Visual Basic® 2005 Programmer's Reference

Rod Stephens

WILEY

Wiley Publishing, Inc.

Visual Basic® 2005 Programmer's Reference

Rod Stephens

WILEY

Wiley Publishing, Inc.

Visual Basic® 2005 Programmer's Reference

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN-13: 978-0-7645-7198-5
ISBN-10: 0-7645-7198-2

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1MA/SS/QZ/QV/IN

Library of Congress Cataloging-in-Publication Data

Stephens, Rod, 1961-
Visual Basic 2005 programmer's reference / Rod Stephens.
p. cm.
Includes index.
ISBN-13: 978-0-7645-7198-5 (paper/website)
ISBN-10: 0-7645-7198-2 (paper/website)
1. Microsoft Visual BASIC. 2. BASIC (Computer program language) I. Title.
QA76.73.B3S83397 2005
005.2'768--dc22
2005008717

About the Authors

Rod Stephens started out as a mathematician but, while studying at MIT, discovered the joys of programming and has been programming professionally ever since. During his career, he has worked on an eclectic assortment of applications in such fields as telephone switching, billing, repair dispatching, tax processing, wastewater treatment, and training for professional football players.

Rod has written 14 books that have been translated into half a dozen different languages, and more than 200 magazine articles covering Visual Basic, Visual Basic for Applications, Delphi, and Java. He is currently a columnist for *Visual Basic Developer* (`www.pinnaclepublishing.com`).

Rod's popular *VB Helper* Web site (`www.vb-helper.com`) receives several million hits per month and contains thousands of pages of tips, tricks, and example code for Visual Basic programmers, as well as example code for this book.

Credits

Executive Editor
Robert Elliott

Development Editor
Kevin Shafer

Technical Editor
John Mueller

Production Editor
Felicia Robinson

Copy Editor
Foxxe Editorial Services

Editorial Manager
Mary Beth Wakefield

Vice President & Executive Group Publisher
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Project Coordinators
Michael Kruzil
Erin Smith

Graphics and Production Specialists
Carrie A. Foster
Lauren Goddard
Denny Hager
Jennifer Heleine
Barbara Moore
Melanee Prendergast
Amanda Spagnuolo
Ron Terry
Julie Trippetti

Quality Control Technician
Leeann Harney
Jessica Kramer
Carl William Pierce

Proofreading and Indexing
TECHBOOKS Production Services

Acknowledgments

Thanks to Bob Elliott, Kevin Shafer, Felicia Robinson, Kathryn Bourgoine, and all of the others who make producing any book possible.

Thanks also to technical editor John Mueller for making sure I wasn't putting my foot too deeply in my mouth and for helping to add extra depth to the book. Visit `http://www.mwt.net/~jmueller` to learn about John's books and to sign up for his free newsletter *.NET Tips, Trends & Technology eXTRA*.

Introduction

When Visual Basic first appeared, it revolutionized Windows programming. By handling many of the tedious details of processing Windows events, it enabled programmers to focus on application details instead of Windows programming trivia.

Unfortunately, early versions of Visual Basic had a few drawbacks. Protection from the underlying Windows details came at the price of reduced flexibility. Using Visual Basic meant you didn't need to mess with the sticky details of Windows event loops, but it also made working directly with those events more difficult when you really wanted to. Advanced programmers could still pry off the cover and work at this lower level, but this was somewhat dangerous. If your code didn't handle all the details correctly, it could crash the program and possibly Windows itself.

Visual Basic also followed a path different from that taken by other Windows programming languages such as C++. It provided a more productive development environment and a generally more intuitive syntax. Its syntax for object-oriented development was more restrictive, however. A developer could still build safe, reliable, extensible applications, but it took some experience and care.

Visual Studio .NET addressed many of these shortcomings. It merged the Visual Basic and C++ development environments into an even more powerful tool. It added the C# language (pronounced "C-sharp") and gave all three a common underlying run-time language called Common Language Runtime (CLR). Visual Basic .NET incorporated changes to bring the language more into line with CLR and the other languages. It included more structured error handling, new syntax for declaring and initializing variables, overloaded functions and subroutines, and a more powerful model for creating classes that include true inheritance.

Visual Basic 2005 adds new features that make Visual Basic a more powerful language than ever. It includes new language features such as unsigned data types, operator overloading, and short-circuit logical operators; object-oriented enhancements such as more flexible property procedure accessibility, generics, and custom events; and coding improvements such as Extensible Markup Language (XML) comments, better IntelliSense, and code snippets.

Visual Basic 2005 is the language's second major release. Most of the obvious bugs in the first release (surprisingly few for such a major reshaping of the language) have been ironed out, so there has never been a better time to learn the language. The first release has proven stable and the current release brings new capabilities to Visual Basic programmers. Developers waiting to see what would become of Visual Basic .NET have their answer: it is here to stay.

Should You Use Visual Basic .NET?

A Visual Basic programmer's joke asks, "What's the difference between Visual Basic .NET and C#? About three months!" The implication is that Visual Basic .NET syntax is easier to understand, and building applications with it is faster. Similarly, C# programmers have their jokes about Visual Basic .NET, implying that C# is more powerful.

In fact, Visual Basic .NET is *not* a whole lot easier to use than C#, and C# is *not* significantly more powerful. The basic form of the two languages is very similar. Aside from a few stylistic differences (Visual Basic is line-oriented; C# uses lots of braces and semicolons), the languages are comparable. Both use the Visual Studio development environment, both provide access to the .NET Framework of support classes and tools, and both provide similar syntax for performing basic programming tasks.

In fact, the languages are so similar that many of Microsoft's Web pages lump the two together. For example, the page `http://msdn.microsoft.com/library/en-us/vbcon/html/vboriWhatsNewVB70.asp` is titled "What's New in Visual Basic and Visual C#."

The main difference between these languages is one of style. If you have experience with previous versions of Visual Basic, you will probably find Visual Basic .NET easier to get used to. If you have experience with C++ or Java, you will probably find C# (or Visual C++ or Visual J#) easy to learn.

Visual Basic does have some ties with other Microsoft products. For example, ASP uses Visual Basic to create interactive Web pages. Microsoft Office applications (Word, Excel, PowerPoint, and so forth) and many third-party tools use Visual Basic for Applications (VBA) as a macro programming language. If you know Visual Basic, you have a head start in using these other languages. Active Server Pages (ASP) and Visual Basic for Application (VBA) are based on pre-.NET versions of Visual Basic, so you won't instantly know how to use them, but you'll have a big advantage if you need to learn ASP or VBA.

If you are new to programming, either Visual Basic .NET or C# is a good choice. I think Visual Basic .NET may be a little easier to learn, but I may be slightly biased because I've been using Visual Basic lately. You won't be making a big mistake either way, and you can easily switch later. Of course, if you have already bought this book, you should stick with Visual Basic to get the most benefit.

Who Should Read This Book

This book is intended for programmers of all levels. It describes the Visual Basic .NET language from scratch, so you don't need experience with previous versions of the language. The book also covers many intermediate and advanced topics. It covers topics in enough depth that even experienced developers will discover new tips, tricks, and language details. After you have mastered the language, you may still find useful tidbits throughout the book, and the reference appendices will help you look up easily forgotten details.

The chapters move quickly through the more introductory material. If you have never programmed before and are intimidated by computers, then you might want to read a more introductory book first. If you are a beginner who's not afraid of the computer, then you should have few problems learning Visual Basic .NET from this book.

If you have programmed in any other language, then fundamentals such as variable declarations, data types, and arrays should be familiar to you, so you should have no problem with this book. The index and reference appendices should be particularly useful in helping you translate from the languages you already know into the corresponding Visual Basic syntax.

How This Book Is Organized

You could divide the chapters in this book into four parts plus appendices. The chapters in each part are described here. If you are an experienced programmer, you can use these descriptions to decide which chapters to skim and which to read in detail.

Part I: Getting Started

The chapters in this part of the book explain the basics of Visual Basic .NET programming. They describe the development environment, basic program syntax, and how to interact with standard controls. More advanced topics include how to build custom controls and how to implement drag and drop.

Chapter 1, "IDE," describes the integrated development environment (IDE). It explains the IDE's windows and how to customize the IDE. It also explains tools that provide help while you're programming such features as the Object Browser and the code window's Intellisense.

Chapter 2, "Controls in General," describes general control concepts. It explains how to add controls to a form, how to read and change a control's properties at design time and at run time, and how to use some of the more complicated control properties (such as `Dock` and `Anchor`). This chapter shows how to catch and respond to events, and how to change event handlers in code.

Chapter 3, "Program and Module Structure," analyzes a simple Visual Basic program and explains the structure created by Visual Studio. It describes the program's code regions and comments, and tells how you can use similar techniques to make your code more readable and manageable.

Chapter 4, "Data Types, Variables, and Constants," explains the standard data types provided by Visual Basic. It shows how to declare and initialize variables and constants, and explains variable scope. It discusses value and reference types, passing parameters by value or reference, and creating parameter variables on the fly. It also explains how to create arrays, enumerated types, and structures.

Chapter 5, "Operators," describes the operators a program uses to perform calculations. These include mathematical operators (`+`, `*`, `\`), string operators (`&`), and Boolean operators (`And`, `Or`). The chapter explains operator precedence and type conversion issues that arise when an expression combines more than one type of operator (for example, arithmetic and Boolean).

Chapter 6, "Subroutines and Functions," explains how you can use subroutines and functions to break a program into manageable pieces. It describes routine overloading and scope.

Chapter 7, "Program Control Statements," describes the statements that a Visual Basic program uses to control code execution. These include decision statements (`If Then Else`, `Select Case`, `IIF`, `Choose`) and looping statements (`For Next`, `For Each`, `Do While`, `While Do`, `Repeat Until`).

Chapter 8, "Error Handling," explains error handling and debugging techniques. It describes the `Try Catch` structured error handler in addition to the older `On Error` statement inherited from earlier versions of Visual Basic. It discusses typical actions a program might take when it catches an error. It also describes techniques for preventing errors and making errors more obvious when they do occur.

Chapter 9, "Introduction to Windows Forms Controls," explains the Visual Basic's standard controls that you can use on Windows forms. It describes the most useful properties, methods, and events provided by these controls, and it gives examples showing how to use them. It also describes cases where these controls rely on each other. For example, several controls such as the ToolBar obtain images from an associated ImageList control.

Chapter 10, "Forms," explains typical uses of forms. It tells how to build partially transparent forms for use as splash, login, and About forms. It describes form cursors and icons, how to override WndProc to intercept a form's Windows messages, how to make a Multiple Document Interface (MDI) application, and how to implement a Most Recently Used (MRU) file list. It does not cover all of the Form object's properties, methods, and events in detail; those are described in Appendix H, "Form Objects."

Chapter 11, "Database Controls and Objects," explains how to use Visual Basic's standard database controls. These include database connection components that handle connections to a database, `DataSet` components that hold data within an application, and data adapter controls that move data between data connections and `DataSets`.

Chapter 12, "Custom Controls," explains how to build your own customized controls that you can then use in other applications. It covers the three main methods for creating a custom control: derivation, composition, and building from scratch. This chapter also provides several examples that you can use as a starting point for controls of your own.

Chapter 13, "Drag and Drop, and the Clipboard," explains how a Visual Basic program can support drag-and-drop operations. It tells how your program can start a drag to another application, how to respond to drag operations started by another application, and how to receive a drop from another application. This chapter also explains how a program can copy data to and from the clipboard. Using the clipboard is similar to certain types of drag-and-drop operations, so these topics fit naturally in one chapter.

Part II: Object-Oriented Programming

The chapters in this part of the book explain fundamental concepts in object-oriented programming (OOP) with Visual Basic. It also describes some of the more important classes and objects that you can use when building an application.

Chapter 14, "OOP Concepts," explains the fundamental ideas behind object-oriented programming. It describes the three main features of OOP: encapsulation, polymorphism, and inheritance. It explains the benefits of these features and tells how you can take advantage of them in Visual Basic.

Chapter 15, "Classes and Structures," explains how to declare and use classes and structures. It explains what classes and structures are, and it describes their differences. It shows the basic declaration syntax and tells how to create instances of classes and structures. It also explains some of the trickier class issues (such as private class scope, declaring events, and shared variables and methods).

Chapter 16, "Namespaces," explains namespaces. It tells how Visual Studio uses namespaces to categorize code and to prevent name collisions. It describes a project's root namespace, tells how Visual Basic uses namespaces to resolve names (such as function and class names), and tells how you can add namespaces to an application yourself.

Chapter 17, "Collection Classes," explains classes included in Visual Studio that you can use to hold groups of objects. It describes the various collection, dictionary, queue, and stack classes; tells how to make strongly typed versions of those classes; and gives some guidance on deciding which class to use under different circumstances.

Chapter 18, "Generics," explains templates that you can use to build new classes designed to work with specific data types. For example, you can build a generic binary tree and then later use it to build classes to represent binary trees of customer orders, employees, or work items.

Part III: Graphics

The chapters in this part of the book describe graphics in Visual Basic .NET. They explain the Graphics Device Interface+ (GDI+) routines that programs use to draw images in Visual Basic. They explain how to draw lines and text; how to draw and fill circles and other shapes; and how to load, manipulate, and save bitmap images. This part also explains how to generate printed output and how to send reports to the screen or to the printer.

Chapter 19, "Drawing Basics," explains the fundamentals of drawing graphics in Visual Basic .NET. It describes the graphics namespaces and the classes they contain. It describes the most important of these classes, `Graphics`, in detail. It also describes the `Paint` event handler and other events that a program should use to keep its graphics up to date.

Chapter 20, "Brushes, Pens, and Paths," explains the most important graphics classes after `Graphics`: `Pen` and `Brush`. It tells how you can use `Pens` to draw solid lines, dashed lines, lines with custom dash patterns, and lines with custom lengthwise stripe patterns. It tells how to use `Brushes` to fill areas with colors, hatch patterns, linear color gradients, color gradients that follow a path, and tiled images. This chapter also describes the `GraphicsPath` class, which represents a series of lines, shapes, curves, and text.

Chapter 21, "Text," explains how to draw strings of text. It shows how to create different kinds of fonts, determine exactly how big text will be when drawn in a particular font, and use GDI+ functions to make positioning text simple. It shows how to use a `StringFormat` object to determine how text is aligned, wrapped, and trimmed, and how to read and define tab stops.

Chapter 22, "Image Processing," explains how to load, modify, and save image files. It shows how to read and write the pixels in an image, and how to save the result in different file formats such as BMP GIF, and JPEG. It tells how to use images to provide auto-redraw features, and how to manipulate an image pixel by pixel, both using a Bitmap's `GetPixel` and `SetPixel` methods and using "unsafe" access techniques that make pixel manipulation much faster than is possible with normal GDI+ methods.

Chapter 23, "Printing," explains different ways that a program can send output to the printer. It shows how you can use the `PrintDocument` object to generate printout data. You can then use the `PrintDocument` to print the data immediately, use a `PrintDialog` control to let the user select the printer and set its characteristics, or use a `PrintPreviewDialog` control to let the user preview the results before printing.

Chapter 24, "Reporting," provides an introduction to Crystal Reports, a tool that makes generating reports in Visual Basic relatively easy. The chapter explains the basics of Crystal Reports and steps through an example that builds a simple report.

Part IV: Interacting with the Environment

The chapters in this part of the book explain how an application can interact with its environment. They show how the program can save and load data in external sources (such as the System Registry, resource files, and text files); work with the computer's screen, keyboard, and mouse; and interact with the user through standard dialog controls.

Chapter 25, "Configuration and Resources," describes some of the ways that a Visual Basic program can store configuration and resource values for use at run time. Some of the most useful of these include environment variables, the Registry, configuration files, and resource files.

Chapter 26, "Streams," explains the classes that a Visual Basic application can use to work with stream data. Some of these classes are `FileStream`, `MemoryStream`, `BufferedStream`, `TextReader`, and `TextWriter`.

Chapter 27, "File-System Objects," describes classes that let a Visual Basic application interact with the file system. These include classes such as `Directory`, `DirectoryInfo`, `File`, and `FileInfo` that make it easy to create, examine, move, rename, and delete directories and files.

Chapter 28, "Useful Namespaces," describes some of the most commonly useful namespaces defined by the .NET Framework. It provides a brief overview of some of the most important System namespaces and gives more detailed examples that demonstrate regular expressions, XML, cryptography, reflection, threading, and Direct3D.

Appendixes

The book's appendices provide a categorized reference of the Visual Basic .NET language. You can use them to quickly review the syntax of a particular command, select from among several overloaded versions of a routine, or refresh your memory of what a particular class can do. The chapters earlier in the book give more context, explaining how to perform specific tasks and why one approach might be preferred over another.

Appendix A, "Useful Control Properties, Methods, and Events," describes properties, methods, and events that are useful with many different kinds of controls.

Appendix B, "Variable Declarations and Data Types," summarizes the syntax for declaring variables. It also gives the sizes and ranges of allowed values for the fundamental data types.

Appendix C, "Operators," summarizes the standard operators such as `+`, `<<`, `OrElse`, and `Like`. It also gives the syntax for operator overloading.

Appendix D, "Subroutine and Function Declarations," summarizes the syntax for subroutine, function, and property procedure declarations.

Appendix E, "Control Statements," summarizes statements that control program flow such as `If Then`, `Select Case`, and looping statements.

Appendix F, "Error Handling," summarizes both structured and "classic" error handling. It describes some useful exception classes and gives an example showing how to build a custom exception class.

Appendix G, “Standard Controls and Components,” describes standard components provided by Visual Basic .NET. It explains the properties, methods, and events that I have found most useful when working with these components.

Appendix H, “Form Objects,” describes forms. In a very real sense, forms are just another type of component. They play such a key role in Visual Basic applications, however, that they deserve special attention in their own appendix.

Appendix I, “Classes and Structures,” summarizes the syntax for declaring classes and structures, and defining their constructors and events.

Appendix J, “Generics,” summarizes the syntax for declaring generic classes.

Appendix K, “Graphics,” summarizes the objects used to generate graphics in Visual Basic .NET. It covers the most useful graphics namespaces.

Appendix L, “Useful Exception Classes,” lists some of the more useful exception classes defined by Visual Basic. You may want to throw these exceptions in your own code.

Appendix M, “Date and Time Format Specifiers,” summarizes specifier characters that you can use to format dates and times. For example, they let you display a time using a 12-hour or 24-hour clock.

Appendix N, “Other Format Specifiers,” summarizes formatting for numbers and enumerated types.

Appendix O, “The `Application` Class,” summarizes the `Application` class that provides properties and methods for controlling the current application.

Appendix P, “The My Namespace,” describes the My namespace, which provides shortcuts to useful features scattered around other parts of the .NET Framework. It provides shortcuts for working with the application, computer hardware, application forms, resources, and the current user.

Appendix Q, “Streams,” summarizes Visual Basic’s stream classes such as `Stream`, `FileStream`, `MemoryStream`, `TextReader`, `CryptoStream`, and so forth.

Appendix R, “File-System Classes,” summarizes methods that an application can use to learn about and manipulate the file system. It explains classic Visual Basic methods such as `FreeFile`, `WriteLine`, and `ChDir`, as well as newer .NET Framework classes such as `FileSystem`, `Directory`, and `File`.

How to Use This Book

If you are an experienced Visual Basic .NET programmer, you may want to skim the language basics covered in the first parts of the book. You may find a few new features that have appeared in Visual Basic 2005, so you probably shouldn’t skip these chapters entirely, but most of the basic language features are the same as in previous versions.

Intermediate programmers and those with less experience with Visual Basic .NET should take these chapters a bit more slowly. The chapters in Part II, “Object-Oriented Programming,” cover particularly tricky topics. Learning all the variations on inheritance and interfaces can be rather confusing.

Beginners should spend more time on these first chapters because they set the stage for the material that follows. It will be a lot easier for you to follow a discussion of file management or regular expressions if you are not confused by the error-handling code that the examples take for granted.

Programming is a skill best learned by doing. You can pick up the book and read through it quickly if you like, but the information is more likely to stick if you open the Visual Basic .NET development environment and experiment with some programs of your own. Normally, when I read a new programming book, I work through every example myself, modifying the code to see what happens if I try different things not covered by the author. I experiment with new variations and pay particular attention to errors, which are hard to cover completely in a book. It's one thing to read about strongly typed collections; it's another to build one yourself using data that is meaningful to you.

Learning by doing may encourage you to skip sections of the book. For example, Chapter 1 covers the interactive development environment in detail. After you've read for a while, you may want to skip some sections and start experimenting with the environment on your own. I encourage you to do so. Lessons learned by doing stick better than those learned by reading. Later, when you have some experience with the development environment, you can go back and examine Chapter 1 in more detail to learn more advanced customization techniques.

The final part of the book is a Visual Basic .NET reference. These appendices present more concise, categorized information about the language. You can use these appendices to recall the details of specific operations. For example, you can read Chapter 9 to learn which controls are useful for different purposes. Then use Appendix G to learn about specific controls' properties, methods, and events.

Throughout your work, you can also refer to the appendices to get information on specific classes, controls, and syntax. For example, you can quickly find the syntax for declaring a generic class in Appendix J. If you need more information on generics, you can find it in Chapter 18 or the online help. If you just need to refresh your memory of the basic syntax, however, scanning Appendix J will be faster.

Necessary Equipment

To read this book and understand the examples, you will need no special equipment. To use Visual Basic .NET and to run the examples found on the book's Web page, you need any computer that can reasonably run Visual Basic .NET. That means a reasonably modern, fast computer with a lot of memory. See the Visual Basic .NET documentation for Microsoft's exact requirements and recommendations.

To build Visual Basic .NET programs, you will also need a copy of Visual Basic .NET. Don't bother trying to run the examples shown here if you have a pre-.NET version of Visual Basic such as Visual Basic 6. The changes between Visual Basic 6 and Visual Basic .NET are huge, and many Visual Basic .NET concepts don't translate well into Visual Basic 6. With some experience in C#, it would be much easier to translate programs into that language.

Much of the Visual Basic 2005 release is compatible with Visual Basic .NET 2003 and earlier versions of Visual Basic .NET, however, so you can make many of the examples work with earlier versions of Visual Basic .NET. You will not be able to load the example programs downloaded from the book's Web site, however. You will need to copy and paste the significant portions of the code into your version of Visual Basic .NET.

The Book's Web Site

On the book's Web site, `www.vb-helper.com/vb_prog_ref.htm`, you can do the following:

- Download the examples in this book
- Download other Visual Basic programming examples
- View updates and corrections
- Read other readers' comments and suggestions

This book was written using beta versions of Visual Basic 2005. Microsoft often makes changes between beta versions and the final release (the whole point of the betas is to identify areas that need fixing or modification) and sometimes even produces patch releases shortly after the main product rollout. The book's Web page will include any modifications that the examples need to handle those changes.

If you have corrections or comments of your own, please send them to me at `RodStephens@vb-helper.com`. I will do my best to keep the Web site as up to date as possible.