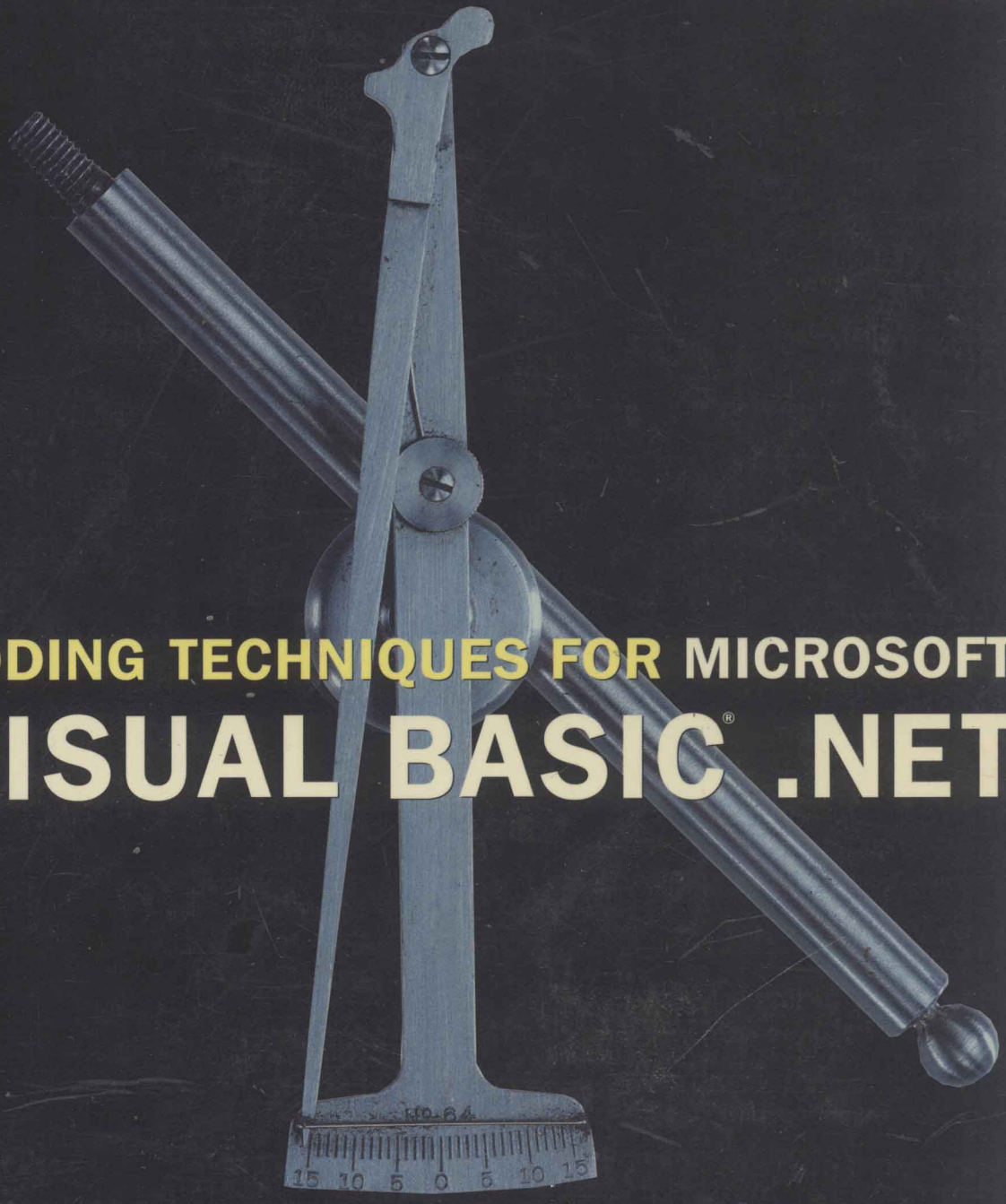
Microsoft



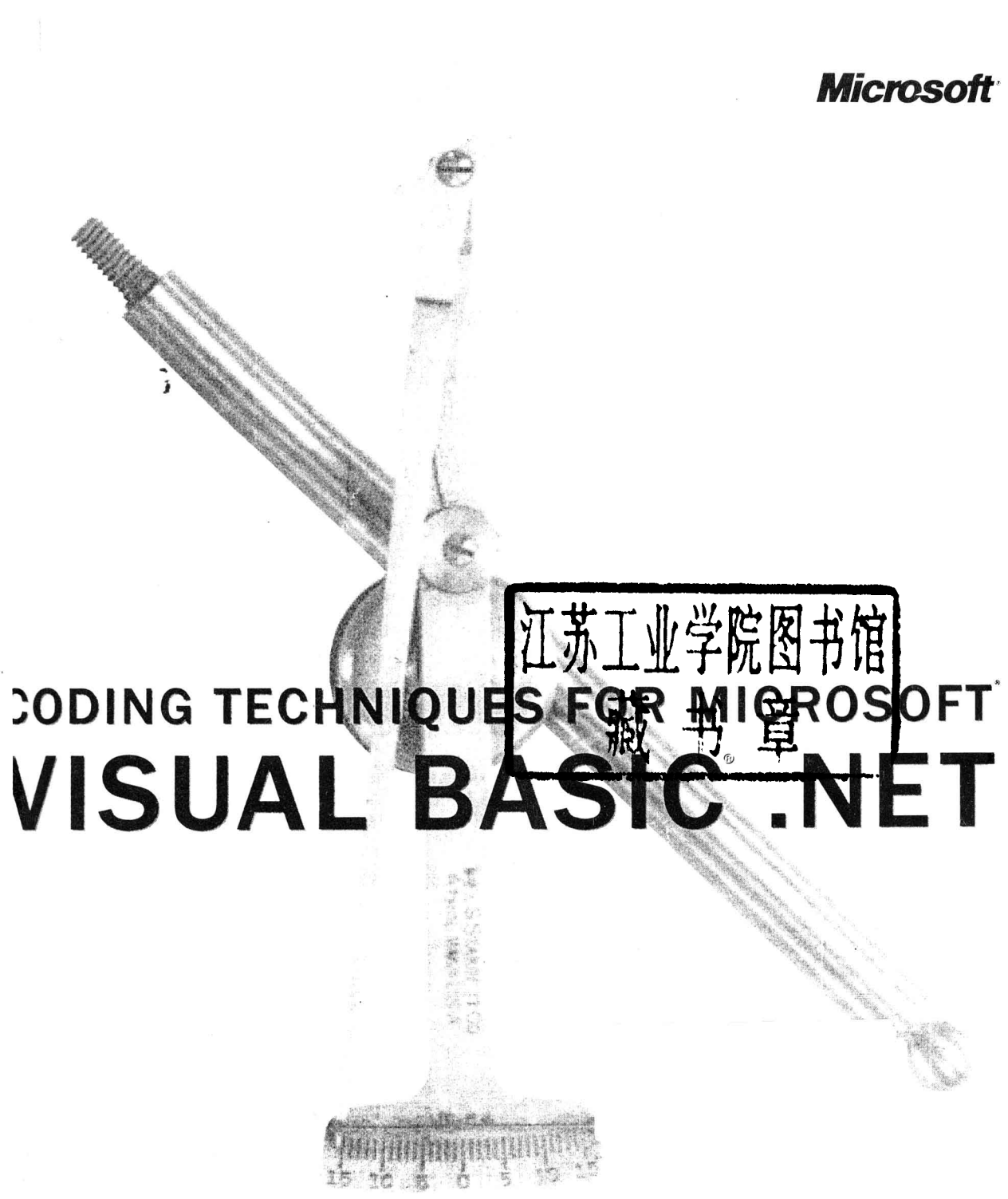
CODING TECHNIQUES FOR MICROSOFT® VISUAL BASIC® .NET

Microsoft .net

世界图书出版公司

John Connell

Microsoft



CODING TECHNIQUES FOR MICROSOFT®

VISUAL BASIC® .NET

crosoft
net

John Connell

This book is dedicated to my mom and dad, Mercedes and John W. Connell. They provided me with unwavering support during this project, as they have in all my pursuits. My two sons, Garrett and Grady, of whom I am immeasurably proud, as well as my baby sister, Patricia, were also there to provide moral support and humor during the long hours.
Thanks, guys, I couldn't have done it without you.

Acknowledgements

Special thanks to Keith D. Adams, childhood friend and Renaissance man; Mr. Morgan Gasior and his computer-scientist wife, Darlene, who provided an endless supply of visionary ideas and showed me the sky; John Dilenschneider, gentleman and computer scientist extraordinaire, for his guidance and friendship; my long-time mentors—Dean Helmut Epp, Ph.D., Dr. Martin Kalin, and Dr. David Miller—for their wit, intelligence, and vision; and Jeff Optholt for seeing things as they really are.

Extra special thanks to John Pierce and Jim Fuchs of Microsoft Press. These two gentlemen worked tirelessly and with impeccable professionalism to make this book into something special. Thanks, John and Jim. In spite of the hard work, you both made this project fun and never lost patience.

Introduction

Over the years, I've been fortunate to design and write production programs with Microsoft Visual Basic that are currently in use at several Fortune 500 companies in the United States. I also teach, hire, and manage technical professionals, which has given me insight into the minds of many Visual Basic programmers. In my university classes, I have seen over and over those aspects of programming with Visual Basic that make sense to programmers, as well as areas such as object-oriented programming that some programmers find a bit confusing. *Coding Techniques for Microsoft Visual Basic .NET* will take you to the next level of programming, lifting the veils from the areas of programming you're unsure of while enhancing your knowledge of the areas that you already work with every day.

Who This Book Is For

This book was written *for* Visual Basic programmers *by* a Visual Basic programmer. In my description of how to work with Visual Basic .NET, I first build a foundation, providing background about the changes in computing and software development that make knowing about the Microsoft .NET Framework of vital interest to programmers as well as a practical necessity. I cover the essentials of object-oriented programming in Visual Basic .NET and explain how to build your own classes and work with the .NET Framework classes, how to work with arrays and collections, and how to debug and handle errors in your programs. From our foundation, we climb to the next level. I cover the details of how to work with .NET assemblies, how to work with files and data streams, and how to monitor files over a network, including how to build a Windows service application that runs on a server. In three full chapters I cover how programming for data access has changed with Visual Basic .NET and ADO.NET. Then we move to the world of Web services—programs and components designed to run on the Internet. In the last chapter, I bring together what's been covered throughout the earlier chapters. Along the way, you'll see plenty of useful and interesting sample code.

Learning Visual Basic .NET by Writing Working Programs

In most computer books I've read, regardless of the programming language they cover, the author provides academic snippets of code to illustrate a point or construct. This approach is helpful, but it leaves readers wondering how one piece of code fits into the larger scheme of a full working program. I've found that the best way to learn a new computer language such as Visual Basic .NET is to write full working programs in that language. Having a goal in mind—and writing a program to solve a problem—engages many dimensions of a programming language and also solidifies how the pieces fit and work together. I take this approach in this book, walking you through several sample applications that illustrate important points about Visual Basic .NET.

If you're coming to Visual Basic .NET from another programming language—such as C, C++, Java, or even COBOL—it won't take long until you feel right at home. The Microsoft .NET Framework is the wave of the future, and Visual Basic programmers are the best prepared to take advantage of this new technology. *Coding Techniques for Microsoft Visual Basic .NET* will make you proficient in the fundamentals of .NET technology, and I'm confident that you'll quickly see the power and ease of what can be accomplished with .NET and will start to look at programming in an exciting new way. Last but not least, you'll have fun in the process.

What's in *Coding Techniques for Microsoft Visual Basic .NET*?

In the list that follows, I describe the highlights of each chapter, summarizing what you'll learn as you progress through the book.

- **Chapter 1, Visual Basic from the Ground Up.** I start off by examining and explaining .NET and why it's a revolutionary (instead of an evolutionary) approach to programming for the twenty-first century. One of the major benefits of the .NET Framework is its capability to write a program once that can automatically target any hardware or operating system. This flexibility is crucial at a time when programmers need to create applications for desktop PCs as well as applications for the Internet. I review the evolution of Visual Basic from the computer language that skyrocketed Windows programming into the mainstream all the way to Visual Basic .NET. I explain at a high level some of the key features of the .NET Framework, such as the class framework, the common language runtime, Web services, assemblies, and the new Visual Studio .NET interactive development environment (IDE)—the cockpit that you'll use when working with these

new capabilities. You'll get your first look at some Visual Basic .NET code to give you a sense of what's required to write a Visual Basic .NET program. After reading Chapter 1, you'll have a good understanding of where we're going and what's important.

- **Chapter 2, Object-Oriented Programming in Visual Basic .NET.** Visual Basic .NET is now a fully object-oriented language, so it finally joins the ranks of the so-called sophisticated languages, such as C++ and Java. If you are new to object-oriented programming, you'll find this chapter an easy way to get up to speed. To illustrate how objects are spawned from classes, I use a simple Visual Basic .NET form and illustrate properties and methods, inheritance, and namespaces. I also cover shared variables, overloading, polymorphism, and encapsulation. Because a large part of the power of .NET comes from the base classes supplied by the .NET Framework, I show how to use the various namespaces and how to access this built-in functionality.

- **Chapter 3, Writing Your First Class.** Following up on Chapter 2, in Chapter 3 I explain how to write your own class and then how to create a second class that inherits from the base class. You'll also learn about the *imports* directive, how to add an assembly to a project, how to work with shared member variables, and why and when to use the *Option Strict* and *Option Explicit* directives. While building a class, you'll use overloaded constructors that permit you to initialize an object upon instantiation. At the conclusion of Chapter 3, object-oriented programming will be demystified, and you'll be pleasantly surprised at how compelling it really is.

- **Chapter 4, Visual Basic .NET Data Types and Features.** While understanding reference and value (primitive) data types in earlier versions of Visual Basic was helpful, understanding these concepts in Visual Basic .NET is crucial because of the way objects—and everything is an object in .NET—are compared and initialized. Data types in .NET are strongly typed (each variable must have a specific data type) and are also type safe (you can only access a variable through its data type). When a variable is no longer needed, it is flagged for deletion by a nondeterministic finalization algorithm, euphemistically known as garbage collection. If you have always set your reference variables to *Nothing*, you'll be interested in the way in which Visual Basic .NET handles freeing up resources and memory. Upon completing Chapter 4, you'll have a good grasp of how variables are brought to life, initialized, and disposed of in Visual Basic .NET.

- **Chapter 5, Examining the .NET Class Framework Using Files and Strings.** The .NET class framework's object-oriented, hierarchical class library is the powerhouse behind .NET. Starting with namespaces, I cover the framework from the ground up, explaining how the framework is organized and using some concrete examples to examine exactly how to work with it. For those of you new to object-oriented programming, this chapter will show precisely how to find what you need in the framework and exactly how to use it. Using a built-in tool, the Windows Class Viewer, you'll master the depth and breadth of the framework and learn techniques to quickly zero in on what you need. In the process, I describe the C# class notation used to designate parameters, overloaded constructors and methods, and return types. In the chapter's main example, I show how the file and stream classes are accessed from the framework and used to read and write to disk. I also examine strings and their new, immutable nature. The techniques for copying, cloning, and formatting strings are illustrated and explained.

- **Chapter 6, Arrays and Collections in Visual Basic .NET.** As you might expect, arrays are handled differently in Visual Basic .NET than in earlier versions of Visual Basic. The .NET Framework class *System.Array* is the base class for all array types. Because arrays are objects (what isn't?), each array you create will have its own knowledge of how many elements it contains, how many dimensions it has included, its boundaries, and so forth. Best of all, by using the *System.Array Sort* method, Visual Basic arrays can now be sorted and reversed automatically. Not only that, .NET arrays can be searched using various approaches such as the built-in binary search. I'll create a calculator program that translates numbers to Roman numerals to illustrate arrays. While an array is really a simple collection, a collection is a group of objects. A collection can be inherited from the *System.Collection* namespace of the .NET Framework. The *Collection* namespace contains interfaces and classes to create new objects such as array lists, hash tables, queues, stacks, and dictionaries. Some of these data structures might be new to Visual Basic programmers. I wrap up Chapter 6 by writing a program that mimics a nondeterministic Rogerian psychologist. I use some advanced features of .NET arrays to accomplish this. Users can run this fun project to examine the psychology of human/machine interaction with Visual Basic .NET.

- **Chapter 7, Handling Errors and Debugging Programs.** Errors don't crash programs, but unhandled errors do. Errors (syntax, runtime, or logic) can occur at any time, and when they do an exception is thrown. Visual Basic .NET uses a structured *Try...Catch...Finally* construct to replace the unstructured *Goto ErrorHandler* used in previous versions of Visual Basic. Structured error handling is built into the core of the .NET Framework, so its power is immediately available to us. In this chapter, I use the calculator program from Chapter 6 and show everything that can go wrong in the program and how to use structured error handling to deal with each potential error. I also examine the debugger. The Visual Studio .NET IDE provides programmers with several debugging windows that give them a clear view on everything from variable values to assembly code in a running program. We'll write a generic error handling class, ErrorTrace.vb, that provides trace logs and can be added to any Visual Basic .NET program. I finish this chapter by showing how to write to the Windows NT or Windows 2000 event log from a Visual Basic .NET program.

- **Chapter 8, Assemblies in Detail.** Assemblies are the building blocks of Visual Basic .NET programs. They are the fundamental unit for deployment, version control, reuse, and security. In this chapter, I build a program, named AssemblySpy, that examines the internals of any .NET assembly written in a .NET-compliant language. This program uses new graphical .NET controls for its user interface and provides information on static and instance fields, properties, events, methods, and constructors. I describe the benefits of private and shared assemblies as well as the reasons for creating "strongly named" assemblies for versioning and sharing. Strongly named assemblies allow for side-by-side execution so that two assemblies with the same name can run in the same directory. Microsoft .NET programs compiled against an assembly with a strong name know which assembly to use, thus eliminating DLL conflicts that have plagued Windows programming.

- **Chapter 9, File System Monitoring.** Built into the .NET Framework, in the *System.IO* namespace, is the *FileSystemWatcher* class. This class fires events when files or directories are changed, created, deleted, or renamed. I'll examine this class in detail by adding it to a class we'll build, named *SystemObserver*, that inherits from *FileSystemWatcher*. I'll also explain the new notion of delegates, which permit

programmers to define and react to their own events. I'll add delegates to the *SystemObserver* class that respond to events of *FileSystemWatcher*. We'll also build a Windows program named File Sentinel and import the *SystemObserver* class. This program provides a user interface that permits a user to select files or directories. File Sentinel can monitor any file or directory (such as a cookies file) and notify you when something important happens. At the end of this chapter, I show you how to develop a Windows Service application (formerly known as an NT service). We'll turn the *SystemObserver* class into a Windows service to illustrate the reusability of our code.

- **Chapter 10, Data Access with ADO.NET.** ADO.NET components have been redesigned to provide a more consistent object model while also providing increased scalability for Internet programming. This is accomplished by using the new disconnected *DataSet* object, which provides a common way to represent and manipulate data on a client. Where traditional data access programming held a connection to the data store, ADO.NET is completely disconnected. This approach uses what are known as *managed providers*, which include a connection object, a command object, a *DataReader* object, and various *DataAdapter* objects. The most compelling aspect of ADO.NET is that the in-memory data set now represents its contents in text-based XML, which can be passed back and forth through any HTTP port 80 firewall, making data sharing between heterogeneous and non-Windows systems a reality.

- **Chapter 11, Data Sets in Detail.** In this chapter, I delve deeper into the ADO.NET object model. I start by writing a program that illustrates how data sets represent their contents in XML, examining both the schema and the XML representation of the data. I show how data is manipulated locally and how changes are then written back to the source. I also demonstrate how to use a tool supplied with Visual Studio .NET, Xsd.exe, that can take an XML file (say from a new vendor) and generate an XML schema definition (XSD) file from it. The XSD file can then be used to create a Visual Basic .NET class that knows how to add, delete, and modify the previously unknown XML file. Next we programmatically build a *DataTable* object in a data set and dynamically add a relationship between fields to build a parent/child relationship. The data from the data set is persisted to disk in XML, and then a data grid control is used to reconstitute the data from the XML file. The data grid does not know or care whether the

information comes from a database or an XML file—in either case it has all the information it needs to build itself. A look at the new *DataView* object illustrates how you can create two separate views of a single data table.

- **Chapter 12, ADO.NET Data Binding.** This chapter completes our examination of ADO.NET by reviewing the *BindingContext* object. Because an ADO.NET disconnected recordset does not have an explicit implementation of a cursor, I describe how to navigate among records with the *BindingContext* object and its related *CurrencyManager* objects. I walk through a program that navigates among records in a single table to show how this is done. I wrap up the chapter by building a table programmatically and adding records to it. We search for specific records in the table and manipulate them.

- **Chapter 13, ASP.NET and Web Services.** In Visual Basic .NET, you can build a Web Form with the same ease as a Windows Form. The same, consistent object model is used. I'll examine how ASP.NET provides a simplified development experience while also providing improved scalability. I look at the new server-side graphical controls as well as the new "code behind" concept, which lets you separate code for business logic from code for the user interface. We build a working ASP.NET loan calculator program to fully review Web Forms, object state, server-side controls, postbacks, field validators, variable caching, data binding to a dynamically built data table, and more. Moving on to Web Services, you'll see how SOAP (Simple Object Access Protocol) breaks down protocol-specific barriers by sending plain text that triggers API methods on remote servers. Web services can advertise their APIs and data types by utilizing the Web Services Description Language (WSDL). I illustrate these concepts with a sample program that uses a Magic 8 ball and a Windows program that consumes the Magic 8 ball Web service. Creating and consuming Web services are my favorite parts of Visual Basic .NET.

- **Chapter 14, Visual Inheritance and Custom Controls.** As you found out early on, everything in Visual Basic .NET is an object, including Windows forms. Because we can inherit from an existing object, in this chapter we'll build a standard form that provides a consistent look and feel and then inherit from the base form to build identical child forms. We will also build a custom Visual Basic .NET control. I wrap up the book with a fun project that takes just about everything you've learned in the book and puts it to use—a program

that places electronic yellow sticky notes on your computer screen. The program automatically saves a note's contents, size, and location and reconstitutes any existing sticky notes when the program is run the next time. I describe how to deploy Visual Basic .NET programs and create a setup project that will deploy our sticky notes project on any Windows 2000, Windows ME, or Windows XP machine, even if the .NET common language runtime is not yet installed.

About the Companion CD

All the sample code is on the companion CD that accompanies this book. The code has been tested using post beta 2 builds of Microsoft Visual Studio .NET and running on Microsoft Windows 2000 Server with Service Pack 2 installed. To use the Web Forms samples, Internet Information Services (IIS) must be installed. You should install IIS before you install Visual Studio .NET. If you install Visual Studio .NET first, you might see the following error message:

```
Error while trying to run project: Unable to start debugging on the Web
server. The server does not support debugging of ASP.NET or ATL Server
applications. Run setup to install the Visual Studio .NET server compo-
nents. If setup has been run verify that a valid URL has been speci-
fied.
You may also want to refer to the ASP.NET and ATL Server debugging
topic in the online documentation. Would you like to disable future
attempts to debug ASP.NET pages for this project?
```

To fix this problem try the following steps (see the Visual Studio .NET online help for more information):

1. Make sure IIS and the Microsoft FrontPage 2000 Server Extensions are installed using Add/Remove Windows Components in Add/Remove Programs.
2. Uninstall the .NET Framework in Add/Remove Programs.
3. Rerun Visual Studio .NET setup, and reinstall the .NET Framework.

The sample programs developed in Chapters 10, 11, and 12, about ADO.NET, also require Microsoft SQL Server 2000, and one sample program requires Microsoft Access 2000 or Microsoft Access 2002. You must have sufficient rights to install and run these applications.

Note that the final version of Visual Studio .NET was not available when this book went to press. The programs were tested against Visual Studio .NET release candidate 2, but changes in the final version might require small modifications to the sample programs.

System Requirements

You'll need the following software to run the samples included on the companion CD:

- Microsoft Visual Studio .NET
- Microsoft Windows 2000 or Microsoft Windows XP
- IIS 5 or later
- Microsoft SQL Server 2000
- Microsoft Access 2000 or Microsoft Access 2002

Do You Have Any Questions?

Every effort has been made to ensure the accuracy of this book and the contents of the companion CD. Should you run into any problems or issues, refer to the following resources:

Microsoft Press provides corrections for books through the World Wide Web at:

http://www.microsoft.com/mspress/support/

If you have comments, questions, or ideas regarding this book or the companion CD, please send them to Microsoft Press using either of the following methods:

E-mail:

mspinput@microsoft.com

Postal Mail:

Microsoft Press

Attn: Coding Techniques for Microsoft Visual Basic .NET Editor

One Microsoft Way

Redmond, WA 98052-6399

Please note that product support is not offered through the above addresses.

上海世界图书出版公司重印出版

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *mspinput@microsoft.com*.

Acquisitions Editor: Danielle Bird
Project Editor: John Pierce
Technical Editor: Jim Fuchs

Body Part No. X08-05019

Table of Contents