Expert C# 2008 Business Objects

Rockford Lhotka

Apress®

Expert C# 2008 Business Objects

ISBN-13 (pbk): 978-1-4302-1019-1

ISBN-13 (electronic): 978-1-4302-1020-7

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

Editor: Matthew Moodie
Technical Reviewers: James Miller, Andrés Villanueva, Joe Fallon
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Richard Dal Porto
Copy Editors: Nicole Abramowitz, Jennifer Whipple
Associate Production Director: Kari Brooks-Copony
Production Editor: Laura Esterman
Compositor: Susan Glinert
Proofreader: Nancy Bell
Indexer: John Collin
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 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `http://www.apress.com/info/bulksales`.

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 `http://www.apress.com`.

To my Mom and Dad. Thank you for all you’ve taught me through the years!

Contents at a Glance

Contents

About the Author

ROCKFORD LHOTKA is the author of numerous books. He is a Microsoft Regional Director, a Microsoft Most Valuable Professional (MVP), and an INETA speaker. He contributes to several major magazines and presents regularly at major conferences around the world, including Microsoft Tech Ed and VS Live. Rockford is the Principal Technology Evangelist for Magenic (`www.magenic.com`), one of the nation's premier Microsoft Gold Certified Partners that is focused on delivering business value through applied technology. For more information, go to `www.lhotka.net`.

About the Technical Reviewers

JAMES MILLER is a senior architect and technical evangelist specializing in enterprise solutions. He has worked in multiple industries and capacities in both the public and private sectors, and he has more than 25 years of programming experience under his belt. He has embraced CSLA since 2002 and has been an active proponent of the framework in his career and on the CSLA forums. He is currently working with an ISV, leading an international group of developers to upgrade their products to the latest .NET technologies, practices, tools, and techniques, while espousing the virtues of OOP, SOA, TDD, agile development, and CSLA. Jim is a proud graduate of the University of Michigan, is Microsoft certified in both VB .NET and C# for Windows and Web-based applications, and jumped at the chance to contribute to the next version of the CSLA framework.

Jim lives in a rural area outside Ann Arbor, Michigan, with his wife, five children, four cats, and three dogs. He fills much of his limited free time as the head coach of his local high school's junior varsity boys' lacrosse team. Jim still has an electric guitar plugged in over in a corner, a bookshelf filled with tech books, and a comfortable chair on the deck, perfect for viewing the deer as they meander by.

ANDRÉS VILLANUEVA is a consultant/developer living in Argentina. After a start in IT at age 15, he moved to the software industry, coding with Visual FoxPro and Visual Basic 6. In 2004, Andrés moved on to .NET and hasn't looked back. His early software experiences were in the banking industry, where he quickly rose as a leader, helping his firm improve consistency by implementing the CSLA framework. Since those days, he has made the leap into the consulting world and now provides software services from his office in Argentina to various clients around the world. He is an open source software fan and the current lead on the CslaGenerator project—an open source code-generation tool that targets development on the CSLA framework. In his little free time, Andrés enjoys playing soccer and relaxing with jazz music.

JOE FALLON is the Director of Framework Development at PurchasingNet, Inc., and is responsible for the development and implementation of the.NET Framework for PNet products. Prior to joining PurchasingNet, he worked at Nestle Chocolate and Confection as an assistant plant engineer after completing five years of service in the U.S. Army as a captain in the field artillery. During his tenure at Nestle USA, he held various positions in industrial engineering and IT.

Joe graduated from the United States Military Academy at West Point, New York, and was the 1981 recipient of the General Omar Bradley Award as the Academy's no. 1 mathematics major. He has been a Microsoft MVP for eight years in a row.

Acknowledgments

This book is a major update to the previous edition. This book, and CSLA .NET 3.6, exist thanks to a lot of work from many people.

I need to acknowledge the support, patience, and love from my wife and sons over the past many years. Without you, this would have been impossible.

I'd also like to thank Greg Frankenfield and Paul Fridman for making Magenic such an awesome place to work. The support that you and the rest of Magenic have provided has been great, and I appreciate it very much. It is an honor to work with everyone there.

CSLA .NET 3.6 is the result of a lot of work by several Magenic colleagues, including Sergey Barskiy, Justin Chase (now at Microsoft), Aaron Erickson, and Nermin Dibek. Jon Stonecash, Mark Steinberg, Grant Breems, and Chris Williams contributed as well. Sandy Fougerousse created the CSLA .NET for Silverlight logo.

A number of people outside Magenic also contributed to CSLA .NET 3.6, including Ricky Supit, Mark Chesney, and Miguel Castro. The CSLA .NET for Windows logo was contributed by Chris Russi.

The Apress editorial team went above and beyond to help shape this book into what you see here and to help get it done as rapidly as possible. I owe them all a debt of gratitude for their fine work.

Finally, I'd like to thank the scores of people who've sent me emails, posted on the forum and my blog with messages of support and encouragement, or just plain asked when the book would be done. The great community that has grown around these books and the CSLA .NET framework is wonderful, and I thank you all. I hope you find this book to be as rewarding to read as it has been for me to write.

Code well and have fun!

Introduction

I have a passion for frameworks. In more than 20 years as a professional developer, I've never worked on a computing platform that did everything I needed it to do to build applications productively. The Microsoft .NET platform is wonderful, but it doesn't always do quite what I want or need. To address those needs, I'm always looking for tools and frameworks, and sometimes I end up creating them myself.

A framework is simply the codification of an architecture or design pattern. Before you can have a good framework, you need to have an architecture. That means you need to have a vision and a set of goals both for the architecture and the kinds of applications it should enable.

This book is about application architecture, design, and development in .NET using object-oriented concepts. The focus is on creating *business objects* and implementing them to work in various distributed environments, including web and client/server configurations. The book makes use of a great many .NET technologies, object-oriented design and programming concepts, and distributed architectures.

Much of the book walks through the thought process I used in designing and creating the CSLA .NET framework to support object-oriented application development in .NET. This includes a lot of architectural concepts and ideas. It also involves some in-depth use of advanced .NET techniques to create the framework.

The book also shows how to make use of the framework to build a sample application with several different interfaces. If you wish, you could skip the framework design chapters and simply make use of the framework to build object-oriented applications.

One of my primary goals in creating the CSLA .NET framework was to simplify .NET development. Developers using the framework in this book don't need to worry about the details of underlying technologies such as remoting, serialization, or reflection. All of these are embedded in the framework, so that a developer using it can focus almost entirely on business logic and application design rather than on getting caught up in "plumbing" issues.

This book is a major update to the previous edition, *Expert C# 2005 Business Objects*. This updated book takes advantage of new features of .NET 3.5 and applies lessons learned by using .NET 2.0 and 3.0 over the past few years.

This book is the most recent expression of concepts I've been working on for more than a dozen years. My goal all along has been to enable the productive use of object-oriented design in distributed n-tier applications. Over the years, both the technologies and my understanding and expression of the concepts have evolved greatly.

From CSLA .NET 2.0 to 3.6

Over the past eight years, the CSLA .NET framework has become one of the most widely used development frameworks on the Microsoft .NET platform. Since I introduced the .NET version in 2001, the framework has grown and evolved quite a lot, in part due to changes to the .NET platform itself, and in part due to feedback from the vibrant community surrounding CSLA .NET.

The CSLA .NET framework is a reflection of an underlying architecture I call *CSLA*, for component-based, scalable, logical architecture. Over the years, I've received hundreds of emails from people who have used CSLA as a basis for their own architectures, as they've built applications ranging from small, single-user programs to full-blown enterprise applications that power major parts of their businesses.

This framework addresses two primary areas of object-oriented software development:

- How to use business objects to efficiently build Windows, web, and service-oriented applications
- How to enable the use of object-oriented design in a distributed computing environment

While .NET supports the use of objects, the author of an object has to do a lot of work to fully support important .NET concepts such as data binding. Much of the focus of CSLA .NET and of this book is on enabling objects to fully support data binding, as well as on other important concepts such as validation and authorization. For most users of CSLA .NET, these are the primary benefits that the framework provides.

Many people build distributed n-tier or service-oriented applications. Using object-oriented design and business objects in a distributed environment has its own challenges, and CSLA .NET uses various techniques to overcome those challenges. For n-tier client/server applications, the framework supports the idea of *mobile objects*—objects that actually move between computers in an n-tier environment. Mobile objects provide a powerful way to implement object-oriented designs in distributed environments. For service-oriented applications, CSLA .NET can be used to build both edge applications and services. The framework is compelling for edge application creation and is often useful for creating services or workflow activities as well.

As the .NET platform and the CSLA .NET framework have evolved, I've made a great many changes and added many new features. In some cases, using the new concepts and features has required making changes to existing business objects and user interface code. I don't take backward compatibility lightly, yet it is important to advance the concepts to keep up with both changes in technology and my views on both object-oriented and distributed computing.

When possible, I have minimized the impact on existing code, so the transition shouldn't be overly complex for most applications. Although there are a few breaking changes from version 3.0 to 3.6, *most* existing code should upgrade easily. Even version 2.1 code should upgrade with relative ease. Business classes written with CSLA .NET versions 1.x or 2.0 will require quite a bit of effort to bring forward.

Over the years, I've received a handful of emails from people for whom CSLA .NET *wasn't* successful, but this isn't surprising. To use CSLA .NET effectively, you must become versed in object-oriented design, understand the concept of mobile objects, and develop a host of other skills. The mobile object architecture has many benefits, but it's not the simplest or the easiest to understand.

However, over that same period of time, I've received countless emails from people who have had tremendous success in building applications using CSLA .NET. These applications range from Windows to web, from small to enterprise, from retail to manufacturing to military environments. I am amazed, pleased, and humbled by these emails and by all the cool places where CSLA .NET has helped organizations and individuals around the world.

Designing CSLA .NET

One of the characteristics of .NET is that it often provides several ways to solve the same problem. Some of the available approaches are better than others, but the best one for a given problem may not be immediately obvious. Over the past eight years, I've spent a lot of time researching many of these options and techniques. Although a variety have proven to work, in the end I've arrived at the one that best matches my original goals.

I have a specific set of goals for the architecture and the book. These goals are important, because they're key to understanding why I made many of the choices I did in terms of which .NET technologies to use and how to use them. The goals are as follows:

- To support a fully object-oriented programming model
- To allow the developer to use the architecture without jumping through hoops
- To enable high scalability
- To enable high performance
- To enable developer productivity when using business objects, including:
  - Support for data binding in Windows and Web Forms
  - Support for many types of UIs based on the same objects
  - Management of validation rules
  - Management of authorization rules
  - N-level undo on a per-object basis (edit, cancel, apply)
  - Integration with distributed transaction technologies such as Enterprise Services and `System.Transactions`
- To support the use of object-oriented design in a distributed environment through the use of mobile objects
- To simplify .NET by handling complex issues such as serialization, reflection, and network communication
- To use the tools provided by Microsoft—notably IntelliSense and the Autocomplete feature in Visual Studio .NET

Of these, saving the developer from jumping through hoops—that is, allowing him or her to do "normal" programming—has probably had the largest impact. To meet all these goals without a framework, the developer would have to write a lot of extra code to track business rules, implement n-level undo, and support serialization of object data. All this code is important, but it adds nothing to the business value of the application.

Fortunately, .NET offers some powerful technologies that help to reduce or eliminate much of this "plumbing" code. If those technologies are then wrapped in a framework, a business developer shouldn't have to deal with them at all. In several cases, this goal of simplicity drove my architectural decisions. The end result is that the developer can, for the most part, simply write a standardized C# class and have it automatically enjoy all the benefits of n-level undo, business rule tracking, and so forth.

It has taken a great deal of time and effort, but I've certainly enjoyed putting this architecture and this book together, and I hope that you will find both valuable during the development of your own applications.

Framework License

LICENSE AND WARRANTY

You can use this Software for any noncommercial purpose, including distributing derivative works.

You can use this Software for any commercial purpose other than you may not use it, in whole or in part, to create a commercial framework product.

In short, you can use CSLA .NET and modify it to create other commercial or business software, you just can't take the framework itself, modify it and sell it as a product.

In return, the owner simply requires that you agree:

This Software License Agreement ("Agreement") is effective upon your use of CSLA .NET ("Software").

1. Ownership.
The CSLA .NET framework is Copyright 2008 by Rockford Lhotka, Eden Prairie, MN, USA.

2. Copyright Notice.
You must not remove any copyright notices from the Software source code.

3. License.
The owner hereby grants a perpetual, non-exclusive, limited license to use the Software as set forth in this Agreement.

4. Source Code Distribution.
If you distribute the Software in source code form you must do so only under this License (i.e. you must include a complete copy of this License with your distribution).

5. Binary or Object Distribution.
You may distribute the Software in binary or object form with no requirement to display copyright notices to the end user. The binary or object form must retain the copyright notices included in the Software source code.

6. Restrictions.
You may not sell the Software. If you create a software development framework based on the Software as a derivative work, you may not sell that derivative work. This does not restrict the use of the Software for creation of other types of non-commercial or commercial applications or derivative works.

7. Disclaimer of Warranty.
The Software comes “as is”, with no warranties. None whatsoever. This means no express, implied, statutory or other warranty, including without limitation, warranties of merchantability or fitness for a particular purpose, noninfringement, or the presence or absence of errors, whether or not discoverable. Also, you must pass this disclaimer on whenever you distribute the Software.

8. Liability.
Neither Rockford Lhotka nor any contributor to the Software will be liable for any of those types of damages known as indirect, special, consequential, incidental, punitive or exemplary related to the Software or this License, to the maximum extent the law permits, no matter what legal theory it’s based on. Also, you must pass this limitation of liability on whenever you distribute the Software.

9. Patents.
If you sue anyone over patents that you think may apply to the Software for a person’s use of the Software, your license to the Software ends automatically. The patent rights, if any, licensed hereunder only apply to the Software, not to any derivative works you make.

10. Termination.
Your rights under this License end automatically if you breach it in any way. Rockford Lhotka reserves the right to release the Software under different license terms or to stop distributing the Software at any time. Such an election will not serve to withdraw this Agreement, and this Agreement will continue in full force and effect unless terminated as stated above.

11. Governing Law.
This Agreement shall be construed and enforced in accordance with the laws of the state of Minnesota, USA.

12. No Assignment.
Neither this Agreement nor any interest in this Agreement may be assigned by Licensee without the prior express written approval of Developer.

13. Final Agreement.
    This Agreement terminates and supersedes all prior understandings or agreements on the subject matter hereof. This Agreement may be modified only by a further writing that is duly executed by both parties.
14. Severability.
    If any term of this Agreement is held by a court of competent jurisdiction to be invalid or unenforceable, then this Agreement, including all of the remaining terms, will remain in full force and effect as if such invalid or unenforceable term had never been included.
15. Headings.
    Headings used in this Agreement are provided for convenience only and shall not be used to construe meaning or intent.

What You Need to Use This Book

The code in this book has been verified to work against Microsoft Visual Studio 2008 Professional Edition SP1 and against version 3.5 SP1 of the .NET Framework. The database is a SQL Server Express database, which is included with Visual Studio 2008 Professional. The Enterprise version of Visual Studio 2008 and the full version of SQL Server are useful but not necessary.

In order to run the tools and products listed previously, you'll need at least one PC with Windows Vista, Windows XP SP2 (or higher), Windows Server 2003, or Windows Server 2008 installed. To test CSLA .NET's support for multiple physical tiers, of course, you'll need an additional PC (or you can use Virtual PC or a similar tool) for each tier that you wish to add.

How This Book Is Structured

This book covers the thought process behind the CSLA .NET for Windows version 3.6 architecture. It describes the construction of the framework that supports the architecture, and it demonstrates how to create WPF, Web Forms, and WCF service applications based on business objects written using the framework.

If you are reading this book to understand the process of designing and constructing a development framework for the .NET platform, then you should read all chapters. If you are reading this book to understand how to use the CSLA .NET framework and are less interested in how the framework itself is designed and implemented, then you should read Chapters 1 through 5 and Chapters 17 through 21.

Chapter 1 introduces some of the concepts surrounding distributed architectures, including logical and physical architectures, business objects, and distributed objects. Perhaps more importantly, this chapter sets the stage, showing the thought process that results in the remainder of the book.

Chapter 2 takes the architecture described at the end of Chapter 1 and uses it as the starting point for a code framework that enables the goals described earlier. By the end of the chapter, you'll have seen the design process for the objects that will be implemented in Chapters 6 through 16; but before that, there's some other business to attend to.

In Chapter 3, I discuss the basics of responsibility-driven object-oriented design. As an example, this chapter lays out the requirements and design for a sample application.

Chapters 4 and 5 discuss how to use each of the primary base classes in the CSLA .NET framework to create your own business objects. I discuss in detail the object-oriented stereotypes supported by the CSLA .NET base classes, along with the code structure for editable and read-only objects, and collections and name/value lists.

Chapters 6 through 16 are all about the construction of the CSLA .NET framework itself. If you're interested in the code behind property declarations, validation rules, authorization rules, n-level undo, mobile object support, and object persistence, then these are the chapters for you. In addition, they make use of some of the more advanced and interesting parts of the .NET Framework, including data binding, serialization, reflection, dynamic method invocation, WCF, .NET security, Enterprise Services, `System.Transactions`, strongly named assemblies, dynamically loaded assemblies, application configuration files, and more.

Chapters 17 and 18 create the business objects for the application. These chapters illustrate how you can use the framework to create a powerful set of business objects rapidly and easily for an application. The end result is a set of objects that not only model business responsibilities, but also support data binding, validation, authorization, n-level undo, and various physical configurations that can optimize performance, scalability, security, and fault tolerance, as discussed in Chapter 1.

Chapter 19 demonstrates how to create a WPF interface to the business objects. Chapter 20 covers the creation of an ASP.NET Web Forms interface with comparable functionality.

Chapter 21 shows how to build WCF services using business objects. This approach enables service-oriented development by providing a programmatic interface to the business objects that any web service or WCF client can call.

By the end, you'll have a framework that supports object-oriented application design in a practical, pragmatic manner. The framework implements a logical model that you can deploy in various physical configurations to optimally support Windows, web, and XML service clients.

Downloading the Code

The code that reflects the contents of this book is available in the Source Code/Download area of the Apress website (`www.apress.com`). For the latest version of the framework and the example application, visit `www.lhotka.net/cslanet/download.aspx`.

Contacting the Author

You may reach Rockford Lhotka on his website, `www.lhotka.net`, which contains his blog, information about the framework and book, and his contact information.