1

Welcome to Visual Basic 2005

The goal of this book is to help you come up to speed with the Visual Basic 2005 language even if you have never programmed before. You will start slowly and build on what you learn. So take a deep breath, let it out slowly, and tell yourself you can do this. No sweat! No kidding!

Programming a computer is a lot like teaching a child to tie his shoes. Until you find the correct way of giving the instructions, not much gets accomplished. Visual Basic 2005 is a language in which you can tell your computer how to do things. But, like a child, the computer will understand only if you explain things very clearly. If you have never programmed before, this sounds like an arduous task, and sometimes it is. However, Visual Basic 2005 gives you a simple language to explain some complex things. Although it never hurts to have an understanding of what is happening at the lowest levels, Visual Basic 2005 frees the programmer from having to deal with the mundane complexities of writing Windows programs. You are free to concentrate on solving problems.

Visual Basic 2005 helps you create solutions that run on the Microsoft Windows operating system. If you are looking at this book, you might have already felt the need or the desire to create such programs. Even if you have never written a computer program before, as you progress through the Try It Out exercises in this book, you will become familiar with the various aspects of the Visual Basic 2005 language, as well as its foundation in Microsoft's .NET Framework. You will find that it is not nearly as difficult as you have been imagining. Before you know it, you will be feeling quite comfortable creating a variety of different types of programs with Visual Basic 2005. Also (as the name .NET implies) Visual Basic 2005 can be used to create applications for use over the Internet. You can also create mobile applications for Pocket PCs and SmartPhones. However, when learning any new technology, you have to walk before you can run, so in this book you will begin by focusing on Windows applications before extending your boundaries to other platforms.

In this chapter, we will cover the following subjects:

- ❑ The installation of Visual Basic 2005
- ❑ A tour of the Visual Basic 2005 Integrated Development Environment (IDE)

- ❑ How to create a simple Windows program
- ❑ How to use and leverage the integrated help system

Windows Versus DOS Programming

A Windows program is quite different from its ancient relative, the MS-DOS program. A DOS program follows a relatively strict path from beginning to end. Although this does not necessarily limit the functionality of the program, it does limit the road the user has to take to get to it. A DOS program is like walking down a hallway; to get to the end you have to walk down the hallway, passing any obstacles that you may encounter. A DOS program would only let you open certain doors along your stroll.

Windows, on the other hand, opened up the world of *event-driven programming*. *Events* in this context include, for example, clicking a button, resizing a window, or changing an entry in a text box. The code that you write responds to these events. To go back to the hallway analogy: In a Windows program, to get to the end of the hall, you just click on the end of the hall. The hallway can be ignored. If you get to the end and realize that is not where you wanted to be, you can just set off for the new destination without returning to your starting point. The program reacts to your movements and takes the necessary actions to complete your desired tasks (Visual Basic 2005).

Another big advantage in a Windows program is the *abstraction of the hardware*; which means that Windows takes care of communicating with the hardware for you. You do not need to know the inner workings of every laser printer on the market just to create output. You do not need to study the schematics for graphics cards to write your game. Windows wraps up this functionality by providing generic routines that communicate with the drivers written by hardware manufacturers. This is probably the main reason that Windows has been so successful. The generic routines are referred to as the Windows *Application Programming Interface* (API).

Before Visual Basic 1.0 was introduced to the world in 1991, developers had to be well versed in C and C++ programming, as well as the building blocks of the Windows system itself, the Windows API. This complexity meant that only dedicated and properly trained individuals were capable of turning out software that could run on Windows. Visual Basic changed all of that, and it has been estimated that there are now as many lines of production code written in Visual Basic as in any other language.

Visual Basic changed the face of Windows programming by removing the complex burden of writing code for the user interface (UI). By allowing programmers to *draw* their own UI, it freed them to concentrate on the business problems they were trying to solve. Once the UI is drawn, the programmer can then add the code to react to events.

Visual Basic has also been *extensible* from the very beginning. Third-party vendors quickly saw the market for reusable modules to aid developers. These modules, or *controls*, were originally referred to as VBXs (named after their file extension). Prior to Visual Basic 5.0, if you did not like the way a button behaved, you could either buy or create your own, but those controls had to be written in C or C++. Database access utilities were some of the first controls available. Version 5 of Visual Basic introduced the concept of *ActiveX*, which allowed developers to create their own *ActiveX controls*.

When Microsoft introduced Visual Basic 3.0, the programming world changed again. Now you could build database applications directly accessible to users (so-called *front-end applications*) completely with

Visual Basic. There was no need to rely on third-party controls. Microsoft accomplished this task with the introduction of *Data Access Objects* (DAO), which allowed programmers to manipulate data with the same ease as manipulating the user interface.

Versions 4.0 and 5.0 extended the capabilities of Version 3.0 to allow developers to target the new Windows 95 platform. Crucially they also made it easier for developers to write code, which could then be manipulated to make it usable to other language developers. Version 6.0 provided a new way to access databases with the integration of *ActiveX Data Objects* (ADO). The ADO feature was developed by Microsoft to aid Web developers using Active Server Pages to access databases. All of the improvements to Visual Basic over the years have ensured its dominant place in the programming world. It helps developers write robust and maintainable applications in record time.

With the release of Visual Basic .NET in February 2002, most of the restrictions that used to exist have been obliterated. In the past, Visual Basic has been criticized and maligned as a "toy" language, as it did not provide all of the features of more sophisticated languages such as C++ and Java. Now, Microsoft has removed these restrictions and made Visual Basic .NET a very powerful development tool. This trend continues with Visual Basic 2005. Although not as drastic a change as from Visual Basic 6 to Visual Basic .NET, there are enough improvements in the language and integrated development environment that Visual Basic 2005 is a welcome upgrade and is a great choice for programmers of all levels.

Installing Visual Basic 2005

You may own Visual Basic 2005 in either of the following forms:

- As part of Visual Studio 2005, a suite of tools and languages that also includes C# (pronounced "C-sharp"), J# (pronounced "J-sharp"), and Visual C++. The Visual Studio 2005 product line includes Visual Studio Standard Edition, Visual Studio Professional Edition, Visual Studio Tools for Office, and Visual Studio Team System. All of these versions come with progressively more tools for building and managing the development of larger, enterprise-wide applications.
- As the Express Edition, which includes a reduced set of the tools and features that are available with Visual Studio 2005.

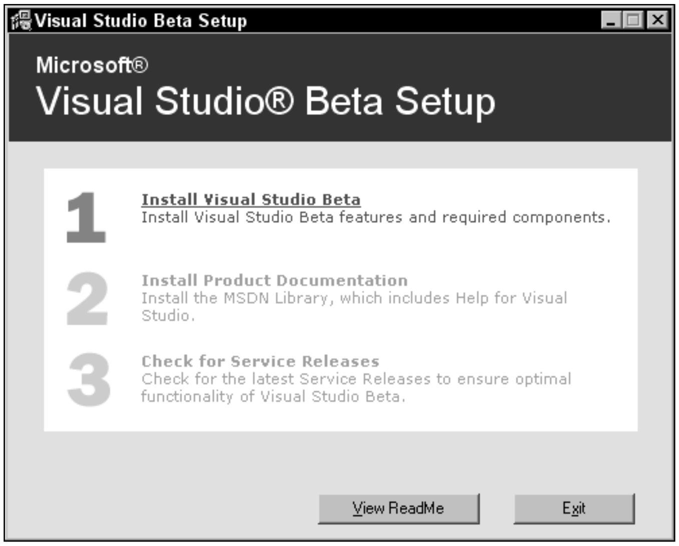
Both enable you to create your own applications for the Windows platform. The installation procedure is straightforward. In fact, the Visual Studio Installer is smart enough to figure out exactly what your computer requires to make it work.

The descriptions in the Try It Out exercise that follows are based on installing Visual Studio 2005 Architect Edition. Most of the installation processes are very straightforward, and you can accept the default installation options for most environments. So, regardless of which edition you are installing, the installation process should be smooth when accepting the default installation options.

Try It Out Installing Visual Basic 2005

1. The Visual Studio 2005 CD has an auto-run feature, but if the Setup screen does not appear after inserting the CD, you have to run `setup.exe` from the root directory of the CD. To do this, go to your Windows Start menu (usually found right at the bottom of your screen) and select Run. Then type ***d*:\ setup.exe** into the Open box, where *d* is the drive letter of your CD drive. After the setup program initializes, you will see the screen as shown in Figure 1-1.

2. This dialog box shows the order in which the installation takes place. To function properly, Visual Basic 2005 requires that several updates be installed on your machine, such as Service Pack 1 for Windows XP. The setup program will inform you if these updates are not installed. You should then install any required updates before proceeding with the installation of Visual Studio 2005. Step 1 installs Visual Studio 2005, so click the Install Visual Studio link.


Visual Studio Beta Setup

Microsoft®

Visual Studio® Beta Setup

1 Install Visual Studio Beta
Install Visual Studio Beta features and required components.

2 Install Product Documentation
Install the MSDN Library, which includes Help for Visual Studio.

3 Check for Service Releases
Check for the latest Service Releases to ensure optimal functionality of Visual Studio Beta.

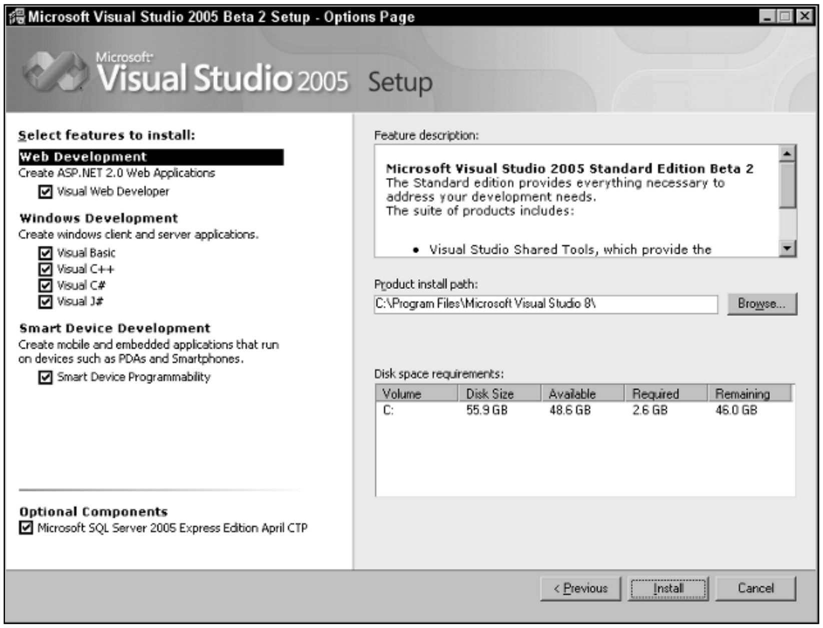
Figure 1-1

3. After agreeing to the End User License agreement, click Continue to proceed to the next step.
4. As with most installations, you will be presented with an option list of components to install (see Figure 1-2). You can choose to install only the features that you need. For example, if your drive space is limited and you have no immediate need for Visual C++ 2005, you can exclude it from the installation. You will also be given the chance to select the location of items (although the defaults should suffice unless your particular machine has special requirements). Any option that is not chosen at the initial setup can always be added later as your needs or interests change. However, if you plan on developing database applications such as those discussed in Chapter 16, you should choose to install SQL Server 2005 Express, which is the last option in the list.

   Three sections of information are given for each feature:

   - ❑ The Feature description box gives you an outline of each feature and its function.
   - ❑ The Feature Install path section outlines where the required files will be installed.
   - ❑ Finally, the Space Allocation section illustrates how the space on your hard drive will be affected by the installation as a whole.

When you are running Visual Basic 2005, a lot of information is swapped from the disk to memory and back again. Therefore, it is important to have some free space on your disk. There is no exact rule for determining how much free space you will need, but if you use your machine for development as well as other tasks, anything less than 100MB free space should be considered a full disk.


Microsoft Visual Studio 2005 Beta 2 Setup - Options Page
Microsoft Visual Studio 2005 Setup
< Previous
Install
Cancel
Browse...

Select features to install:

Web Development
Create ASP.NET 2.0 Web Applications
- [✓] Visual Web Developer

Windows Development
Create windows client and server applications.
- [✓] Visual Basic
- [✓] Visual C++
- [✓] Visual C#
- [✓] Visual J#

Smart Device Development
Create mobile and embedded applications that run on devices such as PDAs and Smartphones.
- [✓] Smart Device Programmability

Optional Components
- [✓] Microsoft SQL Server 2005 Express Edition April CTP

Feature description:

Microsoft Visual Studio 2005 Standard Edition Beta 2
The Standard edition provides everything necessary to address your development needs.
The suite of products includes:

- Visual Studio Shared Tools, which provide the

Product install path:
C:\Program Files\Microsoft Visual Studio 8\

Disk space requirements:

Volume	Disk Size	Available	Required	Remaining
C:	55.9 GB	48.6 GB	2.6 GB	46.0 GB

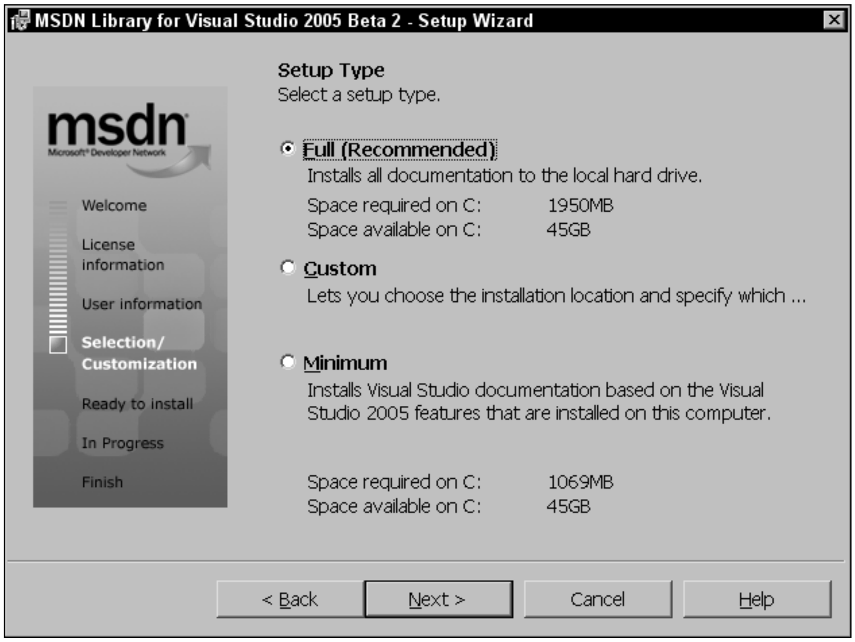
Figure 1-2

5. After you have chosen all the features you want, click Install. Installation will begin and you can sit back and relax for a bit. The setup time varies depending on how many features you chose to install. As a reference, the installation process took around 20 minutes on a 2.4-GHz computer with 512 MB RAM running Windows XP Professional.
6. When installation is completed, you will see a dialog informing you that the installation has completed.

   Here you will see any problems that setup encountered along the way. You are also given the chance to look at the installation log. This log provides a list of all actions taken during the installation process. Unless your installation reported errors, the installation log can safely be ignored. The Visual Studio 2005 setup is nearly complete. Click Done to move on to installing the documentation.
7. The MSDN Library installation is simple and straightforward, and this section covers the highlights. The first screen that you will see is the initial welcome screen. Click Next to proceed.
8. You will be allowed to select the amount of the documentation you want to install, as shown in Figure 1-3. Click Next to start the installation process.

If you have the spare hard drive space, it is a very good idea to install the full documentation. That way you have access to the full library, which will be important if you choose a limited set of options during the install and later add more features.

9. After the MSDN documentation has been installed, you are returned to the initial setup screen again, and the Service Releases option is available.


MSDN Library for Visual Studio 2005 Beta 2 - Setup Wizard

Setup Type
Select a setup type.

(●) Full (Recommended)
Installs all documentation to the local hard drive.
Space required on C: 1950MB
Space available on C: 45GB

( ) Custom
Lets you choose the installation location and specify which ...

( ) Minimum
Installs Visual Studio documentation based on the Visual Studio 2005 features that are installed on this computer.

Space required on C: 1069MB
Space available on C: 45GB

Figure 1-3

It is a good idea to select Service Releases to check for updates. Microsoft has done a good job of making software updates available through the Internet. These updates can include anything from additional documentation to bug fixes. You will be given the choice to install any updates via a Service Pack CD or the Internet. Obviously, the Internet option requires an active connection. Since updates can be quite large, a fast connection is highly recommended.

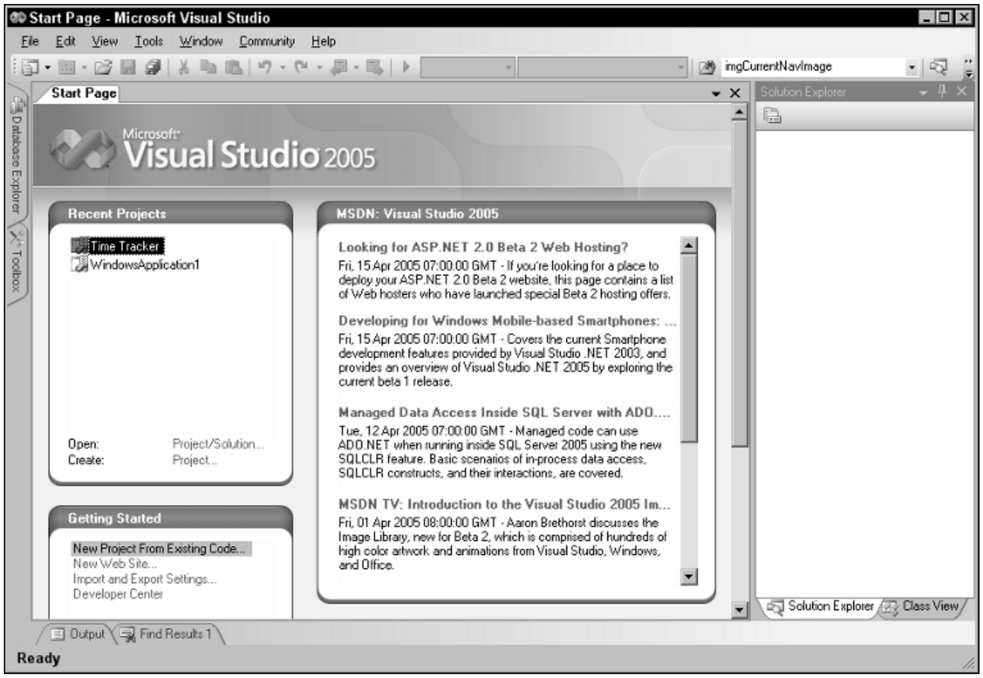
Once you have performed the update process, Visual Studio 2005 is ready to use. Now the real fun can begin! So get comfortable, relax, and let us enter the world of Visual Basic 2005.

The Visual Basic 2005 IDE

You don't actually need the Visual Basic 2005 product to write applications in the Visual Basic 2005 language. The actual ability to run Visual Basic 2005 code is included with the .NET Framework. You could actually just write all of your Visual Basic 2005 using a text editor such as Notepad. You could also hammer nails using your shoe as a hammer, but that slick pneumatic nailer sitting there is probably a lot more efficient. In the same way, by far the easiest way to write in Visual Basic 2005 is by using the Visual Studio 2005 Integrated Development Environment, also known as the IDE. This is what you actually see when working with Visual Basic 2005 — the windows, boxes, and so on. The IDE provides a wealth of features unavailable in ordinary text editors — such as code checking, visual representations of the finished application, and an explorer that displays all of the files that make up your project.

The Profile Setup Page

An IDE is a way of bringing together a suite of tools that makes developing software a lot easier. Fire up Visual Studio 2005 and see what you've got. If you used the default installation, go to your Windows Start menu and then Programs (All Programs on Windows XP and Windows Server 2003) ➪ Microsoft Visual Studio 2005 ➪ Microsoft Visual Studio 2005. A splash screen will briefly appear, and then you should find yourself presented with the Choose Default Environment Settings dialog box. Select the Visual Basic Development Settings option and then click Start Visual Studio. The Microsoft Development Environment will appear, as shown in Figure 1-4.



Looking for ASP.NET 2.0 Beta 2 Web Hosting?
Fri, 15 Apr 2005 07:00:00 GMT - If you're looking for a place to deploy your ASP.NET 2.0 Beta 2 website, this page contains a list of Web hosters who have launched special Beta 2 hosting offers.

Developing for Windows Mobile-based Smartphones: ...
Fri, 15 Apr 2005 07:00:00 GMT - Covers the current Smartphone development features provided by Visual Studio .NET 2003, and provides an overview of Visual Studio .NET 2005 by exploring the current beta 1 release.

Managed Data Access Inside SQL Server with ADO....
Tue, 12 Apr 2005 07:00:00 GMT - Managed code can use ADO.NET when running inside SQL Server 2005 using the new SQLCLR feature. Basic scenarios of in-process data access, SQLCLR constructs, and their interactions, are covered.

MSDN TV: Introduction to the Visual Studio 2005 Im...
Fri, 01 Apr 2005 08:00:00 GMT - Aaron Brethorst discusses the Image Library, new for Beta 2, which is comprised of hundreds of high color artwork and animations from Visual Studio, Windows, and Office.

Figure 1-4

The Menu

By now, you may be a bit eager to start writing some code. But first, begin your exploration of the IDE by looking at the toolbar and menu, which, as you will learn are not really all that different from the toolbars and menus you have seen in other Microsoft software such as Word, Excel, and PowerPoint.

Visual Studio 2005's menu is *dynamic*, meaning that items will be added or removed depending on what you are trying to do. While you are looking at the blank IDE, the menu bar will consist only of the File, Edit, View, Data, Tools, Window, Community, and Help menus. When you start working on a project, however, the full Visual Studio 2005 menu appears as shown in Figure 1-5.

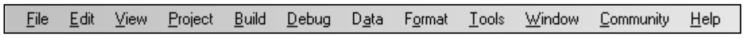

File Edit View Project Build Debug Data Format Tools Window Community Help

Figure 1-5

At this point, there is no need to cover each menu topic in great detail. You will become familiar with each of them as you progress through the book. Here is a quick rundown of what activities each menu item pertains to:

- **File:** It seems every Windows program has a File menu. It has become the standard where you should find, if nothing else, a way to exit the application. In this case, you can also find ways of opening and closing single files and whole projects.
- **Edit:** The Edit menu provides access to the items you would expect: Undo, Redo, Cut, Copy, Paste, and Delete.
- **View:** The View menu provides quick access to the windows that exist in the IDE, such as the Solution Explorer, Properties window, Output window, Toolbox, and so on.
- **Project:** The Project menu allows you to add various files to your application such as forms and classes.
- **Build:** The Build menu becomes important when you have completed your application and want to run it without the use of the Visual Basic 2005 environment (perhaps running it directly from your Windows Start menu, as you would any other application such as Word or Access).
- **Debug:** The Debug menu allows you to start and stop running your application within the Visual Basic 2005 IDE. It also gives you access to the Visual Studio 2005 debugger. The debugger allows you to step through your code while it is running to see how it is behaving.
- **Data**: The Data menu helps you to use information that comes from a database. It appears only when you are working with the visual part of your application (the [Design] tab will be the active one in the main window), not when you are writing code. Chapters 15 and 16 will introduce you to working with databases.
- **Format**: The Format menu also appears only when you are working with the visual part of your application. Items on the Format menu allow you to manipulate how the controls you create will appear on your forms.
- **Tools:** The Tools menu has commands to configure the Visual Studio 2005 IDE, as well as links to other external tools that may have been installed.
- **Window:** The Window menu has become standard for any application that allows more than one window to be open at a time, such as Word or Excel. The commands on this menu allow you to switch between the windows in the IDE.
- **Community:** The Community menu provides access to developer resources, where you can ask questions, search for code snippets, and send product feedback.
- **Help**: The Help menu provides access to the Visual Studio 2005 documentation. There are many different ways to access this information (for example, via the help contents, an index, or a search). The Help menu also has options that connect to the Microsoft Web site to obtain updates or report problems.

The Toolbars

Many toolbars are available within the IDE, including Formatting, Image Editor, and Text Editor, which you can add to and remove from the IDE via the View ➪ Toolbars menu option. Each one provides quick access to often-used commands, preventing you from having to navigate through a series of menu options. For example, the leftmost icon (New Project) on the default toolbar (called the Standard toolbar), shown in Figure 1-6, is available from the menu by navigating to File ➪ New ➪ Project.

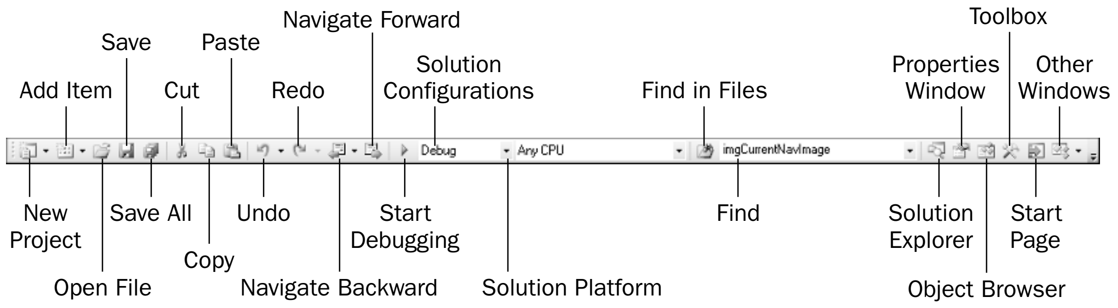


Figure 1-6

The toolbar is segmented into groups of related options, which are separated by a vertical bar. The first five icons provide access to the commonly used project and file manipulation options available through the File and Project menus, such as opening and saving files.

The next group of icons is for editing (Cut, Copy, and Paste). The third group of icons is for undoing and redoing edits and for navigating through your code.

The fourth group of icons provides the ability to start your application running (via the green triangle). You can also choose a configuration for your solution and target specific platforms.

The next section allows you to find text in your code throughout the entire document, project, or solution.

The final group of icons provides quick links back to the Solution Explorer, Properties window, Toolbox, Object Browser, Start Page, and other windows. If any of these windows is closed, clicking the appropriate icon will bring it back into view.

If you forget what a particular icon does, you can hover your mouse pointer over it so that a ToolTip appears displaying the name of the toolbar option.

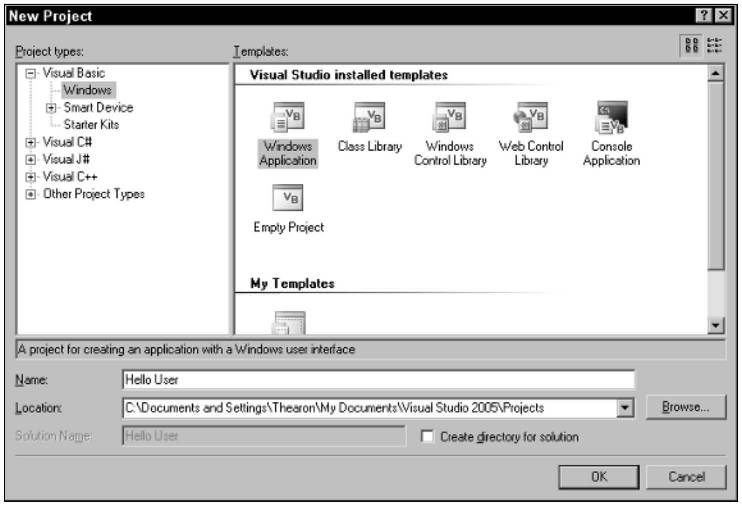
You could continue to look at each of the windows by clicking on the View menu and choosing the appropriate window. But, as you can see, they are all empty at this stage and therefore not too revealing. The best way to look at the capabilities of the IDE is to use it while writing some code.

Creating a Simple Application

To finish your exploration of the IDE, you need to create a project, so that the windows shown earlier in Figure 1-4 actually have some interesting content for you to look at. In the following Try It Out, you are going to create a very simple application called HelloUser that will allow you to enter a person's name and display a greeting to that person in a message box.

Try It Out Creating a HelloUser Project

1. Click the New Project button on the toolbar.
2. The New Project dialog box will open. Make sure you have Visual Basic selected in the Project Types tree-view box to the left. Next, select Windows Application in the Templates box on the right. Finally, type **Hello User** in the Name text box and click OK. Your New Project dialog box should look like Figure 1-7.



A project for creating an application with a Windows user interface

Name: Hello User

Location: C:\Documents and Settings\Thearon\My Documents\Visual Studio 2005\Projects

Solution Name: Hello User

☐ Create directory for solution

Figure 1-7

3. The IDE will then create an empty Windows application for you. So far, your Hello User program consists of one blank window, called a Windows Form (or sometimes just a form), with the default name of `Form1.vb`, as shown in Figure 1-8.

Whenever Visual Studio 2005 creates a new file, either as part of the project creation process or when you create a new file, it will use a name that describes what it is (in this case, a form) followed by a number.

Windows in the Visual Studio 2005 IDE

At this point, you can see that the various windows in the IDE are beginning to show their purposes, and you should take a brief look at them now before you come back to the Try It Out. Note that if any of these windows are not visible on your screen, you can use the View menu to select and show them. Also, if you do not like the location of any particular window, you can move it by clicking on its title bar (the blue bar at the top) and dragging it to a new location. The windows in the IDE can *float* (stand out on their own) or be *docked* (as they appear in Figure 1-8). The following list introduces the most common windows:

- **Database Explorer:** The Database Explorer provides access to your defined database connections. Here you can create new database connections and view existing database connections. In Figure 1-8, the Database Explorer is a tab at the bottom of the Toolbox window.

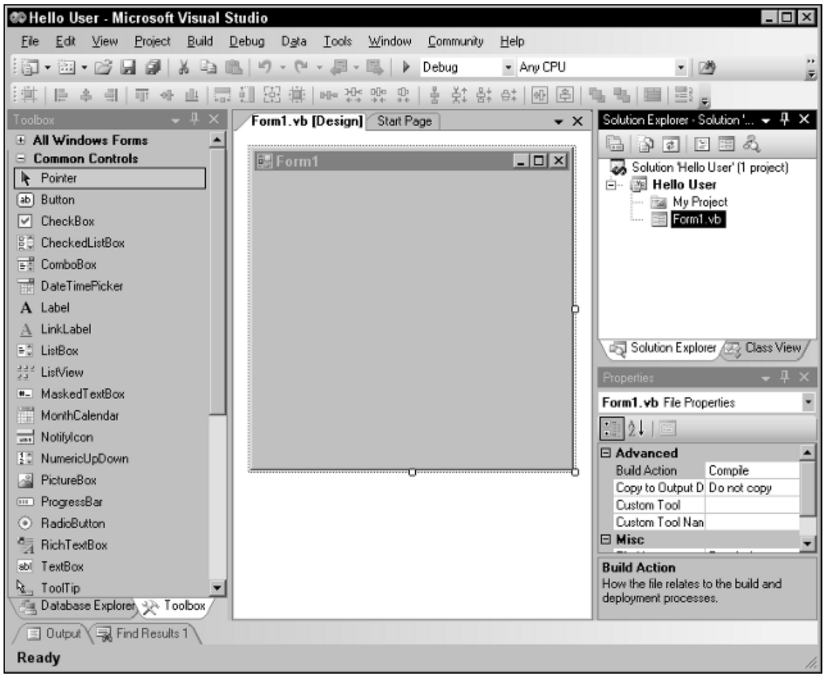
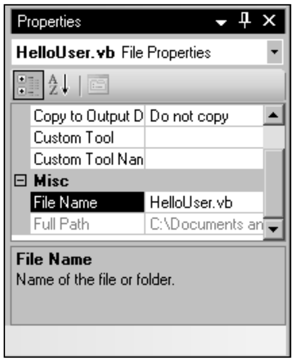


Figure 1-8

- **Toolbox:** The Toolbox contains reusable controls and components that can be added to your application. These can range from buttons to data connectors to customized controls that you have either purchased or developed.
- **Design window:** The Design window is where a lot of the action takes place. This is where you will draw your user interface on your forms. This window is sometimes referred to as the Designer.
- **Solution Explorer:** The Solution Explorer window contains a hierarchical view of your solution. A *solution* can contain many projects, whereas a *project* contains forms, classes, modules, and components that solve a particular problem.
- **Properties:** The Properties window shows what *properties* the selected object makes available. Although you can set these properties in your code, sometimes it is much easier to set them while you are designing your application (for example, drawing the controls on your form). You will notice that the File Name property has the value `Form1.vb`. This is the physical file name for the form's code and layout information.

Try It Out Creating a HelloUser Project (cont.)

1. Change the name of your form to something more indicative of what your application is. Click on `Form1.vb` in the Solution Explorer window. Then, in the Properties window, change the File Name property from `Form1.vb` to **HelloUser.vb** and press Enter, as shown in Figure 1-9. When changing properties you must either press Enter or click off the property for it to take effect.



Copy to Output D	Do not copy
Custom Tool	
Custom Tool Nan	
Misc	
File Name	HelloUser.vb
Full Path	C:\Documents an

File Name
Name of the file or folder.

Figure 1-9

2. Notice that the form's filename has also been updated in the Solution Explorer to read `HelloUser.vb`.

3. Now click the form displayed in the Design window. The Properties window will change to display the form's Form properties (instead of the File properties, which you have just been looking at). You will notice that the Properties window is dramatically different. The difference is the result of two different views of the same file. When the form name is highlighted in the Solution Explorer window, the physical file properties of the form are displayed. When the form in the Design window is highlighted, the visual properties and logical properties of the form are displayed.

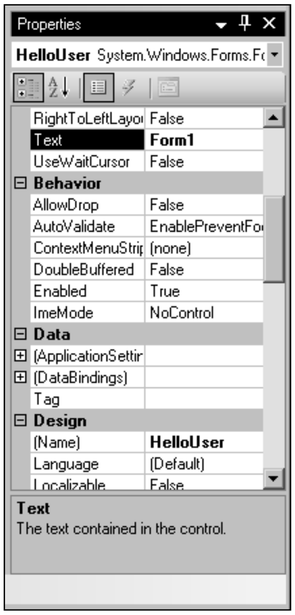
   The Properties window allows you to set a control's properties easily. Properties are a particular object's set of internal data; they usually describe appearance or behavior. In Figure 1-10 you can see that properties are grouped together in categories — Accessibility (not shown), Appearance (header is not shown), Behavior, Data, Design, Focus (not shown), Layout (not shown), Misc (not shown), and Window Style (not shown).

You can see that under the Appearance category (header not shown), even though we changed the file name of the form to `HelloUser.vb`*, the text or caption of the form is still* `Form1`*.*

4. Right now, the title (Text property) of your form (displayed in the bar at the top) is `Form1`. This is not very descriptive, so change it to reflect the purpose of this application. Locate the Text property in the Appearance section of the Properties window. Change the Text property's value to **Hello from Visual Basic 2005** and press Enter. Notice that the form's title has been updated to reflect the change.

If you have trouble finding properties, click the little AZ button on the toolbar toward the top of the Properties window. This changes the property listing from being ordered by category to being ordered by name.

5. You are now finished with the procedure. Click the Start button on the Visual Studio 2005 toolbar (the green triangle) to run the application. As you work through the book, whenever we say "run the project" or "start the project," just click the Start button. An empty window with the title Hello from Visual Basic 2005 is displayed.



Properties

HelloUser System.Windows.Forms.Fc

RightToLeftLayou	False
Text	**Form1**
UseWaitCursor	False
⊟ **Behavior**	
AllowDrop	False
AutoValidate	EnablePreventFo
ContextMenuStrip	(none)
DoubleBuffered	False
Enabled	True
ImeMode	NoControl
⊟ **Data**	
⊞ (ApplicationSettir	
⊞ (DataBindings)	
Tag	
⊟ **Design**	
(Name)	**HelloUser**
Language	(Default)
Localizable	False

Text
The text contained in the control.

Figure 1-10

That was simple, but your little application isn't doing much at the moment. Let us make it a little more interactive. To do this, you are going to add some controls — a label, a text box, and two buttons to the form. This will let you see how the Toolbox makes adding functionality quite simple. You may be wondering at this point when you will actually look at some code. Soon! The great thing about Visual Basic 2005 is that you can develop a fair amount of your application *without* writing any code. Sure, the code is still there, behind the scenes, but, as you will see, Visual Basic 2005 writes a lot of it for you.

The Toolbox

The Toolbox is accessed via the View ➪ Toolbox menu option, the Toolbox icon on the Standard menu bar, or by pressing Ctrl+Alt+X. Alternatively, the Toolbox tab is displayed on the left of the IDE; hovering your mouse over this tab will cause the Toolbox window to fly out, partially covering your form.

The Toolbox contains a Node type view of the various controls and components that can be placed onto your form. Controls such as text boxes, buttons, radio buttons, and combo boxes can be selected and then *drawn* onto your form. For the HelloUser application, you will be using only the controls in the Common Controls node. In Figure 1-11, you can see a listing of common controls for Windows Forms.

Controls can be added to your forms in any order, so it does not matter if you add the label control after the text box or the buttons before the label. In the next Try It Out, you start adding controls.

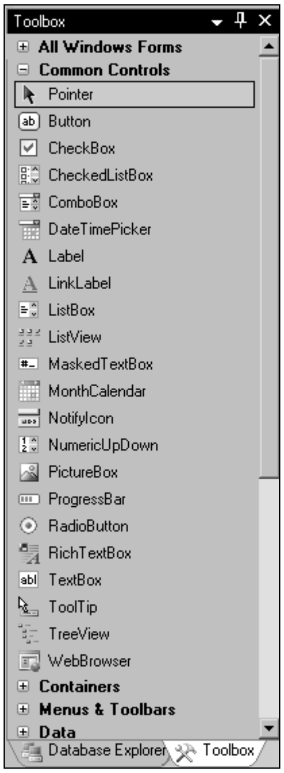


Figure 1-11

Try It Out Adding Controls to the HelloUser Application

1. Stop the project if it is still running, because you now want to add some controls to your form. The simplest way to stop your project is to click the _ button in the top-right corner of the form. Alternatively, you can click the blue square in the IDE (which displays a ToolTip that says “Stop Debugging” if you hover over it with your mouse pointer).

2. Add a Label control to the form. Click Label in the Toolbox and drag it over the form’s Designer and drop it in the desired location. (You can also place controls on your form by double-clicking on the required control in the Toolbox or clicking on the control in the Toolbox and then drawing it on the form.)

3. If the Label control you have just drawn is not in the desired location, it really isn’t a problem. Once the control is on the form, you can resize it or move it around. Figure 1-12 shows what the control looks like after you place it on the form. To move it, click the dotted border and drag it to the desired location. The label will automatically resize itself to fit the text that you enter in the Text property.

4. After drawing a control on the form, you should at least configure its name and the text that it will display. You will see that the Properties window to the right of the Designer has changed to `Label1`, telling you that you are currently examining the properties for it. In the Properties

window, set your new label's Text property to **Enter Your Name**. Notice that, once you press enter or click off of the property, the label on the form has automatically resized itself to fit the Text property. Now set the Name property to **lblName**.

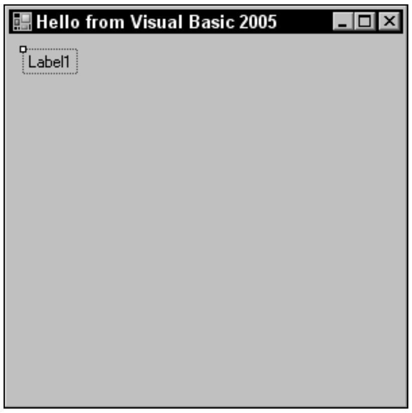


Figure 1-12

5. Now, directly beneath the label, you want to add a text box, so that you can enter a name. You are going to repeat the procedure you followed for adding the label, but this time make sure you select the TextBox control from the toolbar. After you have dragged-and-dropped (or double-clicked) the control into the appropriate position as shown in Figure 1-13, use the Properties window to set its Name property to **txtName**.

Notice the sizing handles on the left and right side of the control. You can use these handles to resize the text box horizontally.

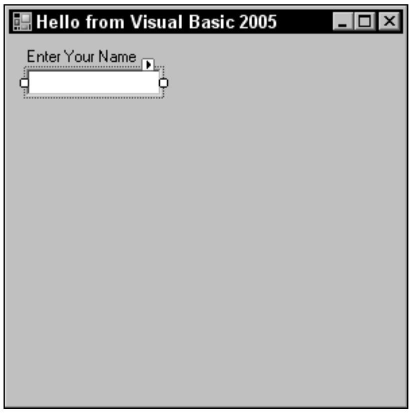


Figure 1-13

6. In the bottom left corner of the form, add a Button control in exactly the same manner as you added the label and text box. Set its Name property to **btnOK** and its Text property to **&OK**. Your form should now look similar to the one shown in Figure 1-14.

The ampersand (&) is used in the Text property of buttons is to create a keyboard shortcut (known as a hot key). The letter with the & sign placed in front of it will become underlined (as shown in Figure 1-14) to signal users that they can select that button by pressing the Alt+letter key combination, instead of using the mouse (on some configurations the underline doesn't appear until the user presses Alt). In this particular instance, pressing Alt+O would be the same as clicking directly on the OK button. There is no need to write code to accomplish this.

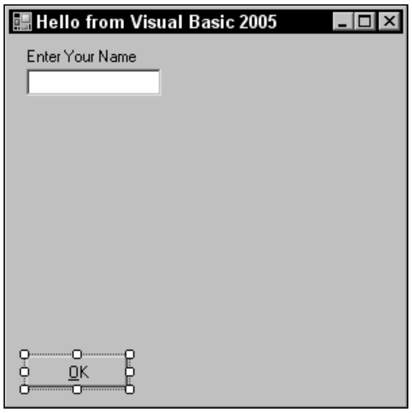


Figure 1-14

7. Now add a second Button control to the bottom right corner of the form by dragging the Button control from the Toolbox onto your form. You'll notice that, as you get close to the bottom right of the form, a blue snap line will appear, as shown in Figure 1-15. This snap line will allow you to align this new Button control with the existing Button control on the form. The snap lines assist you in aligning controls to the left, right, top, or bottom of each other, depending on where you are trying to position the new control. The light blue line provides you with a consistent margin between the edge of your control and the edge of the form. Set the Name property to **btnExit** and the Text property to **E&xit**. Your form should look similar to Figure 1-16.

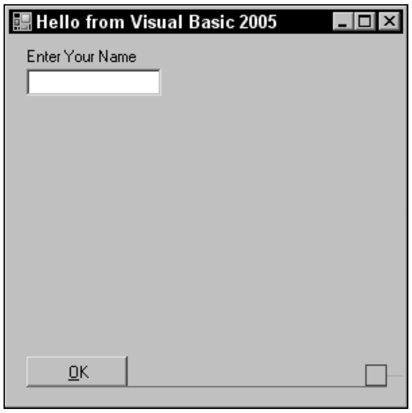


Figure 1-15

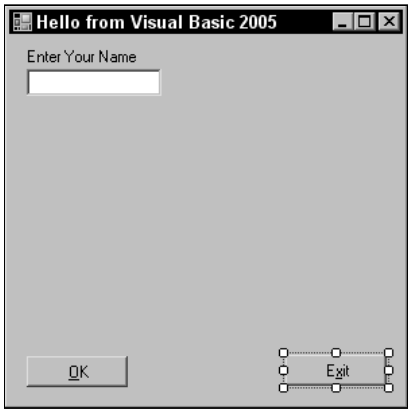


Figure 1-16

Now before you finish your sample application, let us briefly discuss some coding practices that you should be using.

Modified Hungarian Notation

You may have noticed that the names given to the controls look a little funny. Each name is prefixed with a shorthand identifier describing the type of control it is. This makes it much easier to understand what type of control you are working with when you are looking through the code. For example, say you had a control called simply Name, without a prefix of `lbl` or `txt`. You would not know whether you were working with a text box that accepted a name or with a label that displayed a name. Imagine if, in the previous Try It Out, you had named your label Name1 and your text box Name2 — you would very quickly become confused. What if you left your application for a month or two and then came back to it to make some changes?

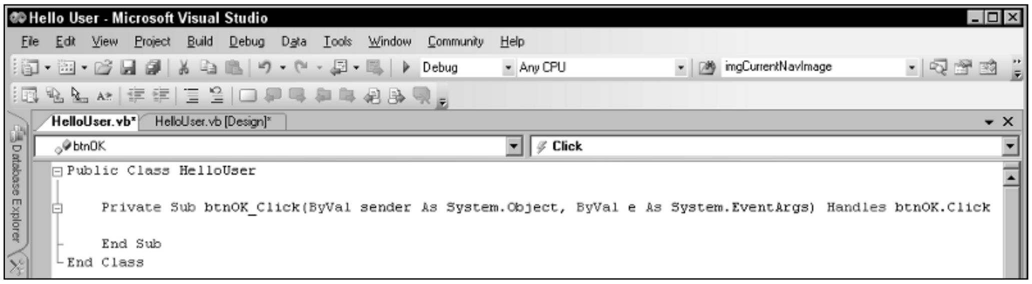
When working with other developers, it is very important to keep the coding style consistent. One of the most commonly used styles for control names within application development in many languages was brought forth by Dr. Charles Simonyi, who worked for the Xerox Palo Alto Research Center (XPARC) before joining Microsoft. He came up with short prefix mnemonics that allowed programmers to easily identify the type of information a variable might contain. Since Simonyi is from Hungary, and the prefixes make the names look a little foreign, the name "Hungarian Notation" came into use for this system. Because the original notation was used in C/C++ development, the notation for Visual Basic 2005 is termed Modified. The following table shows some of the commonly used prefixes that you shall be using in this book.

Hungarian Notation can be a real time-saver when you are looking at code someone else wrote or at code that you wrote months earlier. However, by far the most important thing is to be consistent in your naming. When you start coding, pick a convention for your naming. It is recommended that you use the de facto standard Modified-Hungarian for Visual Basic 2005, but it is not required. Once you pick a convention, stick to it. When modifying others' code, use theirs. A standard naming convention followed throughout a project will save countless hours when the application is maintained. Now let's get back to the application. It's now time to write some actual code.

Control	Prefix
Button	`btn`
ComboBox	`cbo`
CheckBox	`chk`
Label	`lbl`
ListBox	`lst`
MainMenu	`mnu`
RadioButton	`rdb`
PictureBox	`pic`
TextBox	`txt`

The Code Editor

Now that you have the HelloUser form defined, you have to add some code to make it actually do something interesting. You have already seen how easy it is to add controls to a form. Providing the functionality behind those on-screen elements is no more difficult. To add the code for a control, you just double-click the control in question. This will open the code editor in the main window, shown in Figure 1-17.


Hello User - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Debug
Any CPU
imgCurrentNavImage
HelloUser.vb*
HelloUser.vb [Design]*
btnOK
Click
Database Explorer

```
Public Class HelloUser

    Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnOK.Click

    End Sub
End Class
```

Figure 1-17

Notice that an additional tab has been created in the main window. Now you have the Design tab and the Code tab. You draw the controls on your form in the Design tab, and you write code for your form in the Code tab. One thing to note here is that Visual Studio 2005 has created a separate file for the code. The visual definition and the code behind exist in separate files: `HelloUser.Designer.vb` and `Hello User.vb`. This is actually the reason why building applications with Visual Basic 2005 is so slick and easy. Using the Design view you can visually lay out your application, and then, using Code view, you add just the bits of code to implement your desired functionality.

You will also notice that there are two combo boxes at the top of the window. These provide shortcuts to the various parts of your code. Hover your mouse on the combo box on the left, and you'll see a ToolTip appear, telling you that it is the Class Name combo box. If you expand this combo box, you will see a list of all the objects within your application. If you hover your mouse on the combo box on the right, you'll see a ToolTip telling you that this is the Method Name combo box. If you expand this combo box, you will see a list of all defined functions and subroutines for the object selected in the Class Name combo box. If this particular form had a lot of code behind it, these combo boxes would make navigating to the desired area very quick — jumping to the selected area in your code. However, since all of the code fits in the window, there are not a lot of places to get lost.

Try It Out Adding Code to the HelloUser Project

1. To begin adding the necessary code, click the Design tab to show the form again. Then double-click the OK button. The code window will open with the following code. This is the shell of the button's Click event and is the place where you enter the code that you want to run when you click the button. This code is known as an *event handler* and sometimes is also referred to as an *event procedure:*

```
Private Sub btnOK_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnOK.Click

End Sub
```

As a result of the typographic constraints in publishing, it is not possible to put the Sub declaration on one line. Visual Basic 2005 allows you to break up lines of code by using the underscore character (_) to signify a line continuation. The space before the underscore is required. Any whitespace preceding the code on the following line is ignored.

`Sub` is an example of a *keyword*. In programming terms, a keyword is a special word that is used to tell Visual Basic 2005 to do something special. In this case, it tells Visual Basic 2005 that this is a *subroutine*, a procedure that does not return a value. Anything that you type between the lines `Private Sub` and `End Sub` will make up the event procedure for the OK button.

2. Now add the highlighted code into the procedure:

```
Private Sub btnOK_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnOK.Click

    'Display a message box greeting the user
    MessageBox.Show("Hello," & txtName.Text & _
      "! Welcome to Visual Basic 2005.", _
      "Hello User Message")
End Sub
```

Throughout this book, you will be presented with code that you should enter into your program if you are following along. Usually, we will make it pretty obvious where you put the code, but as we go, we will explain anything that looks out of the ordinary. The code with the gray background is code that you should enter.

3. After you have added the preceding code, go back to the Design tab, and double-click the Exit button. Add the highlighted code to the `btnExit_Click` event procedure.

```
Private Sub btnExit_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnExit.Click

    'End the program and close the form
    Me.Close()
End Sub
```

You may be wondering what `Me` is. `Me` is a keyword that refers to the form. Just like the pronoun *me*, it is just shorthand for referring to one's self.

4. Now that the code is finished, the moment of truth has arrived and you can see your creation. First though, save your work by using File ➪ Save HelloUser.vb from the menu or by clicking the Save button on the toolbar.

5. Now click the Start button on the toolbar. You will notice a lot of activity in the Output window at the bottom of your screen. Provided you have not made any mistakes in entering the code, this information just lets you know which files are being loaded to run your application.

   It is at this point that Visual Studio 2005 will *compile* the code. Compiling is the activity of taking the Visual Basic 2005 source code that you have written and translating it into a form that the computer understands. After the compilation is complete, Visual Studio 2005 runs (also known as *executes*) the program, and you'll be able to see the results.

If Visual Basic 2005 encounters any errors, they will be displayed as tasks in the Task List window. Double-clicking a task transports you to the offending line of code. We will learn more about how to debug the errors in our code in Chapter 9.

6. When the application loads, you see the main form. Enter a name and click OK (or press the Alt+O key combination) (see Figure 1-18).

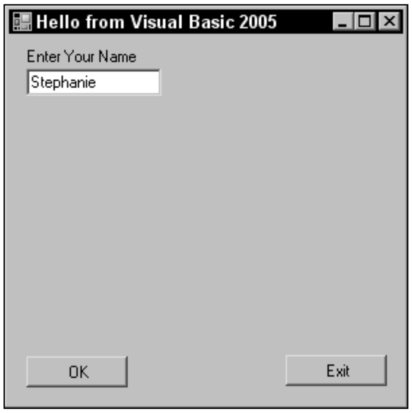
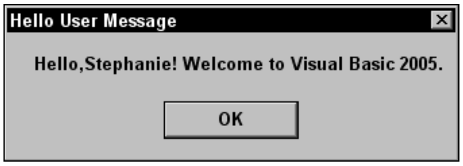


Figure 1-18

7. A window known as a message box appears, welcoming the person whose name was entered in the text box on the form — in this case Stephanie (see Figure 1-19).



Hello,Stephanie! Welcome to Visual Basic 2005.

Figure 1-19

8. After you close the message box by clicking the OK button, click the Exit button on your form. The application closes and you will be returned to the Visual Basic 2005 IDE.

How It Works

The code that you added to the Click event for the OK button will take the name that was entered in the text box and use it as part of the message that was displayed in Figure 1-19.

The first line of text entered in this procedure is actually a *comment*, text that is meant to be read by the human programmer who is writing or maintaining the code, not by the computer. Comments in Visual Basic 2005 begin with a single quote (`'`), and everything following on that line is considered a comment and ignored by the compiler. Comments will be discussed in detail in Chapter 3.

The `MessageBox.Show` method displays a message box that accepts various parameters. As used in your code, you have passed the string text to be displayed in the message box. This is accomplished through the *concatenation* of string constants defined by text enclosed in quotes. Concatenation of strings into one long string is performed through the use of the ampersand (`&`) character.

The code that follows concatenates a string constant of `"Hello,"` followed by the value contained in the Text property of the `txtName` text box control followed by a string constant of `"! Welcome to Visual Basic 2005."` The second parameter being passed to the `MessageBox.Show` method is the caption to be used in the title bar of the Message Box dialog box.

Finally, the underscore (_) character used at the end of the lines in the following code enables you to split your code onto separate lines. This tells the compiler that the rest of the code for the parameter is continued on the next line. This is really useful when building long strings, because it allows you to view the entire code fragment in the Code Editor without having to scroll the Code Editor window to the right to view the entire line of code.

```
Private Sub btnOK_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnOK.Click

    'Display a message box greeting the user
    MessageBox.Show("Hello," & txtName.Text & _
      "! Welcome to Visual Basic 2005.", _
      "Hello User Message")
End Sub
```

The next procedure that you added code for was the Exit button's Click event. Here you simply enter the code: `Me.Close()`. As explained earlier, the `Me` keyword refers to the form itself. The `Close` method of the form closes the form and releases all resources associated with it, thus ending the program.

```
Private Sub btnExit_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnExit.Click

    'End the program and close the form
    Me.Close()
End Sub
```

Using the Help System

The Help system included in Visual Basic 2005 is an improvement over Help systems in previous versions. As you begin to learn Visual Basic 2005, you will probably become very familiar with the Help system. However, it is worthwhile to give you an overview, just to help speed your searches for information.

The Help menu contains the menu items shown in Figure 1-20:

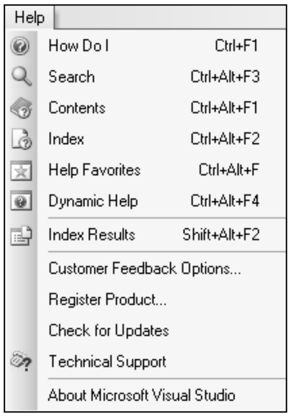


Figure 1-20

As you can see, this menu contains a few more entries than the typical Windows application. The main reason for this is the vastness of the documentation. Few people could keep it all in their heads — but luckily, that is not a problem, because you can always quickly and easily refer to the Help system. Think of it as a safety net for your brain.

One really fantastic feature is Dynamic Help. When you select the Dynamic Help menu item from the Help menu, the Dynamic Help window is displayed with a list of relevant topics for whatever you may be working on. The Dynamic Help window can be displayed by clicking Help ➪ Dynamic Help on the menu bar. The Dynamic Help window is then displayed as a tab behind the Properties window.

Let us say, for example, that you are working with a text box (perhaps the text box in the HelloUser application) and want to find out some information; you just select the text box on our form or in the code window and you can see all the help topics that pertain to text boxes, as shown in Figure 1-21

The other help commands in the Help menu (Search, Contents, and Index), function just as they would in any other Windows application. The How Do I menu item displays the Visual Studio Help collection with a list of common tasks that are categorized. This makes finding help on common tasks fast and efficient.

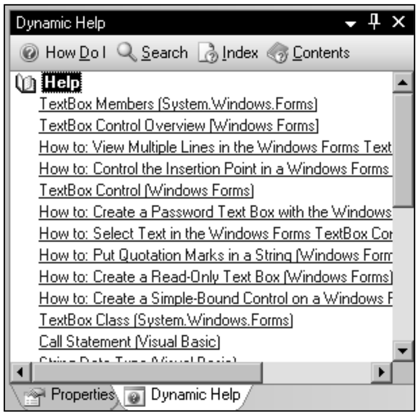

Dynamic Help
How Do I
Search
Index
Contents
Help
TextBox Members (System.Windows.Forms)
TextBox Control Overview (Windows Forms)
How to: View Multiple Lines in the Windows Forms Text
How to: Control the Insertion Point in a Windows Forms
TextBox Control (Windows Forms)
How to: Create a Password Text Box with the Windows
How to: Select Text in the Windows Forms TextBox Cor
How to: Put Quotation Marks in a String (Windows Form
How to: Create a Read-Only Text Box (Windows Forms)
How to: Create a Simple-Bound Control on a Windows F
TextBox Class (System.Windows.Forms)
Call Statement (Visual Basic)
Properties
Dynamic Help

Figure 1-21

Summary

Hopefully, you are beginning to see that developing basic applications with Visual Basic 2005 is not that difficult. You have taken a look at the IDE and saw how it can help you put together software very quickly. The Toolbox enables you to add controls to your form and design a user interface very quickly and easily. The Properties window makes configuring those controls a snap, while the Solution Explorer gives you a bird's eye view of the files that make up your project. You even wrote a little code.

In the coming chapters, you will go into even more detail and get comfortable writing code. Before you go too far into Visual Basic 2005 itself, the next chapter will give you an introduction to the Microsoft .NET Framework. This Framework is what gives all of the .NET languages their ease of use, ease of interoperability, and simplicity in learning.

To summarize, you should now be familiar with:

- ❑ The Integrated Development Environment (IDE)
- ❑ Adding controls to your form in the Designer
- ❑ Setting the properties of your controls
- ❑ Adding code to your form in the code window

Exercise

Create a Windows Application with a Textbox and Button control that will display whatever is typed in the text box when the user clicks on the button.

The answers for this exercise and those at the end of all the other chapters can be found in Appendix D.