Practical Entity Framework Core 6

Database Access for Enterprise Applications

Second Edition

Brian L. Gorman

Apress®

Practical Entity Framework Core 6

Database Access for Enterprise Applications

Second Edition

Brian L. Gorman

Apress®

Practical Entity Framework Core 6: Database Access for Enterprise Applications

Brian L. Gorman
Jesup, IA, USA

ISBN-13 (pbk): 978-1-4842-7300-5 ISBN-13 (electronic): 978-1-4842-7301-2
https://doi.org/10.1007/978-1-4842-7301-2

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Jonathan Gennick
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles 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 Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484273005. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to my wife Cassie and my children Kiera, Karson, Kreighton, and baby K who have all made many sacrifices to give me the space and time to write, as well as for your daily, unceasing love, grace, patience, and encouragement.

This book is further dedicated to you, dear reader. Thank you for allowing me to be part of your journey to greatness.

Table of Contents

About the Author

Brian L. Gorman is a Microsoft Azure MVP, developer, computer science instructor, and trainer and has been working in .Net technologies as long as they have existed. He was originally MCSD certified in .Net 1 and re-certified with MCSA: Web Apps and MCSD: App Builder certifications in 2019. From 2019 to 2022, Brian has earned nine Azure certifications, including Azure and Data Fundamentals, Azure Administrator, Database Administrator, Security Engineer, and Developer Associate certifications, Azure Solutions Architect and DevOps Expert certifications, and an IoT Specialty certification.

Additionally, Brian became an MCT as of April 2019 and is focusing on developing and training developers with full-stack web solutions with .Net Core and Azure, and is also focused on helping small businesses meet certification standards to be able to qualify for Microsoft Partnership. Most recently, Brian was employed as a Senior Training Architect with Opsgility, and is still partnering with Opsgility for a number of training initiatives, including taking on the instructor role for an upcoming MSSA offering in January of 2022. As of October 2021, Brian is now fully self-employed as a trainer and curriculum developer, author, and speaker. Brian's company is called MajorGuidanceSolutions.

In addition to working with .Net technologies, Brian has been an adjunct faculty member in the computer science department for Franklin University for the last 11 years, where his courses have included data structures, algorithms, design patterns, and, more recently, full-stack solutions in the capstone practicum course.

About the Technical Reviewer

André van Meulebrouck has a keen interest in functional programming, especially Haskell and F#.

He also likes data technologies from markup languages to databases and F# type providers.

He lives in Southern California with his wife "Tweety" and is active in athletics: hiking, mountain biking, and gravity/balance sports like freestyle skating (inline and ice), skateboarding, surfing, and sandboarding.

To keep his mind sharp, he does compositional origami, plays classical guitar, and enjoys music notation software.

Acknowledgments

I would not have been able to write this book if it were not for a number of people who have both influenced and helped me throughout my career, as well as the multitudes of grace and support that I have received from my family throughout this process.

I'd like to begin by thanking André van Meulebrouck for his excellent work as a technical reviewer and editor. André's thoughts and comments throughout the process have greatly helped to shape this book over the first and second editions. Also, his incredible patience with working through a couple of bugs with the solution files has been an invaluable resource to help ensure the resources work and the directions are easy to follow. An extra special thanks to André as well for consistently putting up with my misuse of setup vs. set up (you would think I'd be better at this by now).

I'd also like to thank the many friends and acquaintances I've made at various tech conferences in the past few years. I've learned so much from all of you. There are a few that I must mention, however. First, Mike Cole, my peer and friend, thank you for introducing me to AutoMapper projections, and thanks for all your candid conversations around Entity Framework (EF) with me as I wrote this book. Thanks to Mitchel Sellers for your talk on Entity Framework that I got to see at Iowa Code Camp and again at our CVINETA meeting a couple of years ago, which focused on addressing the performance pitfalls that arise from misusing Language Integrated Query (LINQ).

Thank you to Apress and the team who have believed in me and have helped to make this book possible. Thanks to Jonathan Gennick and Jill Balzano for running the project, editing, and overseeing the entire schedule and process.

I would be remiss if I didn't also thank Dustin Behn, the leader of the Inspired Nation, and his life coaching and his Emergence program. Thank you for coaching me these past few years and for helping me get out of my own way to do things like this book.

Last, and most importantly, to my wife Cassie and our kids, to whom the book is also dedicated. Thank you for giving me the time and space to make this book happen and for continually checking on my progress by asking how many chapters I have done and how many I have left.

Introduction

Entity Framework is the object-relational mapper (ORM) of choice for a majority of enterprise application development teams which are leveraging Microsoft .Net technologies. Through the years, EF has gone through a number of changes, and the move into the .Net Core world has seen EF become more performant and more user-friendly.

As this book begins, we'll take a look at the state of things as they are and the state of things to come. We'll begin the real work by touching on the two different approaches to working with a database using EFCore: database first and code first. After the first three chapters, we settle in on the code-first approach with EFCore and approach practical, real-world scenarios to help you and your team develop robust and rugged data solutions while learning the fundamental concepts necessary to effectively work with EFCore.

The great news is that no matter what approach to the database or version of EF you are using, with just a few minor exceptions, things will generally work in a similar fashion, so all of the information in this book is relevant to anyone working with Entity Framework.

Who this book is for

Practical Entity Framework is written for anyone that is new to Entity Framework or is still learning and wants to become much better with Entity Framework.

If you are already an expert or a well-established developer with a few years of EF under your belt, this book will likely not have a lot of new information for you, but there may be a couple of concepts that you would still benefit from reviewing.

Overall, the book is designed as a practical approach – which means that there is a lot of hands-on work to step through the moving pieces that are necessary to understand and work with EFCore, as well as how to approach architecting SOLID solutions around EFCore.

The practical nature of each activity will give you many examples and cover a lot of the basic and advanced topics you will likely encounter in real-world applications.

PART I

Getting Started

CHAPTER 1

Introduction to Entity Framework

In this chapter, we are going to cover the history and origins of Entity Framework and then continue into discussions of where Entity Framework is and where it is headed. We'll conclude with what it takes to get Entity Framework into any .Net project.

One, two, three, four versions? Oh my!

Before we begin doing anything, it's important to note that at the time that I'm writing this book, there are currently three active versions of Entity Framework in play that organizations likely have deployed across various solutions, and by the time you are reading this text, you are likely to encounter at least two of them on a regular basis. The good news is that, for the most part, they all work in a very similar fashion, with just a few slight differences in some of the commands and available functionality.

As this is the second edition of this text, this book is an improvement and update on the original *Practical Entity Framework*, which was released in July 2020. The original version was written with EF6 and EFCore for .Net Core 3.1. If you need information that is more specific to these original versions (and still very valid versions) of Entity Framework, I would encourage you to pick up a copy of the first edition. Again, almost everything in this text would also apply to the original versions of Entity Framework – EF6, EFCore3, and EFCore5 – but there are some improvements that will be highlighted in this text that would not work in previous versions.

Before we dive into the meat of EFCore6, in the next few pages, we'll examine where we came from, how we got to this situation of having multiple, active versions, and where we're going from here. Let's start at the very beginning.

B. L. Gorman, *Practical Entity Framework Core 6*, https://doi.org/10.1007/978-1-4842-7301-2_1

When it all began

Microsoft SQL databases have been around for quite some time. In fact, they existed before .Net was created.

OLEDb and spaghetti database access

Prior to the .Net Framework, often a database connection was handled through code in an `Object Linking and Embedding Database Object` (OLEDb). Developers would often write `SQL` queries inline and then connect to the database and perform actions using these tools. Furthermore, queries often lacked any kind of security and organization. Similar or identical calls might be written from multiple pages. As if this approach didn't have enough problems to begin with, SQL queries might have even existed within the html, which is easily viewable from a simple "right-click and view-source" operation. In the most egregious situations, database credentials might have even been easily viewable in this same source. Finally, and yes it gets even worse, often the user credentials that were used in these pages had full access to everything in the database, perhaps even multiple databases.

In addition to the problems of having a spaghetti code approach to database operations, exposing queries and credentials to the world leads to extremely dangerous security breaches. One of the most common security risks when working with data, even to this day, is an attack known as a SQL Injection query.

Imagine that your update statement was fully exposed in the source on your web page. All it would take to compromise the database is a savvy hacker to use their knowledge to "inject" a few statements along with your query, and they could accomplish anything from performing destructive actions like dropping tables or other schema objects to mining operations like exporting your data for their own use. Even if your query wasn't directly exposed, if you had given the user a form text field to work with, then the attacker could easily place SQL code right in that form text and potentially hijack or corrupt your database. Obviously, some better approaches to prevent issues like these were critically needed.

ADO.Net – A better tool for application database interaction

For .Net developers, the next step in working with a database relied on a technology known as *ADO.Net*. Believe it or not, ADO.Net is still in use, and it's even possible to use ADO.Net in your greenfield projects, even today (and there may even be some developers who might even die on the hill of the efficiency of this approach).

ADO.Net was developed to help prevent a few of the problems we've previously discussed. One of the most important aspects of the ADO.Net library was the ability to easily parameterize queries. With parameterized queries, developers no longer had to create inline SQL queries directly in the application code. Rather, the ADO.Net approach allowed (and still allows) developers to create a base connection object, `SQLConnection`, with credentials obscured and the connection string stored in one common, secure location. The connection object is directly referenced by a command object, `SQLCommand`. The command object had settings allowing developers to toggle the command to work as a regular query or to execute a database object such as a stored procedure. Most importantly, the query allows the parameters to be defined and constrained by type, as well as allows for automatically replacing bad characters often used in SQL Injection attacks.

Once the queries were executed from the command, the results could be used to hydrate a result set, such as a `DataReader` or a `DataSet`. These results-oriented objects were then used to transport the relevant data and provide access to the data to render it back to the end user. This approach was the best tool we had as developers before Entity Framework (or other ORMs such as *NHibernate*).

A brief note about ADO.Net

As mentioned previously, it is still possible to program database operations directly with ADO.Net. At this point, however, ADO.Net is rarely used directly in current enterprise-level applications. In modern development, we almost always want to wrap our database operations with a unit of work and also potentially provide access through repositories (e.g., the *unit of work* and *repository patterns*), which is generally provided by most *object-relational mappers (ORMs)*. Entity Framework takes ADO.Net to the next level by abstracting the need to directly interact with ADO.Net. Additionally, as a fully capable ORM, EF utilizes both the unit of work and repository patterns by default.

Entity Framework makes its debut

In 2008, when EF was created, the only version of the .Net Framework in play was just that – the .Net Framework (version 3.5 at the time of the first release of EF). The framework had already been released in version 2.0 and then 3.0, and finally, some additional tools came in the framework version 3.5 release, including the first version of Entity Framework. The final release of the .Net Framework came with version 4.8 in late September of 2019. At that point, the version of EF was EF6 (which means there was an EF5 and an EF6, which is why we now have EFCore5 and EFCore6 even though the "core" moniker is now officially dropped from .Net – there will be more on this later).

With each iteration of the .Net Framework, Microsoft revolutionized the way we program in relation to the database with the introduction of Entity Framework and the query syntax known as *LINQ* (Language Integrated Query).

Entity Framework and LINQ

In tandem, *EF* and *LINQ* made it possible to not only work against our database objects using C# or VB.Net code but also gave us the ability to define database structures directly in code. Using code to create database objects rather than traditional SQL scripts is known as working in a *code-first database approach*.

Being able to define and work with objects in memory that modeled the database object while also directly tracking changes against the database was quite a powerful revolution. Directly tracking the changes in memory also leads to a new level of understanding of concurrency issues for those of us who were used to working with disconnected data. This transition from disconnected to connected data was a very good thing, even if it was a slightly painful transition. Additionally, EF provides the ability to easily work with the data in a disconnected fashion, which is also a valid and valuable option. We will examine working with both disconnected and connected data in this text.

While EF and LINQ were some of the more important database tools that were made available to us with each iteration of the .Net Framework, there was more going on than just these language and paradigm changes. Ultimately, the introduction of a new CEO would start to take Microsoft down an entirely different path.