The Addison-Wesley Microsoft Technology Series

LINQ to Objects Using C# 4.0

Using and Extending LINQ to Objects and Parallel LINQ (PLINQ)

TROY MAGENNIS

Foreword by BARRY VANDEVIER,
Chief Information Officer, Sabre Holdings

LINQ to Objects Using C# 4.0

Using and Extending LINQ to Objects and Parallel LINQ (PLINQ)

Troy Magennis

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data:

Magennis, Troy, 1970-
LINQ to objects using C# 4.0 : using and extending LINQ to objects and parallel LINQ (PLINQ) / Troy Magennis.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-321-63700-0 (pbk. : alk. paper) 1. Microsoft LINQ. 2. Query languages (Computer science) 3. C#
(Computer program language) 4. Microsoft .NET Framework. I. Title.
QA76.73.L228M345 2010
006.7'882—dc22
2009049530

ISBN-13: 978-0-321-63700-0
ISBN-10: 0-321-63700-3

Text printed in the United States on recycled paper at RR Donnelly in Crawfordsville, Indiana.

First printing March 2010

To my wife, Janet Doherty, for allowing me to spend those extra hours tapping away on the keyboard; thank you for your support and love.

This page intentionally left blank

CONTENTS

FOREWORD

I have worked in the software industry for more than 15 years, the last four years as CIO of Sabre Holdings and the prior four as CTO of Travelocity. At Sabre, on top of our large online presence through Travelocity, we transact $70 billion in annual gross travel sales through our network and serve over 200 airline customers worldwide. On a given day, we will process over 700 million transactions and handle 32,000 transactions per second at peak. Working with massive streams of data is what we do, and finding better ways to work with this data and improve throughput is my role as CIO.

Troy is our VP over Architecture at Travelocity, where I have the pleasure of watching his influence on a daily basis. His perspective on current and future problems and depth of detail are observed in his architectural decisions, and you will find this capability very evident in this book on the subject of LINQ and PLINQ.

Developer productivity is a critical aspect for every IT solution-based business, and Troy emphasizes this in every chapter of his book. Languages and language features are a means to an end, and language features like LINQ offer key advances in developer productivity. By simplifying all types of data manipulation by adding SQL-style querying within the core .NET development languages, developers can focus on solving business problems rather than learning a new query language for every data source type. Beyond developer productivity, the evolution in technology from individual processor speed improvements to multi-core processors opened up a big hole in run-time productivity as much of today's software lacks investment in parallelism required to better utilize these new processors. Microsoft's investment in Parallel LINQ addresses this hole, enabling much higher utilization of today's hardware platforms.

Open-standards and open-frameworks are essential in the software industry. I'm pleased to see that Microsoft has approached C# and LINQ in an open and inclusive way, by handing C# over as an ECMA/ISO

standard, allowing everyone to develop new LINQ data-sources and to extend the LINQ query language operators to suit their needs. This approach showcases the traits of many successful open-source initiatives and demonstrates the competitive advantages openness offers.

Decreasing the ramp-up speed for developers to write and exploit the virtues of many-core processors is extremely important in today's world and will have a very big impact in technology companies that operate at the scale of Sabre. Exposing common concurrent patterns at a language level offers the best way to allow current applications to scale safely and efficiently as core-count increases. While it was always possible for a small percentage of developers to reliably code concurrency through OpenMP or hand-rolled multi-threading frameworks, parallel LINQ allows developers to take advantage of many-core scalability with far fewer concerns (thread synchronization, data segmentation, merging results, for example). This approach will allow companies to scale this capability across a much higher percentage of developers without losing focus on quality. So roll up your sleeves and enjoy the read!

—Barry Vandevier
Chief Information Officer, Sabre Holdings

PREFACE

LINQ to Objects Using C# 4.0 takes a different approach to the subject of Language Integrated Query (LINQ). This book focuses on the LINQ syntax and working with in-memory collections rather than focusing on replacing other database technologies. The beauty of LINQ is that once you master the syntax and concepts behind how to compose clever queries, the underlying data source is mostly irrelevant. That's not to say that technologies such as LINQ to SQL, LINQ to XML, and LINQ to Entities are un-important; they are just not covered in this book.

Much of the material for this book was written during late 2006 when Language Integrated Query (LINQ) was in its earliest preview period. I was lucky enough to have a window of time to learn a new technology when LINQ came along. It became clear that beyond the clever data access abilities being demonstrated (DLINQ at the time, LINQ to SQL eventually), LINQ to Objects would have the most impact on the day-to-day developers' life. Working with in-memory collections of data is one of the more common tasks performed, and looking through code in my previous projects made it clear just how complex my for-loops and nested if-condition statements had evolved. LINQ and the language enhancements being proposed were going to change the look and feel of the way we programmed, and from where I was sitting that was fantastic.

The initial exploration was published on the HookedOnLINQ.com Wiki (120 odd pages at that time), and the traffic grew over the next year or two to a healthy level. Material could have been pulled together for a publication at that time (and been first to market with a book on this subject, something my Addison-Wesley editor will probably never forgive me for), but I felt knowing the syntax and the raw operators wasn't a book worth reading. It was critical to know how LINQ works in the real world and how to use it on real projects before I put that material into ink. The first round of books for any new programming technology often go slightly deeper than the online-documentation, and I wanted to wait and see how

the LINQ story unfolded in real-world applications and write the first book of the second-generation—the book that isn't just reference, but has integrity that only real-world application can ingrain.

The LINQ story is a lot deeper and has wider impact than most people realize at first glance of any TechEd session recording or user-group presentation. The ability to store and pass code as a data structure and to control when and how that code is executed builds a powerful platform for working with all matter of data sources. The few LINQ providers shipped by Microsoft are just the start, and many more are being built by the community through the extension points provided. After mastering the LINQ syntax and understanding the operators' use (and how to avoid misuse), any developer can work more effectively and write cleaner code. This is the purpose of this book: to assist the reader in beginning the journey, to introduce how to use LINQ for more real-world examples and to dive a little deeper than most books on the subject, to explore the performance benefits of one solution over another, and to deeply look at how to create custom operators for any specific purpose.

I hope you agree after reading this book that it does offer an insight into how to use LINQ to Objects on real projects and that the examples go a step further in explaining the patterns that make LINQ an integral part of day-to-day programming from this day forward.

Who Should Read This Book

The audience for this book is primarily developers who write their applications in C# and want to understand how to employ and extend the features of LINQ to Objects. LINQ to Objects is a wide set of technology pieces that work in tandem to make working with in-memory data sources easier and more powerful. This book covers both the initial C# 3.0 implementation of LINQ and the updates in C# 4.0. If you are accustomed to the LINQ syntax, this book goes deeper than most LINQ reference publication and delves into areas of performance and how to write custom LINQ operators (either as sequential algorithms or using parallel algorithms to improve performance).

If you are a beginning C# developer (or new to C# 3.0 or 4.0), this book introduces the code changes and syntax so that you can quickly master working with objects and collections of objects using LINQ. I've tried to

strike a balance and not jump directly into examples before covering the basics. You obviously should know how to build a LINQ query statement before you start to write your own custom sequential or parallel operators to determine the number of mountain peaks around the world that are taller than 8,000 meters (26,000 feet approximately). But you will get to that in the latter chapters.

Overview of the Book

LINQ to Objects Using C# 4.0 starts by introducing the intention and benefits LINQ offers developers in general. Chapter 1, "Introducing LINQ," talks to the motivation and basic concepts LINQ introduces to the world of writing .NET applications. Specifically, this chapter introduces before and after code makeovers to demonstrate LINQ's ability to simplify coding problems. This is the first and only chapter that talks about LINQ to SQL and LINQ to XML and does this to demonstrate how multiple LINQ data sources can be used from the one query syntax and how this powerful concept will change application development. This chapter concludes by listing the wider benefits of embracing LINQ and attempts to build the big picture view of what LINQ actually is, a more complex task than it might first seem.

Chapter 2, "Introducing LINQ to Objects," begins exploring the underlying enabling language features that are necessary to understand how the LINQ language syntax compiles. A fast-paced, brief overview of LINQ's features wraps up this chapter; it doesn't cover any of them in depth but just touches on the syntax and capabilities that are covered at length in future chapters.

Chapter 3, "Writing Basic Queries," introduces reading and writing LINQ queries in C# and covers the basics of choosing what data to project, in what format to select that data, and in what order the final result should be placed. By the end of this chapter, each reader should be able to read the intention behind most queries and be able to write simple queries that filter, project, and order data from in-memory collections.

Chapter 4, "Grouping and Joining Data," covers the more advanced features of grouping data in a collection and combining multiple data sources. These partitioning and relational style queries can be structured and built in many ways, and this chapter describes in depth when and why to use one grouping or joining syntax over another.

Chapter 5, "Standard Query Operators," lists the many additional standard operators that can be used in a LINQ query. LINQ has over 50 operators, and this chapter covers the operators that go beyond those covered in the previous chapters.

Chapter 6, "Working with Set Data," explores working with set-based operators. There are multiple ways of performing set operations over in-memory collections, and this chapter explores the merits and pitfalls of both.

Chapter 7, "Extending LINQ to Objects," discusses the art of building custom operators. The examples covered in this chapter demonstrate how to build any of the four main types of operators and includes the common coding and error-handling patterns to employ in order to closely match the built-in operators Microsoft supplies.

Chapter 8, "C# 4.0 Features," is where the additional C# 4.0 language features are introduced with particular attention to how they extend the LINQ to Objects story. This chapter demonstrates how to use the dynamic language features to make LINQ queries more fluent to read and write and how to combine LINQ with COM-Interop in order to use other applications as data sources (for example, Microsoft Excel).

Chapter 9, "Parallel LINQ to Objects," closely examines the motivation and art of building application code that can support multi-core processor machines. Not all queries will see a performance improvement, and this chapter discusses the expectations and likely improvement most queries will see. This chapter concludes with an example of writing a custom parallel operator to demonstrate the thinking process that goes into correctly coding parallel extensions in addition to those provided.

Conventions

There is significant code listed in this book. It is an unavoidable fact for books about programming language features that they must demonstrate those features with code samples. It was always my intention to show lots of examples, and every chapter has dozens of code listings. To help ease the burden, I followed some common typography conventions to make them more readable. References to classes, variables, and other code entities are distinguished in a `monospace` font. Short code listings that are to be read

inline with the surrounding text are also presented in a `monospace` font, but on their own lines, and they sometimes contain code comments (lines beginning with `//` characters) for clarity.

```
// With line-breaks added for clarity
var result = nums
                .Where(n => n < 5)

                .OrderBy (n => n);
```

Longer listings for examples that are too big to be inline with the text or samples I specifically wanted to provide in the sample download project are shown using a similar `monospace` font, but they are denoted by a listing number and a short description, as in the following example, Listing 3-2.

Listing 3-2 Simple query using the Query Expression syntax

```
List<Contact> contacts = Contact.SampleData();

var q = from c in contacts
        where c.State == "WA"
        orderby c.LastName, c.FirstName
        select c;

foreach (Contact c in q)
    Console.WriteLine("{0} {1}",
        c.FirstName, c.LastName);
```

Each example should be simple and consistent. For simplicity, most examples write their results out to the Console window. To capture these results in this book, they are listed in the same font and format as code listings, but identified with an output number, as shown in Output 3-1.

Output 3-1

```
Stewart Kagel
Chance Lard
Armando Valdes
```

Sample data for the queries is listed in tables, for example, Table 2-2. Each column maps to an object property of a similar legal name for queries to operate on.

Words in **bold** in normal text are defined in the Glossary, and only the first occurrence of the word gets this treatment. When a `bold monospace` font in code is used, it is to draw your attention to a particular key point being explained at that time and is most often used when an example evolves over multiple iterations.

Sample Download Code and Updates

All of the samples listed in the book and further reference material can be found at the companion website, the HookedOnLINQ.com reference wiki and website at http://hookedonlinq.com/LINQBook.ashx.

Some examples required a large sample data source and the Geonames database of worldwide geographic place names and data. These data files can be downloaded from http://www.geonames.org/ and specifically the http://download.geonames.org/export/dump/allCountries.zip file. This file should be downloaded and placed in the same folder as the executable sample application is running from to successfully run those specific samples that parse and query this source.

Choice of Language

I chose to write the samples in this book using the C# language because including both C# and VB.Net example code would have bloated the number of pages beyond what would be acceptable. There is no specific reason why the examples couldn't have been in any other .NET language that supports LINQ.

System Requirements

This book was written with the code base of .NET 4 and Visual Studio 2010 over the course of various beta versions and several community technical previews. The code presented in this book runs with Beta 2. If the release

copy of Visual Studio 2010 and .NET 4 changes between this book publication and release, errata and updated code examples will be posted on the companion website at http://hookedonlinq.com/LINQBook.ashx.

To run the samples available from the book's companion website, you will need to have Visual Studio 2010 installed on your machine. If you don't have access to a commercial copy of Visual Studio 2010, Microsoft has a freely downloadable version (Visual Studio 2010 Express Edition), which is capable of running all examples shown in this book. You can download this edition from http://www.microsoft.com/express/.

Acknowledgments

It takes a team to develop this type of book, and I want our team members to know how appreciated their time, ideas, and effort have been. This team effort is what sets blogging apart from publishing, and I fully acknowledge the team at Addison-Wesley, in particular my editors Joan Murray and Olivia Basegio for their patience and wisdom.

To my technical reviewers, Nick Paldino, Derik Whittaker, Steve Danielson, Peter Ritchie, and Tanzim Saqib—thank you for your insights and suggestions to improve accuracy and clarity. Each of you had major impact on the text and code examples contained in this book.

Some material throughout this book, at least in spirit, was obtained by reading the many blog postings from Microsoft staff and skilled individuals from our industry. In particular I'd like to thank the various contributors to the Parallel FX team blog (http://blogs.msdn.com/pfxteam/), notably Igor Ostrovsky (strongly influenced my approach to aggregations), Ed Essey (helped me understand the different partitioning schemes used in PLINQ), and Stephen Toub. Stephen Toub also has my sincere thanks for giving feedback on the Parallel LINQ chapter during its development (Chapter 9), which dramatically improved the content accuracy and depth.

I would also like to acknowledge founders and contributors to Geonames.org (http://geonames.org), whose massive set of geographic data is available for free download under creative commons attribution license. This data is used in Chapter 9 to test PLINQ performance on large data sets.

Editing isn't easy, and I'd like to acknowledge the patience and great work of Anne Goebel and Chrissy White in making my words flow from post-tech review to production. I know there are countless other staff who touched this book in its final stages of production, and although I don't know your names, thank you.

Finally, I'd like to acknowledge readers like you for investing your time to gain a deeper understanding of LINQ to Objects. I hope after reading it you agree that this book offers valuable insights on how to use LINQ to Objects in real projects and that the examples go that step further in explaining the patterns that make LINQ an integral part of day-to-day programming from this day forward. Thank you.

About the Author

Troy Magennis is a Microsoft Visual C# MVP, an award given to industry participants who dedicate time and effort to educating others about the virtues of technology choices and industry application.

A keen traveler, Troy currently works for Travelocity, which manages the travel and leisure websites travelocity.com, lastminute.com, and zuji. As vice president of Architecture, he leads a talented team of architects spread across four continents committed to being the traveler's companion.

Technology has always been a passion for Troy. After cutting his teeth on early 8-bit personal computers (Vic20s, Commodore 64s), he moved into electronics engineering, which later led to positions in software application development and architecture for some of the most prominent corporations in automotive, banking, and online commerce.

Troy's first exposure to LINQ was in 2006 when he took a sabbatical to learn it and became hooked, ultimately leading him to publish the popular HookedOnLINQ website.

CHAPTER 1

INTRODUCING LINQ

Goals of this chapter:

- Define "Language Integrated Query" (LINQ) and why it was built.
- Define the various components that make up LINQ.
- Demonstrate how LINQ improves existing code.

This chapter introduces LINQ—from Microsoft's design goals to how it improves the code we write for data access-based applications. By the end of this chapter, you will understand why LINQ was built, what components makeup the LINQ family, and LINQ's advantages over previous technologies. And you get a chance to see the LINQ syntax at work while reviewing some before and after code makeovers.

Although this book is primarily about LINQ to Objects, it is important to have an understanding of the full scope and goals of all LINQ technologies in order to make better design and coding decisions.

What Is LINQ?

Language Integrated Query, or LINQ for short (pronounced "link"), is a set of **Microsoft .NET Framework** language enhancements and libraries built by Microsoft to make working with data (for example, a collection of in-memory objects, rows from a database table, or elements in an XML file) simpler and more intuitive. LINQ provides a layer of programming abstraction between .NET languages and an ever-growing number of underlying data sources.

Why is this so inviting to developers? In general, although there are many existing programming interfaces to access and manipulate different sources of data, many of these interfaces use a specific language or syntax of their own. If applications access and manipulate data (as most do), LINQ allows developers to query data using similar **C#** (or **Visual**