Beginning Visual Basic 2005 Express Edition

From Novice to Professional

Peter Wright

Apress®

Beginning Visual Basic 2005 Express Edition: From Novice to Professional

Copyright © 2006 by Peter Wright

ISBN-13 (pbk): 978-1-59059-622-7

ISBN-10 (pbk): 1-59059-622-6

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Jim Sumser
Technical Reviewer: Jason Bock
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Keir Thomas, Matt Wade
Associate Publisher, Production Director, and Project Manager: Grace Wong
Copy Edit Manager: Nicole LeClerc
Copy Editors: Sharon Wilkey, Ami Knox
Assistant Production Director: Kari Brooks-Copony
Senior Production Editor: Laura Cheu
Compositor: Pat Christenson
Proofreader: Nancy Riddiough
Indexer: Broccoli Information Management
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code section. You will need to answer questions pertaining to this book in order to successfully download the code.

CHAPTER 1

Welcome to Visual Basic Express

Visual Basic is the original Microsoft Windows Rapid Application Development (RAD) tool. When it first hit the market way back in '91, it started a revolution in how people write computer programs.

Prior to Visual Basic (VB) arriving, writing a program to run on Windows—complete with all the bells and whistles of the Windows graphical user interface—was an exercise in pain. Windows is, after all, a hideously complex beast to work with in code. Visual Basic, though, simplified the whole thing. If you wanted a window with a button in it, all you had to do was drag and drop controls from a Toolbox onto a window the program gave you, and you were finished.

Visual Basic 2005 Express carries this tradition forward. It's just as easy today to write programs for the very latest versions of Windows as it was back then to create programs for Windows 3.0. Visual Basic 2005, though, while strikingly similar to classic Visual Basic in many areas, is radically different in others. The language has evolved and is now a truly *object-oriented* language. Because Visual Basic is now also a .NET-enabled language, when you sit down to write your Visual Basic masterpieces today, you have the full backing and power of Microsoft's legendary .NET Framework at your disposal. Of all the Express tools, I still feel happiest in Visual Basic 2005 Express (or VB Express—I use the names interchangeably). It's the most descriptive language to use in many cases, bearing more than a passing resemblance to English in terms of its syntax and structure.

In this chapter I'm going to set the stage a little. If you've never programmed before and you've already installed and taken a look at VB Express, you may feel a little daunted by all the strange icons, words, and images that the user interface has plastered all over it. I'll demystify it all for you in this chapter.

If you're an old hand, perhaps an accomplished Visual Basic or Java developer, or perhaps a .NET developer looking to learn new things with Express, this is the chapter where you'll see some of the most obvious and stunning changes that Microsoft has made to its development environments in the Express tools. A lot of the functionality in Visual Basic 2005 Express comes from Visual Studio .NET 2005, so you'll get a glimpse into just what that product can provide, if perhaps you are viewing it as a target for the future.

Whoever, and whatever, you are though, this is the chapter where I hope I can show you just some of why VB Express is, in my mind, one of the most significant product releases Microsoft has ever made.

Visual Basic has had a bad rap since the release of .NET, with many people calling classic Visual Basic a toy, an amateur programming environment that's great for prototyping ideas, but not really that great when it comes to producing high-performance, easy-to-maintain applications of any complexity. That's ignoring something vital, though. Visual Basic was designed to make Windows programming accessible to everyone. It didn't matter whether you were a professional programmer, a graduate, a high-school dropout, or a retired garbage collector, Visual Basic was designed to put everyone on an even playing field when it came to making great-looking, functional software.

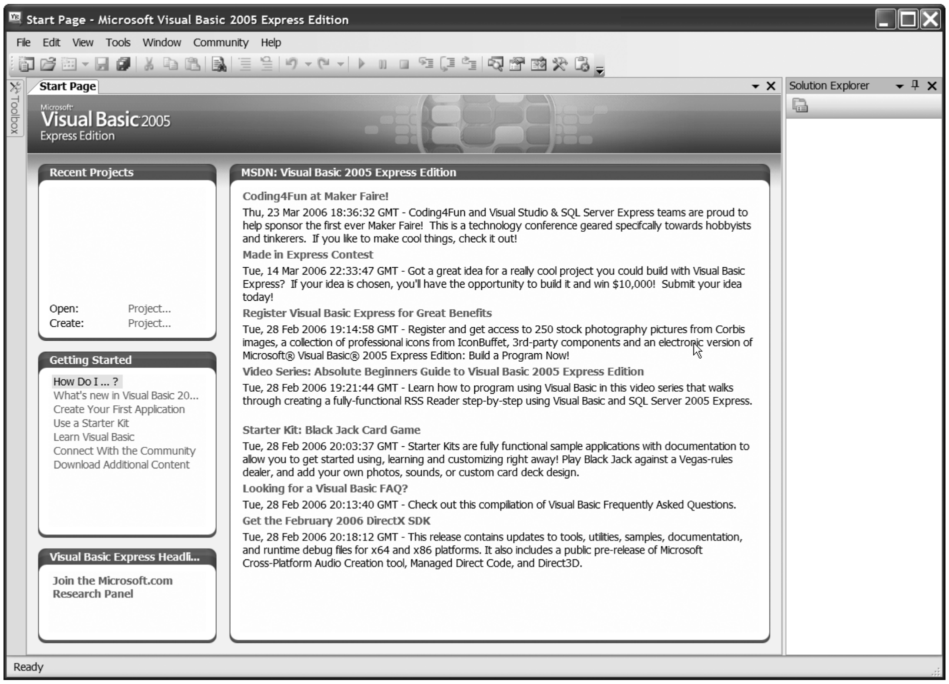
In addition, Visual Basic was an extremely focused piece of software. Visual Basic let you do one thing (create Windows applications) and do it very well indeed. It was only later in Visual Basic's life that it was integrated into "Visual Studio," and as a result had access to facilities for creating server-side components and web applications.

Because of its easy-to-use features and its inherent goal of focusing on doing just one thing, and doing it very well, Visual Basic brought a few million new developers to the world of Windows, and helped not only propel Windows even further into the hearts and minds of millions of people all over the world, but also set the benchmark for just what writing a computer program should really be like. You only have to look around the market today at products such as Delphi, JBuilder, C#Builder, and of course Visual Studio .NET to instantly spot similarities between those tools and good old-fashioned classic Visual Basic.

You can see the warm welcoming UI of Visual Basic 2005 Express in Figure 1-1.

Visual Basic 2005 Express is the result of the years of experience Microsoft has had with VB as a whole. Everything that made Visual Basic great is still there. You can still rapidly build user interfaces for your applications just by dragging and dropping controls. The programming language is still beautifully descriptive and easy to read (if not fully understood) by all. Therefore, Visual Basic 2005 (the language) is perhaps the least error prone of all the languages Microsoft supports, particularly for beginners.

Conversely, all the arguments that were ever leveled at Visual Basic have been addressed. Visual Basic 2005 is a .NET language. You write code in VB and then compile it. When it's compiled, the compiler spits out Microsoft Intermediate Language (MSIL), an intermediary language that *all* the .NET compilers (yes, including the C++ and C# ones) produce. The net result is that Visual Basic programs now run at pretty much exactly the same speed as their C++ .NET and C# counterparts. In fact, a common selling point of .NET (which applies to VB now) is that .NET programs in general can be faster than pure C or C++ written ones because at runtime the .NET system will optimize the code for the processor in your machine. Most classic C and C++ compilers, on the other hand, will target a base processor compatible with all machines to let the programs run on the widest possible range of hardware.


Start Page - Microsoft Visual Basic 2005 Express Edition
File Edit View Tools Window Community Help
Toolbox
Start Page
Solution Explorer
Microsoft Visual Basic 2005 Express Edition
Recent Projects
Open: Project...
Create: Project...
Getting Started
How Do I ... ?
What's new in Visual Basic 20...
Create Your First Application
Use a Starter Kit
Learn Visual Basic
Connect With the Community
Download Additional Content
Visual Basic Express Headli...
Join the Microsoft.com Research Panel
MSDN: Visual Basic 2005 Express Edition
Ready

Coding4Fun at Maker Faire!
Thu, 23 Mar 2006 18:36:32 GMT - Coding4Fun and Visual Studio & SQL Server Express teams are proud to help sponsor the first ever Maker Faire! This is a technology conference geared specifcally towards hobbyists and tinkerers. If you like to make cool things, check it out!

Made in Express Contest
Tue, 14 Mar 2006 22:33:47 GMT - Got a great idea for a really cool project you could build with Visual Basic Express? If your idea is chosen, you'll have the opportunity to build it and win $10,000! Submit your idea today!

Register Visual Basic Express for Great Benefits
Tue, 28 Feb 2006 19:14:58 GMT - Register and get access to 250 stock photography pictures from Corbis images, a collection of professional icons from IconBuffet, 3rd-party components and an electronic version of Microsoft® Visual Basic® 2005 Express Edition: Build a Program Now!

Video Series: Absolute Beginners Guide to Visual Basic 2005 Express Edition
Tue, 28 Feb 2006 19:21:44 GMT - Learn how to program using Visual Basic in this video series that walks through creating a fully-functional RSS Reader step-by-step using Visual Basic and SQL Server 2005 Express.

Starter Kit: Black Jack Card Game
Tue, 28 Feb 2006 20:03:37 GMT - Starter Kits are fully functional sample applications with documentation to allow you to get started using, learning and customizing right away! Play Black Jack against a Vegas-rules dealer, and add your own photos, sounds, or custom card deck design.

Looking for a Visual Basic FAQ?
Tue, 28 Feb 2006 20:13:40 GMT - Check out this compilation of Visual Basic Frequently Asked Questions.

Get the February 2006 DirectX SDK
Tue, 28 Feb 2006 20:18:12 GMT - This release contains updates to tools, utilities, samples, documentation, and runtime debug files for x64 and x86 platforms. It also includes a public pre-release of Microsoft Cross-Platform Audio Creation tool, Managed Direct Code, and Direct3D.

Figure 1-1. *The clean, welcoming user interface of Visual Basic 2005 Express*

Visual Basic 2005 is also a fully object-oriented (OO) language now. If that means nothing at all to you, don't worry. We'll go into the full details of object-oriented programming starting in Chapter 4. There is nothing that C# and the other OO languages can do now that VB can't. In fact, in some instances Visual Basic makes life easier. For example, if you are writing a Microsoft Office automation program, VB is the best choice. Why? Well, many of the components of Office still expect variable-length parameter lists to be passed to them, and VB is still the best language on the planet for doing that. I'm getting a little ahead of myself here, but the time will come when a friend or colleague will denigrate Visual Basic and you for learning it. When that time comes, you'll remember this paragraph.

In short, Visual Basic is now a completely modern, high-performance language. In Microsoftspeak it's also a first-class .NET language and fully able to use and take advantage of all that the .NET Framework has to offer.

Just What Is Express?

Express is the name given by Microsoft to a range of entry-level .NET 2.0 development tools. Each tool (there are six in all) is focused on allowing you to learn how to develop one specific kind of application. For example, Visual Web Developer 2005 Express is focused on developing web applications. Visual Basic 2005 Express and Visual C# 2005 Express are both focused on producing standard Windows-style applications, either with the Visual Basic or C# programming language.

Each tool also includes a lot of the tools and technologies that you can find in the full Visual Studio 2005 package. Visual C# Express, for example, includes some fantastic tools for restructuring the code in your programs (a process called *refactoring*). The user interface of all the Express products also have a lot in common with all previous versions of Visual Studio .NET, as well as the new Visual Studio 2005.

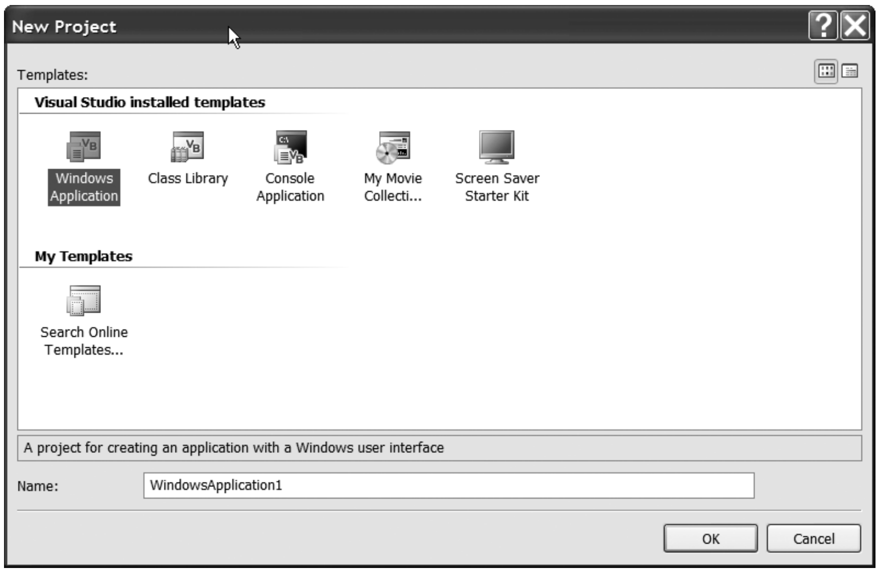
The best way to learn everything the package can do, and to get comfortable with it, is to use it. So, if you haven't installed Visual Basic 2005 Express already, now is the time to do so.

Exploring the Visual Basic 2005 Express IDE

It's a tired tradition that the first program you write when learning a new programming language or tool is "Hello, World!" Traditionally it has been a great way to become familiar with how to write your program's code, figure out how to display something on the screen, compile the code, and then run the resulting program. Visual Basic 2005 Express makes programming so easy that this little exercise is almost a no-brainer. In fact, in Charles Petzold's book *Programming Windows* (Microsoft Press, 1998), Charles had us write a program that displays a window, puts "Hello, World" in the center of it, and then made the text always stay in the center of the window no matter where the user moved it or resized it. The resulting code was around 80 lines. Let's do the same thing in VB Express.

Try It Out: Hello, World, VB Express Style

First, open Visual Basic 2005 Express. When the welcome screen appears (you saw what this looked like in Figure 1-1), click File ➤ New Project on the menu bar. The New Project dialog box appears, just as in Figure 1-2.



A project for creating an application with a Windows user interface

Name: WindowsApplication1

Figure 1-2. *The New Project dialog box*

If you've used Visual Studio .NET before, you'll probably be surprised by just how few options appear. As I said earlier, VB Express is focused on doing just one thing, very well.

For now, just click Windows Application and then click the OK button.

After a bit of a pause (how long you wait depends on how powerful your machine is), you'll be dropped into Visual Basic's form-editing mode. You can see this in Figure 1-3.

Don't panic if your screen looks a little different from mine. The user interface of the IDE (integrated development environment—the thing you should be looking at right now) is highly customizable, so chances are that if you've already been playing around with it, it may look slightly different.

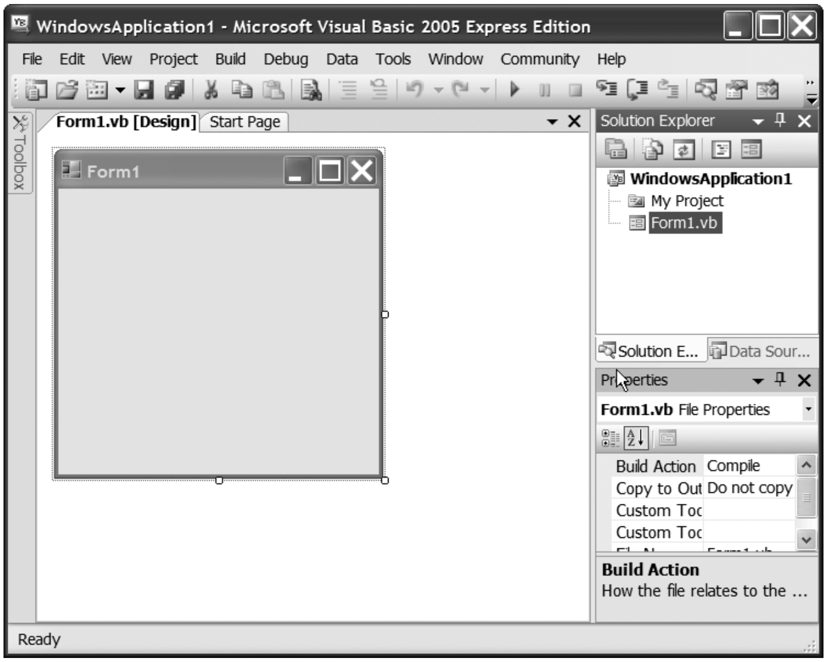
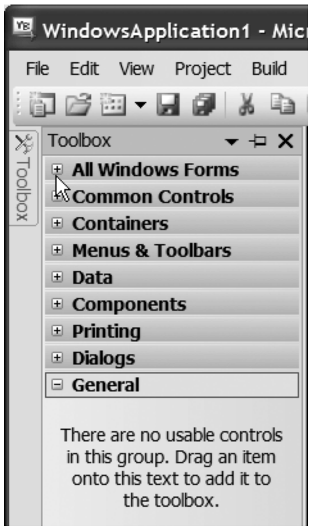


Figure 1-3. *The form editor in Visual Basic 2005 Express*

What you are actually looking at here is the form that represents the main window of your application. You candrag and drop controls from the Toolbox on the left onto the surface of the window to build up a nice professional-looking user interface for your application. The Properties window on the right lets you customize those controls to give your application a unique look and feel, and also provides options that relate to the code that you'll need to write to get a more complex application off the ground.

If your Toolbox is not showing (mine isn't in Figure 1-3), the first thing you'll need to do is display it. Move the mouse over the word *Toolbox* on the left side of the IDE and you'll see the Toolbox slide out, as in Figure 1-4.

When you move the mouse out of the Toolbox area, the pane will slide shut again. To prevent this from happening (some people like it that way, some don't), click the pushpin at the top of the pane, to the left of the Close icon (the X), to lock the Toolbox open. Finally, click on the plus symbol (+) next to the words "Common Controls" to display the list of the most common controls. The Toolbox will look like Figure 1-5 when you do this.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Figure 1-4. *If the Toolbox is not showing, just hover the mouse over the Toolbox tab on the left of the IDE.*

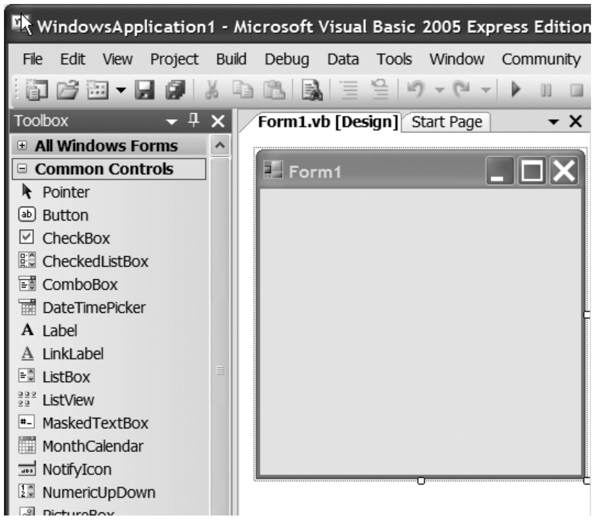


Figure 1-5. *The controls in the Toolbox are grouped. Clicking the + sign next to each group's name expands the group to show the controls it contains.*

Move your mouse over the Label control in the Toolbox, and then drag and drop it onto the form (a window in design mode is called a *form*). Your form should look a lot like Figure 1-6 when you're finished.

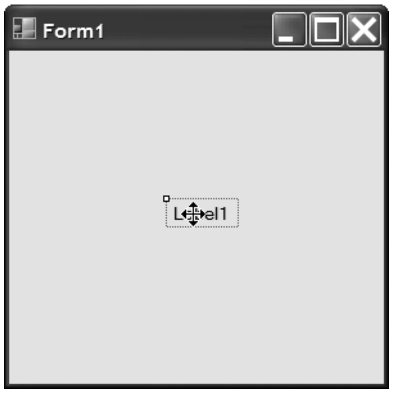


Figure 1-6. *Your form with the label on*

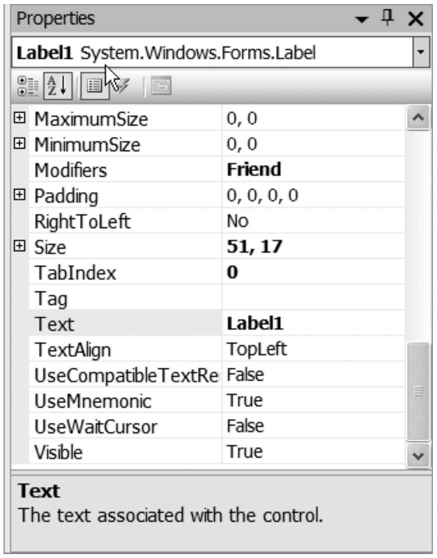
Looking good so far; you've created a form that at runtime will become a window, and it has some text in it. Best of all, you haven't written any code yet, so let's carry on.

The Properties window on the right of the IDE allows you to customize pretty much anything in your application. Click on the label you just dropped onto the form and you'll see the text at the top of the Properties window change to show that you are now looking at the properties of the Label control that you just added, as in Figure 1-7.

Properties are easy to understand, and in fact bear a lot in common with the real world. Take me, for example. I'm a pale and pasty Englishman with black hair. You could say that my SkinColor property (property names don't have spaces in them) is White, my HairColor property is Black, and my Name property is Pete.

In the case of our Label control, a couple of its properties are quite obvious. Its name is `label1` (you can see this at the top of the Properties list), because it's the first Label control that you have dropped on the form. The text that you can see inside the label on the form is the `Text` property, and its value is also `label1`. You'll need this text to show the message "Hello, World," so obviously you'll need to change that `Text` property.

Scroll the Properties list down until you can see the `Text` property, click it, and type in **Hello, World**. The property should look like Figure 1-8 when you're finished.



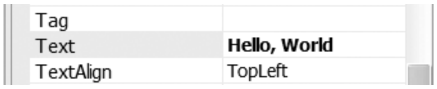
Properties

Label1 System.Windows.Forms.Label

⊞ MaximumSize	0, 0
⊞ MinimumSize	0, 0
Modifiers	**Friend**
⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**51, 17**
TabIndex	**0**
Tag	
Text	**Label1**
TextAlign	TopLeft
UseCompatibleTextRe	False
UseMnemonic	True
UseWaitCursor	False
Visible	True

Text
The text associated with the control.

Figure 1-7. *The top of the Properties window always shows you the object that you are working with.*



Tag	
Text	**Hello, World**
TextAlign	TopLeft

Figure 1-8. *Change the Text property from label1 to Hello, World.*

So far, so good. Now, Charles Petzold's application had the text always centered in the window, but you'll find that what you've done so far won't achieve that. To demonstrate, click the Run button on the toolbar at the top of the IDE (it looks like the Play symbol on a VCR or DVD player), or press F5 on your keyboard to do the same thing. A flurry of activity takes place while VB Express compiles the application and then runs it. You should now see a brand new window on screen, probably overlapping VB Express itself as mine did in Figure 1-9.

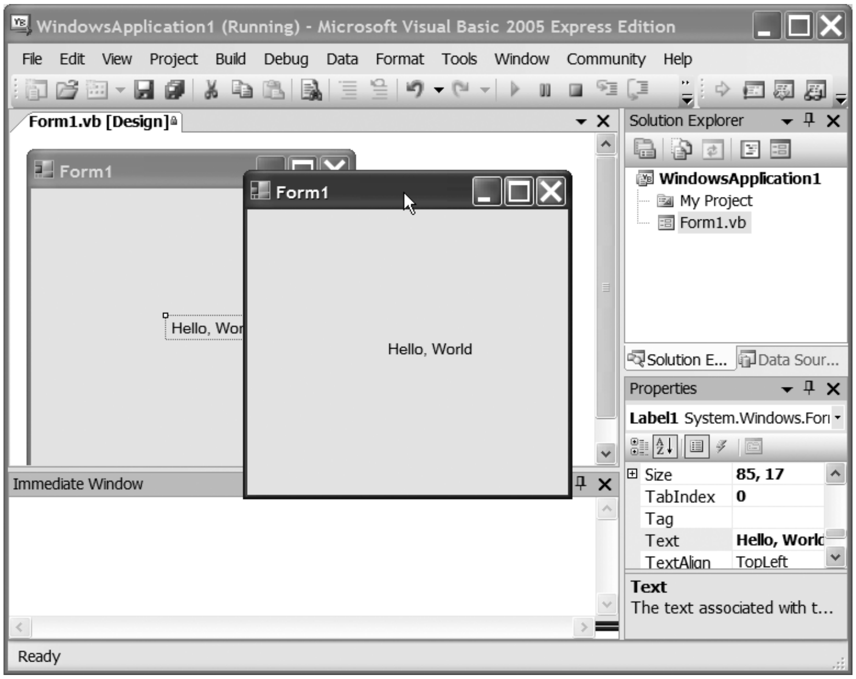

WindowsApplication1 (Running) - Microsoft Visual Basic 2005 Express Edition
File Edit View Project Build Debug Data Format Tools Window Community Help
Form1.vb [Design]
Form1
Hello, World
Solution Explorer
WindowsApplication1
My Project
Form1.vb
Solution E... Data Sour...
Properties
Label1 System.Windows.Fori
Size 85, 17
TabIndex 0
Tag
Text Hello, Worlc
TextAlign TopLeft
Text
The text associated with t...
Immediate Window
Ready

Figure 1-9. *The Hello, World program, running for the first time*

If you try making the window smaller or larger, you'll find that the text always remains in the same position and could even vanish if you made the window small enough. That needs to change. Close the window (by clicking its Close icon—the X) and you'll be returned to the VB Express IDE.

At this point, even in classic Visual Basic, you'd have to do a bunch of typing to get that text to stay centered on the form. Not so anymore. Click on the label once to make sure it's selected and then take another look at the Properties window.

First, scroll up to find the property called `AutoSize`. This is set to `True`, which means that the label will always be just big enough to hold all the text inside. No bigger, no smaller. Double-click the word `True` next to the property name to set it to `False`.

Now find the `Dock` property. `Dock` lets you lock a control, like our label, to a specific position on the form. You want the label to always be the same size as the form. If you click on the word `None` next to the `Dock` property name, you'll see a down arrow appear. Click it, and you'll see a Dock property editor appear as in Figure 1-10.

Each box in the editor represents a position inside the form. Click on the middle one to make the label fill the entire form. You should see the `Dock` property value change to the word `Fill`, and the label will grow to take up the entire form. You can see this in Figure 1-11.

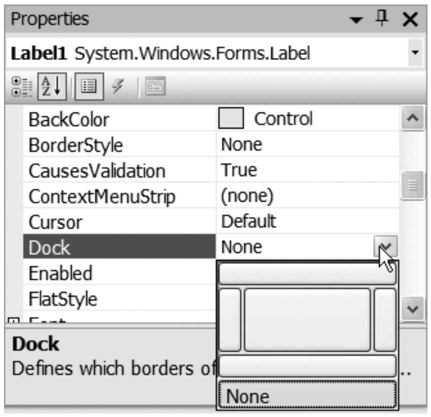


Figure 1-10. *The Dock property editor*

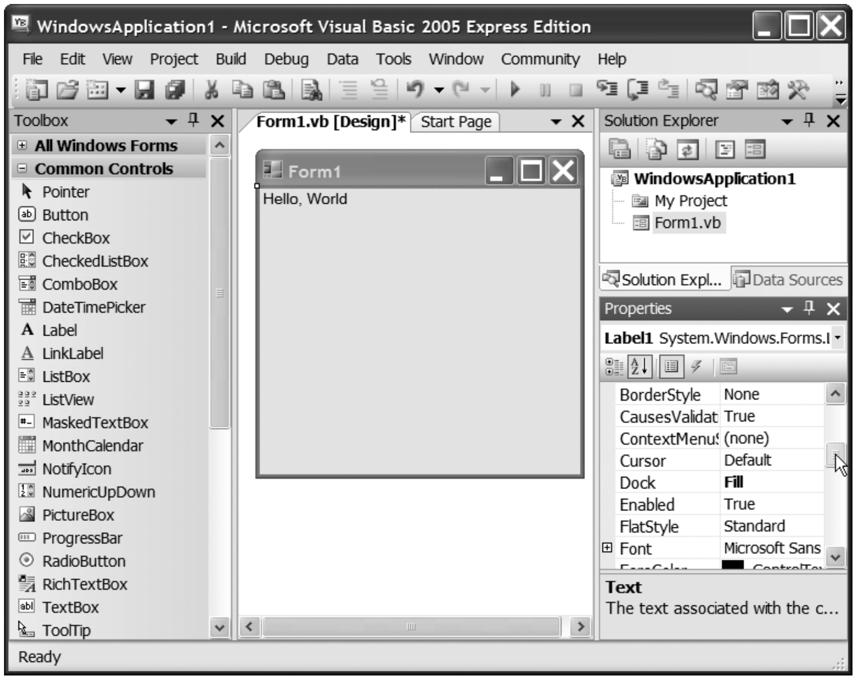
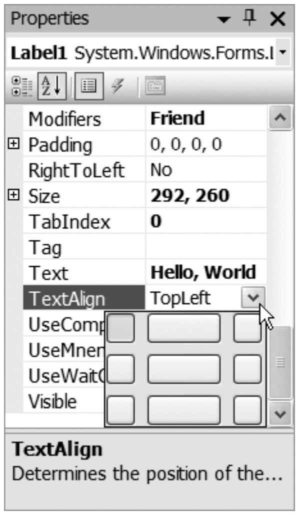


Figure 1-11. *The label should now fill the form (if you look closely, you'll see a subtle border inside the form itself).*

Now all that remains is to align the text properly within the now massive Label control. This is achieved in much the same way as setting up the `Dock` property. Find the `TextAlign` property, and just as you did with the `Dock` property, click the property value and then click the arrow that appears, as in Figure 1-12.



Modifiers	Friend
Padding	0, 0, 0, 0
RightToLeft	No
Size	292, 260
TabIndex	0
Tag	
Text	Hello, World
TextAlign	TopLeft
UseComp	
UseMner	
UseWait(	
Visible	

TextAlign
Determines the position of the...

Figure 1-12. *The TextAlign property editor works in a similar way to the Dock property editor.*

Again, click the middlemost button to center the text in the control, and then press F5 to run the program once again. This time the text remains centered, no matter how big or small you make the form. You just did, with a few mouse clicks, what traditionally would have taken quite some code to achieve.

It's a very simple example, but it does show how Visual Basic 2005 Express is totally geared toward making your life easier as a developer.

When you're finished exploring the example, close it from the File menu by selecting File ➤ Close Solution. A dialog box appears, asking whether you want to Save or Discard the project. Click Discard (unless of course you really want to save it to your disk for posterity).

ALREADY USED .NET?

If you've used .NET before, you'll know that creating simple projects to try things out was something of a pain. You'd create a new Windows application, for example, and call it `WindowsApplication1`, and Visual Studio would save that to your hard disk. This meant of course that if you created a bunch of simple example programs to test things out, you'd end up with a bunch of unused directories and files on your hard disk.

The Express tools and Visual Studio 2005 get around that problem with *temporary* projects. Any project you create is classed as a temporary project and will be discarded when you close it or shut down the IDE, unless you explicitly save it.

When using temporary projects, you can create as many test programs as you like without worrying about cluttering up your hard disk.

FOR THE VISUAL BASIC DEVELOPERS

If you did press Save instead of Discard at the end of the preceding “Try It Out,” you would have noticed that you were asked just two things: what you want to call the project, and where you want it stored. This is completely different from Visual Basic, in which saving a project resulted in a series of dialog boxes appearing and asking you to name every file in the project and the project itself.

The reason for this can be found in the Solution Explorer. When you first create a project, you are asked to give it a name. This name appears in the Solution Explorer. Similarly, each file you create is also assigned a name, which you can change by right-clicking the file in the Solution Explorer and choosing Rename. For example, our form from the earlier “Try It Out” is a file called `Form1.vb`.

When you save a project in Express, a directory is created for the project based on its name, and all the files in the project are saved into that directory by using the names shown in the Solution Explorer. Isn’t that so much easier?

Exploring the IDE a Little More

I’ve just scratched the surface of what the Visual Basic 2005 Express IDE can do for you. This is only Chapter 1, after all. So, let’s take a minute to look at some of the other cool features it has before you dive into a much bigger “Try It Out” example.

At the start of this section I mentioned that your IDE might look a little different from mine. Let me show you why (even if you have used Visual Studio before, keep watching—this is pretty cool).

Each of the things around the form editor (the Properties window, the Toolbox, and so on) is actually a docked window. You can make them hide and appear on demand, you can undock them and leave them floating around on the desktop, and you can even stick them to different edges of the IDE.

Try it—grab the title of the Properties window and drag it left. The screen changes to look like the one in Figure 1-13.

Notice all the arrow shapes. These let you tell VB Express exactly where you want the window that you’re dragging around docked. Simply drag over the appropriate arrow, and hey, presto, the window docks. Prior to this marvelous invention, I always found docking and moving windows to be a real pain in the neck; I could never position the floating window in just the right place for it to dock to where I really wanted it to end up.

You’ve probably already noticed that there are more windows docked around the edges of the IDE than just the Solution Explorer and the Properties window. In fact, if you tell the IDE to display every possible window, you soon end up with a huge mess of tabs everywhere. VB Express, like its big brother Visual Studio, has a ton of windows that you can use to gain different views into your application and the things going on with it.

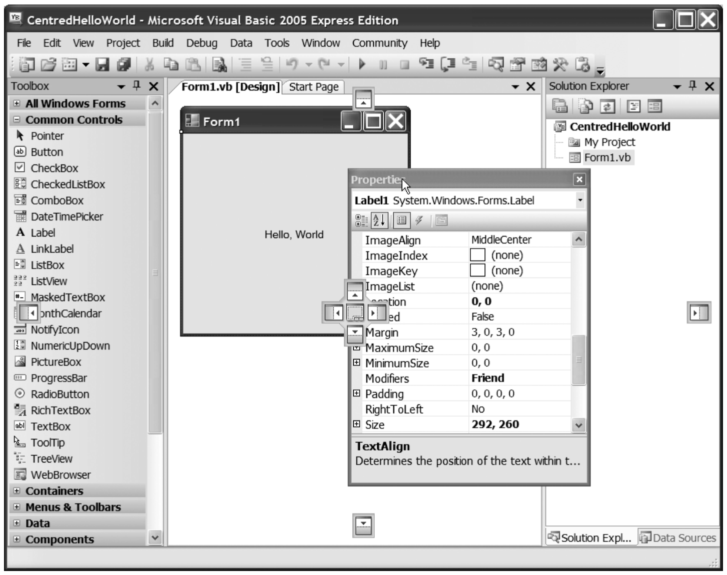

CentredHelloWorld - Microsoft Visual Basic 2005 Express Edition
File Edit View Project Build Debug Data Tools Window Community Help
Toolbox
Form1.vb [Design]
Start Page
Solution Explorer
Form1
Hello, World
Properties
Label1 System.Windows.Forms.Label
TextAlign
Determines the position of the text within t...

Figure 1-13. *The new docking arrows make it simple to put a floating window exactly where you want it.*

You can see the full list of windows by clicking the View menu at the top of the IDE, as shown in Figure 1-14.

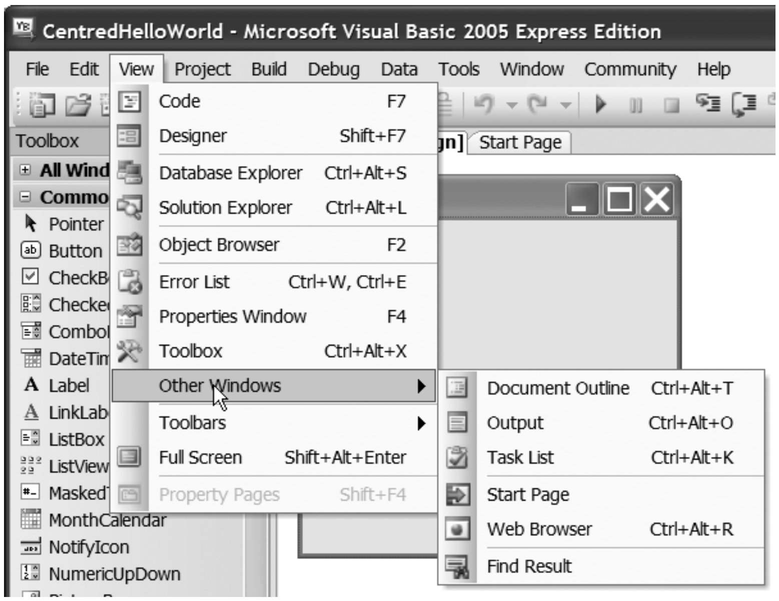

CentredHelloWorld - Microsoft Visual Basic 2005 Express Edition
File Edit View Project Build Debug Data Tools Window Community Help
Code F7
Designer Shift+F7
Database Explorer Ctrl+Alt+S
Solution Explorer Ctrl+Alt+L
Object Browser F2
Error List Ctrl+W, Ctrl+E
Properties Window F4
Toolbox Ctrl+Alt+X
Other Windows
Toolbars
Full Screen Shift+Alt+Enter
Property Pages Shift+F4
Document Outline Ctrl+Alt+T
Output Ctrl+Alt+O
Task List Ctrl+Alt+K
Start Page
Web Browser Ctrl+Alt+R
Find Result

Figure 1-14. *The full list of windows in Visual Basic 2005 Express*

Rather than bore you to tears explaining each one in excruciating detail, Table 1-1 summarizes what each window does. You'll look in more detail at each of them as you come across them throughout the book.

Table 1-1. *The Full List of Standard Windows Available in Visual Basic 2005 Express*

Window Name	Description
Code	Shows you the actual Visual Basic code behind a form. When writing programs in VB Express, you'll find yourself switching between the form designer and the code behind the form a lot.
Designer	You've seen the designer already. The designer is the view you'll use to drag controls from the Toolbox onto your forms to build up your application's user interface.
Database Explorer	When you start programming database-aware applications, the Database Explorer comes into play, showing you the tables, stored procedures, and other artifacts inside the database you're working with.
Solution Explorer	The Solution Explorer, as you've seen, shows you all the files in your project. A solution, though, can contain more than one project, and in that case the Solution Explorer will show you every single project in the solution, and all the files in each project.
Object Browser	As you'll see throughout the rest of this book (and in the "Try It Out" shortly), everything you do when creating programs in VB Express revolves around using objects. The Object Browser shows you just which ones are available to you (think of it as a glimpse into the .NET LEGO box)
Error List	Inevitably as you start to work on your own programs (and even when working through some of the examples in this book), you'll make mistakes that will show up as errors when you try to run the program. When that happens, the Error List automatically comes into view, letting you double-click on each error to automatically jump to the corresponding problem in the code.
Properties	The Properties window gives you a way to customize the look and feel, and behavior, of the various components of your program and its user interface.
Toolbox	The Toolbox is a dynamic window in that its contents change based on what you are doing. If you are designing a form for your application's user interface, for example, the Toolbox shows you all the various user interface controls that you can drop onto the form to customize it.
Document Outline	The Document Outline window gives you a great way to keep track of everything on your forms as you build them. For example, if you add a group box to the form, and then a button inside the group box, the Document Outline window will show a nice hierarchical tree indicating exactly which controls and UI elements contain which other UI elements.
Output	When you compile and run a program, this window shows you the output of the Visual Basic compiler. It can also be used by your own program to output debugging information while it's running.

Continued

Table 1-1. *Continued*

Window Name	Description
Task List	The Task List is a flexible window, showing you the errors that occurred at compile time, as well as tasks that you have added yourself. You can actually write little notes into your application to remind you to do things, and these appear in the Task List window.
Start Page	This window displays the start page, the view that you see when you first start VB Express. The Start Page window shows you news and other interesting tidbits from the Microsoft Express communities.
Web Browser	VB Express includes a built-in web browser, and that is just what the Web Browser window is. This is a great tool for quickly looking something up on the Internet if you come across some problems in your code. It's also the window that's used to display the online help system.
Find Results	Your programs can get huge. If VB Express had a search tool like that in Microsoft Word, finding things would be a nightmare (you'd have to keep repeatedly clicking through a Find dialog box to get to just the thing you really need). Instead then, VB Express has a Find Results window that shows you every match that a Find finds. Just double-click an entry in the Find Results window to jump right to the located element.

As you work through the next "Try It Out," use the View menu to display some of the windows in Table 1-1 and watch how they update as you work through the example.

The main toolbar of the IDE also provides you quick access to some of the most common windows. The View icons are at the right-hand end of the toolbar, as shown in Figure 1-15.

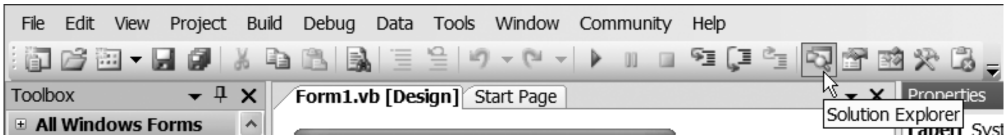


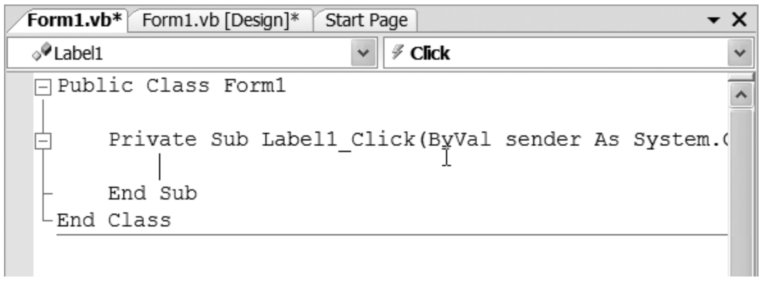
Figure 1-15. *Icons on the main toolbar provide quick access to some of the most common windows.*

Simply hover the mouse over an icon to see a ToolTip appear, explaining what the button does.

Working with the Editors

You already experienced working with the form designer in the previous "Try It Out." Let's take a more detailed look before you dive into the next "Try It Out."

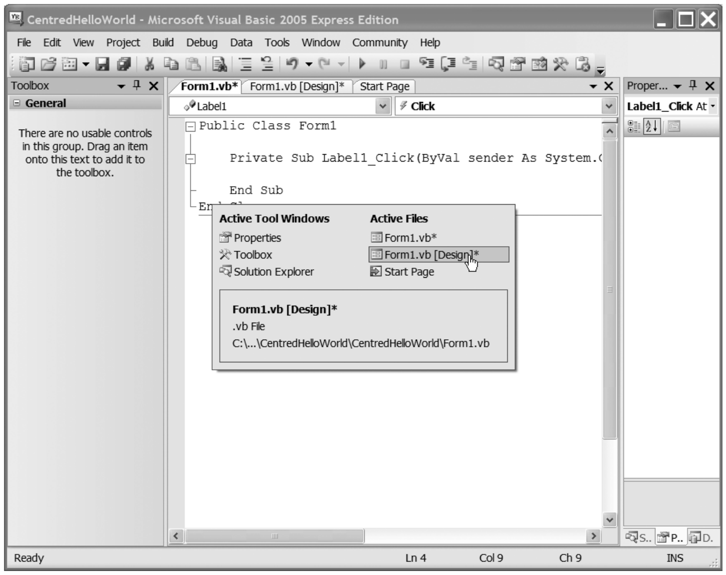
Most of the nontrivial programs you'll work on will have more than one form and definitely more than one source code file. It's not uncommon, therefore, to have many editors open at once. VB Express shows each currently open "document" as a tab across the top of the main editor window, as in Figure 1-16.



```
Public Class Form1

    Private Sub Label1_Click(ByVal sender As System.(

    End Sub
End Class
```

Figure 1-16. *When the editor has more than one document open, tabs at the top of the editor window let you quickly select which document to jump to.*

You can switch between the various open documents either by clicking the tab you want or by holding down the Ctrl key and pressing Tab. When you do that, a dialog box appears, as in Figure 1-17, showing you all the open documents and windows in the IDE.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

```
Public Class Form1

    Private Sub Label1_Click(ByVal sender As System.(

    End Sub
End Cl
```

Form1.vb [Design]*
.vb File
C:\...\CentredHelloWorld\CentredHelloWorld\Form1.vb

Figure 1-17. *Holding down the Ctrl key and then pressing Tab shows you which documents and windows are currently open in the IDE.*

If you keep tapping the Tab key while holding down the Ctrl key, you'll find that the highlight moves between each document in the Open Files list. Alternatively, while still holding down the Ctrl key, you can click the document or window that you want to activate, or even use the arrow keys on the keyboard.

VB Express also offers you a stunning amount of flexibility in how you can configure the editors (form editor, code editor, and so on) to suit you best. For example, I develop on a machine with two monitors, and I like to have a form or source code displayed on one monitor, and another source file on a separate monitor. This is really easy to set up in VB Express. Just right-click one of the editor tabs, and a context menu appears, as you can see in Figure 1-18.

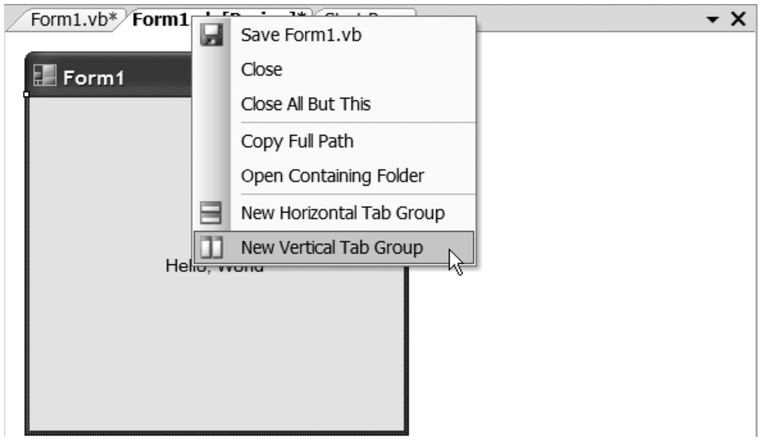
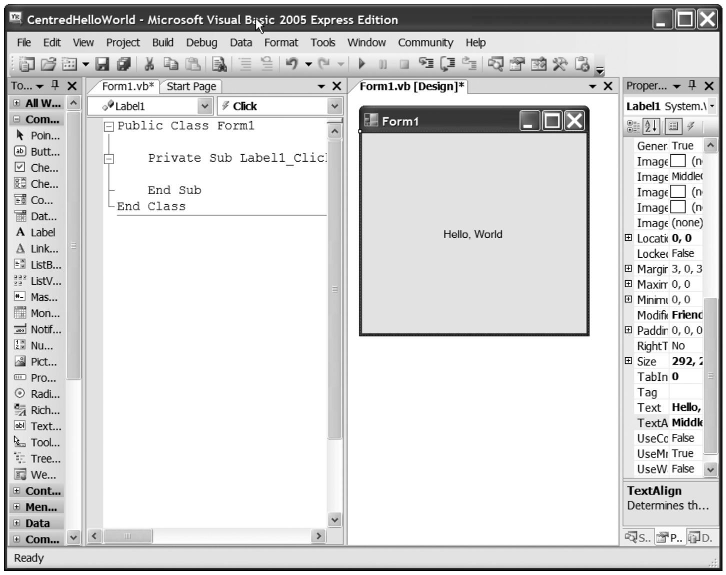


Figure 1-18. *Right-clicking an editor tab pops up a context menu.*

If you then select New Vertical Tab Group, the editor splits in two vertically (I would have preferred Microsoft to call this New Horizontal Tab Group, because it creates a new group of tabs listed horizontally across the top of the screen). You can see this in Figure 1-19.

So, on my machine I simply make VB Express span both monitors, create a new vertical tab group, and then click and drag the divider bar that splits the editor window in two until I have each half of the editor taking up exactly one monitor.

If you really wanted to go crazy with this, you could keep creating new vertical and horizontal groups and end up with a bunch of source files on display at once. Realistically though, even on two monitors, that makes the display very cramped indeed (I yearn for a couple of Apple Cinema 23-inch flat screen monitors, but until I get them, I'll just relegate this feature to the “neat to have” box).


CentredHelloWorld - Microsoft Visual Basic 2005 Express Edition
File Edit View Project Build Debug Data Format Tools Window Community Help
Form1.vb* Start Page
Label1
Click
Form1.vb [Design]*
Form1
Hello, World
Proper...
Label1 System.
TextAlign
Determines th...
Ready

```
Public Class Form1

    Private Sub Label1_Clic

    End Sub
End Class
```

Figure 1-19. *Creating a new vertical tab group splits the current editor view in half, vertically.*

As I said, this is a "neat to have" feature, but it doesn't really offer developers like you that much in the way of productivity benefits. IntelliSense on the other hand does.

All versions of Microsoft's "Visual" development tools have had IntelliSense for years. In fact, even Microsoft's Office tools now have IntelliSense, but that's really only because office workers using Excel and Word get jealous when they see all the cool toys that we developers have to play with.

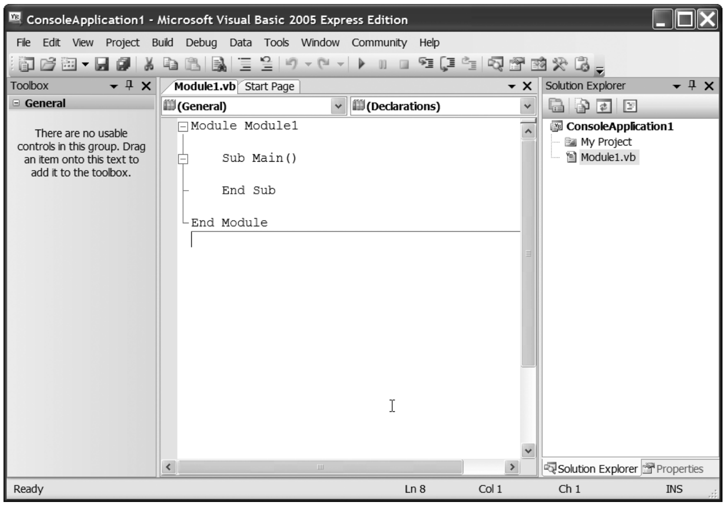
IntelliSense basically is a way for Visual Basic 2005 Express to guess what it is you're trying to do and offer assistance as you write your program code. It's awesome. In fact, in VB Express it's beyond awesome thanks to another neat technology called code snippets, which you'll look at shortly.

Try It Out : Using IntelliSense

Create a new project in Visual Basic 2005 Express by selecting File ➤ New Project from the menu bar, or by pressing Ctrl+N.

When asked what kind of project to create, select Console Application and click the OK button in the dialog box. A console application has no windows-based user interface and simply runs in a text window. This interface is ideal for giving you a flavor of how IntelliSense and the other code development features of the IDE work.

After a short pause, the project will be created and your IDE will look very similar to the one in Figure 1-20.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

```
Module Module1

    Sub Main()

    End Sub

End Module
```

Figure 1-20. *The IDE after starting up a new console application*

Because a console application has no real graphical user interface in the traditional Windows application sense, the IDE automatically drops you into the code editor. If you have never written any Visual Basic applications before, don't worry if the text on the screen looks worryingly cryptic. You'll work through just how a Visual Basic program looks and what it all means, in detail, in the next chapter.

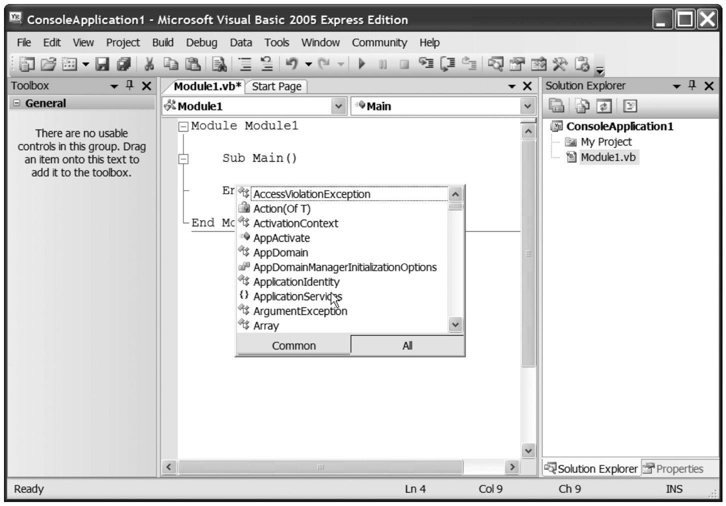
When you write a program in Visual Basic, you actually spend most of your time either creating or using objects and classes (think of a *class* as a blueprint for an object). Visual Basic also provides you with something called *modules*. These are just containers for short snippets of code. Take a look at the code on-screen and you'll see that we're already working with a module:

```
Module Module1

    Sub Main()

    End Sub

End Module
```

As you can see, your program consists of a single module called `Module1`. Inside that module is an empty *subroutine* called `Main`. This is the subroutine that will run when you hit F5 to start the program. It all looks pretty simple, doesn't it. The complexity comes in when you consider what it means to work with the .NET Framework.

The .NET Framework consists of hundreds of classes that each contain a bunch of subroutines that you can call. That all adds up to a lot of things to remember, which is exactly what IntelliSense is designed to help you with.

Position your cursor on the blank line between `Sub Main()` and `End Sub` and then hit Ctrl+spacebar. The IntelliSense window appears just beneath the cursor, as shown in Figure 1-21.



```
Module Module1

    Sub Main()

    Er
End Mc
```

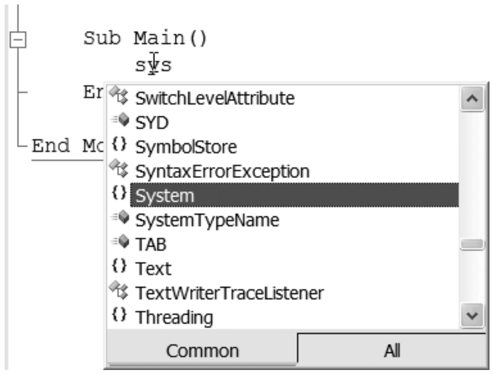
Figure 1-21. *The IntelliSense window appears when you press Ctrl+spacebar.*

Here you'll find all the classes that are available for you to use, as well as namespaces. Because there are so many classes in the framework, .NET uses *namespaces* to group them. For example, if you were about to write code to talk to the Internet, you'd need the classes in the `System.Net` namespace. We'll look more at IntelliSense and namespaces in the next chapter.

If you were to run the program at this point, all you'd see is a text window (called the console window—this is a console application after all) appear briefly and disappear. Because this is a console application, running it causes the console to appear, and because you don't have any code in your application, the console window immediately disappears. There's a way to stop that from happening.

The `System` namespace contains a class called `Console`. This has lumps of code in it (methods) to do various tasks, such as display text in the console window and other such console-related goodies. There's also a method in there called `ReadLine()` that waits for the user to enter some text and press Enter before continuing. Let's use IntelliSense to add a line of code into your `Main ()` subroutine to do just that.

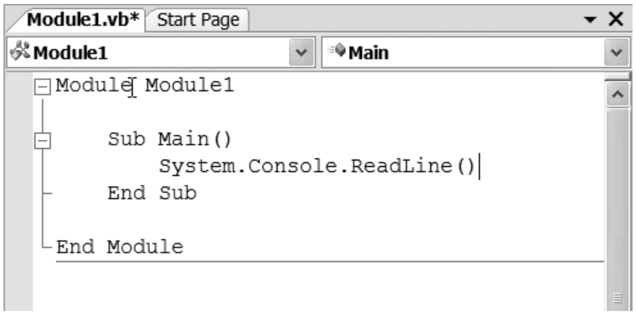
If the IntelliSense window is not currently displayed, press Ctrl+spacebar once again to show it, and then type the letters **Sys**. The IntelliSense window automatically jumps to the first entry it can find that begins with those letters—the `System` namespace (as shown in Figure 1-22).



```
    Sub Main()
        sys
    En
End Mc
```

Figure 1-22. *Typing while the IntelliSense window is open causes it to automatically jump to the first match closest to whatever you typed.*

You can select the entry shown in the list by clicking it with the mouse, pressing Enter, typing **.,** or **(,** or by pressing Tab. Why so many choices? Well, to use a method in the `System` namespace's `Console` class, you'll need to list the namespace followed by a period, followed by the name of the class, followed by a period, followed by the method name. So, try it out. Erase the text you just typed and press Ctrl+spacebar once again. This time, type exactly this: **Sys.Cons.ReadL**. Your editor will look like the one in Figure 1-23.



```
Module Module1

    Sub Main()
        System.Console.ReadLine()
    End Sub

End Module
```

Figure 1-23. *Notice how IntelliSense automatically fills in the rest of the text for you.*

As you can see, IntelliSense can save you an awful lot of typing. More to the point though, it can save you an awful lot of grief by avoiding spelling mistakes. Notice also that IntelliSense changes when you start to call a method on a class by showing you a description of that method.

Finish the line of code, and your program should look like this:

```
Module Module1

    Sub Main()
        System.Console.ReadLine()
    End Sub

End Module
```

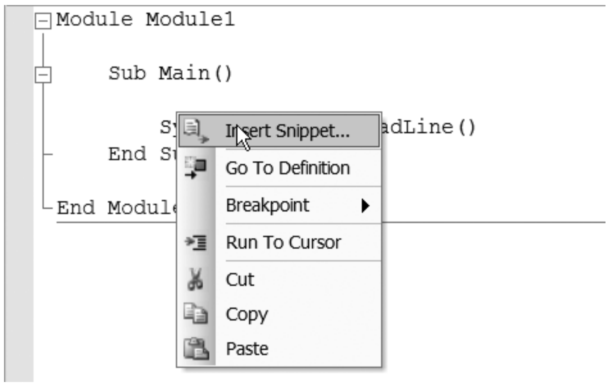
If you run the program now (F5), you'll see the console window appear, and it will stay visible until you click in it and press Enter. `System.Console.ReadLine()` is waiting for you to do just that, and until you do, the program won't end. When it does end, the console is closed down and you are returned to the normal IDE view.

IntelliSense does far more than this, though, and can actually help you add structure to your program. For example, let's get this program to count from 1 to 10 and display those numbers in the console.

Now, just to play dumb here, say I know that the Visual Basic command I need to use to count is the `For` command, but I can't remember how to use it. I do know that IntelliSense can help me.

Place the cursor on the `S` of `System` on the line of code you just typed in, and press Enter to insert a blank line.

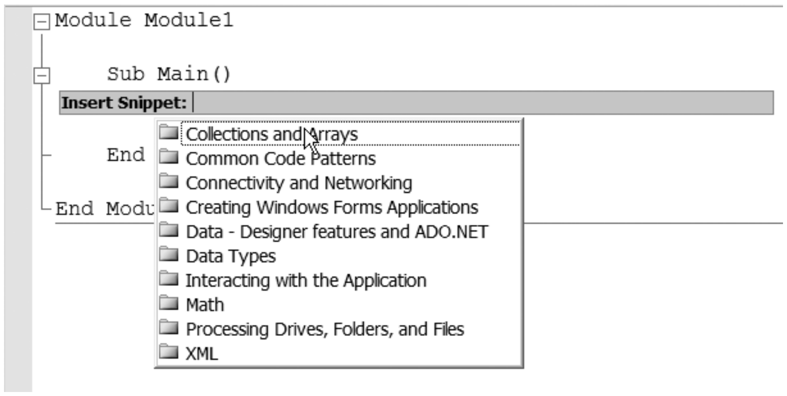
Click in that blank line and right click with the mouse. A menu appears, as in Figure 1-24.



```
Module Module1

    Sub Main()

        Sy                    adLine()
    End Su

End Modul
```

Figure 1-24. *The context menu in the code editor*

Now go ahead and select Insert Snippet. Another list appears, showing you a list of expansions. These are actually code snippets, lumps of code written by Microsoft (you can add your own to the list as well) that you can insert into your own projects as you need them. You can see the snippets list in Figure 1-25.

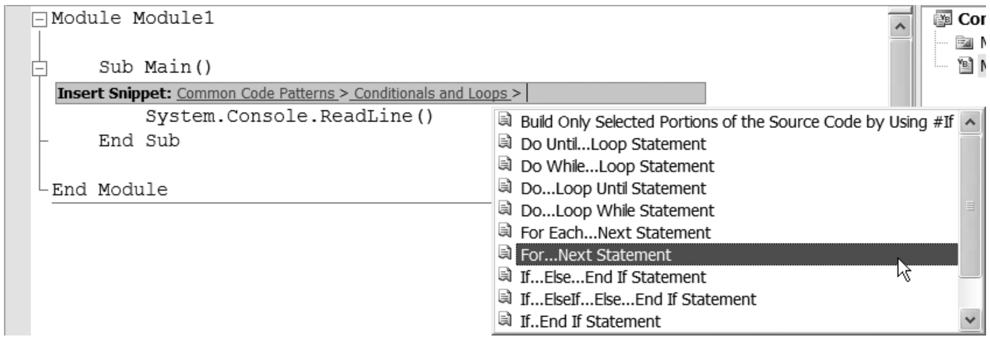


```
Module Module1

    Sub Main()
Insert Snippet:

    End

End Modu
```

Collections and Arrays
Common Code Patterns
Connectivity and Networking
Creating Windows Forms Applications
Data - Designer features and ADO.NET
Data Types
Interacting with the Application
Math
Processing Drives, Folders, and Files
XML

Figure 1-25. *The list of code snippets*

Click on the code snippet group called Common Code Patterns. The list will change to show you more snippet categories. Click Conditionals and Loops, and the list will change once again, this time showing you actual code snippets, as in Figure 1-26.

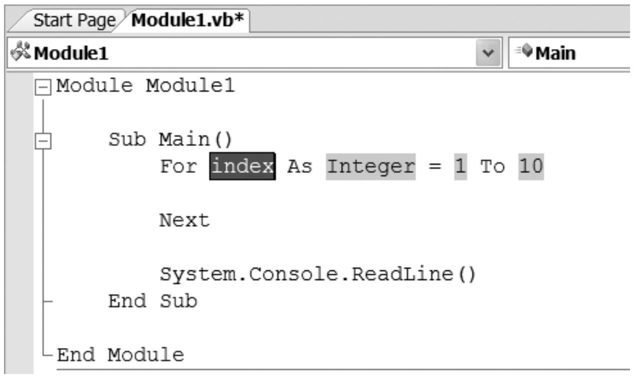


```
Module Module1

    Sub Main()
Insert Snippet: Common Code Patterns > Conditionals and Loops >
        System.Console.ReadLine()
    End Sub

End Module
```

Build Only Selected Portions of the Source Code by Using #If
Do Until...Loop Statement
Do While...Loop Statement
Do...Loop Until Statement
Do...Loop While Statement
For Each...Next Statement
For...Next Statement
If...Else...End If Statement
If...ElseIf...Else...End If Statement
If..End If Statement

Figure 1-26. *You need to click through numerous code snippet groups to get at the actual code snippets themselves.*

Select the snippet called `For...Next Statement` and you'll see the code window suddenly change, as in Figure 1-27.



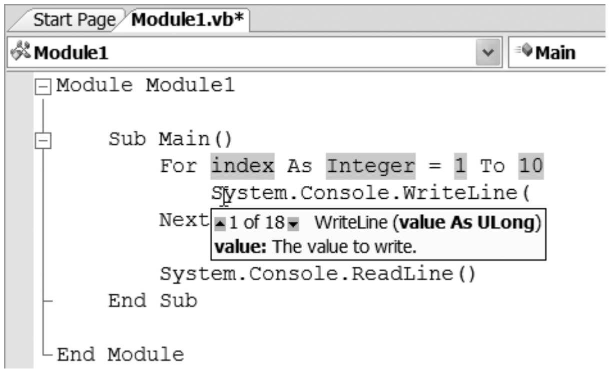
```
Module Module1

    Sub Main()
        For index As Integer = 1 To 10

        Next

        System.Console.ReadLine()
    End Sub

End Module
```

Figure 1-27. *The selected code snippet is inserted into your own code.*

Notice that there are two elements in the new code highlighted in green. These are called *replacements* and are meant for you to, umm, replace. For now, ignore the highlighted "index" thing—it's called a *variable*, and if you've never come across them before, then you'll be introduced to them in the next chapter.

You want your loop to count from 1 to 10, and that's exactly what this snippet does. If you wanted to count from 10 to 100, you would just change the numbers on that first line of code accordingly.

All that remains is to put some code in to display the numbers. You can use IntelliSense for this as well. Click so that your cursor is on the blank line inside the new code snippet block. Press Ctrl+spacebar to bring up the IntelliSense window and type **Sys.Cons.WriteL(**. Your code window will look like Figure 1-28.



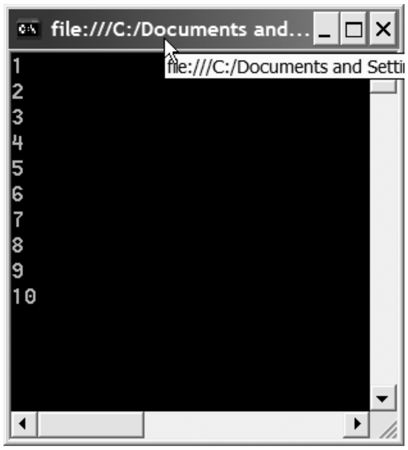
```
Module Module1

    Sub Main()
        For index As Integer = 1 To 10
            System.Console.WriteLine(
        Next
                 1 of 18  WriteLine (value As ULong)
                 value: The value to write.
        System.Console.ReadLine()
    End Sub

End Module
```

Figure 1-28. *Use IntelliSense to make your code window look like this.*

`System.Console.WriteLine` displays text in the console window when the program is running. You simply put the item that you want displayed inside the parentheses at the end of the command. Now, your code snippet uses something called `Index` to count from 1 to 10. Inside your loop then (The `For...Next` thing is called a *loop*), you'll just have `WriteLine` display `Index`. Change the code so that it looks like this:

```
Module Module1

    Sub Main()
        For index As Integer = 1 To 10
            System.Console.WriteLine(index)
        Next

        System.Console.ReadLine()
    End Sub

End Module
```

Run the application now, and your console window will display the numbers 1 to 10 and wait for you to press Enter before closing down. You can see this in Figure 1-29.



```
1
2
3
4
5
6
7
8
9
10
```

Figure 1-29. *When you run the program, it will visibly count from 1 to 10 and then wait for you to press Enter on your keyboard before shutting down.*

Writing Your Own Web Browser

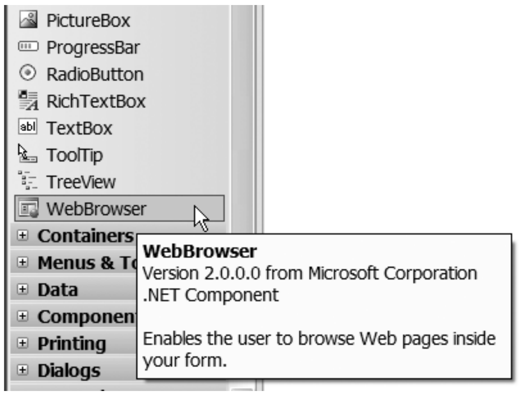
I'm going to close out this chapter with a nice big "Try It Out" that not only shows you even more about the IDE and what it can do, but also focuses on what Visual Basic is best at: writing great applications.

In this example you're going to produce a really simple but functional web browser, something that took Microsoft many years to produce. You're going to do it in about 15 minutes.

Try It Out: Writing a Web Browser

Start a new Windows application project in the usual way (click File ➤ New Project and then choose Windows Application from the Project Type dialog box when it appears).

After the form appears in the editing area, take a look at the Toolbox, shown in Figure 1-30.



WebBrowser
Version 2.0.0.0 from Microsoft Corporation
.NET Component

Enables the user to browse Web pages inside your form.

Figure 1-30. *The Toolbox contains a ready-made WebBrowser control.*

Scroll down the Toolbox. Near the bottom you'll see a WebBrowser control. Double-click it to add it to the form.

The form will appear to change color, but other than that, there's no obvious sign that the form is now a web browser. By default, when you drop the WebBrowser control onto a form, it just shows a blank white page and docks itself to the entire form area. Let's fix that.

Many of the controls in VB Express support something called *smart tags*, a technology originally developed to make people's lives in the Office applications more productive. The WebBrowser control has a smart tag attached, and you can access it by clicking its icon (a small left-pointing arrow at the top right of the browser). Figure 1-31 shows this.

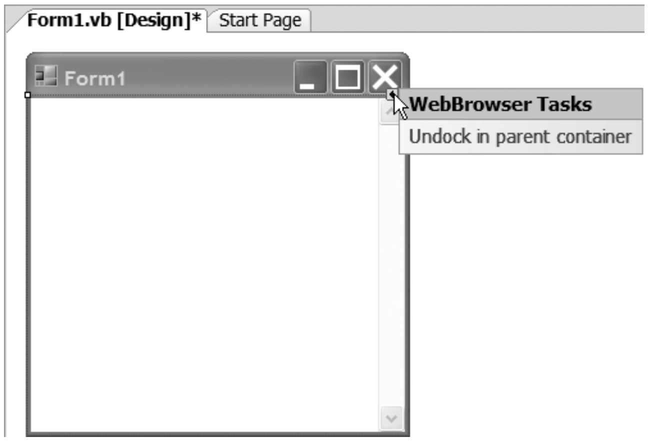


Figure 1-31. *Clicking the smart tag icon on a control pops up an additional menu.*

Select Undock in Parent Container (the only menu item that appears), and the web browser will undock itself to take up a lot less of the form, as shown in Figure 1-32.

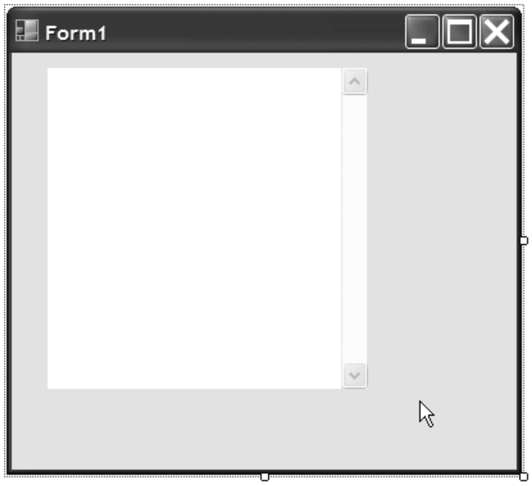


Figure 1-32. *Undocking the web browser makes it take up less room and allows you to move and resize it.*

Go ahead now and drop some labels, a text box, and a few buttons onto the form so that it looks like mine in Figure 1-33. If you can't remember how to do this or to set properties on a control, refer to the first "Try It Out" in this chapter.

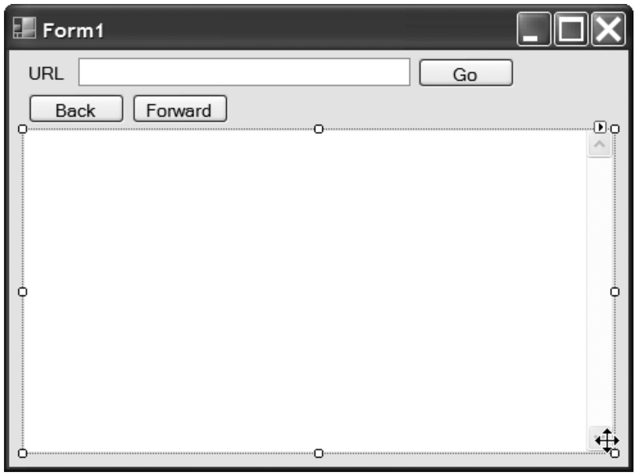


Figure 1-33. *Your web browser form should look like this.*

The next thing that you need to do is to set up anchor points for the various controls. If you were to resize the form right now (please don't), you would find that the form would grow or shrink in size but the controls would remain exactly where they are. What you really want to happen is that when the user resizes the form, the controls move to match the new form size. This is achieved by setting up the `Anchor` property of a control.

In the case of our web browser form, set the `Anchor` property for the text box holding the URL to `Top`, `Left`, and `Right`. This will make the URL stay in the same place on the form, but grow in width as the form is resized. Set the `Anchor` property for the Go button to `Top Right`, which means it will always appear at its current location, in relation to the top-right corner of the form.

Finally, set the `Anchor` property for the browser control to `Top`, `Left`, `Right`, and `Bottom`. This means that the web browser will never move, but as the form changes in width, so too will the browser. Similarly, as the form changes in height, so too will the browser.

Try resizing the form now and you should see the effects of the `Anchor` property changes, like mine in Figure 1-34.

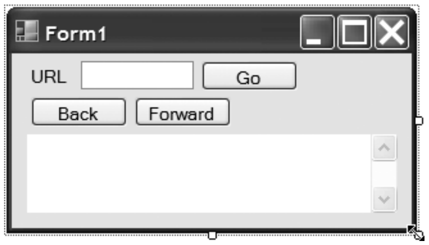


Figure 1-34. *Anchoring controls lets you move and/or resize them as the form changes in size.*

So far you've managed to set up a pretty neat-looking user interface for the web browser that will actually respond to users changing the size of the form. The next step is to add some functionality to this, to make the program useable.

However, before diving into writing some code, you first need to rename some of those controls. The reason for this is that you will be writing code that tells the controls what to do by name. Having a button called `Button1` is okay for just messing around and trying things out, but your code will be a whole lot clearer and easier to follow if you give everything a decent name. So with that in mind, set the `Name` properties as shown in Table 1-2.

Table 1-2. *Name Properties*

Control	Name Property
URL text box (not the label)	`urlBox`
Go button	`goButton`
Back button	`backButton`
Forward button	`forwardButton`
The WebBrowser control	`browser`

Perhaps the most important bit of code you need to write is the code that responds to the user hitting that Go button. When the user clicks it, you want the web browser to navigate straight to the URL that the user entered into the text box.

Double-click the Go button, and the code window appears.

The way to make a WebBrowser control navigate to a new page is a method called `Navigate`. All you need to do is tell that `Navigate` method where to go.

Type in code so that your `goButton_Click` code looks like this:

```
Public Class Form1

    Private Sub goButton_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles goButton.Click
        browser.Navigate(urlBox.Text)
    End Sub

End Class
```

What's happening here? Well, when you double-clicked the button on the form, you told VB Express that you wanted to write some code to deal with the button's default event. An *event* is something that happens in response to something else. For example, when a user clicks a button, a Click event happens, and you can write code to deal with that.

The code that's most interesting here is the bit that you have to write. You're simply using the "browser's" `Navigate` function to navigate to the text entered into the text box. `Text` is a property of the TextBox control that you can look at in the Properties window. You can refer to it in code as shown.

Go ahead and run the application now. Type in a URL, click the Go button, and watch what happens—see Figure 1-35.

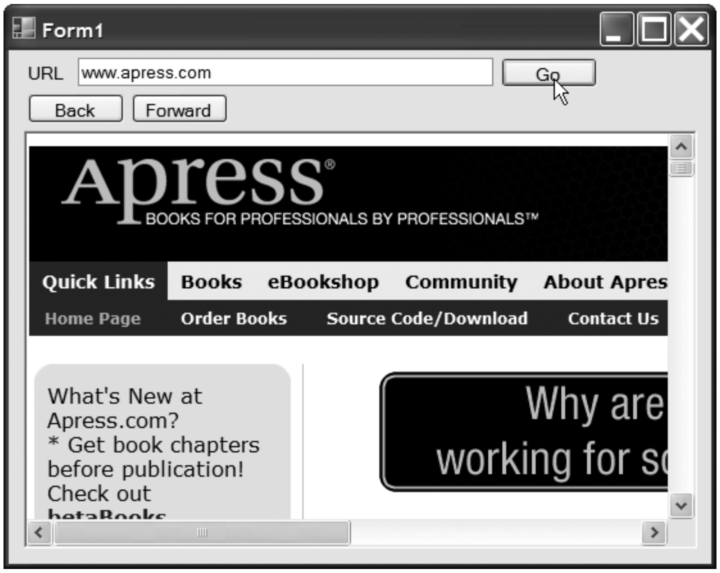


Figure 1-35. *The Go button's Click event brings the web browser to life, telling it exactly where to navigate.*

To stop the application, just close the form as you would any other window.

All that remains now is to get those Back and Forward buttons working.

Double-click the Back button to edit its Click event (you'll be looking at the code window, so to get back to the form in order to double-click the Back button, click the tab labeled Form1.vb [Design] at the top of the code editor).

To get the browser to go back a page (assuming there is a page to go back to), you just need to tell it `GoBack()`. Type in code so that the Back button's Click event looks like this:

```
Private Sub backButton_Click(ByVal sender As System.Object, _
     ByVal e As System.EventArgs) Handles backButton.Click
        browser.GoBack()
    End Sub
```

Do the same thing for the Forward button, but set its code to look like this:

```
    Private Sub forwardButton_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles forwardButton.Click
        browser.GoForward()
    End Sub
```

Voilà – you're done. Run the program, navigate to a web page, click some links, and then start using those Back and Forward buttons.

Those of you who have never written a program before in your life are probably quite worried by the typing that went on. Aside from that, what this example demonstrates is the very essence of Visual Basic 2005 Express. Express makes application development easy. In fact, the .NET Framework 2.0 makes application development easy, and VB Express just adds even more to the mix to make the whole deal so much more satisfying.

Summary

You covered a lot of ground in this introduction to Visual Basic 2005 Express. You saw how to change the look and feel of the IDE, how to create projects, and how to write code. You also took a look at IntelliSense, and hopefully I managed to give you a sense of just what a great time-saver that feature is in VB Express. Finally, we wrapped everything up by developing a simple but fully functioning web browser application.

After this whirlwind tour of the IDE, you'll be pleased to know that in the next chapter I slow things down a bit so you can take a long hard look at just what a Visual Basic program is and what it all means. Those of you who have never written code will soon understand all the things that may have worried you in the earlier examples. Those of you coming here from other languages are about to wonder just how on earth you ever got along before Visual Basic 2005 Express came along.

See you in Chapter 2.