Part I

Integrated Development Environment

1

A Quick Tour

Ever since we have been developing software, there has been a need for tools to help us write, compile, and debug our applications. Microsoft Visual Studio 2008 is the next iteration in the continual evolution of a best-of-breed integrated development environment (IDE). If this is your first time using Visual Studio, then you will find this chapter a useful starting point. Even if you have worked with a previous version of Visual Studio, you may want to quickly skim it.

This chapter introduces the Visual Studio 2008 user experience and will show you how to work with the various menus, toolbars, and windows. It serves as a quick tour of the IDE, and as such it won't go into detail about what settings can be changed or how to go about customizing the layout, as these topics will be explored in the following chapters.

Let's Get Started

Each time you launch Visual Studio you will notice the Microsoft Visual Studio 2008 splash screen appear. Like a lot of splash screens, it provides information about the version of the product and to whom it has been licensed, as shown in Figure 1-1.

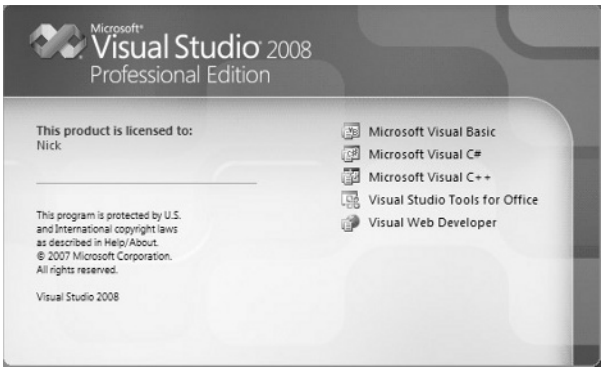

Figure 1-1

More importantly, the Visual Studio splash screen includes a list of the main components that have been installed. If you install third-party add-ins, you may see those products appear in this list.

The first time you run Visual Studio 2008, you will see the splash screen only for a short period before you are prompted to select the default environment settings. It may seem unusual to ask those who haven't used a product before how they imagine themselves using it. As Microsoft has consolidated a number of languages and technologies into a single IDE, that IDE must account for the subtle (and sometimes not so subtle) differences in the way developers work.

If you take a moment to review the various options in this list, as shown in Figure 1-2, you'll find that the environment settings that will be affected include the position and visibility of various windows, menus, and toolbars, and even keyboard shortcuts. For example, if you select the General Development Settings option as your default preference, this screen describes the changes that will be applied.

Figure 1-2

A tip for Visual Basic .NET developers coming from previous versions of Visual Studio is that they *should NOT use the Visual Basic Development Settings option. This option has been configured for VB6 developers and will only infuriate Visual Basic .NET developers, as they will be used to different shortcut key mappings. We recommend that* you *use the general development settings, as these will use the standard keyboard mappings without being geared toward another development language.*

The Visual Studio IDE

Depending on which set of environment settings you select, when you click the Start Visual Studio button you will most likely see a dialog indicating that Visual Studio is configuring the development environment. When this process is complete, Visual Studio 2008 will open, ready for you to start work, as shown in Figure 1-3.

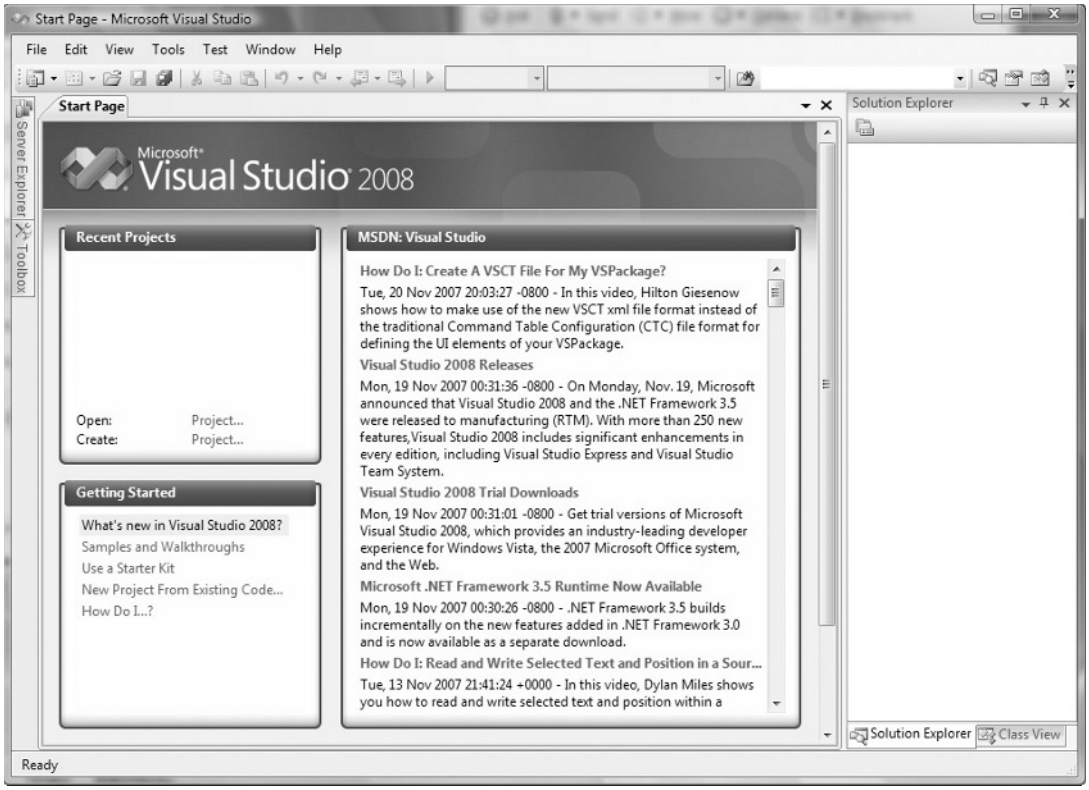

Figure 1-3

Regardless of the environment settings you selected, you will see the Start Page in the center of the screen. However, the contents of the Start Page and the surrounding toolbars and tool windows can vary. At this stage it is important to remember that your selection only determined the default settings, and that over time you can configure Visual Studio to suit your working style.

> *The contents shown in the right-hand portion of the Start Page are actually just the contents of an RSS feed. You can change this to be your favorite blog, or even a news feed (so you can catch up on the latest news while your solution is loading), by changing the news channel property on the Environment ⇨ Startup node in the Options dialog, accessible via the Options item on the Tools menu.*

Before we launch into building our first application, it's important that we take a step back and look at the components that make up the Visual Studio 2008 IDE. Menus and toolbars are positioned along the top of the environment (as in most Windows applications), and a selection of sub-windows, or panes, appears on the left and right of the main window area. In the center is the main editor space: Whenever you open a code file, an XML document, a form, or some other file, it will appear in this space for editing. With each file you open, a new tab is created so that you can toggle among opened files.

On either side of the editor space is a set of tool windows: These areas provide additional contextual information and functionality. In the case of the general developer settings, the default layout includes the Solution Explorer and Class View on the right, and the Server Explorer and Toolbox on the left. The tool windows on the left are in their collapsed, or *unpinned,* state. If you click on a tool window's title, it will expand; it will collapse again when it no longer has focus or you move the cursor to another area of the screen. When a tool window is expanded you will see a series of three icons at the top right of the window, similar to those shown in the left image of Figure 1-4.

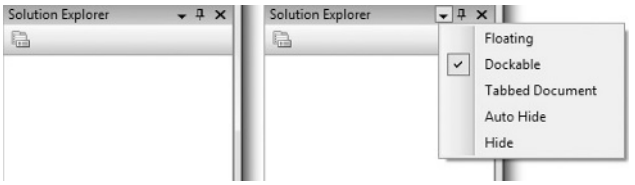

Figure 1-4

If you want the tool window to remain in its expanded, or *pinned,* state, you can click the middle icon, which looks like a pin. The pin will rotate 90 degrees to indicate that the window is now pinned. Clicking the third icon, the X, will close the window. If later you want to reopen this or another tool window, you can select it from the View menu.

> *Some tool windows are not accessible via the View menu, for example those having to do with debugging, such as threads and watch windows. In most cases these windows are available via an alternative menu item: in the case of the debugging windows it is the Debug menu.*

The right image in Figure 1-4 shows the context menu that appears when the first icon, the down arrow, is clicked. Each item in this list represents a different way of arranging the tool window. In the left image of Figure 1-5 the Solution Explorer is set as dockable, whereas in the right image the floating item has been selected. The latter option is particularly useful if you have multiple screens, as you can move the various tool windows onto the additional screen, allowing the editor space to use the maximum screen real estate. Selecting the Tabbed Document option will make the tool window into an additional tab in the editor space. In Chapter 4 you will learn how to effectively manage the workspace by docking and pinning tool windows.

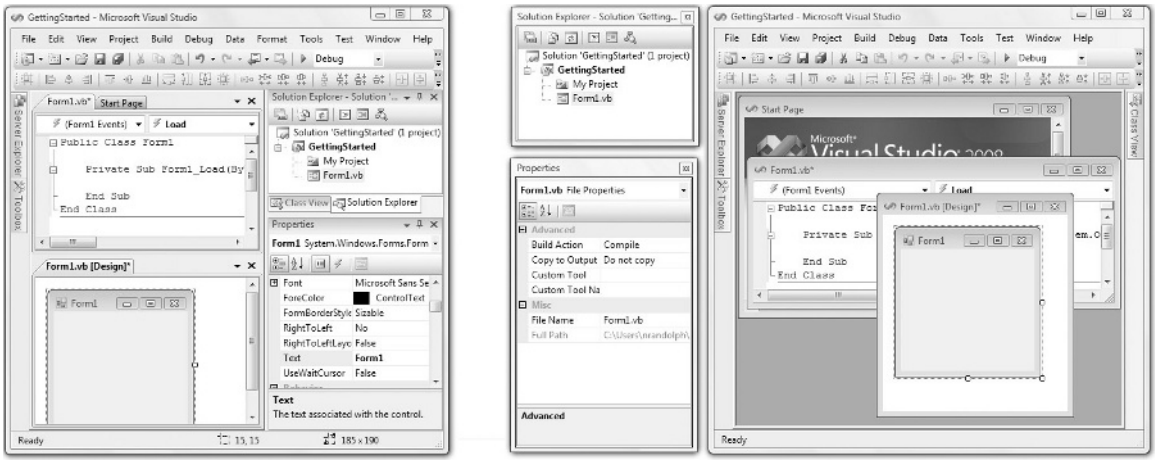

Figure 1-5

The other thing to note about the left image of Figure 1-5 is that the editor space has been divided into two horizontal regions. If you right-click an existing tab in the editor space, you can elect to move it to a new horizontal or vertical tab group. This can be particularly useful if you are working on multiple forms, or if you want to view the layout of a form while writing code in the code-behind file.

In the right image of Figure 1-5 the editor space is no longer rendered as a series of tabs. Instead, it is a series of child windows, in classic multiple-document-interface style. Unfortunately, this view is particularly limiting, because the child windows must remain within the bounds of the parent window, making it unusable across multiple monitors. To toggle between tabbed and multiple document window layouts, simply select the Environment ⇨ General node from the Options dialog.

Develop, Build, and Debug Your First Application

Now that you have seen an overview of the Visual Studio 2008 IDE, let's walk through creating a simple application that demonstrates working with some of these components. This is, of course, the mandatory "Hello World" sample that every developer needs to know, and it can be done in either Visual Basic .NET or C#, depending on what you feel more comfortable with.

1. Start by selecting File ⇨ New ⇨ Project. This will open the New Project dialog, as shown in Figure 1-6. A couple of new features are worth a mention here. Based on numerous feedback requests, this dialog is now resizable. More importantly, there is an additional drop-down box in the top right-hand corner, which is used to select the version of the .NET Framework that the application will target. The ability to use a single tool to create applications that target different framework versions means that developers can use fewer products and can take advantage of all the new features, even if they are maintaining an older product.

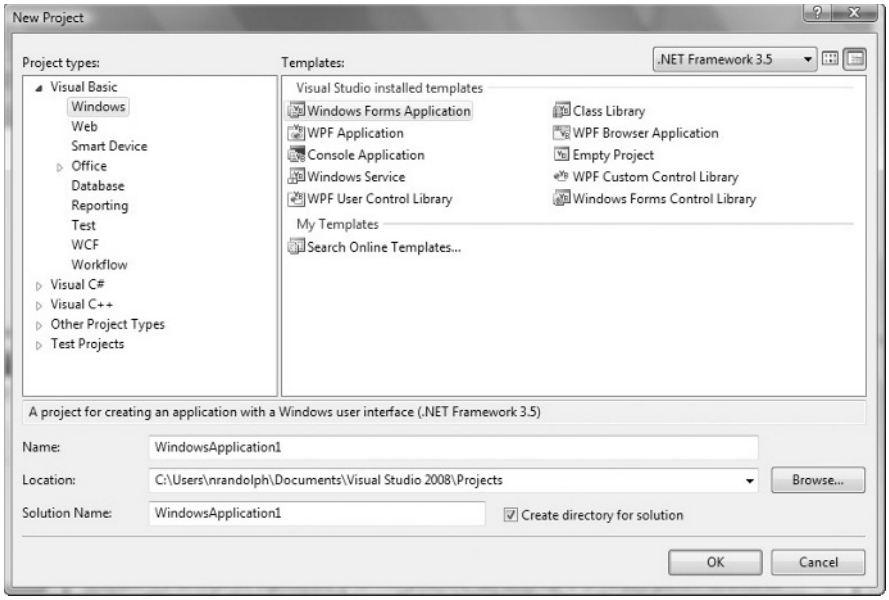

Figure 1-6

Select the Windows Forms Application from the Templates area (this item exists under the root Visual Basic and Visual C# nodes, or under the sub-node Windows) and set the Name to "GettingStarted," before selecting OK. This should create a new windows application project, which includes a single startup form and is contained within a "GettingStarted" solution, as shown in the Solution Explorer window of Figure 1-7. This startup form has automatically opened in the visual designer, giving you a graphical representation of what the form will look like when you run the application. You will notice that there is now an additional command bar visible and that the Properties tool window is in the right tool windows area.

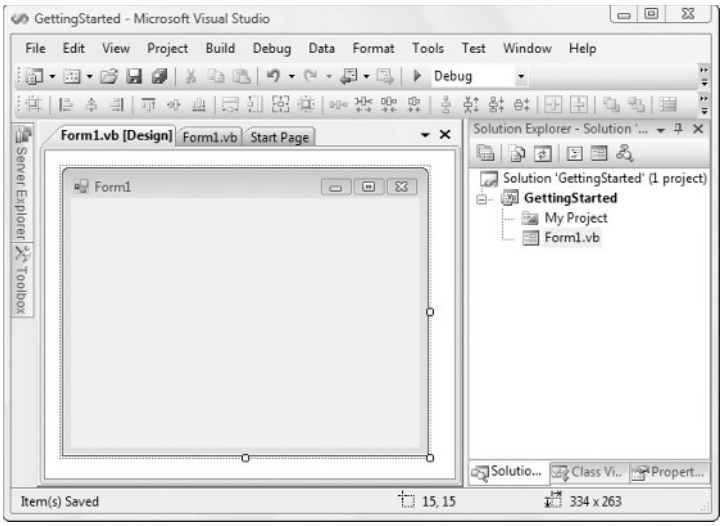

Figure 1-7

2. Click on the Toolbox tool window, which will cause the window to expand, followed by the pin icon, which will pin the tool window open. To add controls to the form, select the appropriate items from the Toolbox and drag them onto the form. In Figure 1-8, you can see how the Toolbar tool window appears after being pinned and the result of clicking and dragging a button onto the form visual designer.

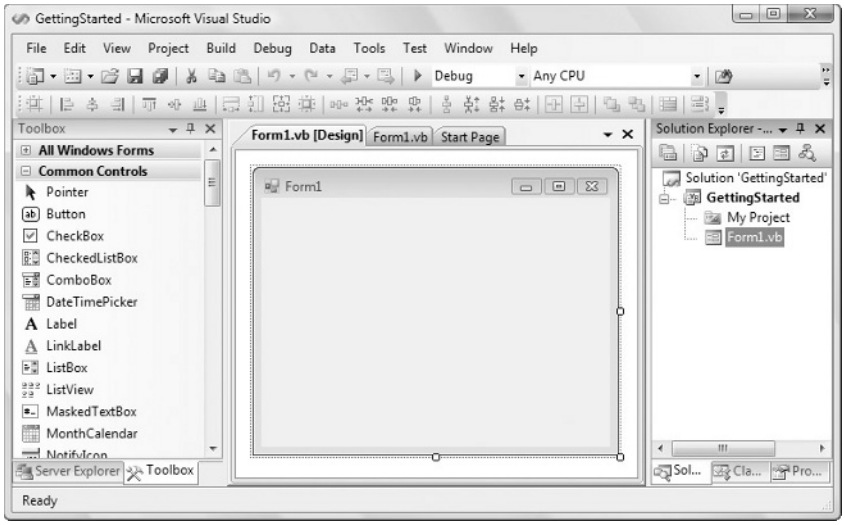

Figure 1-8

3. Add a button and textbox to the form so that the layout looks similar to the one shown in Figure 1-9. Select the textbox and select the Properties tool window (you can press F4 to automatically open the Properties tool window). Use the scrollbar to locate the (`Name`) property and set it to `txtToSay`. Repeat for the button control, naming it `btnSayHello` and setting the `Text` property to `"Say Hello!"`

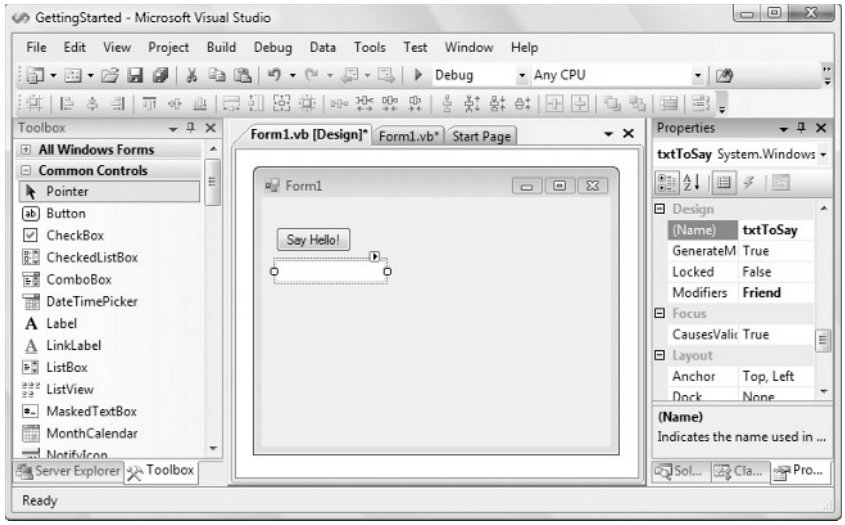

Figure 1-9

4. When a form is opened in the editor space, an additional command bar is added to the top of Visual Studio 2008. If you select both controls on the form, you will see that certain icons on this command bar are enabled. Selecting the Make Same Width icon will align the edges of the two controls, as illustrated in Figure 1-10.

   You will also notice that after you add controls to the form the tab will be updated with an asterisk (*) after the text to indicate that there are unsaved changes to that particular item. If you attempt to close this item while changes are pending, you will be asked if you want to save the changes. When you build the application, any unsaved files will automatically be saved as part of the build process.

One thing to be aware of is that some files, such as the solution file, are modified when you make changes within Visual Studio 2008 without your being given any indication that they have changed. If you try to exit the application or close the solution, you will still be prompted to save these changes.

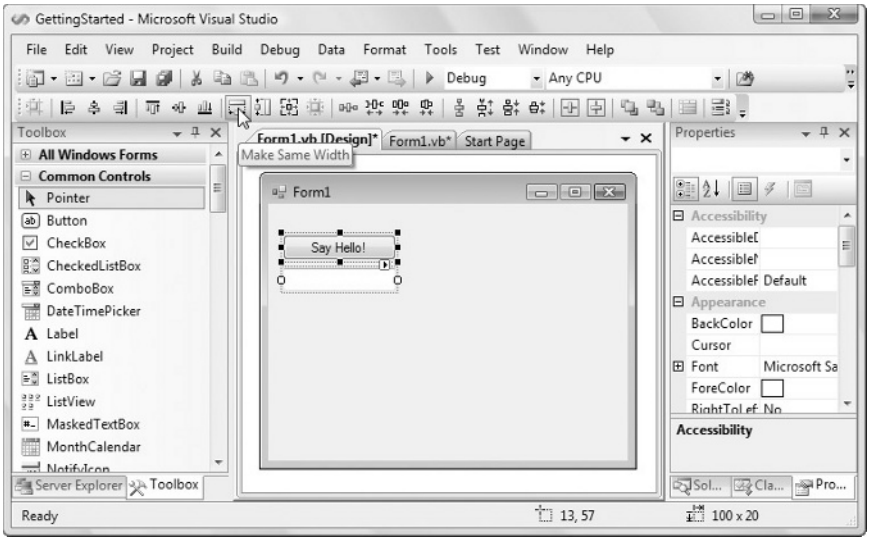

Figure 1-10

5. Deselect all controls and then double-click the button. This will not only open the code editor with the code-behind file for this form; it will also create and wire up an event handler for the `Click Event` on the button. Figure 1-11 shows the code window after we have added a single line to echo the message to the user.

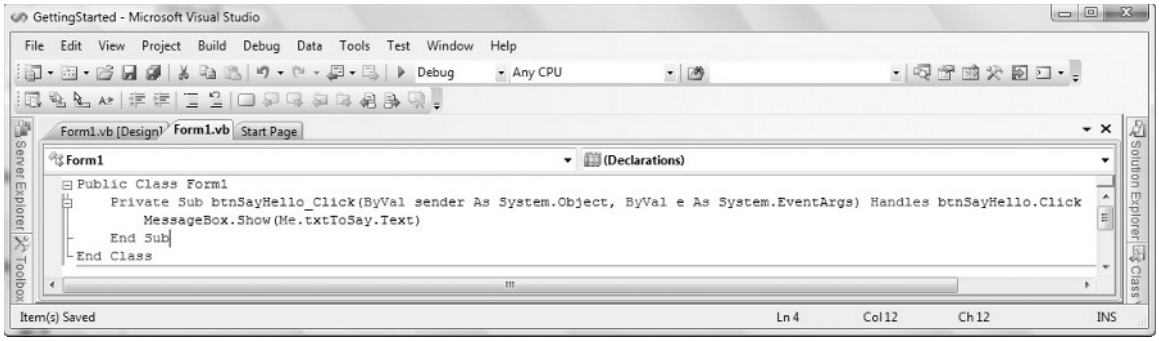

Figure 1-11

6. The last step in the process is to build and execute the application. Before doing so, place the cursor somewhere on the line containing `Messagebox.Show` and press F9. This will set a breakpoint — when you run the application by pressing F5 and then click the Say Hello! button, the execution will halt at this line. Figure 1-12 illustrates this breakpoint being reached. The data tip, which appears when the mouse hovers over the line, shows the contents of the `txtToSay.text` property.

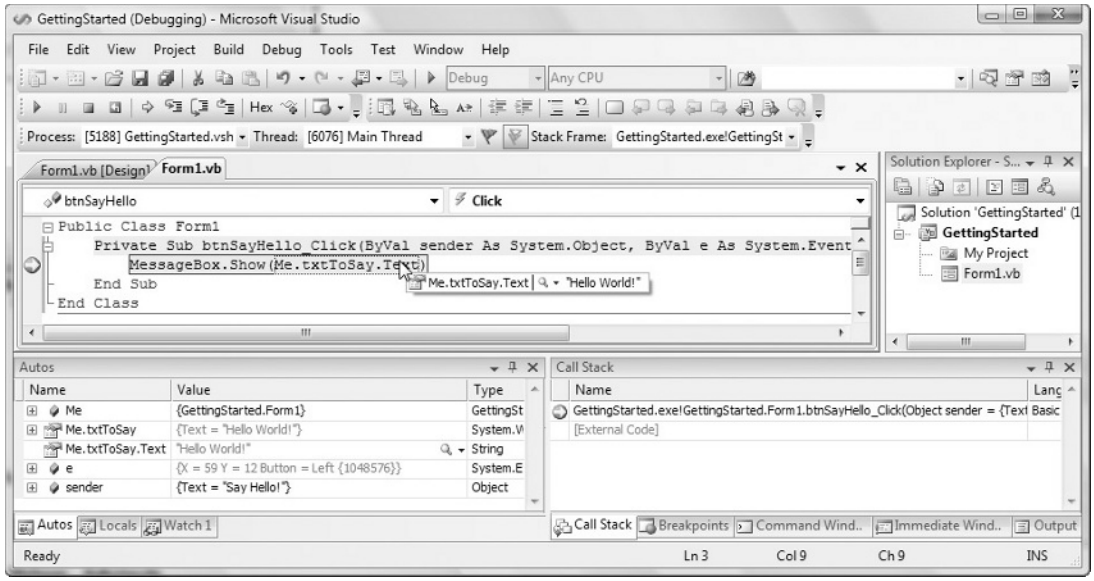

Figure 1-12

The layout of Visual Studio in Figure 1-12 is significantly different from the previous screenshots, as there are a number of new tool windows visible in the lower half of the screen and new command bars at the top. When you stop the application you will notice that Visual Studio returns to the previous layout. Visual Studio 2008 maintains two separate layouts: design

time and runtime. Menus, toolbars, and various windows have default layouts for when you are editing a project, whereas a different setup is defined for when a project is being executed and debugged. You can modify each of these layouts to suit your own style and Visual Studio 2008 will remember them.

It's always a good idea to export your layout and settings (see Chapter 3) once you have them set up just the way you like them. That way you can take them to another PC or restore them if your PC gets rebuilt.

Summary

You've now seen how the various components of Visual Studio 2008 work together to build an application. As a review of the default layout for Visual Basic programs, the following list outlines the typical process of creating a solution:

1. Use the File menu to create a solution.
2. Use the Solution Explorer to locate the form that needs editing and click the View Designer button to show it in the main workspace area.
3. Drag the necessary components onto the form from the Toolbox.
4. Select the form and each component in turn, and edit the properties in the Properties window.
5. Use the Solution Explorer to locate the form and click the View Code button to access the code behind the form's graphical interface.
6. Use the main workspace area to write code and design the graphical interface, switching between the two via the tabs at the top of the area.
7. Use the toolbars to start the program.
8. If errors occur, review them in the Error List and Output windows.
9. Save the project using either toolbar or menu commands, and exit Visual Studio 2008.

While many of these actions can be performed in other ways (for instance, right-click the design surface of a form and you'll find the View Code command), this simplified process shows how the different sections of the IDE work in conjunction with each other to create a comprehensive application design environment.

In subsequent chapters, you'll learn how to customize the IDE to more closely fit your own working style, and how Visual Studio 2008 takes a lot of the guesswork out of the application development process. You will also see a number of best practices for working with Visual Studio 2008 that you can reuse as a developer.