

.NET 2.0 for Delphi Programmers



Jon Shemitz

.NET 2.0 for Delphi Programmers

Copyright © 2006 by Jon Shemitz

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13: 978-1-59059-386-8

ISBN-10: 1-59059-386-3

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

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Jim Sumser

Technical Reviewer: Hallvard Vassbotn

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

Project Manager: Sofia Marchant

Copy Edit Manager: Nicole LeClerc

Copy Editor: Ami Knox

Assistant Production Director: Kari Brooks-Copony

Production Editor: Lori Bring

Compositor: Susan Glinert

Proofreader: Liz Welch

Indexer: Rebecca Plunkett

Artist: April Milne

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 information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at www.apress.com in the Source Code section.

To Anders Hejlsberg, for Turbo Pascal, Delphi, and now C#;

*And to the vegetable garden that I didn't grow in 2005
so that I'd have time to finish this book;*

*And, most of all, to Tané, Sam, and Arthur, with thanks
for all your patience and encouragement.*

Contents at a Glance

Table Cross-Reference	xvii
About the Author	xviii
About the Technical Reviewer	xix
Acknowledgments	xx
Preface	xxi

PART 1 ■■■ Common Language Runtime

■ CHAPTER 1	Managed Code	3
■ CHAPTER 2	The Object Model	17
■ CHAPTER 3	Garbage Collection	59
■ CHAPTER 4	JIT and CIL	81

PART 2 ■■■ C# and Delphi

■ CHAPTER 5	C# Primitive Types	107
■ CHAPTER 6	C# Control Structures	125
■ CHAPTER 7	C# Objects	139
■ CHAPTER 8	C# Interfaces and Delegates	179
■ CHAPTER 9	C# Topics	201
■ CHAPTER 10	Delphi for .NET	221

PART 3 ■■■ The Framework Class Library

■ CHAPTER 11	Strings and Files	257
■ CHAPTER 12	Collections	305
■ CHAPTER 13	Reflection	329
■ CHAPTER 14	Serialization and Remoting	353
■ CHAPTER 15	WinForms Basics	373

■ CHAPTER 16	Graphics	391
■ CHAPTER 17	Threads and Synchronization	413
■ CHAPTER 18	XML	439

PART 4 ■■■ Appendixes

■ APPENDIX 0	Unsafe C# Code	457
■ APPENDIX 1	NUnit	465
■ APPENDIX 2	Assembly Loading and Signing	473
■ APPENDIX 3	Configuration Files	477
■ APPENDIX 4	Glossary	479
■ APPENDIX 5	Bibliography	485
■ INDEX	489

Contents

Table Cross-Reference	xvii
About the Author	xviii
About the Technical Reviewer	xix
Acknowledgments	xx
Preface	xxi

PART 1 ■■■ Common Language Runtime

■ CHAPTER 1	Managed Code	3
	Beyond Delphi	3
	Intermediate Code	4
	Garbage Collection	7
	Run-time Checking	8
	Checked Casts	9
	Pointer Arithmetic	10
	Unsafe Code	11
	Language Independence	13
	Common Type System	14
	More Jobs	16
	Key Points	16
■ CHAPTER 2	The Object Model	17
	Farther Beyond Delphi	17
	What's New	19
	Generics	19
	Single Object Model	24
	No More Globals	27
	Attributes	29
	Nested Classes	30
	Type Initializers	32
	Sealed Classes and Sealed Methods	34

What's Different	35
Reference Types vs. Value Types	35
Strings	39
Arrays	40
Delegates	41
Namespaces	43
Enums	44
What's Missing	45
Subranges	46
Array Types	46
Sets	47
Metaclasses	48
Common Language Specification	52
CLS Rules	53
Cross Language Programming	55
Key Points	58
CHAPTER 3 Garbage Collection	59
Performance	60
Detecting Live Data	63
Pathological Cases	64
Finalization	69
Disposing and Finalizing	70
Disposing and Not Finalizing	72
Complications	72
Large Object Heap	73
Self-Tuning	74
Multithreading	75
Multiprocessors	76
Weak References	77
Key Points	79
CHAPTER 4 JIT and CIL	81
.NET Is Not Interpreted	81
Real Pointers	82
Demand Loading	82
Code Quality	83
Inlining and Properties	84
Precompilation	85

JIT Benefits	86
Productivity	86
Portability	87
CIL	87
Type-safe Assembler	88
CIL and the CLR	89
Actual CIL	89
Expressions	92
Logical Operations	94
Methods and Results	95
ILDASM	95
Key Points	103

PART 2 ■■■ C# and Delphi

■ CHAPTER 5	C# Primitive Types	107
	Types and Expressions	107
	Aliases for System Types	108
	Numeric Literals	109
	Numeric Expressions	110
	Operators	111
	Assignment Operators	113
	The Conditional Operator	114
	The Null Coalescing Operator	114
	The Increment and Decrement Operators	115
	Operator Precedence	116
	Strings and Characters	117
	Arrays	119
	Enums	120
	Boxing	122
	Nullable Types	122
	Key Points	124
■ CHAPTER 6	C# Control Structures	125
	Blocks and Statements	125
	Conditionals	126
	The if Statement	127
	The switch Statement	127

Loops	128
The for Statement	129
The foreach Statement	130
The while Statement	132
The do Statement	132
Exception Handling	133
Special Blocks	134
The using Statement	134
The lock Statement	136
Key Points	137

CHAPTER 7	C# Objects	139
	No Headers	139
	Generics	141
	Inline Types	142
	Constraints	143
	C# Object Types	144
	Access	145
	Modifiers	147
	Fields	148
	Static Fields	148
	Constant Fields	148
	Read-only Fields	149
	Volatile Fields	150
	The new Modifier	150
	Methods	151
	Inheritance	157
	Polymorphism	158
	Properties	159
	Indexers	161
	Mixed Access	162
	Parameterized Properties	162
	Constructors	163
	Optional Initializer	165
	Default Constructors	168
	Value Types	169
	Finalizers	169

Operator Overloading	170
Background and Warning	171
Infix Operators.....	172
Type Conversion	173
Truth.....	175
Nested Types	175
Which Object Type?	176
Key Points	177
CHAPTER 8 C# Interfaces and Delegates	179
Interfaces	179
Iterators	182
Delegates	186
Events	188
Delegate Value Equality.....	190
Anonymous Methods	191
Covariance and Contravariance	193
Asynchronous Execution	195
Key Points	199
CHAPTER 9 C# Topics	201
The Main Method	201
Namespaces	201
Name Resolution.....	203
Aliases	204
Namespace Versioning.....	206
Attributes	207
Attribute Targets.....	210
Compile-time Attributes.....	213
The @ Escape	214
Preprocessor Directives	215
Conditional Compilation.....	215
Warnings and Errors.....	217
Folding Regions.....	218
Partial Classes	218
Key Points	220

CHAPTER 10 Delphi for .NET	221
Adapting to Change	221
The Object Model	222
Other Language Changes	229
.NET Platform Support	236
Obsolete Features	246
Win32 and .NET Differences	247
Delphi vs. C#	250
Delphi Language Highlights	251
C# Language Highlights	252
Key Points	253

PART 3 ■■■ The Framework Class Library

CHAPTER 11 Strings and Files	257
Learning the FCL	257
Strings	259
The String Class	260
Concatenation Methods	260
The Format Method	262
Substrings	267
Compare Methods	268
Search and Replace	269
Split and Join	270
Miscellaneous Methods	272
Constructors	272
Interning	273
String Conversions	275
The StringBuilder Class	275
Regular Expressions	277
Regex Introduction	277
The Regex Engine	279
Regex Pattern Language	280
The Regex Class	290
Files	298
File System Information	299
File IO	301
The .NET Console	304
Key Points	304

CHAPTER 12 Collections	305
Arrays	306
Copy	307
Sort	307
Search	311
Miscellaneous	312
Lists	313
Late-bound Lists	313
Early-bound Lists	315
Hash Tables	316
Late-bound Hashes	317
Early-bound Hashes	319
Stacks and Queues	319
Enumerations	320
Fundamentals	321
Threading	322
Multiple Enumerators	324
Delegates	325
Iterators	326
Other Collection Interfaces	326
Key Points	328
CHAPTER 13 Reflection	329
Run-time Type Information	329
Type Values	330
The typeof() Operator	331
GetType	333
Get Type by Name	334
Type Details	335
Member Access	335
Type Metadata	342
Assemblies	345
Emit	347
Key Points	352

CHAPTER 14	Serialization and Remoting	353
	Standard Streaming	354
	XML Streaming	358
	Different Representation	360
	Different Technology	360
	More Attributes	361
	SOAP Bubbles	362
	.NET Remoting	363
	Interprocess Communication	364
	Application Domains	368
	Key Points	372
CHAPTER 15	WinForms Basics	373
	Form Design and Loading	374
	Docking	376
	Events	379
	Event Multiplexing	379
	Low-level GUI Access	380
	Threads	381
	The Small Stuff	382
	The Biggest Small Stuff	382
	VCL-to-FCL Map	388
	Key Points	389
CHAPTER 16	Graphics	391
	Familiar, but Not Identical	391
	GDI+ Details	393
	Colors	393
	Pens	395
	Brushes	396
	Fonts and Text	400
	Bitmaps	405
	Paths and Regions	409
	Printing	410
	Key Points	411

CHAPTER 17	Threads and Synchronization	413
	Thread Basics	413
	Threads and Processes	414
	Synchronization	415
	.NET Threads	419
	Thread Priority	422
	Foreground and Background Threads	422
	Thread-local Storage	423
	Aborting and Interrupting Threads	424
	Synchronization	426
	Managed Locking	426
	The .NET “Memory Model”	429
	Interlocked Access	431
	Wait Handles	432
	Thread Pool	435
	Worker Threads	436
	Wait Callbacks	437
	GUI Issues	438
	Key Points	438
CHAPTER 18	XML	439
	XML Writer	439
	XML Reader	443
	The XML DOM	446
	XSLT	447
	Key Points	454
PART 4	Appendixes	
APPENDIX 0	Unsafe C# Code	457
APPENDIX 1	NUnit	465
APPENDIX 2	Assembly Loading and Signing	473

■ APPENDIX 3	Configuration Files	477
■ APPENDIX 4	Glossary	479
■ APPENDIX 5	Bibliography	485
■ INDEX	489

Table Cross-Reference

Table 2-1.	Set Operators and Their Bitmapped Equivalents	48
Table 3-1.	Results from the Chapter3\MakingTrouble Project	65
Table 5-1.	System Types	108
Table 5-2.	C# Operators That Are Different Than Delphi Operators	112
Table 5-3.	Operator Precedence	116
Table 5-4.	Symbolic Character Escapes	117
Table 5-5.	Hexadecimal Character Escapes	118
Table 7-1.	Constructor Syntax	163
Table 8-1.	Interface and Delegate Tradeoffs	188
Table 10-1.	Obsolete Features	246
Table 10-2.	Delphi Language Features	251
Table 10-3.	C# Language Features	252
Table 11-1.	Standard Numeric Formats	265
Table 11-2.	Standard <i>DateTime</i> Formats	266
Table 11-3.	Miscellaneous <i>String</i> Methods	272
Table 11-4.	Regex Pattern Characters	281
Table 11-5.	Perl-compatible Predefined Character Classes	282
Table 11-6.	Two-character Regex Assertions	283
Table 11-7.	Default <i>Regex</i> Behaviors, and Their <i>RegexOptions</i> Overrides	297
Table 11-8.	Selected <i>Path</i> Members	299
Table 12-1.	The Five Main Collection Interfaces	327
Table 13-1.	A Few Type Categorization Members	344
Table 15-1.	FCL Equivalents for Common VCL Constructs	389
Table 17-1.	A Race Condition	416
Table 17-2.	Deadlock	418
Table 17-3.	No Deadlock	418
Table A1-1.	Class (Test Fixture) Attributes	470
Table A1-2.	Method Attributes	470
Table A1-3.	NUnit Assertions	471

About the Author



JON SHEMITZ has been programming since he was 12, when he learned Focal on a PDP-8. He's been programming professionally since he graduated from Yale in 1981, and has done everything from shrink-wrap programming to consulting. Jon has used Borland Pascals since Turbo Pascal 1, and has been doing .NET programming in C# since 2002. This is Jon's second book: he's written dozens of programming articles; contributed to four other books; and has given programming talks on two continents.

Jon does contract programming, consulting, and training—you can contact him at www.midnightbeach.com.

About the Technical Reviewer



■ **HALLVARD VASSBOTN** is a senior systems developer at and partial owner of Infront AS (www.infront.no), developing state-of-the-art real-time financial information and trading systems (www.theonlinetrader.com). Hallvard has been a professional programmer since 1985, and has written numerous articles for *The Delphi Magazine* and tech edited several popular Delphi books.

You can read his technical blog at hallvards.blogspot.com/.

Hallvard lives in Oslo, Norway, with the three diamonds of his heart, Nina, Ida, and Thea. You can reach him at hallvardvassbotn@c2i.net.

Acknowledgments

This book represents a lot of effort over several years. It wouldn't have been possible without the help of many talented people—most of whom I've never met.

Dan Appleman and the editorial board at Apress had the good taste to agree that a book about .NET for Delphi programmers would sell better than the Delphi for .NET reference that they originally agreed to publish. More importantly, they've been willing to wait for a book that's longer and later than they originally expected. Sofia Marchant, my project manager, has been answering my questions for nearly three-and-a-half years, and she put together a production team that smoothly and painlessly turned my 4 meg of Word and TIF files into a printed book. Ami Knox, the copy editor, made my punctuation and capitalization consistent, and caught awkward phrases that made it past all my rewrites; I'd especially like to thank Ami for the extra effort involved in dealing with my 'scare quotes.' Liz Welch, the proofreader, did a great job transferring the syntax highlighting from my manuscript to the printed page, and she also caught several mistakes that made it past both Ami and me.

Google made it much easier for me to answer various questions. Without Google, the research would have taken much longer, and I might never even have found some of the more interesting details. VMWare generously supplied me with a free "authors copy" of *VMware Workstation*, which made it much easier (and safer) to install and uninstall beta software. My orthopedist, Dr. Howard Schwartz, was very patient with my impatience when a bike accident tore shoulder ligaments and disabled me for three months near the end of the first draft.

I have the good fortune to live with three fine writers—my partner, Tané Tachyon, and our sons, Sam and Arthur Shemitz. All three of them have had to put up with innumerable problem paragraphs, and inevitably made good suggestions that helped move things along.

Weyert de Boer, Mark Boler, Marcel van Brakel, Alessandro Federici, Marc Hoffman, Chuck Jazdzewski, Ian Marteens, Jon Skeet, and Danny Thorpe all read drafts of various chapters. Their comments helped improve the book, and their positive feedback helped keep me working.

I'd particularly like to thank Jon Skeet and Marcel van Brakel. Jon helped me understand the .NET *memory model* (Chapter 17) and how it affects interprocessor synchronization. Marcel read every chapter at least once, and made detailed comments on most of them. I've benefited greatly from his deep knowledge, his helpful suggestions, and his uncanny ability to find every point where I waved my hands vaguely and hoped no one would notice.

Finally, I can't say enough about Hallvard Vassbotn, my technical reviewer. This project was much more work (and took much longer) than he could possibly have anticipated when he signed on, yet he read every chapter two or three times—and caught errors and made suggestions, each time. Hallvard also wrote the Delphi syntax chapter when I was considering dropping it after my bike accident. I've enjoyed working with him, and have been thoroughly impressed by his intelligence, energy, and diligence. Naturally, any mistakes that remain are entirely my fault.

April 2006
Santa Cruz, California

Preface

It's rough being a Delphi programmer. We know we have a wonderful, productive environment—but jobs are few and far between. We know that we can write any sort of application with Delphi—yet Delphi is seen as a GUI builder and a database front-end. We've all seen (or at least heard of) systems where the 'interesting parts' are written in C or C++, in DLLs, and Delphi is just used for the GUI interface. We may know C++ and have significant Win32 experience—and yet not been considered for C++ jobs because we didn't know MFC or ATL.

.NET changes that. All .NET languages use the same Framework Class Library (FCL). Learn the FCL—in any language—and you're a .NET programmer. “Learn once, work anywhere.”

What split the Windows programming world into mutually incompatible Delphi shops, VB shops, and C++ shops was never the languages themselves. Picking up any particular language has always been easy. The barriers to entry have always been the different libraries. Using a different language meant learning a new library. Learning a new library meant that every little thing required a documentation search; your productivity was near zero for weeks on end. But with .NET, once you learn the Framework Classes, you can easily move from project to project and from job to job.

What's more, in this bigger, broader job market, Delphi skills are a big advantage. .NET is not a knock-off or successor to Delphi, and there are significant differences between Delphi and .NET—but .NET is a lot like Delphi. .NET has components, events, exceptions, interfaces, properties, and objects that descend via single inheritance from a common ancestor. All just like Delphi. .NET has more in common with Delphi than it does with either MFC, ATL, or VB, and so Delphi programmers will find .NET easier to learn than VB or C++ programmers will.

This book presents .NET from a Delphi programmer's viewpoint. It doesn't ask you to plow through things you already know in the hopes of picking up a few choice bits of new information; it presents the core concepts of the .NET world in terms of the Delphi concepts you're familiar with. The examples are in either C# or Delphi, not both—unless I'm trying to highlight a syntax difference.

From your employer's point of view, .NET offers managed code plus most of Delphi's traditional productivity advantages, without Delphi's traditional drawback of being a niche product that few programmers know. From your point of view, .NET offers something like a hundred times as many possible jobs—and it puts the fun back in programming. Garbage collection frees us from the tyranny of Free What You Create and all the petty discipline of avoiding memory leaks. We can write functions that return objects; we never have to worry about a “tombstoned pointer” to a prematurely freed object leading to memory corruption.

.NET is fun. .NET is productive. .NET offers what you've always loved about Delphi, without locking you into a narrow ghetto. This book will help you transfer your Delphi skills to the broader, brighter world outside the ghetto walls.

Organization

As I wrote this book, I tried to write the book I wish I'd had when I was learning .NET. I tried to remember what I found confusing, and what key points made for Aha! moments. At the same time, I imagined the reactions of people I've worked with, or met online, and went into more detail on the points where they would be confused or argumentative. Hopefully, the result will spare you a lot of trial and error.

I assume you know Delphi well enough to get paid to write it, but I've tried very hard to avoid ambiguity and unexplained jargon. You should be able to read this book straight through, and understand it all well enough to go out and get yourself in trouble—you should not have to reread any section two or three times before it makes sense. (I also know that many people will **not** read the book straight through, and have provided plenty of parenthetical cross-references for the reader who wants to skip around, or who will only open the book on an “as needed” basis.)

The first part of the book is for a native code programmer (i.e., Win32 or Linux) with no *managed code* experience. Chapter 1 explains what managed code is, and how it makes you even more productive than you are with Delphi. Chapter 2 introduces the .NET programming model, and how it differs from the familiar Delphi programming model. Chapter 3 has garbage collection details, while Chapter 4 goes into more detail about how Just In Time (JIT) compilation works, and why .NET uses JIT compilation. Most of the examples in the first four chapters are in Delphi, except where I use a little C# to introduce generics in Chapter 2.

The second part of the book is (mostly) a Delphi programmer's introduction to C#. While you can probably decipher C# examples on your own, I think you'll find that Part 2 makes it easier—and that reading the C# chapters will make it **much** easier to actually write C#.

The third part of the book covers the .NET Framework Class Library, or FCL. This part is nearly as long as the first two parts put together, and is very much the heart of the book. The Microsoft documentation is a fine reference when you know what class to use, but it's not a particularly good introduction. I've tried to provide the conceptual overview that you need to make sense of the documentation and/or to ask questions that Google can answer. After reading Chapters 11 through 18, you should understand the FCL design philosophy well enough that you'll find it easy to learn new parts of the library. There are Delphi examples in every chapter, but most of the FCL examples are in C#.

■ **Note** This is not a Delphi book—this is a book about .NET, for Delphi programmers.

Typography

Inline code looks like this, and I use bold for **emphasis** and italics as a sort of quote, to introduce *new terms*.

I also distinguish single quotes from double quotes. A double quote is a ‘strong’ or ‘true’ quote, while a single quote is a *scare quote*—a ‘weak’ or ‘sort of’ quote. (Other former philosophy majors will find this convention familiar; my copy editor suggested that I may need to explain it to everyone else.)

That is, if I say that Benjamin Franklin said “Thank you,” I’m saying that I am 100% sure that Benjamin Franklin said “Thank you” on at least one occasion. I use double quotes when I’m actually quoting something I’ve read or heard. By contrast, if I say that Benjamin Franklin said ‘Those who will sacrifice Freedom for the sake of Security will soon find they have Neither,’ I’m saying that Benjamin Franklin said something like that. I use single quotes when I’m paraphrasing, or when I’m using slang or a neologism.

The Sample Code

There are over 150 sample projects mentioned in this book. For the most part, I only print the few most interesting lines of each. In some cases, I don’t even do that—I describe a technique, and refer you to a sample project for the details. To run the projects and/or read the code that I don’t print, you’ll have to download the appropriate zip file from the Apress web site, and install it on a machine with a .NET development environment.

You can get the code by going to the Source Code section of the Apress web site, www.apress.com, where you’ll find complete download and installation instructions. I urge you to download the sample code. Reading the code and pressing F1 on various identifiers is a great way to dip into the .NET documentation. More importantly, while I’ve made every effort to keep the book self-contained so that you can read it away from a computer, some techniques are best grasped by experimentation. Using my working code as a starting point can be very helpful here. (Most of the projects are just snippets that demonstrate a single point, but there are a few that contain code you may want to borrow.) All the sample code—from the code that demonstrates various useful techniques to the utility units in my `common` directory—is distributed under a license that lets you use my code in any way you like, so long as you leave my copyright notice in the source code.