Chapter 1

Wading into Visual Basic

In This Chapter

- Seeing where Visual Basic fits in with .NET
- Writing your first Visual Basic 2008 program
- Exploiting the newfound power of Visual Basic

To get started with Visual Basic 2008, I recommend that you jump right in and write software! And to help you with such an assertive approach, this chapter gives you just what you need to test the waters of the Visual Basic pool and get comfortable with its place in the larger Visual Studio environment.

Then you can really get your feet wet as you build *Hello World* — your first VB 2008 Windows Forms application — right here in the first few pages! You find out how to launch Visual Studio 2008 (the development tool for your VB applications), how to start a new project, and how to build a form visually and make it work with code.

Also in this chapter, I give you a glimpse into the deeper power of Visual Basic. Specifically, I introduce how VB 2008 integrates with the Microsoft .NET Framework and offer insight into what that means to you as a programmer.

Visual Basic's Role in the Framework

Microsoft created the .NET Framework to make development for the various Windows operating systems easier. But because of the differences between Visual Basic 6.0 and Visual Basic 7.0 (the first .NET version), most VB developers found development much harder. For example, VB 7.0 made all variables into objects, which removed the programmer's ability to define a variable type on the fly.

But developing applications in .NET doesn't have to be harder than it was in VB 6.0. The .NET Framework and Visual Basic 2008 can be powerful tools, and the trick is discovering how they work together through the Visual Studio Integrated Development Environment (IDE).

Part of the difficulty that many programmers face when moving to the .NET Framework is the terminology, which can get confusing. I'd like to put the problem with terminology to bed right now, so check out this list of the potentially confusing terms used in .NET development:

- **Visual Basic 2008:** The programming language described throughout this whole book. No longer can you run or load Visual Basic as a separate entity. It is simply one programming language that speaks to the Microsoft .NET Framework, which is the next term in the list.
- **.NET Framework:** The layer that sits between the language (in this case, Visual Basic) and the operating system, which can be Windows 98, Windows Me, Windows 2000, Windows XP, Windows Server 2003, or any of the subversions of those (such as the Tablet PC edition). The .NET Framework layer serves to provide functionality based on the operation of the Windows system on which it resides, as well as to provide libraries for other functionality (such as math computations and database access). Figure 1-1 is a visual representation of the relationship of all the layers in the framework.
- **Visual Studio 2008:** The tool that you use to create any kind of application using any compatible programming language. Visual Studio replaces the Visual Basic 6.0 program that was formerly part of the Visual Studio suite (all individual suite components were labeled Version 6.0). When you go to write a new program in the .NET environment, you run Visual Studio 2008 and select the kind of program you want to write in the programming language you want to use. For example, you may choose to create a Windows Forms program using the Visual Basic language, just like the old days. Or you might want to write an application for a smart device using C#. You can also mix languages, for example, writing the forms in VB and the classes in C#. In this book, I will be using VB for everything — because it is a book about VB!

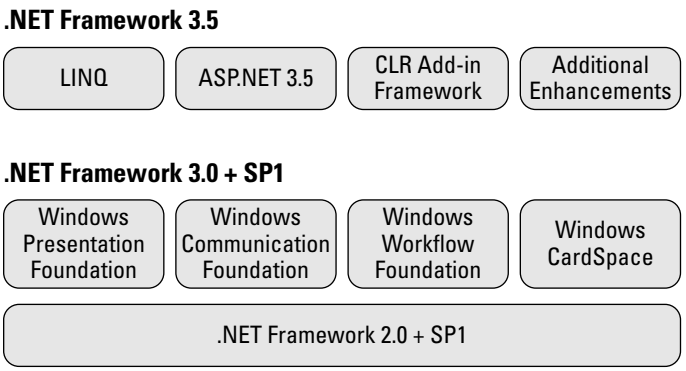

Figure 1-1: The .NET Framework hierarchy.

- **Windows Forms:** The new term for an old-fashioned Visual Basic application. This term refers to an application that is written using the .NET Framework and has a Windows user interface.
- **Web Forms:** The term for an application with a Web page interface written using the .NET Framework. Creating a Web Forms application is very similar to writing a Windows Forms application.
- **Web services:** The class libraries that are written using a standard defined by the same people who defined standards for the World Wide Web. Web services are used for interaction between divergent systems.

The .NET Framework is what you may already know as the Win32 layer in the old Windows DNA system. Like the new .NET Framework, the Win32 layer gave you the ability to get to the functions of the operating system when developing for a Windows platform. Also, the .NET Framework includes a lot of adjunct functionality, such as math and data libraries, that makes programming a more cohesive experience.

Basically, everything that Windows does is exposed by the .NET Framework. Specifically, the .NET Framework gives a programmatic name to nearly every object and event that Windows can control. A programmer can use that name to refer to anything existing in the operating system. Do you need to tell the printer to make two copies of your document? Try referring to `My.Computer.Printers.DefaultPrinter.PrinterSettings.Copies = 2`. Do you need to paint some item on the screen blue? Try referring to `System.Drawing.Brushes.Blue`.

In this .NET world, the programming language becomes just a way to interact with the framework and, therefore, with the Windows operating system. All programs need a set of established rules to handle the flow (decisions, loops, and so on) within programs. Visual Basic provides one such set of rules, and the framework provides the objects and events to interact with.

Saying Hello to VB 2008!

In the following sections, I get you started with the classic Hello World program. Although this isn't the single most exciting application you can build, it helps to make sure that your development environment is set up the best way possible.

How VB 2008 differs from VB 6

Visual Basic 6 was a stand-alone program, and Visual Basic 2008 is one language in a larger development system. To go back to VB's roots, Basic was a programming language used 20 years ago as part of MS-DOS. In 1985, Basic became Visual Basic and was made into a part of the Windows application-building tool. You find a lot more to the Visual Basic 6 program than just the language — its form-building software, for example, is called *Ruby.*

Visual Basic has gone through a few revisions since VB 6. VB 2002 (a.k.a. VB 7), VB 2003 (VB 7.1), and VB 2005 (VB 8) are all just revisions of the language as it uses the .NET Framework. VB 2002 brought on board a whole new way to think about building applications in Windows, and VB 2005 brought back a lot of the features that VB 6 programmers depended on — like ease of use.

In Visual Basic 2008, you have a new way to build user experiences and, with it, a new way to interact with the Windows operating system. The real reason to understand the extent of this larger development system — and the complexity of the .NET Framework that surrounds VB 2008 — is so that reading related books and documentation is easier.

Installing Visual Studio

To follow this example, you need to start by running Visual Studio 2008, which is the development environment used throughout this book to build applications in Visual Basic. Before you can run Visual Studio, you need to install it!

Visual Studio comes in a number of editions:

- **Team System:** Designed for full programming staffs in large corporations, this edition includes large-scale application system design tools such as test-driven development and Team Foundation Server.
- **Professional Edition:** Designed for the developers working with users in a stand-alone setting. The Professional Edition is more common for the solo developer or for mid-sized application development. This is the edition I use in this book.
- **Standard Edition:** Designed for building smaller, stand-alone applications, this version is perfectly functional for 80 percent of applications built. But if you plan to build large systems that need to be enterprise quality and may have many users, go for the Professional Edition.
- **Express Edition:** Designed for students and hobbyists. This version lacks a lot of the project types that the other versions have. If you are running Express, some of the examples in this book won't work for you. On this book's Web site (`www.vbfordummies.net`), I have posted a few Express articles and some projects that I have altered to work in Express edition.

If you don't have access to the MSDN Library (Microsoft's handy technical archive), I highly recommend getting it. You can load up a machine with your choice of sample code, documentation, and other reference material on Visual Studio editions, operating systems, and server software. You can find out about the library at `http://msdn.microsoft.com`, and you can buy subscriptions from several resellers, including your favorite software dealer.

Installing Visual Studio can be rough, so I recommend going with the defaults for your first time. The installation process takes quite a while, too. Even if you are using the DVD, expect to spend two hours installing the program. If you are working from the CDs, expect to spend four hours.

After installing Visual Studio, you can run it by choosing Start➪All Programs➪Microsoft Visual Studio 2008➪Microsoft Visual Studio 2008. The environment loads, and you can get started on a program by choosing File➪New➪Project. Next, you need to make choices about your project type and language, as described in the next section.

Starting a Windows Forms project

After you choose File➪New➪Project in Visual Studio, the New Project dialog box appears, as shown in Figure 1-2. In the Project Types pane, you find a folder structure that lists the languages loaded with your installation and the project types available for those languages. I suggest beginning with a plain old Windows Forms Application — which is the Visual Basic 2008 answer to the traditional (and perhaps familiar) VB 6.0 application.

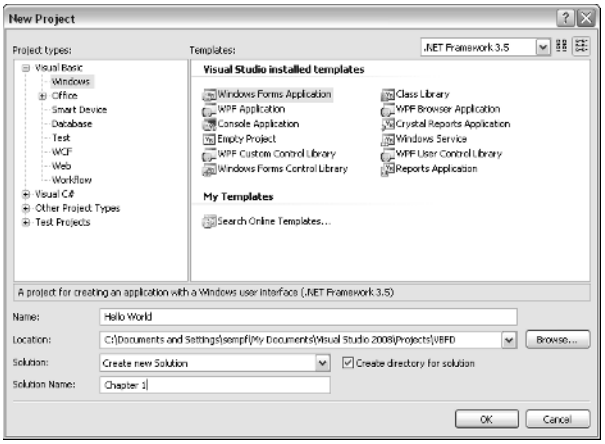

Figure 1-2: The New Project dialog box.

To get started building your Hello World application, follow these steps:

1. **Select the project type from the Templates pane in the New Project dialog box.**

   For this example, select Windows Forms Application. Also, make sure that Visual Basic is the language selected in the Project Types pane. If you loaded other languages during installation, you may have other choices.

2. **Type the name you want to give your project to replace the default name in the Name text box.**

   In this example, I type **Hello World** in the text box.

3. **Click the OK button.**

   Visual Basic loads the default form (called `Form1`) and presents it to you in the Design View. The default form comes complete with a workspace, the title bar, and familiar Windows elements like the Resize buttons and the Close button. You do most of the work to customize your form using this visual view.

4. **Click the word Toolbox on the left side of the screen and open the Common Controls tree.**

   The Toolbox appears, with Windows Forms controls loaded, as shown in Figure 1-3.

5. **Double-click the Button control.**

   Visual Studio loads a button onto the default form in Design View.

6. **On the default `Form1`, click the Button control and drag it to reposition it on the form.**

   Figure 1-4 shows the result of dragging the button to the middle of the `Form1` window.

This step list gives you the beginnings of the Windows Forms application, which you see as `Form1` in the Design View. But to see where Visual Basic comes in, you have to find the code behind the form. Visual Studio offers you (surprise!) the Code View when you're ready to use Visual Basic to add functionality to your form.

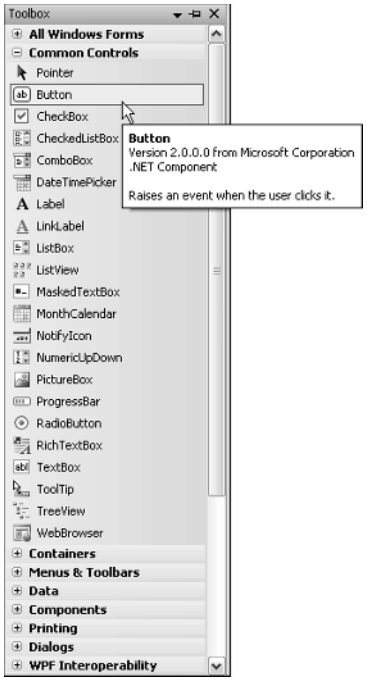

Figure 1-3: Choosing the Button control from the Toolbox.

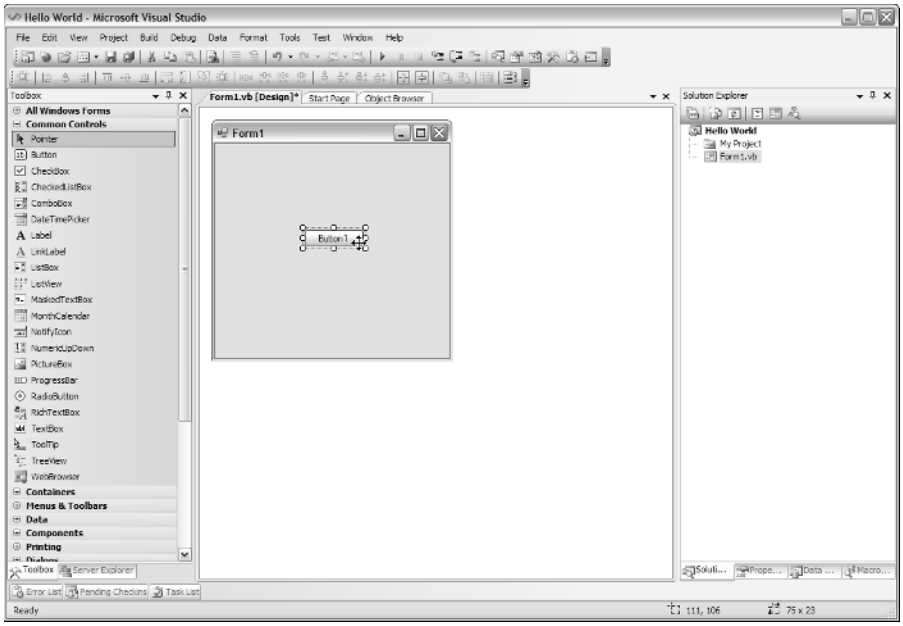

Figure 1-4: Moving the button around the form.

Adding functionality to the form with VB code

To add a little functionality to the Windows form you build in the preceding section, follow these steps:

1. **Double-click the Button control to enter Code View.**

   In the Code View window, you see basic button-click code that looks like the following:

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As _
        System.Object, ByVal e As System.EventArgs) _
        Handles Button1.Click

    End Sub
End Class
```

   This code is a template that wraps the code that will be run when you click the button. Visual Studio does the hard part for you, making sure that the formatting of the `Sub` is correct!

2. **In the Code View window, type a line of code to change the text that appears on the Button control to *Hello World*.**

   Specifically, type the following code on the line preceding the `End Sub` line:

```
Button1.Text = "Hello World"
```

   Your button's code now looks like the following:

```
Public Class Form1
Private Sub Button1_Click(ByVal sender As _
            System.Object, ByVal e As System.EventArgs)
        _
            Handles Button1.Click

 Button1.Text = "Hello World"

 End Sub
End Class
```

Running and operating your Windows form

So this experience is pretty cool, right? Programming with Visual Basic is so easy that, here in Chapter 1, you can already write a Windows Forms application. But what can you do with it? Check out the following:

- **Run your Windows Forms application within the Visual Studio environment.** Press F5, and Visual Studio opens your active project as a Windows program. It appears on your taskbar and everything. Click the button on your form, and the button text changes to "Hello World" (or whatever text you specified in the code). Pretty neat, huh? Your Windows form should look something like the image in Figure 1-5.
- **Run your application outside of the Visual Studio environment.** If you are still in Debug mode, you will need to stop your program first by using the Stop button on the toolbar or by closing the form window. Then you can save and move on.

  The very simple way to run an application outside of Visual Studio is as follows:

  1. **Choose File➪Save All.**

     Visual Studio will save your project using the defaults you supplied in the Add New Project dialog box.

  2. **Choose Build➪Build Program Name.**

     In this example, choose Build➪Build Solution, and Visual Studio compiles your application into a usable Windows program (with the file extension `.exe`) and stores it in the default folder.

  3. **Navigate to the default folder containing your new Windows application.**

     For my application, the path is `C:\Documents and Settings\sempf\My Documents\Visual Studio 2008\Projects\VBFD\Chapter1\Hello World\bin\Debug`.

Figure 1-5: Your Hello World application.

If your local configuration for the project happens to be set to Release mode (not recommended for this book), you might find it in `C:\Documents and Settings\sempf\My Documents\Visual Studio 2008\Projects\VBFD\Chapter1\Hello World\bin\Release`.

4. **Double-click the filename for the compiled program to run it.**

   You may see a lot of files in the default folder, but in the example, `Hello World.exe` is the file you're looking for.

There is a more complex method for running your VB programs outside the Visual Studio environment. You use a Setup Project, which is a very cool tool but beyond the scope of this book. Research the term *Setup Project* in the MDSN Library when you're ready to find out more about this device, which helps you distribute your application to other users.

Finding More Power in Visual Studio

Earlier in this chapter, I showed you the Windows Forms application development environment and a little of the new Visual Basic 2008 code. If you are familiar with VB 6.0, the form and the code look pretty familiar at this point. In fact, the major Windows development tools for any programming language work pretty much this way.

But when you look beyond the Windows form and the code structure, a few more details become evident. For instance, Visual Studio takes your VB code beyond the Windows form. The following sections give you an overview of the development power that you find in Visual Studio.

Visual Studio doesn't just do Windows!

The first evident change that sets Visual Studio apart as a development tool is this: You can use Visual Studio to write programs that run on the World Wide Web as well as on Windows computers. When you click the File menu to add a new project, notice the second option in the menu. As shown in Figure 1-6, the second project option is a new Web site.

Choose this option to create a Web application, which incorporates a whole host of technologies — the .NET Framework, ASP.NET, Visual Basic, and HTML — that each have essential roles for enabling an application to run online.

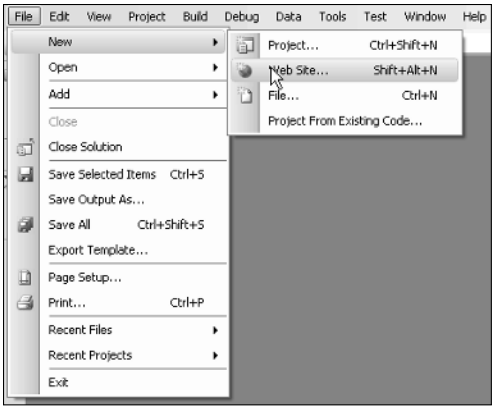

Figure 1-6: The File menu in Visual Studio.

Visual Basic goes mobile

Mobile computing made its appearance in Visual Basic 2005 and really shines now in Visual Studio 2008. If you follow development for mobile devices, you may have noticed the plethora of releases from the Mobile team over the past few years. They are all baked right into Visual Studio 2008. Pocket PC 2003, Mobile 5.0, and Mobile 6.0 all make their appearance in Visual Studio 2008, and can be programmed in VB 2008 — just like every other type of project.

I don't give examples of these specific project types in this book because you can create a mobile device application in the same manner that you create a Windows Forms application (like the Hello World program discussed earlier in the chapter). You should know that getting familiar with the Visual Basic language as presented in this book puts you on the right track for creating applications for a Pocket PC. Mobile computing applications require some special programming practices, so make sure to grab some device-specific information when you work on those project types.

Writing routines to use with other software is easier with Visual Basic 2008. You can write add-ins for Microsoft Office apps, including Excel and Word templates with VB code running behind them. These routines don't use the VBScript that you may have seen before; a completely new part of Office 2007 allows you to write templates with special, built-in functionality. For example, I've built a Word template that automates a reporting process by asking the user for a report number, checking that number against a database of all the reports filed, and filling out part of the document-in-process with the relevant information from the database. You can also customize the ribbon bar and create and deploy add-ins easily.

VB as your one-stop development shop

Generally, Visual Studio and the .NET Framework are designed to be the one-stop shop for any kind of development on Windows machines. But in this version, Visual Basic 2008 can also do it all. The language can now touch all the parts of the .NET Framework that any of the other languages can get to, without resorting to the cryptic function calls necessary in prior versions of VB.

The new features covered in this book include the following:

- **The Windows Presentation Foundation:** Microsoft has updated the formula to design new user experiences again with even more power.
- **The Windows Communication Foundation:** Making interconnected applications even more powerful, the WCF is an advanced step that I'll touch on later in the book.
- **Language Integrated Query:** LINQ brings data constructs right into your code with new query mechanisms for collections of objects.
- **System.XML:** If you are working with Extensible Markup Language, VB 2008 brings new meaning to the word *simple*.