



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

<https://doi.org/10.1007/978-1-4842-7301-2>

ISBN-13 (electronic): 978-1-4842-7301-2

Copyright © 2022 by Brian L. Gorman

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

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

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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 Authorxxiii
- About the Technical Reviewerxxv
- Acknowledgmentsxxvii
- Introductionxxix
- Part I: Getting Started 1
- Chapter 1: Introduction to Entity Framework 3
 - One, two, three, four versions? Oh my! 3
 - When it all began..... 4
 - ADO.Net – A better tool for application database interaction 5
 - A brief note about ADO.Net 5
 - Entity Framework makes its debut 6
 - Entity Framework and LINQ 6
 - A new direction and a new visionary leader 7
 - Microsoft goes all in for all developers 7
 - A new vision requires a new path 7
 - What is .Net 5 and why is Entity Framework called EFCore5 instead of EF5, and why are we already on .Net 6 and EFCore6..... 8
 - The state of the union..... 9
 - The future..... 9
 - Activity 1-1: Getting started with EFCore6 10
 - Task 1: Create a new project and add the EF packages 10
 - Task 2: Add the EFCore6 packages to your project 15
 - Activity summary..... 25

TABLE OF CONTENTS

Chapter summary 26

 Important takeaways..... 26

 Closing thoughts..... 26

Chapter 2: Working with an Existing Database 27

 Reverse-engineering or database first 27

 Why would we approach Entity Framework in this manner? 27

 Reverse-engineered solutions..... 28

 Keeping everything in sync 28

 Interacting with the existing database 29

 Activity 2-0: Working with a pre-existing database 29

 Task 1: Prerequisites 29

 Task 2: Download and restore the backup file for the latest version of the
AdventureWorks database to your machine..... 32

 Activity 2-1: Reverse-engineering an existing database with EFCore5 43

 Task 1: Creating the solution with a new project and referencing the DBLibrary project 43

 Task 2: Ensure .Net 6 and update all of the NuGet packages for both projects..... 49

 Task 3: Scaffold a new database context using the Scaffold-Context command..... 55

 Task 4: Create a settings file and leverage it from code 67

 Task 5: Connect to the database and show results in code 74

 Activity summary 79

 Chapter summary 79

 Important takeaways..... 79

 Closing thoughts..... 79

Chapter 3: Entity Framework: Code First..... 81

 Code first doesn't always mean code first..... 81

 When not to use the code-first approach 81

 When to use the code-first approach 83

 Code first in an existing project..... 83

 Code first in a new project against a mature database..... 84

 Code first in a new project with a new database 84

The benefits of a well-executed code-first development effort.....	84
Ability to get up and running quickly.....	85
A complete record of database changes in source control	85
Agility when needing to revert to a previous state.....	86
Shifting from declarative to imperative database programming.....	87
It's time to see code-first database programming in action	88
A final thought before diving into the activities.....	88
Activity 3-1: Creating a new code-first project against an existing database in EFCore6	89
Use the starter files or your project from Chapter 2.....	89
Task 1: Getting started with the activity	90
Task 2: Creating and reviewing the initial migration	93
Task 3: Comment out the initial migration, run the update, and review the database	96
Task 4: Add a new migration, review it, remove it, and then add a real migration.....	102
Activity 3-1 summary.....	110
Activity 3-2: Creating a new code-first project in EFCore6.....	110
Task 1: Begin a new project for managing inventory	111
Task 2: Add a new library for your database models – the “code” of code first	123
Task 3: Reference the InventoryModels project and use it to create a migration	127
Task 4: Update and review the database.....	135
Task 5: Add code to insert and query a list of items.....	137
Activity 3-2 summary.....	143
Chapter summary	143
Important takeaways.....	144
Closing thoughts.....	144
Part II: Building the Data Solution	145
Chapter 4: Models and the Data Context	147
What is the database context and why do we need it?	147
DbContext vs.ObjectContext.....	148
What is the DbContext?.....	149
Constructing a new DbContext.....	150

TABLE OF CONTENTS

Critical properties and methods available when working with the DbContext	152
Important properties on the DbContextOptionsBuilder object	152
Important properties on the DbContextOptions object	153
Important properties on the DbContext object	153
Methods available on the DbContext	154
Methods and extensions on the DbSet<TEntity> object	156
Working with models	157
Two immediate benefits of code-first models	157
Building a database entity model	157
A final thought about models	158
Activity 4-1: Modifying the Item model	158
Practical application for your daily routine	159
Starter files	159
Task 1: Creating the base project	159
Task 2: Add properties to the Item class, and then use a migration to update the database with fields to match the properties	166
Task 3: Add auditing to entities via inheritance	172
Activity 4-1 summary	178
Activity 4-2: Using the ChangeTracker to inject some automated auditing	179
Remember how you already set up the DbContext	179
Common critical underlying objects	180
The ChangeTracker is the lifeblood of our interaction with the Entity Framework	180
Task 1: Getting started	180
Task 2: Use the change tracker to inject auditing information on calls to save changes	183
Task 3: Add an update method to validate last modified auditing is working as expected	189
Activity 4-2 summary	193
Chapter summary	193
Important takeaways	194
Closing thoughts	194

Chapter 5: Constraints, Keys, and Relationships	195
Constraining your data to enhance your solutions	195
Size limitations	196
Value constraints	198
Default values	198
Other data annotations	199
Using keys in database tables for unique and relational results	200
Working with relational data	201
First, second, and third normal form	201
First normal form (1NF)	202
Second normal form (2NF)	203
Third normal form (3NF)	205
Types of relationships	207
One-to-one relationships	208
One-to-many relationships	209
Many-to-many relationships	210
Some final thoughts about relationships and normalization	212
Activity 5-1: Add length, range, and other constraints to the Item model	212
Creating constraints	212
Prerequisite: Get set up for this activity	213
Task 1: Setting length constraints on columns	213
Task 2: Creating a range on numeric fields	220
Task 3: Ensuring a field is a key, making fields required, and setting default values on a column	225
Task 4: Add a new migration and apply these changes to the database	228
Activity 5-1 summary	232
Activity 5-2: Working with relationships	233
Creating a one-to-many relationship	233
Task 0: Getting started	233
Task 1: Create the Categories in a one-to-many relationship with Items	234

TABLE OF CONTENTS

Task 2: Create a one-to-one relationship from Category to CategoryDetail	242
Task 3: Create a many-to-many relationship.....	249
Activity 5-2 summary.....	256
Activity 5-3: Using a non-clustered, unique index	256
Soft delete or hard delete, either way, just make sure it works	257
Task 0: Getting started.....	257
Task 1: Create the Genre	257
Task 2: Create the ItemGenre and the many-to-many relationship.....	259
Task 3: Use the Index attribute to create a unique, non-clustered index.....	263
Activity 5-3 summary.....	266
Chapter summary	267
Important takeaways.....	268
Closing thoughts.....	268
Chapter 6: Data Access (Create, Read, Update, Delete)	269
CRUD	269
LINQ	269
Basic interactions	269
Leverage the DbSet<T> objects.....	270
Common commands.....	271
A final thought before diving into the activities.....	273
Activity 6-1: Quick CRUD with scaffolded controllers	273
Task 0: Getting started.....	274
Task 1: Creating the new MVC project.....	274
Task 2: Start working with the ASP.Net MVC project	282
Task 3: Create CRUD for the items.....	289
Activity 6-1 summary.....	296
Chapter summary	296
Important takeaways.....	297
Closing thoughts.....	297

Chapter 7: Stored Procedures, Views, and Functions.....	299
Understanding stored procedures, views, and functions.....	299
Stored procedures	300
Functions	300
Views	301
Setting up the database to run scripts efficiently	302
The problem	302
The solution	306
Fluent API	307
What can you do with the Fluent API	307
How do you work with the Fluent API	307
Working with the database objects	309
A final thought before diving into the activities	309
Activity 7-1: Working with stored procedures	309
Task 0: Getting started	310
Task 1: Create a new stored procedure using inline code in your migration	310
Task 2: Create the extension method to use local files for scripting	316
Task 3: Apply the migration	321
Task 4: Leverage the stored procedure in code	324
Activity 7-1 summary	331
Activity 7-2: Working with functions, the FluentAPI, and seed data	331
Task 0: Getting started	332
Task 1: Script out a new scalar-valued function	332
Task 2: Leverage the new function from code	337
Task 3: Create a new table-valued function	339
Task 4: Seed data with the Fluent API	346
Task 5: Seed data with a custom solution	348
Task 6: Seed the Players and Items data	358
Activity 7-2 summary	363

TABLE OF CONTENTS

Activity 7-3: Working with views..... 364

 Task 0: Getting started..... 364

 Task 1: Create the view 364

 Task 2: Expose the view data from the UI layer..... 367

Activity 7-3 summary..... 370

Chapter summary 370

 Important takeaways..... 371

 Closing thoughts..... 371

Chapter 8: Sorting, Filtering, and Paging..... 373

 It's time to learn LINQ 373

 LINQ is generally not the problem 373

 Use a profiler or another tool..... 374

 Issues and solutions 374

 Issue #1: Pre-fetching results and then iterating in code to filter the results 375

 Issue #2: Not disconnecting your data 376

 Issue #3: IEnumerable vs. IQueryable 377

 Practical application 378

Activity 8-1: Sorting, paging, and filtering..... 379

 Task 0: Getting started..... 379

 Task 1: Compare the execution efficiency of two queries 381

 Task 2: Filtering our results 393

 Task 3: Paging the filtered results 398

 Task 4: Disconnecting the result sets..... 401

Activity 8-1 summary..... 403

Chapter summary 403

 Important takeaways..... 403

 Closing thoughts..... 403

Part III: Enhancing the Data Solution.....	405
Chapter 9: LINQ for Queries and Projections	407
Data in the real world.....	407
LINQ vs. stored procedures	407
Complex data and the code-first approach	408
DTOs, view models, or domain models	409
Decoupling your business or view logic from the database.....	409
Sometimes, a pre-defined object is overkill	410
One tool to rule them all.....	411
AutoMapper	412
Chapter 9 activities: Using LINQ, decoupled DTO classes, projections, anonymous types, and AutoMapper.....	412
Activity 9-1: Working with LINQ in complex queries	413
Task 0: Getting started.....	413
Task 1: Get all the salespeople	414
Task 2: Use projections to get more efficient queries	422
Activity 9-1 summary.....	437
Activity 9-2: Setting up AutoMapper	438
Task 0: Getting started.....	438
Task 1: Get AutoMapper packages and configure the solution.....	438
Task 2: Create the DTO objects.....	441
Activity 9-2 summary.....	446
Activity 9-3: Working with AutoMapper.....	446
Task 0: Getting started.....	447
Task 1: Perform a more advanced query	450
Task 2: Using AutoMapper and DTO projections	456
Activity 9-3 summary.....	461
Chapter summary	462
Important takeaways.....	462
Closing thoughts.....	462

Chapter 10: Encryption of Data..... 463

 Keeping your system’s data secure 463

 Data at rest..... 463

 Encryption in the past vs. encryption today 463

 Passwords 464

 SSO via social logins 464

 ASP.Net built-in authentication..... 464

 Salting and hashing..... 465

 Protecting sensitive user information..... 466

 Encryption basics 467

 Which type should you use..... 467

 Chapter 10 activities: Using Always Encrypted and Transparent Data Encryption 469

 Activity 10-1: Using Always Encrypted..... 469

 Task 0: Getting started..... 469

 Task 1: Enable Always Encrypted on the InventoryManagerDb..... 470

 Activity 10-1 summary..... 491

 Activity 10-2: Using Transparent Data Encryption..... 491

 Task 0: Getting started..... 492

 Task 1: Plan the migration strategy 493

 Task 2: Create the backup columns..... 495

 Task 3: Create the keys and certificates..... 499

 Task 4: Drop constraints on the targeted columns 502

 Task 5: Drop the columns that are going to be targeted for encryption,
 and then recreate them 506

 Task 6: Select the backup data, transform it for encryption, and store it in
 the original columns..... 514

 Task 7: Clean up the table 516

 Activity 10-2 summary..... 517

 Chapter summary 517

 Important takeaways..... 517

 Closing thoughts 518

Chapter 11: Repository and Unit of Work Patterns	519
The repository (Repo) pattern	519
Sources of information about the repository pattern	519
The repository pattern abstracts the database plumbing code from the implementation	520
Entity Framework's built-in repository	520
The unit of work pattern	521
Using a unit of work	521
Combining the repository and the unit of work	521
The one-two punch	522
A couple of drawbacks.....	522
In general, rely on EF	524
Separation of concerns	524
Logical separation of concerns	524
Final benefits of separation of concerns	525
Chapter 11 activities	525
Activity 11-1: Layering your solution.....	525
Task 0: Getting started.....	526
Task 1: Creating the database layer	526
Task 2: Creating the business layer.....	530
Task 3: Create and implement database operations in the database layer	531
Task 4: Create and implement business operations in the service layer	538
Task 5: Refactor the console program.....	541
Activity 11-1 summary.....	546
Activity 11-2: Rolling your own UoW	546
Transactions are easy and effective	546
Use the using statement for transaction lifecycles	547
Task 0: Getting started.....	547
Task 1: Modify the InventoryDatabaseLayer	548
Task 2: Modify the InventoryBusinessLayer	554

TABLE OF CONTENTS

Task 3: Build the insert logic 558

Task 4: Build the update logic 562

Task 5: Build the delete logic..... 566

Task 6: Update the transaction scope..... 569

Activity 11-2 summary..... 572

Chapter summary 573

Important takeaways..... 573

Closing thoughts..... 574

Chapter 12: Unit Testing, Integration Testing, and Mocking 575

Testing your code is a must-have, not a nice-to-have 575

 The code needs to be changed..... 575

 The database is the lifeblood of the application..... 576

 Testing saves your sanity and protects the system..... 576

Two different approaches leading to the ability to test changes 576

 Unit testing 576

 Libraries utilized 577

 Integration testing 577

Activities for Chapter 12 578

Activity 12-1: Unit testing with mocking 579

 Mocking for your tests..... 579

 Task 0: Getting started..... 579

 Task 1: Add the unit testing project to the solution 580

 Task 2: Write your first unit test..... 584

 Task 3: Get and implement Moq 586

 Task 4: Refactor the InventoryBusinessLayer to be context independent..... 592

 Task 5: Run the unit test and refactor 594

Activity 12-1 summary..... 599

Activity 12-2: Integration testing with an in-memory database..... 599

 Task 0: Getting started..... 600

 Task 1: Create a new xUnit project..... 600

Task 2: Set up the expected data for seeding and integration testing	603
Task 3: Write integration tests	610
Task 4: Refactor the code	612
Activity 12-2 summary	612
Chapter summary	613
Unit tests	613
Integration tests	613
Shouldly and xUnit	614
Dependencies and injection to decouple layers	614
Chapter 13: Asynchronous Data Operations and Multiple Database Contexts	615
Asynchronous operations	615
Multithreaded programming	616
Async, await, and the TaskParallelLibrary	616
Responsive solutions for the end user	617
Asynchronous database operations	617
Basic asynchronous syntax	618
Multiple database contexts	618
Single sign-on (SSO)	618
Business units	619
Multiple contexts require a bit more work	620
Putting it into practice	620
Activity 13-1: Asynchronous database operations	621
Task 0: Getting started	621
Task 1: Refactor the database layer	621
Task 2: Refactor the integration tests	629
Task 3: Refactor the business layer	632
Task 4: Refactor the unit tests	636
Task 5: Refactor the main program	638
Task 6: Fix a broken integration test	649

TABLE OF CONTENTS

Activity 13-1 summary..... 651

Activity 13-2: Multiple database contexts..... 651

 Task 0: Getting started..... 651

 Task 1: Inject both contexts into the solution, and learn about working with
 multiple contexts 652

 Task 2: Scaffold Category pages 658

 Task 3: Ensure solid learning on the database context 668

Activity 13-2 summary..... 671

Chapter summary 671

 Important takeaways..... 672

 Closing thoughts..... 672

Part IV: Recipes for Success..... 673

Chapter 14: .Net 5 and EFCore5..... 675

 One framework to rule them all, with more coming 675

 EF6, EFCore, and .Net 5/6/7/... 675

 .Net 6/7 and EFCore6/7 676

 Changes with EFCore5..... 676

Activity 14-1: Many-to-many navigation properties..... 677

 Task 0: Getting started..... 677

 Task 1: Review the existing relationships..... 678

 Task 2: Explore this implicit mapping 683

Activity 14-1 summary..... 690

Activity 14-2: Filtered include 690

 Task 0: Getting started..... 691

 Task 1: Create the method and set up the filtered include query..... 691

 Task 2: Fix the original query..... 698

Activity 14-2 summary..... 701

Activity 14-3: Split queries..... 701

 Task 0: Getting started..... 702

Task 1: Create the query.....	702
Task 2: Use the new split query functionality.....	705
Activity 14-3 summary.....	709
Activity 14-4: Simple logging and tracking queries with the DBCommandInterceptor.....	710
Task 0: Getting started.....	710
Task 1: Add a method to use for demonstration, and then add logging	710
Task 2: Use the ToQueryString output.....	714
Task 3: Implement the DBCommandInterceptor to log slow running queries	717
Activity 14-4 summary.....	721
Activity 14-5: Flexible entity mapping.....	722
Task 0: Getting started.....	722
Task 1: Use flexible entity mapping to retrieve the results of a view	722
Activity 14-5 summary.....	725
Activity 14-6: Table-per-type (TPT) inheritance mapping.....	725
Task 0: Getting started.....	726
Task 1: Create the inheritance hierarchy	727
Task 2: Move data.....	730
Activity 14-6 summary.....	734
Chapter summary	735
Important takeaways.....	735
Closing thoughts.....	735
Chapter 15: .Net 6 and EFCore6.....	737
Planned highly requested features and enhancements.....	737
SQL Server temporal tables.....	737
JSON columns	738
ColumnAttribute.Order	738
Compiled models.....	739
Migrations bundles.....	739
.Net integration improvements.....	739

TABLE OF CONTENTS

Additional new features	740
More flexible free text search.....	740
UnicodeAttribute.....	740
PrecisionAttribute.....	741
EntityTypeConfigurationAttribute.....	741
Translate ToString on SQLite.....	741
EF.Functions.Random	741
Support for SQL Server sparse columns	742
Command timeout in the connection string for SQLite.....	742
In-memory database – Validate required parameters.....	742
Savepoints API – Use partial transactions to roll back to a previous savepoint.....	742
Reverse-engineering preserves database comments in code	742
Chapter 15 activities	743
Activity 15-1: New attributes	743
Task 0: Getting started.....	743
Task 1: Use the Precision attribute	743
Task 2: Leverage the EntityTypeConfigurationAttribute	748
Task 3: Use the new Unicode attribute	751
Activity 15-1 summary.....	755
Activity 15-2: Changes to how text and searching are handled, null or whitespace translated to SQL, sparse columns, nullable reference types, and a new random function	756
Task 0: Getting started.....	756
Task 1: Improved free text search	756
Task 2: Review the upgrade to string.Concat	766
Task 3: Review the use of EF.Functions.Random.....	768
Task 4: Reviewing improved SQL Server translation for IsNullorWhiteSpace	769
Task 5: Support for sparse columns	770
Activity 15-2 summary.....	774

Chapter summary	775
Important takeaways	775
Closing thoughts	775
Chapter 16: Appendix A: Troubleshooting	777
Migrations	777
Objects exist/objects don't exist	778
Comment out code	778
Manual insert to the database	779
Change DB connection	779
Starter packs	780
General starter pack creation	780
What you should do every time	781
Simple instructions	782
Final packs	785
Review your solution	786
Use a diff tool like GitHub, VSCode, or WinMerge	786
Index	787

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.

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.