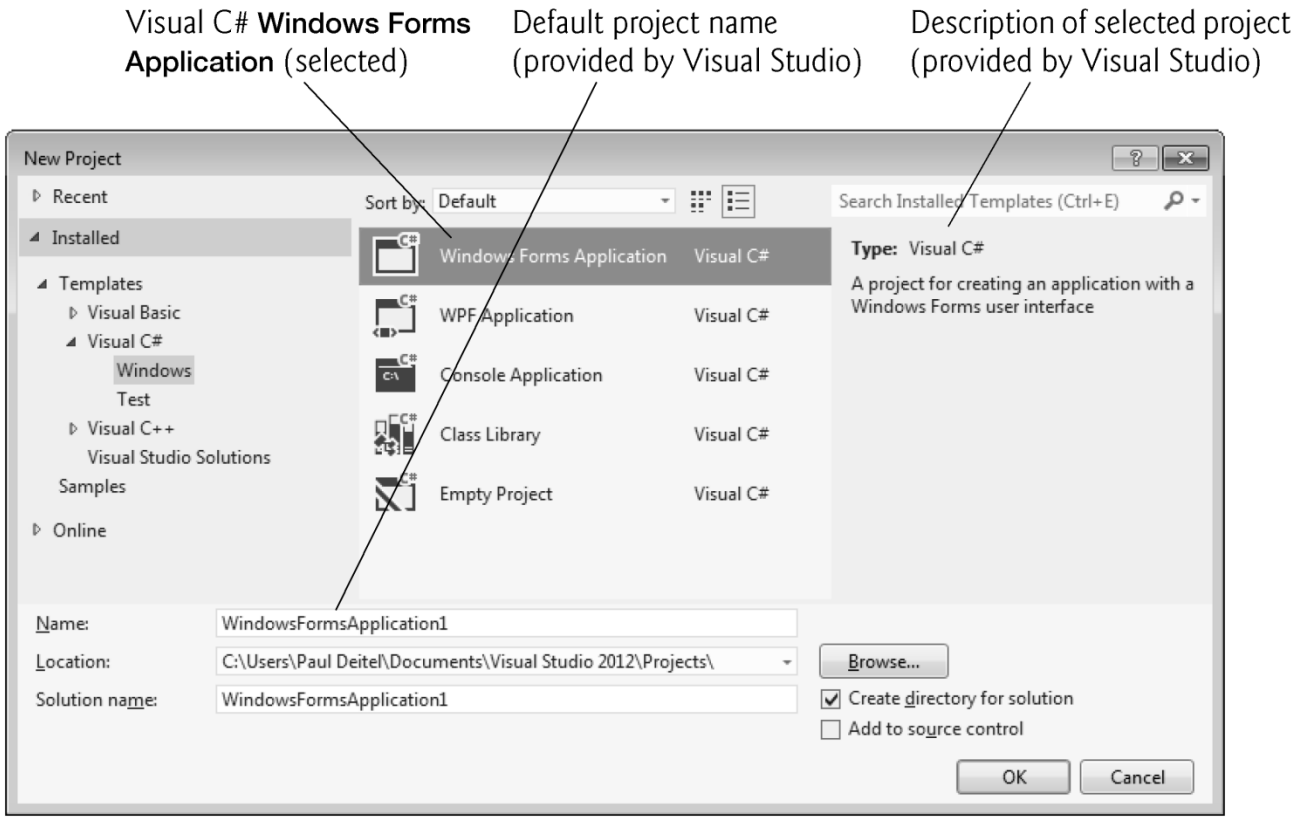


Fig. 2.3 | **New Project** dialog.

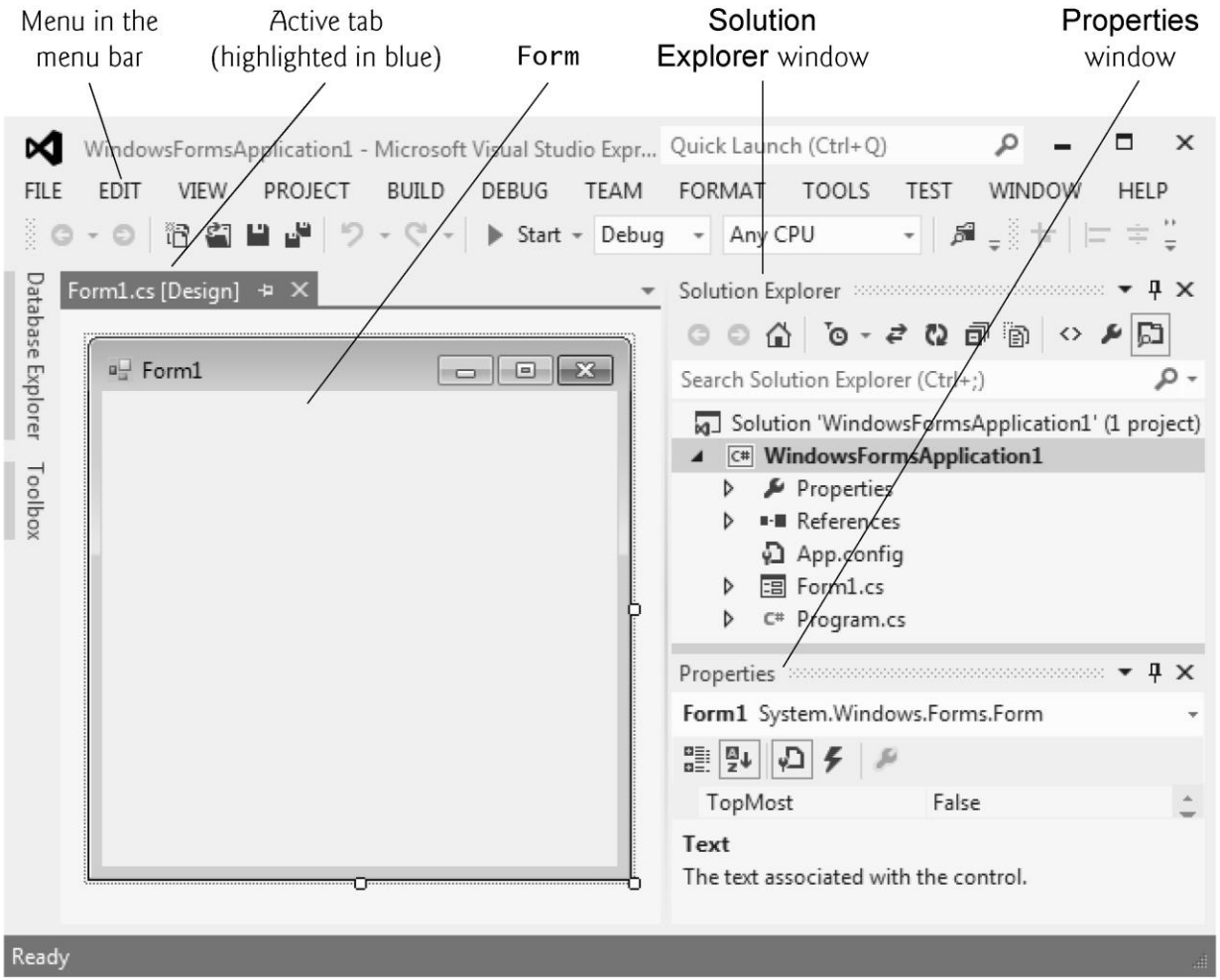


Fig. 2.4 | **Design** view of the IDE.

2.2 Overview of the Visual Studio 2012 IDE

Forms and Controls

- The rectangle in the `Design` area titled `Form1` (called a Form) represents the main window of the Windows `Forms` app that you're creating.
- Visual C# apps can have multiple `Forms` (windows)—but we'll use use only one `Form`.
- You'll learn how to customize the `Form` by adding GUI controls—in this example, you'll add a `Label` and a `PictureBox` (as you'll see in Fig. 2.20).
- A Label typically contains descriptive text (for example, `"Welcome to Visual C#!"`), and a Picture-Box displays an image.

2.2 Overview of the Visual Studio 2012 IDE

- Visual Studio has many preexisting controls and other components you can use to build and customize your apps.
- In this chapter, you'll work with preexisting controls from the .NET Framework Class Library.
- As you place controls on the `Form`, you'll be able to modify their properties (discussed in Section 2.4).
- For example, Fig. 2.5 shows where the `Form`'s title can be modified and Fig. 2.6 shows a dialog in which a control's font properties can be modified.

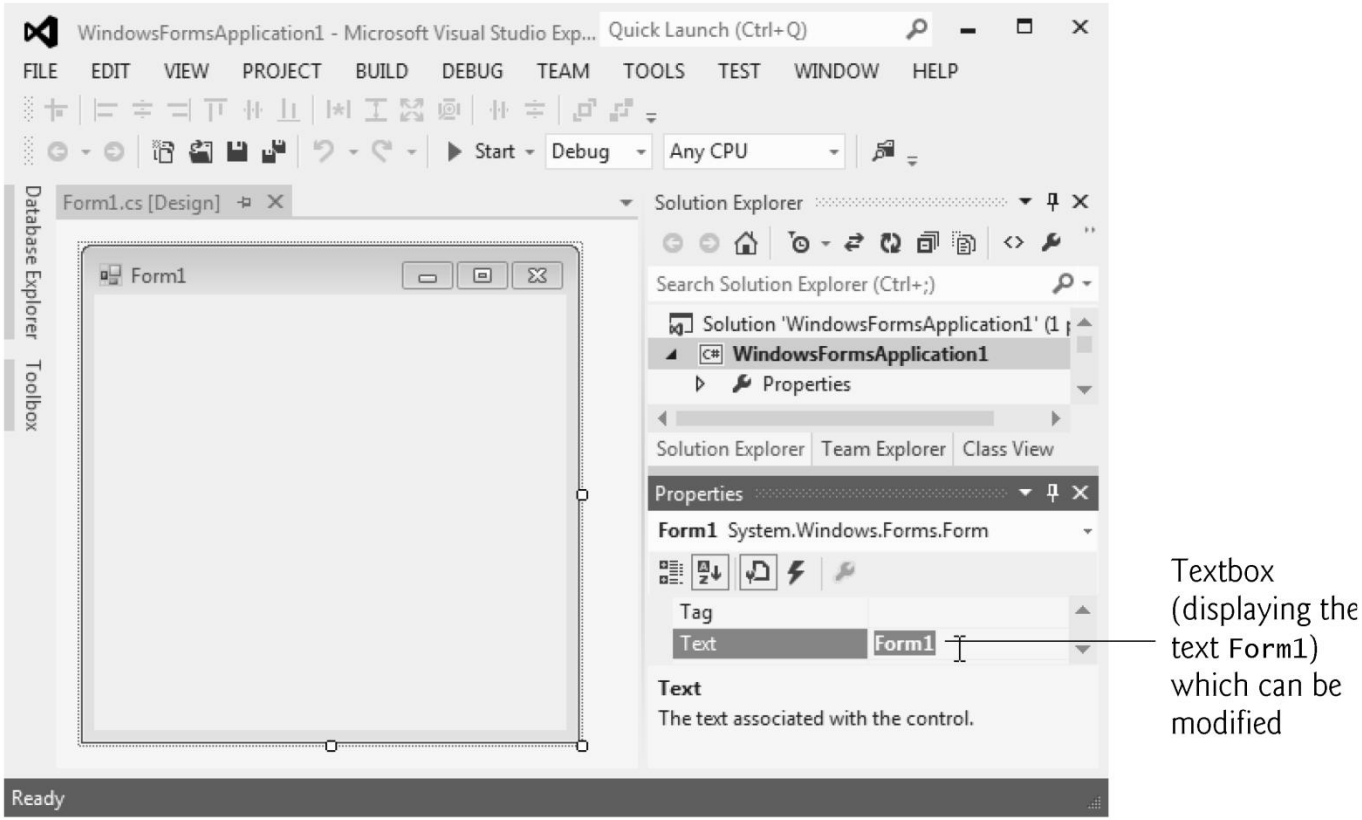


Fig. 2.5 | Textbox control for modifying a property in the Visual Studio IDE.

2.2 Overview of the Visual Studio 2012 IDE

- Collectively, the `Form` and controls make up the app's GUI.
- Users enter data into the app by typing at the keyboard, by clicking the mouse buttons and in a variety of other ways.
- Programs use the GUI to display instructions and other information for users to view.
- For example, the New Project dialog in Fig. 2.3 presents a GUI where the user clicks the mouse button to select a template type, then inputs a project name from the keyboard (the figure is still showing the default project name WindowsFormsApplication1 supplied by Visual Studio).

2.2 Overview of the Visual Studio 2012 IDE

- Each open document's name is listed on a tab.
- To view a document when multiple documents are open, click its tab.
- The active tab (the tab of the currently displayed document) is highlighted in blue (for example, Form1.cs [Design] in Fig. 2.4).

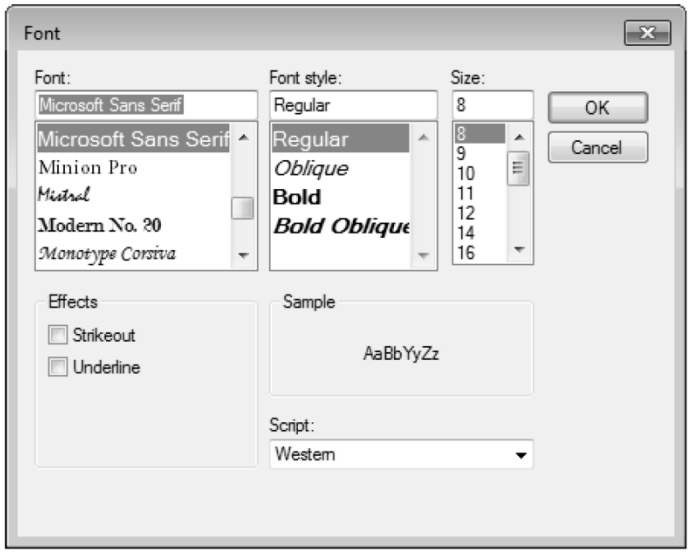


Fig. 2.6 | Dialog for modifying a control's font properties.

2.3 Menu Bar and Toolbar

- Commands for managing the IDE and for developing, maintaining and executing apps are contained in menus, which are located on the menu bar of the IDE
- The set of menus displayed depends on what you're currently doing in the IDE.

Fig. 2.7 | Visual Studio menu bar.

2.3 Menu Bar and Toolbar

- Menus contain groups of related commands (also called menu items) that, when selected, cause the IDE to perform specific actions—for example, open a window, save a file, print a file and execute an app.
- For example, new projects are created by selecting FILE > New Project….
- The menus depicted in Fig. 2.7 are summarized in Fig. 2.8.

Menu	Description
FILE	Contains commands for opening, closing, adding and saving projects, as well as printing project data and exiting Visual Studio.
EDIT	Contains commands for editing apps, such as cut, copy, paste, undo, redo, delete, find and select.
VIEW	Contains commands for displaying IDE windows (for example, **Solution Explorer**, **Toolbox**, **Properties** window) and for adding toolbars to the IDE.
PROJECT	Contains commands for managing projects and their files.
BUILD	Contains options for turning your app into an executable program.
DEBUG	Contains commands for compiling, debugging (that is, identifying and correcting problems in apps) and running apps.
TEAM	Allows you to connect to a Team Foundation Server—used by development teams that typically have multiple people working on the same app.
FORMAT	Contains commands for arranging and modifying a `Form`'s controls. The **Format** menu appears *only* when a GUI component is selected in **Design** view.

Fig. 2.8 | Summary of Visual Studio menus that are displayed when a `Form` is in **Design** view. (Part 1 of 2.)

Menu	Description
TOOLS	Contains commands for accessing additional IDE tools and options for customizing the IDE.
TEST	Contains options for performing various types of automated testing on your app.
WINDOW	Contains commands for hiding, opening, closing and displaying IDE windows.
HELP	Contains commands for accessing the IDE's help features.

Fig. 2.8 | Summary of Visual Studio menus that are displayed when a `Form` is in **Design** view. (Part 2 of 2.)

2.3 Menu Bar and Toolbar

- You can access many common menu commands from the toolbar, which contains icons, that graphically represent commands.
- By default, the standard toolbar is displayed when you run Visual Studio for the first time—it contains icons for the most commonly used commands, such as opening a file, adding an item to a project, saving files and running apps (Fig. 2.9).
- Some commands are initially disabled (grayed out or unavailable to use) and are enabled only when needed.
- For example, Visual Studio enables the command for saving a file once you begin editing a file.

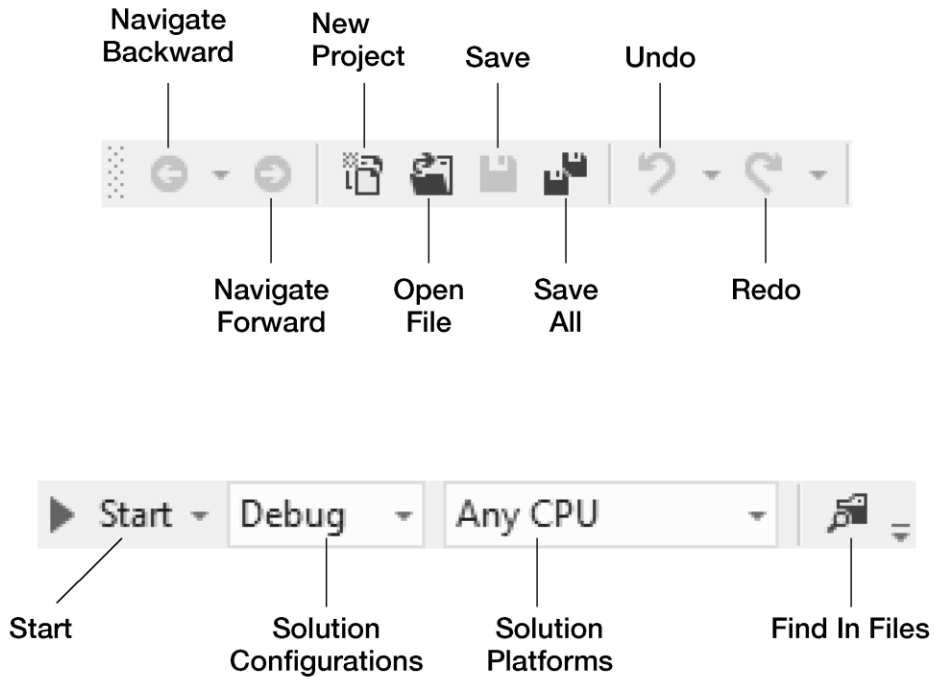


Fig. 2.9 | Standard Visual Studio toolbar.

2.3 Menu Bar and Toolbar

- You can customize which toolbars are displayed by selecting VIEW > Toolbars then selecting a toolbar from the list in Fig. 2.10.
- Each toolbar you select is displayed with the other toolbars at the top of the Visual Studio window.
- You move a toolbar by dragging its handle at the left side of the toolbar.
- To execute a command via the toolbar, click its icon.

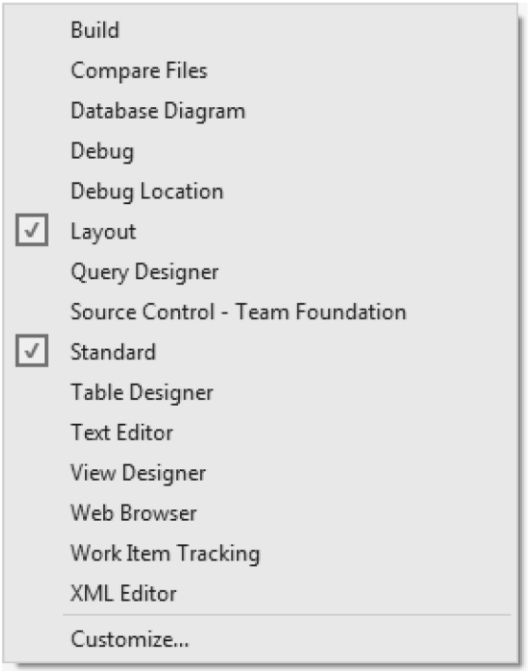


Fig. 2.10 | List of toolbars that can be added to the top of the IDE.

2.3 Menu Bar and Toolbar

- It can be difficult to remember what each toolbar icon represents.
- Hovering the mouse pointer over an icon highlights it and, after a brief pause, displays a description of the icon called a tool tip .
- Tool tips help you become familiar with the IDE's features and serve as useful reminders for each toolbar icon's functionality.

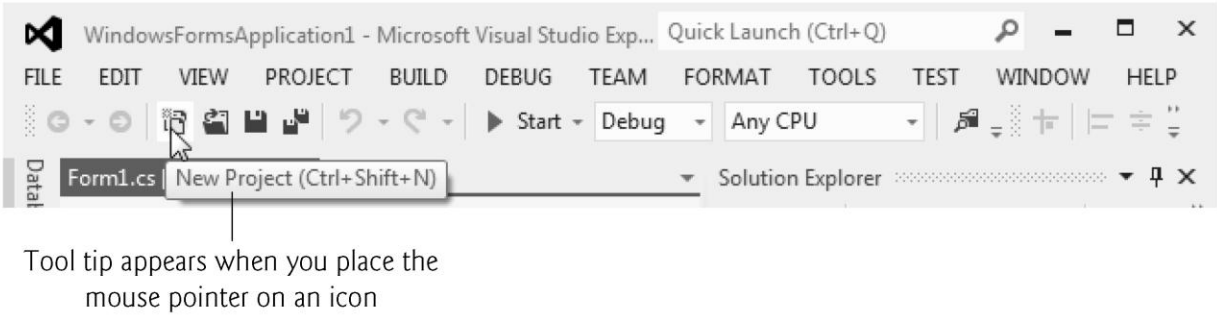


Fig. 2.11 | Tool tip demonstration.

2.4 Navigating the Visual Studio IDE

- The IDE provides windows for accessing project files and customizing controls.
- This section introduces several windows that you'll use frequently when developing Visual C# apps.
- Each of the IDE's windows can be accessed by selecting its name in the VIEW menu.

2.4 Navigating the Visual Studio IDE

Auto-Hide

- Visual Studio provides a space-saving feature called auto-hide.
- When auto-hide is enabled for a window, a tab containing the window's name appears along either the left, right or bottom edge of the IDE window (Fig. 2.12).
- Clicking the name of an auto-hidden window displays that window (Fig. 2.13).
- Clicking the name again (or clicking outside) hides the window.
- To "pin down" a window (that is, to disable auto-hide and keep the window open), click the pin icon.
- When auto-hide is enabled, the pin icon is horizontal—when a window is "pinned down," the pin icon is vertical.

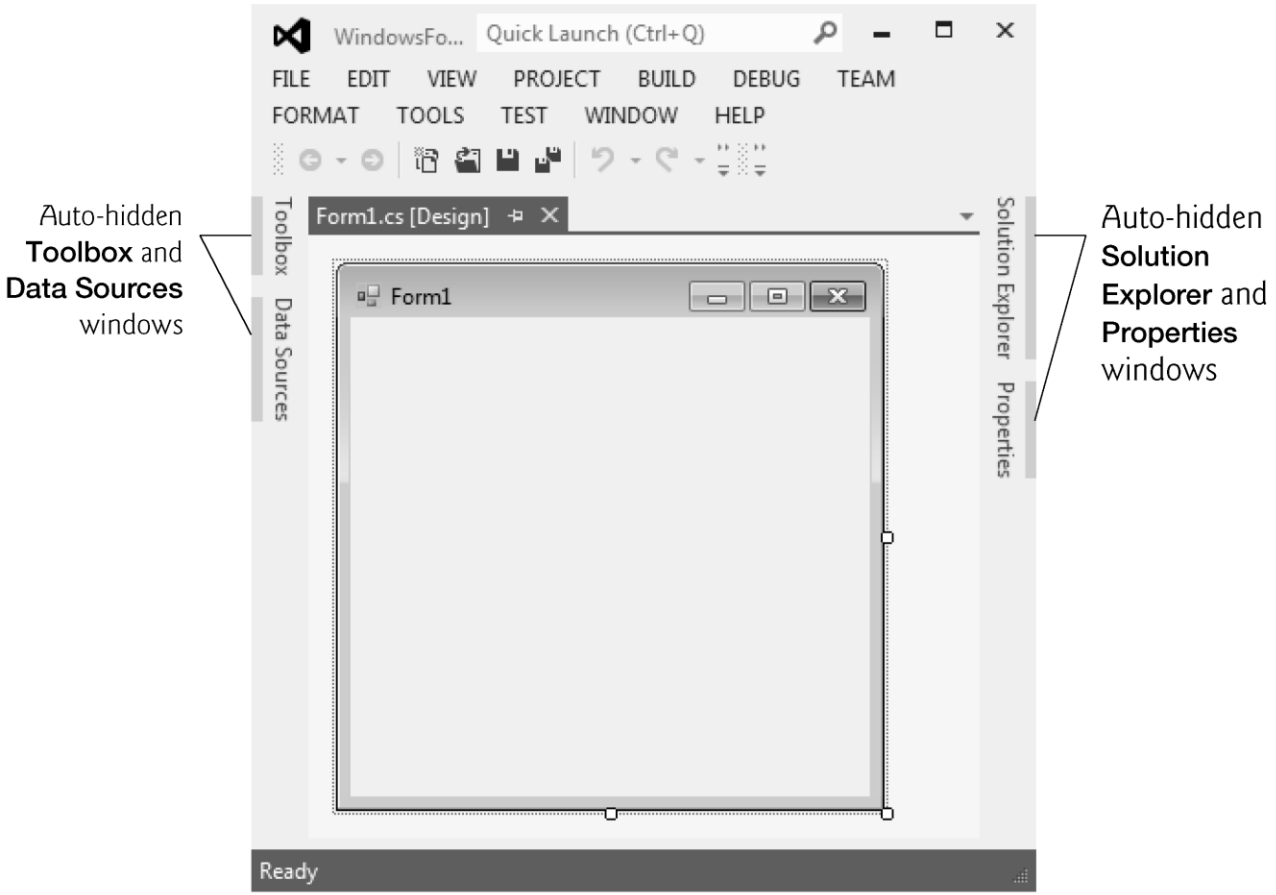


Fig. 2.12 | Auto-hide feature demonstration.

2.4 Navigating the Visual Studio IDE

- The next few sections cover three of Visual Studio's main windows—the Solution Explorer, the Properties window and the Toolbox.
- These windows display project information and include tools that help you build your apps.

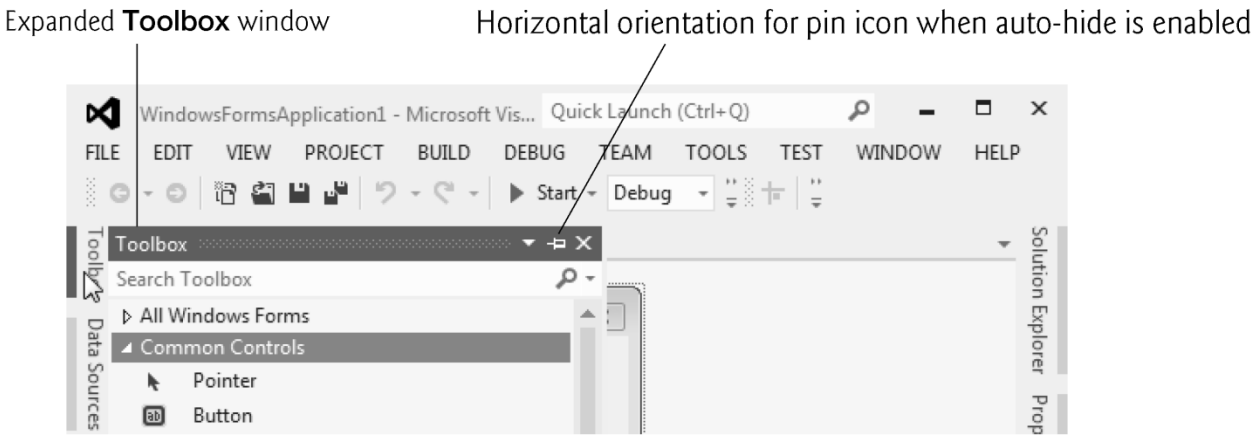


Fig. 2.13 | Displaying the hidden **Toolbox** window when auto-hide is enabled.

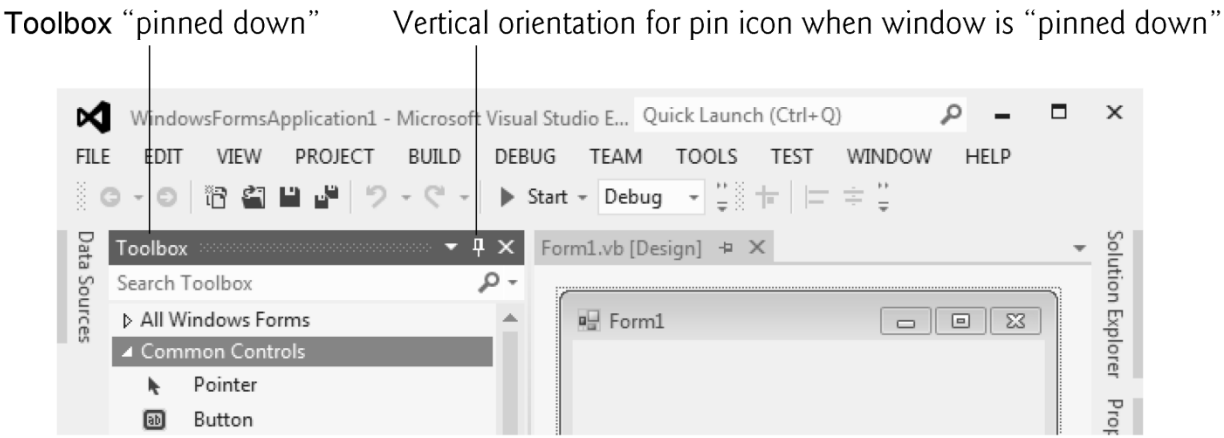


Fig. 2.14 | Disabling auto-hide—"pinning down" a window.

2.4.1 Solution Explorer

- The Solution Explorer window (Fig. 2.15) provides access to all of a solution's files.
- If it's not shown in the IDE, select VIEW > Solution Explorer.
- When you open a new or existing solution, the Solution Explorer displays the solution's contents.

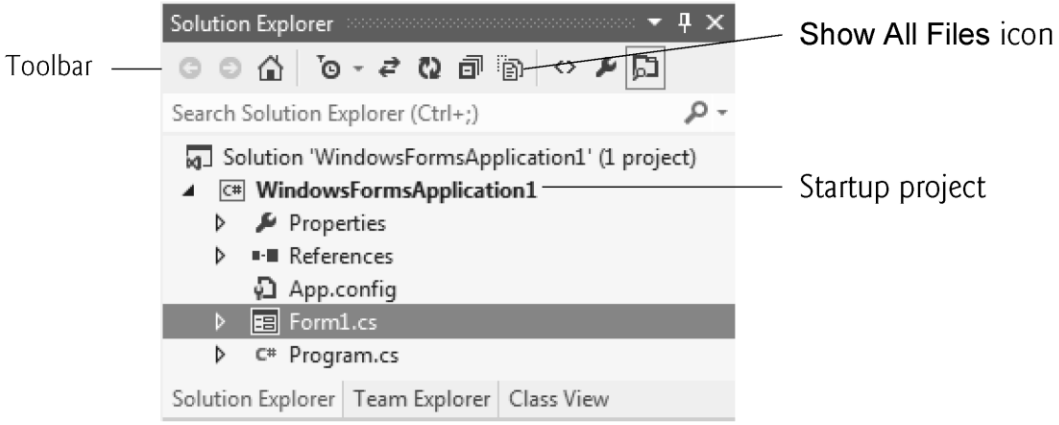


Fig. 2.15 | **Solution Explorer** window with an open project.

2.4.1 Solution Explorer

- The solution's startup project is the one that runs when you select DEBUG > Start Debugging (or press the *F5 key)*.
- For a single-project solution like the examples in this book, the startup project is the only project (in this case, WindowsFormsApplication1).
- The startup project's name appears in bold text in the Solution Explorer window.

2.4.1 Solution Explorer

- When you create an app for the first time, the Solution Explorer window appears as shown in Fig. 2.15.
- The Visual C# file that corresponds to the `Form` shown in Fig. 2.4 is named `Form1.cs` (selected in Fig. 2.15).
- Visual C# files use the `.cs` file-name extension, which is short for "Visual C#."

2.4.1 Solution Explorer

- By default, the IDE displays only files that you may need to edit—other files that the IDE generates are hidden.
- The Solution Explorer window includes a toolbar that contains several icons.
- Clicking the Show All Files icon (Fig. 2.15) displays all the solution's files, including those generated by the IDE.
- Clicking the arrows to the left of a node *expands* or *collapses* that node.

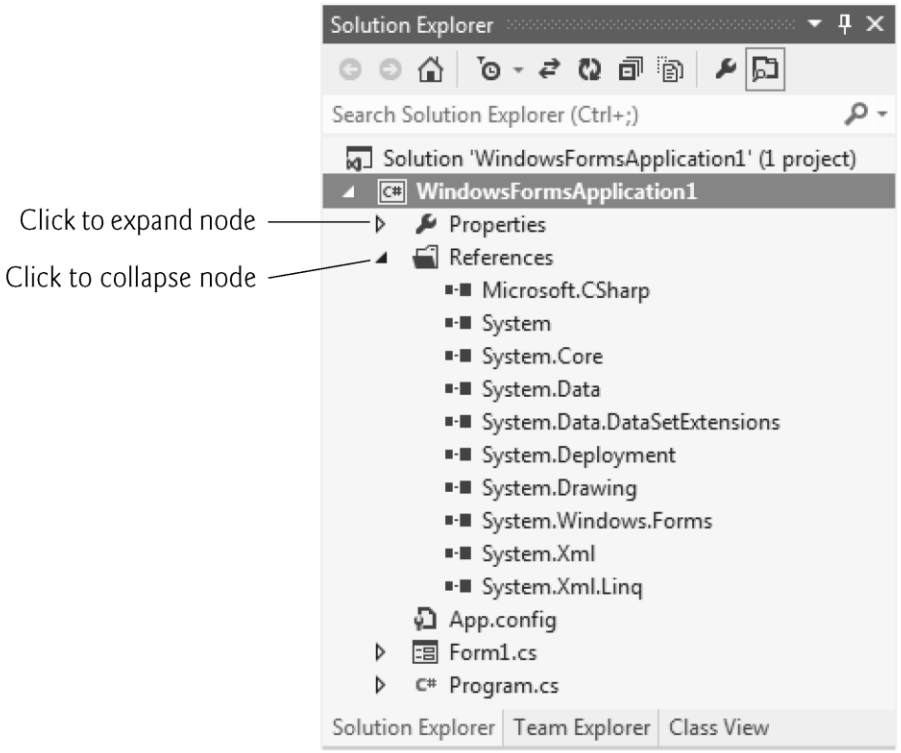


Fig. 2.16 | **Solution Explorer** with the **References** node expanded.

2.4.1 Solution Explorer

- Try clicking the arrow to the left of References to display items grouped under that heading (Fig. 2.16).
- Click the arrow again to collapse the tree.
- Other Visual Studio windows also use this convention.

2.4.1 Solution Explorer

- You can right-click on a project or solution name in Solution Explorer for an option to open the directory in Window's File Explorer
- You will see that the file structure is identical (or nearly identical) to the one in VS

2.4.1 Solution Explorer

- It is important to understand the structure of these directories
- The solution is in a directory, and that directory has a different directory for every project within the solution
- The solution directory also contains the .sln file which has the information about the structure of the solution
- In our class we will probably not add more than one project to any solution. However, it is important to check the entire solution into GitHub, so check in the full **solution** directory, not just the project directory.
- The bin and obj directories are for binary files and can be ignored by git

2.4.2 Toolbox

- To display the Toolbox window, select VIEW > Toolbox.
- The Toolbox contains the controls used to customize `Form`s (Fig. 2.17).
- With visual app development, you can "drag and drop" controls onto the `Form` and the IDE will write the code that creates the controls for you.
- Just as you don't need to know how to build an engine to drive a car, you don't need to know how to build controls to use them.
- *Reusing* preexisting controls saves time and money when you develop apps.
- You'll use the Toolbox when you create your first app later in the chapter.

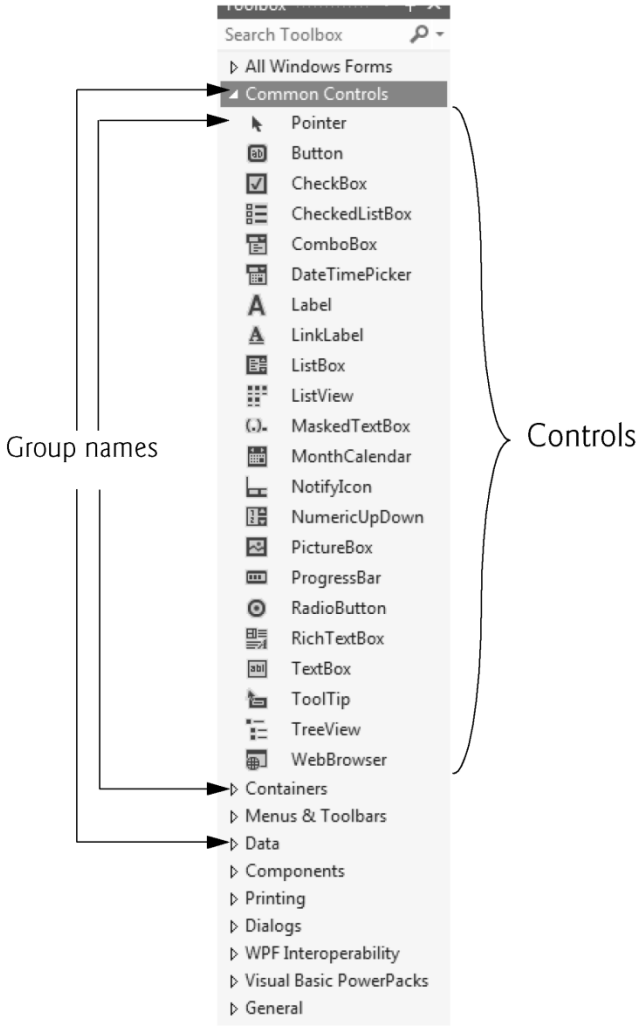


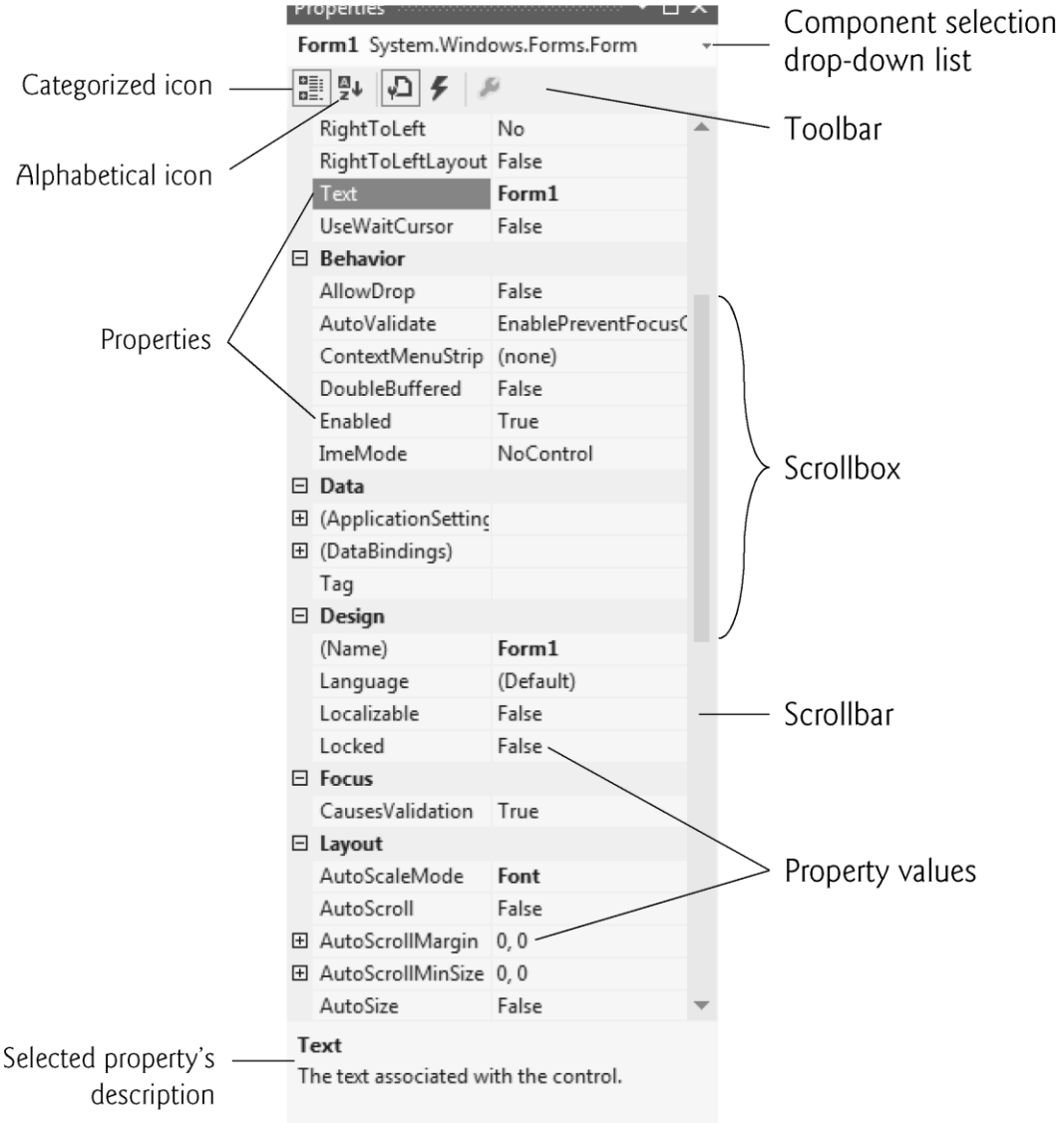
Fig. 2.17 | **Toolbox** window displaying the **Common Controls** group.

2.4.2 Toolbox

- The Toolbox groups the prebuilt controls into categories—All Windows Forms, Common Controls, Containers, Menus & Toolbars, Data, Components, Printing, Dialogs, WPF Interoperability, Visual C# PowerPacks and General are listed in Fig. 2.17.
- Again, note the use of arrows, which can expand or collapse a group of controls.

2.4.3 Properties Window

- If the Properties window is not displayed below the Solution Explorer, select VIEW > Properties Window to display it.
- The Properties window contains the properties for the currently selected `Form`, control or file in the IDE.
- Properties specify information about the `Form` or control, such as its size, color and position.
- Each `Form` or control has its own set of properties—a property's description is displayed at the bottom of the Properties window whenever that property is selected.



2.4.3 Properties Window

- Fig. 2.18 shows `Form1`'s Properties window.
- The left column lists the `Form`'s properties—the right column displays the current value of each property.
- You can sort the properties either *alphabetically* (by clicking the Alphabetical icon) or *categorically* (by clicking the Categorized icon).
- Depending on the size of the Properties window, some of the properties may be hidden from view on the screen.

2.4.3 Properties Window

- You can scroll through the list of properties by dragging the scrollbox up or down inside the scrollbar, or by clicking the arrows at the top and bottom of the scrollbar.
- We show how to set individual properties later in this chapter.
- The Properties window is crucial to visual app development—it allows you to modify a control's properties visually, without writing code.
- You can see which properties are available for modification and, in many cases, can learn the range of acceptable values for a given property.

2.4.3 Properties Window

- The Properties window displays a brief description of the selected property, helping you understand its purpose.
- A property can be set quickly using this window, and no code needs to be written.
- At the top of the Properties window is the component selection drop-down list, which allows you to select the `Form` or control whose properties you wish to display in the Properties window.
- Using the component selection drop-down list is an alternative way to display a `Form`'s or control's properties without clicking the actual `Form` or control in the GUI.

2.5 Using Help

- ***Context-Sensitive Help***
  - Visual Studio provides context-sensitive help pertaining to the "current content" (that is, the items around the location of the mouse cursor).
  - To use context-sensitive help, click an item, then press the *F1* key.
  - The help documentation is displayed in a web browser window.
  - To return to the IDE, either close the browser window or select the IDE's icon in your Windows task bar.
  - Figure 2.19 shows the help page for a `Form`'s `Text` property.
  - You can view this help by selecting the `Form`, clicking its `Text` property in the Properties window and pressing the *F1* key.

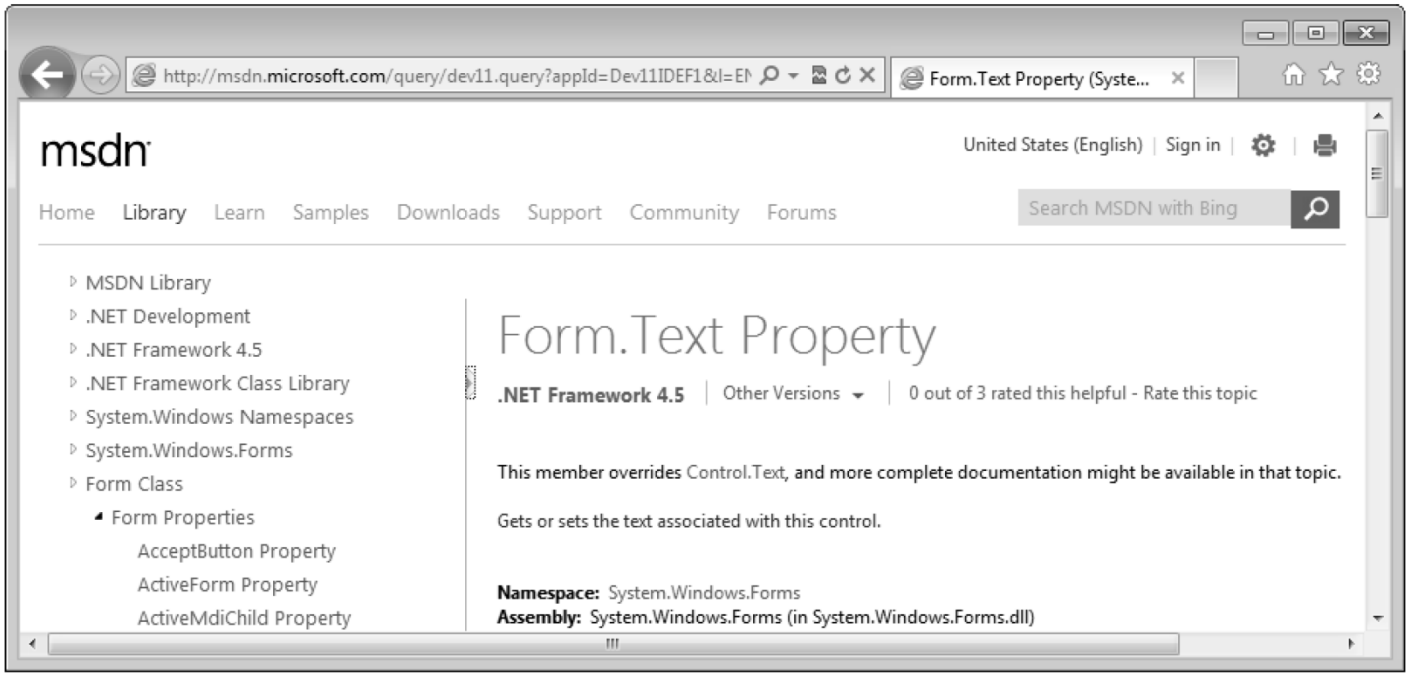


Fig. 2.19 | Using context-sensitive help.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- Next, we create an app that displays the text `"Welcome to Visual C#!"` and an image of the Deitel & Associates bug mascot.
- The app consists of a `Form` that uses a `Label` and a `PictureBox`.
- Figure 2.20 shows the final app executing.
- The app and the bug image are available with this chapter's examples.
- See the Before You Begin section following the Preface for download instructions.
- We assume the examples are located at `C:\examples` on your computer.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- You won't write a single line of code.
- Instead, you'll use visual app development techniques.
- Visual Studio processes your actions (such as mouse clicking, dragging and dropping) to generate app code.
- Chapter 3 begins our discussion of writing app code.
- Throughout the book, you produce increasingly substantial and powerful apps that usually include a combination of code written by you and code generated by Visual Studio.
- The generated code can be difficult for novices to understand—but you'll rarely need to look at it.

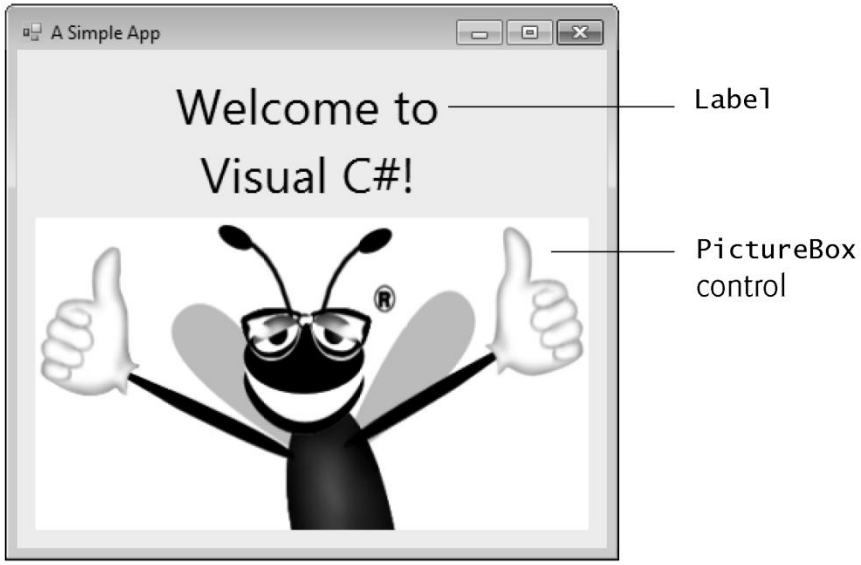


Fig. 2.20 | Simple app executing.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- Visual app development is useful for building GUI-intensive apps that require user interaction.
- To create, save, run and terminate this first app, perform the following steps:

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Closing the open project***. If the project you were working on earlier in this chapter is still open, close it by selecting FILE > Close Solution. A dialog asking whether to save the current project might appear. Click Save to save your changes or Discard to ignore them.
- ***Creating the new project.*** To create a new Windows Forms app, select FILE > New Project… to display the New Project dialog (Fig. 2.21). Select Windows Forms Application. Name the project ASimpleApp, specify the Location where you want to save it (we used the default location) and click OK.

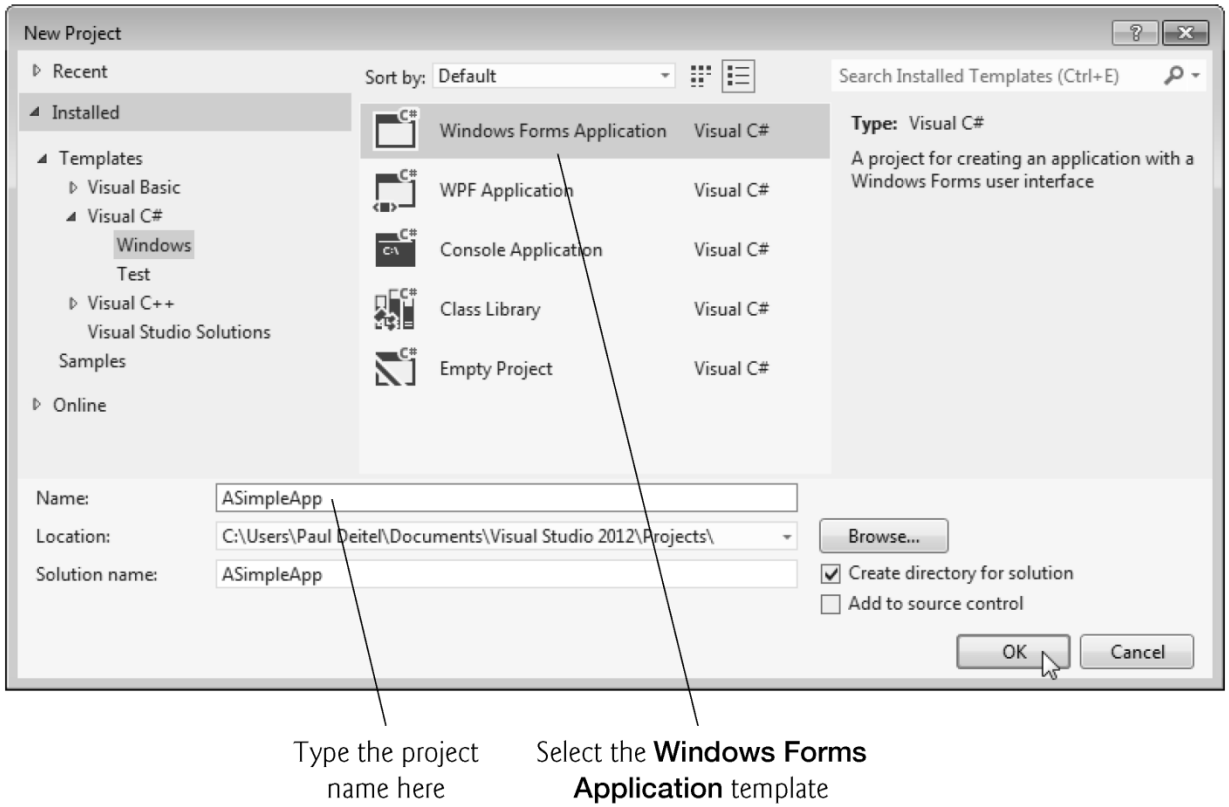


Fig. 2.21 | **New Project** dialog.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Setting the text in the Form's title bar.*** The text in the `Form`'s title bar is determined by the `Form`'s Text property (Fig. 2.22).
- If the Properties window is not open, click the properties icon in the toolbar or select VIEW > Properties Window.
- Click anywhere in the `Form` to display the `Form`'s properties in the Properties window.
- In the textbox to the right of the `Text` property, type `"A Simple App"`, as in Fig. 2.22. Press the *Enter* key—the `Form`'s title bar is updated immediately.

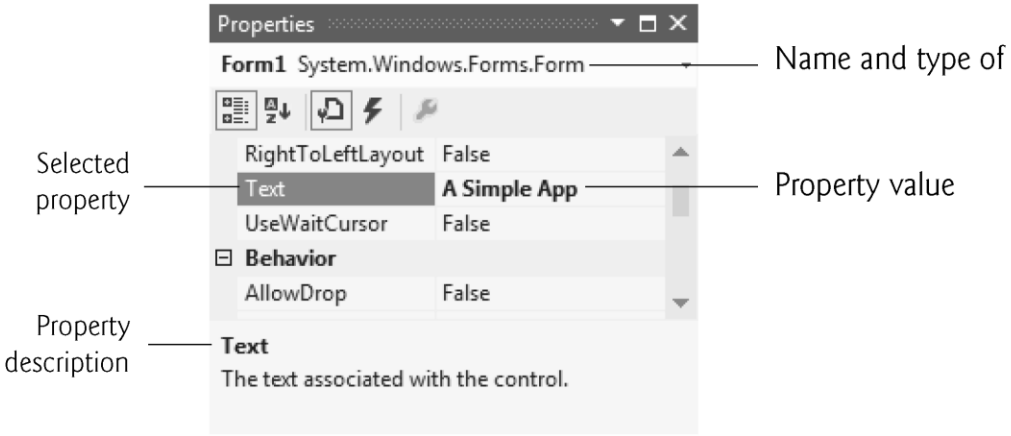


Fig. 2.22 | Setting the `Form`'s `Text` property in the Properties window.

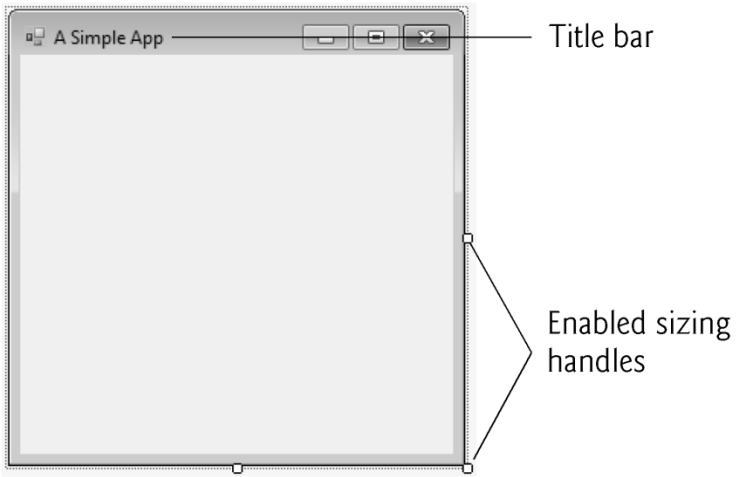


Fig. 2.23 | `Form` with enabled sizing handles.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Resizing the Form.*** Click and drag one of the `Form`'s enabled sizing handles (the small white squares that appear around the `Form`).
- Using the mouse, select the bottom-right sizing handle and drag it down and to the right to make the `Form` larger.

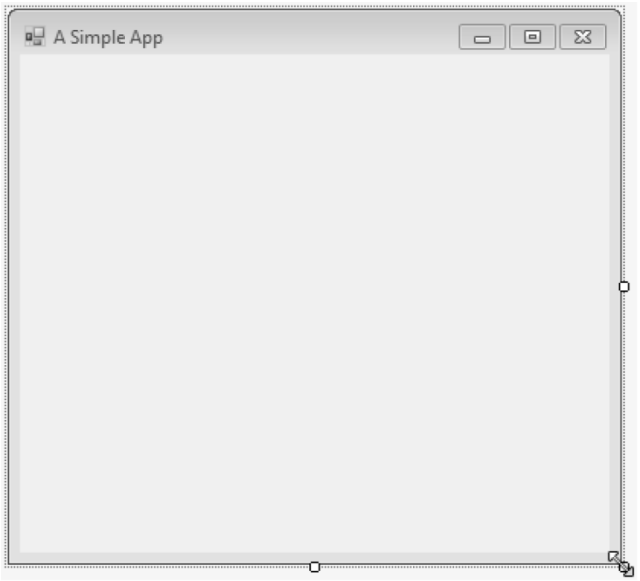


Fig. 2.24 | Resized Form.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Changing the `Form`'s background color.*** The BackColor property specifies a `Form`'s or control's background color. Clicking `BackColor` in the Properties window causes a down-arrow button to appear next to the value of the property. When clicked, the down-arrow button displays other options, which vary depending on the property. In this case, the arrow displays tabs for Custom, Web and System (the default). Click the Custom tab to display the palette (a grid of colors). Select the box that represents light blue. Once you select the color, the palette closes and the `Form`'s background color changes to light blue.

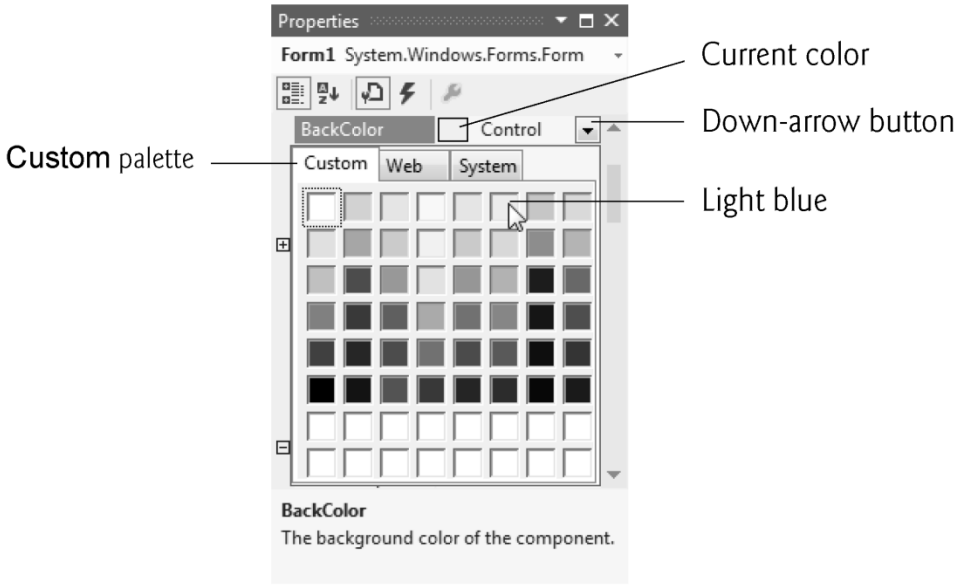


Fig. 2.25 | Changing the `Form`'s `BackColor` property.

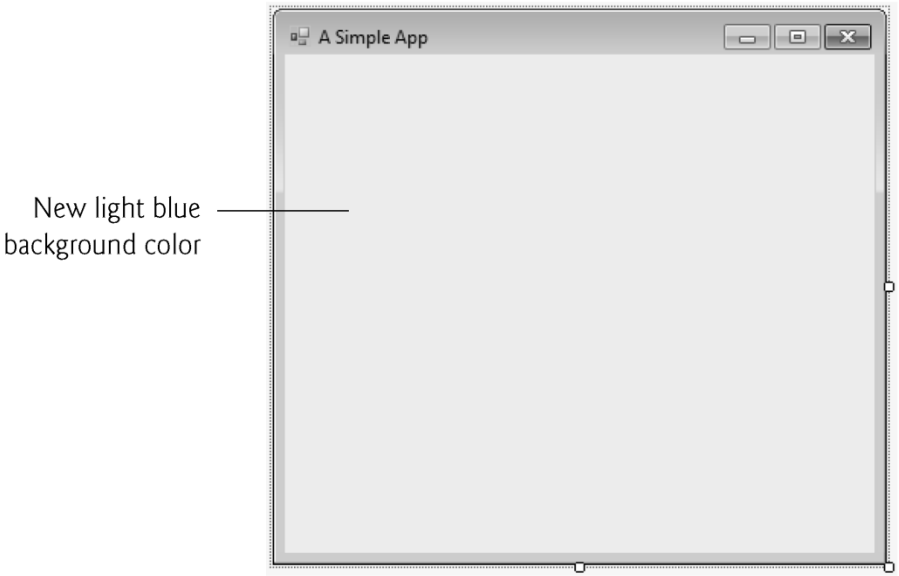


Fig. 2.26 | Form with new BackColor property applied.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Adding a Label control to the Form.*** If the Toolbox is not already open, select VIEW > Toolbox to display the set of controls you'll use for creating your apps. For the type of app we're creating in this chapter, the typical controls we use are located in either the All Windows Forms group of the Toolbox or the Common Controls group. If either group name is collapsed, expand it by clicking the arrow to the left of the group name.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- Next, double click the `Label` control in the Toolbox. This action causes a `Label` to appear in the upper-left corner of the `Form`.
  - [*Note*: If the `Form` is behind the Toolbox, you may need to hide the Toolbox to see the `Label`.]
- Although double clicking any Toolbox control places the control on the `Form`, you also can "drag" controls from the Toolbox to the `Form`—you may prefer dragging the control because you can position it wherever you want.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- The `Label` displays the text `label1` by default. When you add a `Label` to the `Form`, the IDE sets the `Label`'s `BackColor` property to the `Form`'s `BackColor`. You can change the `Label`'s background color by changing its `BackColor` property.

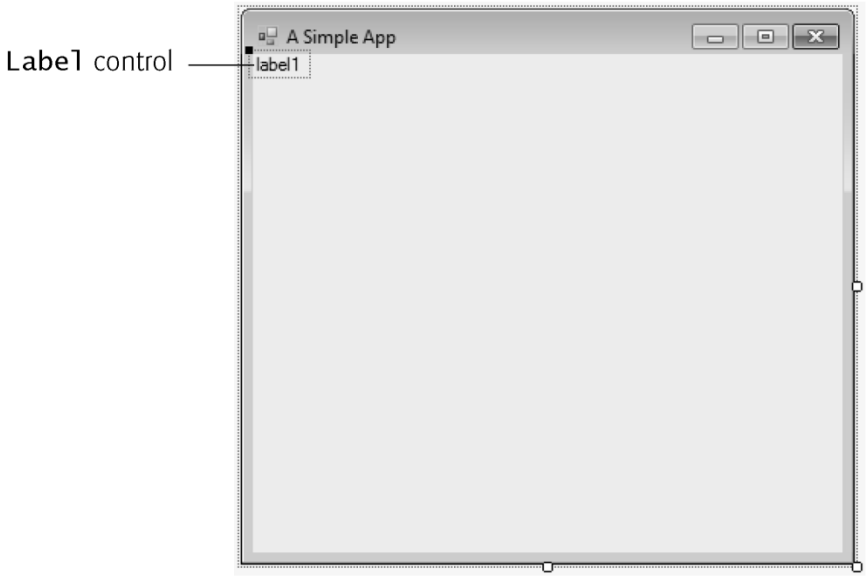


Fig. 2.27 | Adding a `Label` to the `Form`.

2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- ***Customizing the Label's appearance.*** Select the `Label` by clicking it. Its properties now appear in the Properties window. The `Label`'s `Text` property determines the text (if any) that the `Label` displays. The `Form` and `Label` each have their own `Text` property—`Form`s and controls can have the *same* property names (such as `BackColor`, `Text`, etc.) without conflict. Set the `Label`'s `Text` property to `Welcome to Visual C#!`. The `Label` resizes to fit all the typed text on one line.

2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- By default, the AutoSize property of the `Label` is set to `True`, which allows the `Label` to update its size to fit all of the text if necessary. Set the `AutoSize` property to `False` so that you can resize the `Label` on your own. Resize the `Label` (using the sizing handles) so that the text fits. Move the `Label` to the top center of the `Form` by dragging it or by using the keyboard's left and right arrow keys to adjust its position. Alternatively, when the `Label` is selected, you can center the `Label` control horizontally by selecting FORMAT > Center In Form > Horizontally.

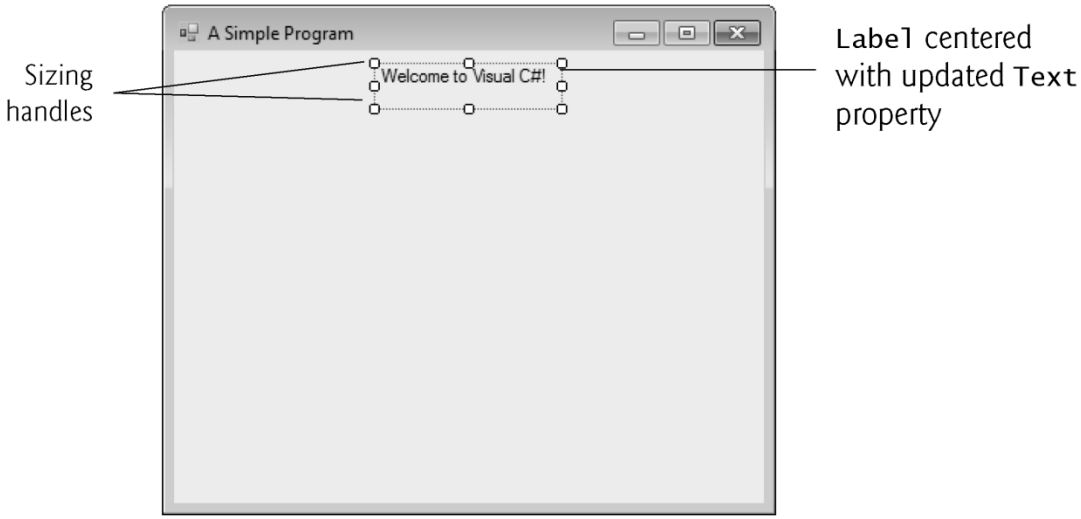


Fig. 2.28 | GUI after the `Form` and `Label` have been customized.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Setting the Label's font size.*** To change the font type and appearance of the `Label`'s text, select the value of the Font property, which causes an ellipsis button to appear next to the value. When the ellipsis button is clicked, a dialog that provides additional values—in this case, the Font dialog—is displayed. You can select the font name (the font options may be different, depending on your system-), font style (Regular, Italic, Bold, etc.) and font size (16, 18, 20, etc.) in this dialog.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- The Sample text shows the elected font settings. Under Font, select Segoe UI, Miscrosoft's recommended font for user interfaces. Under Size, select 24 points and click OK. If the `Label`'s text does not fit on a single line, it wraps to the next line. Resize the `Label` so that the words "`welcome to`" appear on the Label's first line and the words "`Visual C#!`" appear on the second line. Re-center the `Label` horizontally.

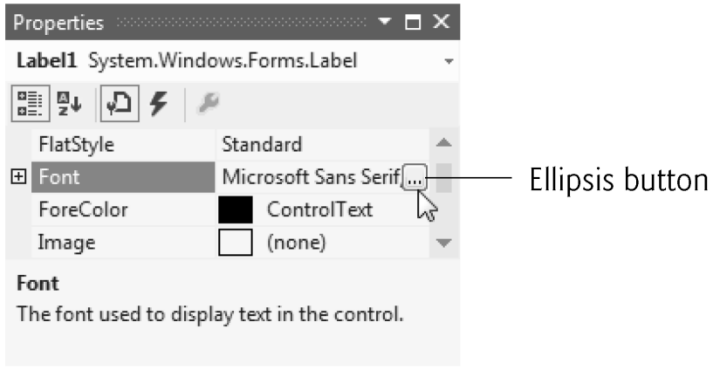


Fig. 2.29 | **Properties** window displaying the `Label`'s `Font` property.

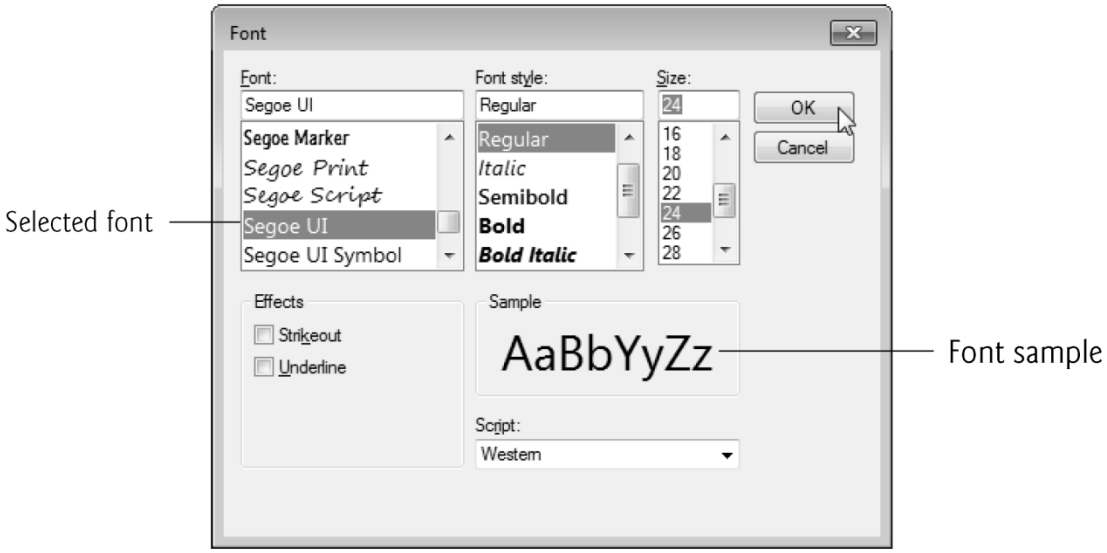


Fig. 2.30 | **Font** dialog for selecting fonts, styles and sizes.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Aligning the Label's text.*** Select the `Label`'s TextAlign property, which determines how the text is aligned within the `Label`. A three-by-three grid of buttons representing alignment choices is displayed. The position of each button corresponds to where the text appears in the `Label`.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- For this app, set the `TextAlign` property to `MiddleCenter` in the three-by-three grid—this selection centers the text horizontally and vertically within the `Label`. The other `TextAlign` values, such as `TopLeft`, `TopRight`, and `BottomCenter`, can be used to position the text anywhere within a `Label`. Certain alignment values may require that you resize the `Label` to fit the text better.

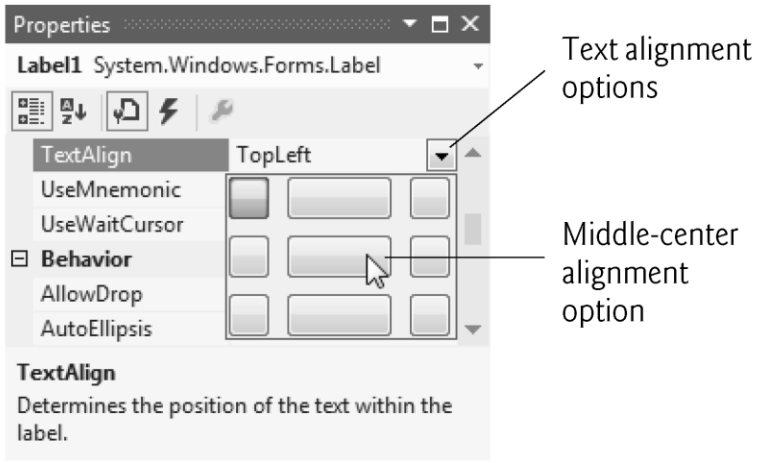


Fig. 2.31 | Centering the Label's text.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Adding a PictureBox to the Form.*** The `PictureBox` control displays images. The process involved in this step is similar to that of Step 6, in which we added a `Label` to the `Form`. Locate the `PictureBox` in the Toolbox and double click it to add it to the `Form`. When the `PictureBox` appears, move it underneath the `Label`, either by dragging it or by using the arrow keys.

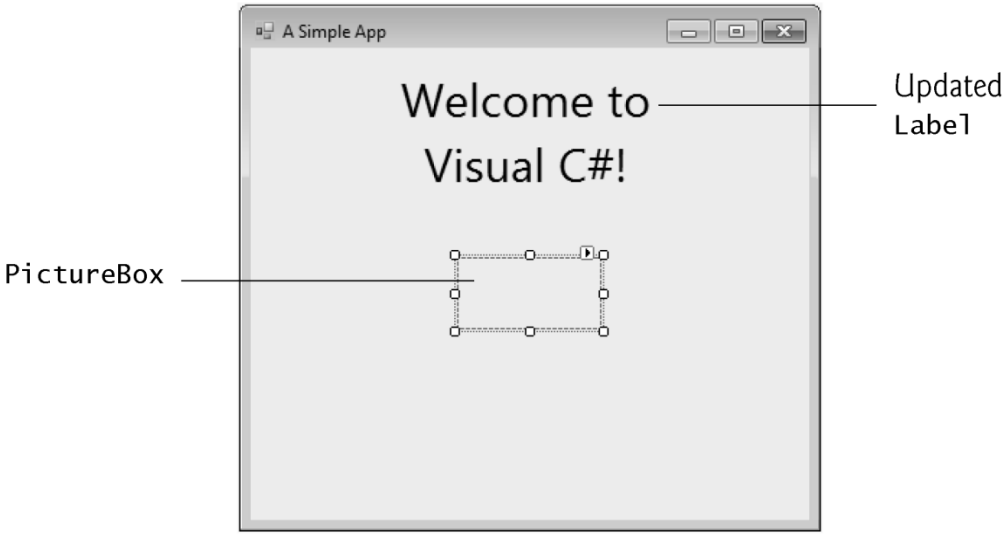


Fig. 2.32 | Inserting and aligning a `PictureBox`.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Inserting an image***. Click the `PictureBox` to display its properties in the Properties window. Locate and select the Image property, which displays a preview of the selected image or (none) if no image is selected. Click the ellipsis button (or the Choose Image… link above the property description) to display the Select Resource dialog, which is used to import files, such as images, for use in an app. Click the Import… button to browse for an image to insert, select the image file and click OK.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- We used `bug.png` from this chapter's examples folder. The image is previewed in the Select Resource dialog. Click OK to use the image. Supported image formats include PNG (Portable Network Graphics), GIF (Graphic Interchange Format), JPEG (Joint Photographic Experts Group) and BMP (Windows bitmap). To scale the image to the `PictureBox`'s size, change the SizeMode property to StretchImage. Resize the `PictureBox`, making it larger.

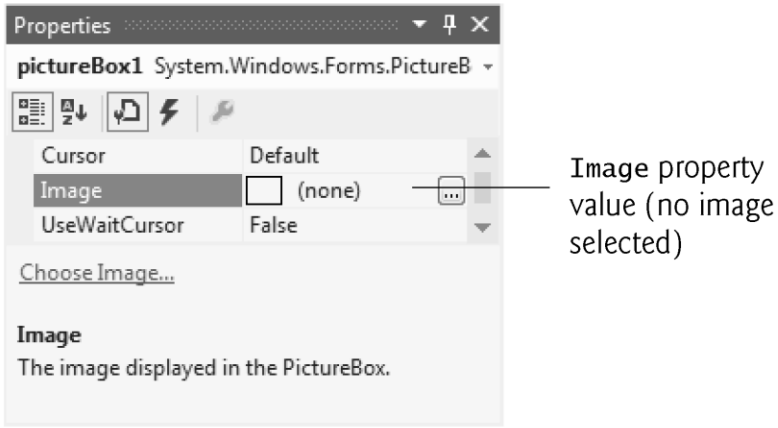


Fig. 2.33 | `Image` property of the `PictureBox`.

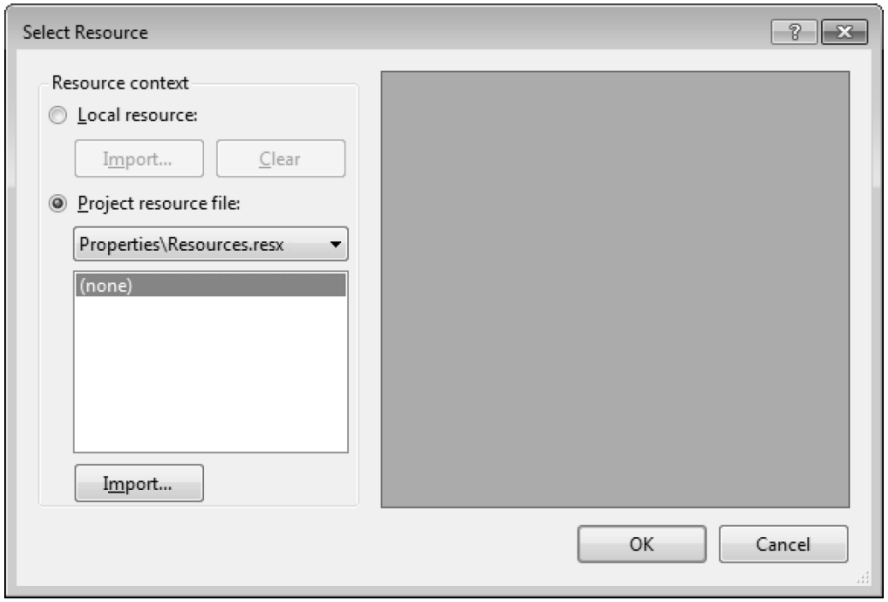


Fig. 2.34 | **Select Resource** dialog to select an image for the `PictureBox`.

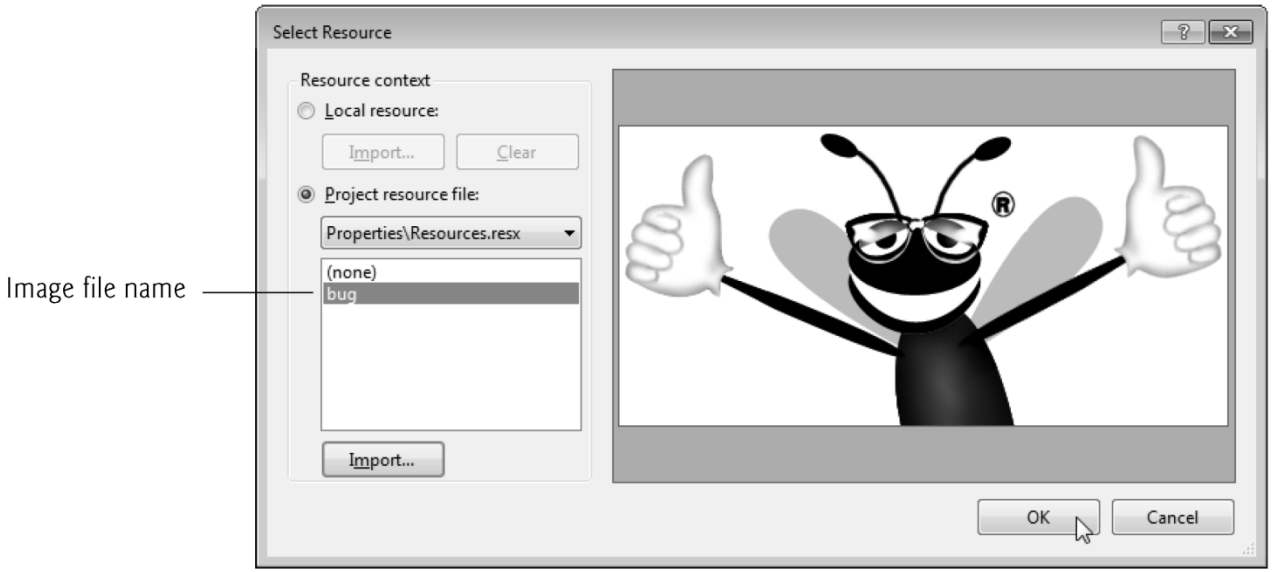


Fig. 2.35 | **Select Resource** dialog displaying a preview of selected image.

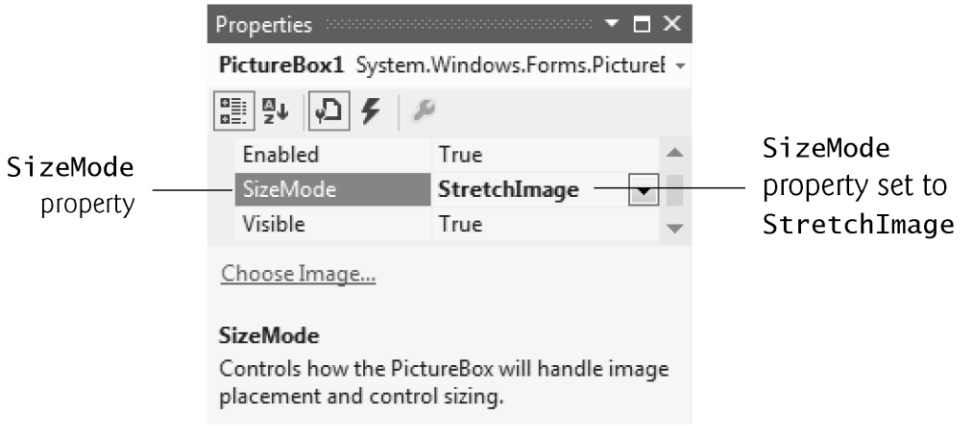


Fig. 2.36 | Scaling an image to the size of the `PictureBox`.

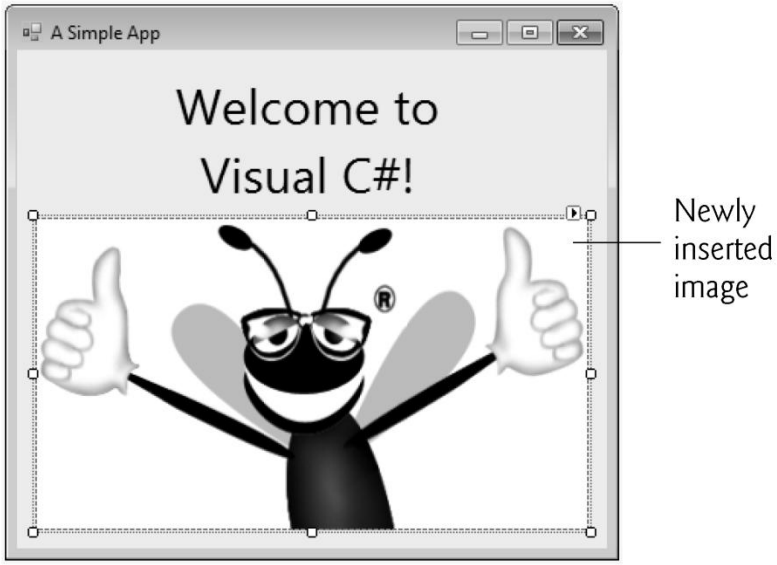


Fig. 2.37 | `PictureBox` displaying an image.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Saving the project.*** Select FILE > Save All to save the entire solution. The solution file (which has the filename extension `.sln`) contains the name and location of its project, and the project file (which has the filename extension `.csproj`) contains the names and locations of all the files in the project. If you want to reopen your project at a later time, simply open its `.sln` file.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Running the project.*** Recall that up to this point we have been working in the IDE design mode (that is, the app being created is not executing). In run mode, the app is executing, and you can interact with only a few IDE features—features that are not available are disabled (grayed out). The text `Form1.cs [Design]` in the project tab means that we're designing the `Form` *visually* rather than *programmatically*. If we had been writing code, the tab would have contained in the tab, the file has been changed and should be saved.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- Select DEBUG > Start Debugging- to execute the app (or you can press the *F5* key). shows the IDE in run mode (indicated by the title-bar text ASimpleApp (Running) – Microsoft Visual Studio Express 2012 for Windows Desktop). Many toolbar icons and menus are disabled, since they cannot be used while the app is running.

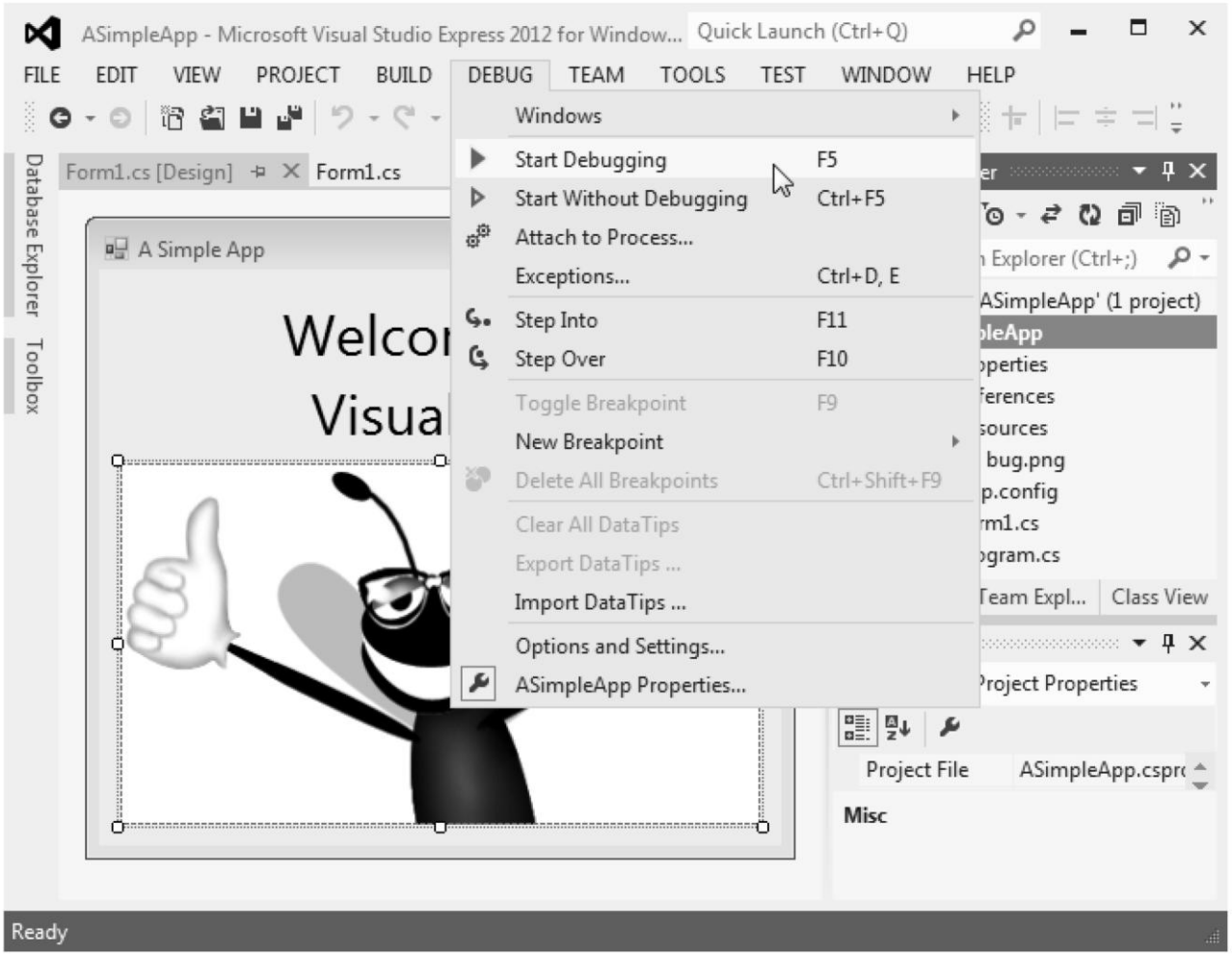


Fig. 2.38 | Debugging a solution.

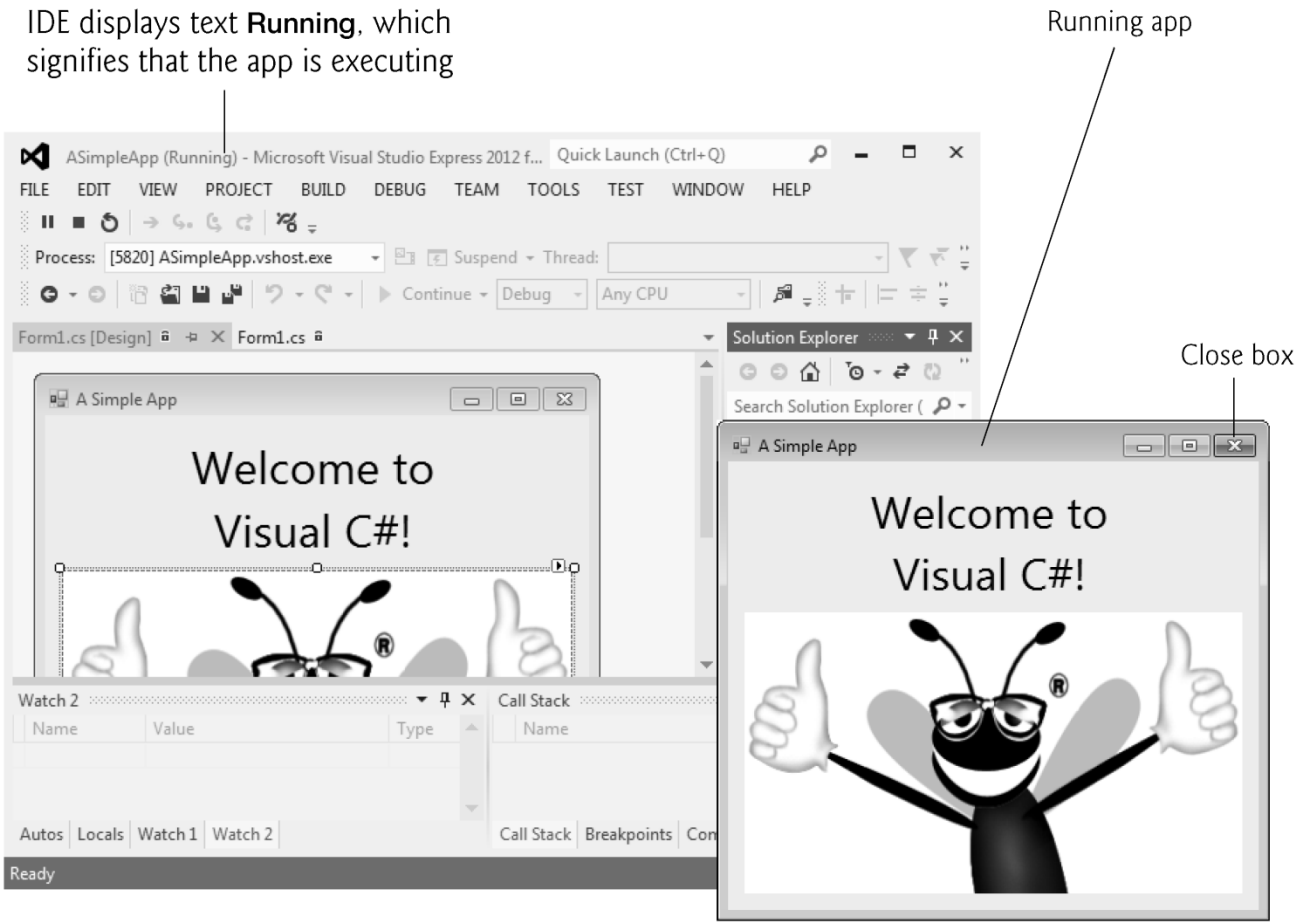


Fig. 2.39 | IDE in run mode, with the running app in the foreground.

2.6 Using Visual App Development to Create a Simple App that Displays Text and an Image

- ***Terminating execution.*** Click the running app's close box (the X in the top-right corner of the running app's window). This action stops the app's execution and returns the IDE to design mode. You can also select DEBUG > Stop Debugging to terminate the app.