

C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals

Seventh Edition

Start building websites and services with ASP.NET Core 7, Blazor, and EF Core 7

Mark J. Price

C# 11 and .NET 7 – Modern Cross-Platform Development Fundamentals

Seventh Edition

Copyright © 2022 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Senior Publishing Product Manager: Suman Sen

Acquisition Editor – Peer Reviews: Saby Dsilva

Project Editor: Janice Gonsalves

Content Development Editor: Lucy Wan

Copy Editor: Safis Editing

Technical Editor: Tejas Mhasvekar

Proofreader: Safis Editing

Indexer: Tejal Daruwale Soni

Presentation Designer: Pranit Padwal

First published: March 2016

Second edition: March 2017

Third edition: November 2017

Fourth edition: October 2019

Fifth edition: November 2020

Sixth edition: November 2021

Seventh edition: November 2022

Production reference: 1011122

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80323-780-0

www.packt.com

Quick Chapter Reference

Chapter 1: Hello, C#! Welcome, .NET!	1
Chapter 2: Speaking C#	45
Chapter 3: Controlling Flow, Converting Types, and Handling Exceptions	99
Chapter 4: Writing, Debugging, and Testing Functions	145
Chapter 5: Building Your Own Types with Object-Oriented Programming	199
Chapter 6: Implementing Interfaces and Inheriting Classes	249
Chapter 7: Packaging and Distributing .NET Types	311
Chapter 8: Working with Common .NET Types	355
Chapter 9: Working with Files, Streams, and Serialization	395
Chapter 10: Working with Data Using Entity Framework Core	435
Chapter 11: Querying and Manipulating Data Using LINQ	489
Chapter 12: Introducing Web Development Using ASP.NET Core	531
Chapter 13: Building Websites Using ASP.NET Core Razor Pages	561
Chapter 14: Building Websites Using the Model-View-Controller Pattern	601
Chapter 15: Building and Consuming Web Services	657
Chapter 16: Building User Interfaces Using Blazor	703
Chapter 17: Epilogue	747
Index	751

Table of Contents

Preface

xxxiii

Chapter 1: Hello, C#! Welcome, .NET! 1

Setting up your development environment 3

Choosing the appropriate tool and application type for learning • 3

Pros and cons of the .NET Interactive Notebooks extension • 4

Using Visual Studio Code for cross-platform development • 4

Using GitHub Codespaces for development in the cloud • 4

Using Visual Studio for Mac for general development • 5

Using Visual Studio for Windows for general development • 5

What I used • 5

Deploying cross-platform • 6

Downloading and installing Visual Studio 2022 for Windows • 6

Microsoft Visual Studio for Windows keyboard shortcuts • 7

Downloading and installing Visual Studio Code • 7

Installing other extensions • 8

Managing Visual Studio Code extensions at the command line • 9

Understanding Microsoft Visual Studio Code versions • 9

Microsoft Visual Studio Code keyboard shortcuts • 9

Understanding .NET 10

Understanding .NET Framework • 10

Understanding the Mono, Xamarin, and Unity projects • 10

Understanding .NET Core • 11

Understanding the journey to one .NET • 11

Understanding Blazor WebAssembly versioning • 12

Understanding .NET support • 12

Understanding .NET Runtime and .NET SDK versions • 14

<i>Listing and removing versions of .NET</i>	• 14
What is different about modern .NET?	• 15
<i>Windows desktop development</i>	• 15
<i>Web development</i>	• 15
<i>Database development</i>	• 15
Understanding .NET Standard	• 16
.NET platforms and tools used by the C# and .NET book editions	• 17
Topics covered by Apps and Services with .NET 7	• 18
Understanding intermediate language	• 18
Comparing .NET technologies	• 18
Building console apps using Visual Studio 2022	19
Managing multiple projects using Visual Studio 2022	• 19
Writing code using Visual Studio 2022	• 19
Compiling and running code using Visual Studio	• 21
<i>Understanding the compiler-generated folders and files</i>	• 22
Understanding top-level programs	• 22
<i>Implicitly imported namespaces</i>	• 23
<i>Revealing the hidden code by throwing an exception</i>	• 24
Adding a second project using Visual Studio 2022	• 24
Building console apps using Visual Studio Code	25
Managing multiple projects using Visual Studio Code	• 26
Writing code using Visual Studio Code	• 26
Compiling and running code using the dotnet CLI	• 29
Adding a second project using Visual Studio Code	• 30
Exploring code using .NET Interactive Notebooks	32
Using .NET Interactive Notebooks for the code in this book	• 32
Reviewing the folders and files for projects	32
Understanding the common folders and files	• 33
Understanding the solution code on GitHub	• 33
Making good use of the GitHub repository for this book	33
Raising issues with the book	• 34
Giving me feedback	• 34
Downloading solution code from the GitHub repository	• 35
Using Git with Visual Studio Code and the command line	• 35
<i>Cloning the book solution code repository</i>	• 36
Looking for help	36
Reading Microsoft documentation	• 36

Getting help for the dotnet tool • 36	
Getting definitions of types and their members • 37	
Looking for answers on Stack Overflow • 40	
Searching for answers using Google • 40	
Subscribing to the official .NET blog • 40	
Watching Scott Hanselman’s videos • 41	
A companion book to continue your learning journey • 41	
Practicing and exploring	42
Exercise 1.1 – Test your knowledge • 42	
Exercise 1.2 – Practice C# anywhere with a browser • 42	
Exercise 1.3 – Explore topics • 42	
Exercise 1.4 – Explore themes of modern .NET • 43	
Summary	43
 Chapter 2: Speaking C#	 45

Introducing the C# language	45
Understanding language versions and features • 46	
<i>Project COOL • 46</i>	
C# 1 • 46	
C# 1.2 • 46	
C# 2 • 46	
C# 3 • 46	
C# 4 • 47	
C# 5 • 47	
C# 6 • 47	
C# 7.0 • 47	
C# 7.1 • 48	
C# 7.2 • 48	
C# 7.3 • 48	
C# 8 • 48	
C# 9 • 48	
C# 10 • 49	
C# 11 • 49	
Understanding C# standards • 49	
Discovering your C# compiler versions • 50	
<i>How to output the SDK version • 51</i>	
<i>Enabling a specific language version compiler • 51</i>	

Switching the C# compiler for .NET 6 • 52

Understanding C# grammar and vocabulary 52

Showing the compiler version • 53

Understanding C# grammar • 54

Statements • 54

Comments • 54

Blocks • 55

Examples of statements and blocks • 56

Understanding C# vocabulary • 56

Comparing programming languages to human languages • 56

Changing the color scheme for C# syntax • 57

Help for writing correct code • 57

Importing namespaces • 58

Implicitly and globally importing namespaces • 59

Verbs are methods • 62

Nouns are types, variables, fields, and properties • 63

Revealing the extent of the C# vocabulary • 63

Working with variables 66

Naming things and assigning values • 66

Literal values • 67

Storing text • 67

Verbatim strings • 68

Raw string literals • 68

Raw interpolated string literals • 69

Summarizing options for storing text • 70

Storing numbers • 70

Storing whole numbers • 71

Exploring whole numbers • 72

Storing real numbers • 72

Writing code to explore number sizes • 73

Comparing double and decimal types • 74

Storing Booleans • 76

Storing any type of object • 76

Storing dynamic types • 77

Declaring local variables • 78

Specifying the type of a local variable • 78

Inferring the type of a local variable • 79

<i>Using target-typed new to instantiate objects</i> • 80	
Getting and setting the default values for types • 81	
Exploring more about console apps	82
Displaying output to the user • 82	
<i>Formatting using numbered positional arguments</i> • 82	
<i>Formatting using interpolated strings</i> • 83	
<i>Understanding format strings</i> • 84	
Getting text input from the user • 85	
Simplifying the usage of the console • 86	
<i>Importing a static type for a single file</i> • 87	
<i>Importing a static type for all code files in a project</i> • 87	
Getting key input from the user • 88	
Passing arguments to a console app • 89	
Setting options with arguments • 91	
Handling platforms that do not support an API • 93	
Understanding async and await	95
Improving responsiveness for console apps • 95	
Practicing and exploring	96
Exercise 2.1 – Test your knowledge • 96	
Exercise 2.2 – Test your knowledge of number types • 97	
Exercise 2.3 – Practice number sizes and ranges • 97	
Exercise 2.4 – Explore topics • 98	
Summary	98
 Chapter 3: Controlling Flow, Converting Types, and Handling Exceptions	 99
Operating on variables	99
Exploring unary operators • 100	
Exploring binary arithmetic operators • 101	
Assignment operators • 102	
Exploring logical operators • 103	
Exploring conditional logical operators • 104	
Exploring bitwise and binary shift operators • 105	
Miscellaneous operators • 107	
Understanding selection statements	108
Branching with the if statement • 108	
<i>Why you should always use braces with if statements</i> • 109	
Pattern matching with the if statement • 109	

Branching with the switch statement • 110	
Pattern matching with the switch statement • 112	
Simplifying switch statements with switch expressions • 114	
Understanding iteration statements	115
Looping with the while statement • 115	
Looping with the do statement • 116	
Looping with the for statement • 116	
Looping with the foreach statement • 117	
<i>Understanding how foreach works internally • 117</i>	
Storing multiple values in an array	118
Working with single-dimensional arrays • 118	
Working with multi-dimensional arrays • 119	
Working with jagged arrays • 121	
List pattern matching with arrays • 123	
Summarizing arrays • 125	
Casting and converting between types	126
Casting numbers implicitly and explicitly • 126	
Converting with the System.Convert type • 127	
Rounding numbers • 128	
<i>Understanding the default rounding rules • 128</i>	
Taking control of rounding rules • 129	
Converting from any type to a string • 130	
Converting from a binary object to a string • 130	
Parsing from strings to numbers or dates and times • 131	
<i>Errors using Parse • 132</i>	
<i>Avoiding exceptions using the TryParse method • 132</i>	
Handling exceptions	133
Wrapping error-prone code in a try block • 133	
<i>Catching all exceptions • 135</i>	
<i>Catching specific exceptions • 136</i>	
<i>Catching with filters • 137</i>	
Checking for overflow	137
Throwing overflow exceptions with the checked statement • 138	
Disabling compiler overflow checks with the unchecked statement • 139	
Practicing and exploring	140
Exercise 3.1 – Test your knowledge • 140	
Exercise 3.2 – Explore loops and overflow • 141	

Exercise 3.3 – Practice loops and operators • 141	
Exercise 3.4 – Practice exception handling • 142	
Exercise 3.5 – Test your knowledge of operators • 142	
Exercise 3.6 – Explore topics • 142	
Summary	143
Chapter 4: Writing, Debugging, and Testing Functions	145
Writing functions	145
Understanding top-level programs and functions • 146	
Times table example • 148	
<i>Writing a times table function • 148</i>	
A brief aside about arguments and parameters • 150	
Writing a function that returns a value • 152	
<i>Converting numbers from cardinal to ordinal • 154</i>	
Calculating factorials with recursion • 155	
Documenting functions with XML comments • 159	
Using lambdas in function implementations • 160	
Debugging during development	163
Using the Visual Studio Code integrated terminal during debugging • 163	
Creating code with a deliberate bug • 164	
Setting a breakpoint and starting debugging • 164	
<i>Using Visual Studio 2022 • 165</i>	
Navigating with the debugging toolbar • 166	
<i>Using Visual Studio Code • 167</i>	
Debugging windows • 168	
Stepping through code • 168	
Customizing breakpoints • 169	
Hot reloading during development	171
Hot reloading using Visual Studio 2022 • 172	
Hot reloading using Visual Studio Code and the command line • 172	
Logging during development and runtime	173
Understanding logging options • 173	
Instrumenting with Debug and Trace • 173	
<i>Writing to the default trace listener • 174</i>	
Configuring trace listeners • 175	
Switching trace levels • 177	
<i>Adding packages to a project in Visual Studio 2022 • 177</i>	

<i>Adding packages to a project in Visual Studio Code</i> • 178	
<i>Reviewing project packages</i> • 178	
Logging information about your source code • 182	
Unit testing	183
Understanding types of testing • 184	
Creating a class library that needs testing • 184	
Writing unit tests • 186	
<i>Running unit tests using Visual Studio 2022</i> • 188	
<i>Running unit tests using Visual Studio Code</i> • 188	
<i>Fixing the bug</i> • 189	
Throwing and catching exceptions in functions	189
Understanding usage errors and execution errors • 189	
Commonly thrown exceptions in functions • 190	
Understanding the call stack • 191	
Where to catch exceptions • 194	
Rethrowing exceptions • 194	
Implementing the tester-doer pattern • 196	
<i>Problems with the tester-doer pattern</i> • 196	
Practicing and exploring	197
Exercise 4.1 – Test your knowledge • 197	
Exercise 4.2 – Practice writing functions with debugging and unit testing • 197	
Exercise 4.3 – Explore topics • 198	
Summary	198
 Chapter 5: Building Your Own Types with Object-Oriented Programming	 199
Talking about OOP	199
Building class libraries	200
Creating a class library • 201	
Defining a class in a namespace • 202	
Understanding members • 203	
Instantiating a class • 204	
Importing a namespace to use a type • 205	
<i>Avoiding a namespace conflict with a using alias</i> • 206	
<i>Renaming a type with a using alias</i> • 206	
Understanding objects • 207	
<i>Inheriting from System.Object</i> • 207	

Storing data within fields	208
Defining fields • 208	
Understanding access modifiers • 208	
Setting and outputting field values • 209	
Storing a value using an enum type • 210	
Storing multiple values using an enum type • 211	
Storing multiple values using collections • 213	
Understanding generic collections • 213	
Making a field static • 214	
Making a field constant • 215	
Making a field read-only • 216	
Initializing fields with constructors • 217	
<i>Defining multiple constructors • 217</i>	
Writing and calling methods	218
Returning values from methods • 218	
Combining multiple returned values using tuples • 219	
<i>C# language support for tuples • 220</i>	
<i>Naming the fields of a tuple • 220</i>	
<i>Deconstructing tuples • 221</i>	
<i>Deconstructing types • 221</i>	
Defining and passing parameters to methods • 222	
Overloading methods • 223	
Passing optional and named parameters • 224	
<i>Naming parameter values when calling methods • 225</i>	
Controlling how parameters are passed • 225	
<i>Simplified out parameters • 226</i>	
Understanding ref returns • 227	
Splitting classes using partial	227
Controlling access with properties and indexers	227
Defining read-only properties • 228	
Defining settable properties • 229	
Requiring properties to be set during instantiation • 231	
Defining indexers • 233	
More about methods	235
Implementing functionality using methods • 235	
Implementing functionality using operators • 238	
Implementing functionality using local functions • 240	

Pattern matching with objects	241
Defining flight passengers • 241	
Enhancements to pattern matching in C# 9 or later • 243	
Working with records	244
Init-only properties • 244	
Understanding records • 245	
Positional data members in records • 246	
<i>Simplifying data members in records</i> • 246	
Practicing and exploring	247
Exercise 5.1 – Test your knowledge • 247	
Exercise 5.2 – Explore topics • 247	
Summary	247
 Chapter 6: Implementing Interfaces and Inheriting Classes	 249
Setting up a class library and console application	250
Making types safely reusable with generics	251
Working with non-generic types • 251	
Working with generic types • 252	
Raising and handling events	253
Calling methods using delegates • 254	
Defining and handling delegates • 255	
Defining and handling events • 257	
Implementing interfaces	258
Common interfaces • 259	
Comparing objects when sorting • 259	
Comparing objects using a separate class • 263	
Implicit and explicit interface implementations • 265	
Defining interfaces with default implementations • 266	
Managing memory with reference and value types	267
Defining reference and value types • 268	
How reference and value types are stored in memory • 268	
Equality of types • 269	
Defining struct types • 271	
Defining record struct types • 272	
Releasing unmanaged resources • 273	
Ensuring that Dispose is called • 275	

Working with null values	276
Making a value type nullable • 276	
Understanding null-related initialisms • 278	
Understanding nullable reference types • 278	
Controlling the nullability warning check feature • 279	
Declaring non-nullable variables and parameters • 279	
Checking for null • 282	
Checking for null in method parameters • 283	
Inheriting from classes	284
Extending classes to add functionality • 285	
Hiding members • 285	
Overriding members • 286	
Inheriting from abstract classes • 287	
Preventing inheritance and overriding • 288	
Understanding polymorphism • 289	
Casting within inheritance hierarchies	291
Implicit casting • 291	
Explicit casting • 291	
Avoiding casting exceptions • 292	
<i>Using is to check a type</i> • 292	
<i>Using as to cast a type</i> • 293	
Inheriting and extending .NET types	293
Inheriting exceptions • 293	
Extending types when you can't inherit • 295	
<i>Using static methods to reuse functionality</i> • 295	
<i>Using extension methods to reuse functionality</i> • 296	
Writing better code	297
Treating warnings as errors • 297	
Understanding warning waves • 300	
Using an analyzer to write better code • 301	
<i>Suppressing warnings</i> • 304	
<i>Fixing the code</i> • 305	
<i>Understanding common StyleCop recommendations</i> • 307	
Practicing and exploring	308
Exercise 6.1 – Test your knowledge • 308	
Exercise 6.2 – Practice creating an inheritance hierarchy • 308	

Exercise 6.3 – Explore topics • 309	
Summary	309
Chapter 7: Packaging and Distributing .NET Types	311
The road to .NET 7	311
.NET Core 1.0 • 312	
.NET Core 1.1 • 312	
.NET Core 2.0 • 312	
.NET Core 2.1 • 313	
.NET Core 2.2 • 313	
.NET Core 3.0 • 313	
.NET Core 3.1 • 313	
.NET 5.0 • 314	
.NET 6.0 • 314	
.NET 7.0 • 315	
Improving performance with .NET 5 and later • 315	
Checking your .NET SDKs for updates • 315	
Understanding .NET components	316
Assemblies, NuGet packages, and namespaces • 316	
<i>What is a namespace?</i> • 316	
<i>Dependent assemblies</i> • 316	
Microsoft .NET project SDKs • 317	
Namespaces and types in assemblies • 318	
NuGet packages • 318	
Understanding frameworks • 319	
Importing a namespace to use a type • 319	
Relating C# keywords to .NET types • 320	
<i>Mapping C# aliases to .NET types</i> • 321	
<i>Understanding native-sized integers</i> • 321	
<i>Revealing the location of a type</i> • 321	
Sharing code with legacy platforms using .NET Standard • 322	
Understanding defaults for class libraries with different SDKs • 323	
Creating a .NET Standard 2.0 class library • 324	
Controlling the .NET SDK • 324	
Publishing your code for deployment	325
Creating a console app to publish • 326	
Understanding dotnet commands • 328	

<i>Creating new projects</i> • 328	
Getting information about .NET and its environment • 328	
Managing projects • 329	
Publishing a self-contained app • 330	
Publishing a single-file app • 331	
Reducing the size of apps using app trimming • 333	
<i>Enabling assembly-level trimming</i> • 333	
<i>Enabling type-level and member-level trimming</i> • 333	
Decompiling .NET assemblies 334	
Decompiling using the ILSpy extension for Visual Studio 2022 • 334	
Viewing source links with Visual Studio 2022 • 338	
No, you cannot technically prevent decompilation • 339	
Packaging your libraries for NuGet distribution 340	
Referencing a NuGet package • 340	
<i>Fixing dependencies</i> • 341	
Packaging a library for NuGet • 342	
<i>Publishing a package to a public NuGet feed</i> • 344	
<i>Publishing a package to a private NuGet feed</i> • 346	
Exploring NuGet packages with a tool • 346	
Testing your class library package • 347	
Porting from .NET Framework to modern .NET 348	
Could you port? • 348	
Should you port? • 349	
Differences between .NET Framework and modern .NET • 350	
.NET Portability Analyzer • 350	
.NET Upgrade Assistant • 350	
Using non-.NET Standard libraries • 351	
Working with preview features 352	
Requiring preview features • 353	
Enabling preview features • 353	
Practicing and exploring 353	
Exercise 7.1 – Test your knowledge • 353	
Exercise 7.2 – Explore topics • 354	
Exercise 7.3 – Explore PowerShell • 354	
Summary 354	

Chapter 8: Working with Common .NET Types	355
Working with numbers	355
Working with big integers • 356	
Working with complex numbers • 357	
Understanding quaternions • 358	
Generating random numbers for games and similar apps • 358	
Working with text	359
Getting the length of a string • 359	
Getting the characters of a string • 360	
Splitting a string • 360	
Getting part of a string • 360	
Checking a string for content • 361	
Joining, formatting, and other string members • 362	
Building strings efficiently • 363	
Pattern matching with regular expressions	364
Checking for digits entered as text • 364	
Regular expression performance improvements • 365	
Understanding the syntax of a regular expression • 366	
Examples of regular expressions • 366	
Splitting a complex comma-separated string • 367	
Activating regular expression syntax coloring • 369	
Improving regular expression performance with source generators • 372	
Storing multiple objects in collections	374
Common features of all collections • 374	
Improving performance by ensuring the capacity of a collection • 376	
Understanding collection choices • 376	
<i>Lists</i> • 376	
<i>Dictionaries</i> • 377	
<i>Stacks</i> • 378	
<i>Queues</i> • 379	
<i>Sets</i> • 379	
<i>Collection methods summary</i> • 379	
Working with lists • 380	
Working with dictionaries • 381	
Working with queues • 382	
Sorting collections • 385	

More specialized collections • 386	
<i>Working with a compact array of bit values • 386</i>	
<i>Working with efficient lists • 386</i>	
Working with immutable collections • 386	
Good practice with collections • 387	
Working with spans, indexes, and ranges 387	
Using memory efficiently using spans • 388	
Identifying positions with the Index type • 388	
Identifying ranges with the Range type • 388	
Using indexes, ranges, and spans • 389	
Working with network resources 390	
Working with URIs, DNS, and IP addresses • 390	
Pinging a server • 392	
Practicing and exploring 393	
Exercise 8.1 – Test your knowledge • 393	
Exercise 8.2 – Practice regular expressions • 393	
Exercise 8.3 – Practice writing extension methods • 394	
Exercise 8.4 – Explore topics • 394	
Summary 394	
 Chapter 9: Working with Files, Streams, and Serialization 395	
Managing the filesystem 395	
Handling cross-platform environments and filesystems • 395	
Managing drives • 398	
Managing directories • 398	
Managing files • 400	
Managing paths • 401	
Getting file information • 402	
Controlling how you work with files • 403	
Reading and writing with streams 404	
Understanding abstract and concrete streams • 404	
<i>Understanding storage streams • 405</i>	
<i>Understanding function streams • 405</i>	
<i>Understanding stream helpers • 405</i>	
Writing to text streams • 406	
Writing to XML streams • 408	

Simplifying disposal by using the using statement • 410	
Compressing streams • 411	
Working with tar archives • 413	
<i>Reading and writing tar entries • 417</i>	
Encoding and decoding text	417
Encoding strings as byte arrays • 418	
Encoding and decoding text in files • 421	
Reading and writing with random access handles	421
Serializing object graphs	422
Serializing as XML • 422	
Generating compact XML • 425	
Deserializing XML files • 426	
Serializing with JSON • 426	
High-performance JSON processing • 428	
Controlling JSON processing • 429	
New JSON extension methods for working with HTTP responses • 432	
Migrating from Newtonsoft to new JSON • 432	
Practicing and exploring	432
Exercise 9.1 – Test your knowledge • 433	
Exercise 9.2 – Practice serializing as XML • 433	
Exercise 9.3 – Explore topics • 434	
Summary	434
 Chapter 10: Working with Data Using Entity Framework Core	 435
Understanding modern databases	435
Understanding legacy Entity Framework • 436	
<i>Using the legacy Entity Framework 6.3 or later • 436</i>	
Understanding Entity Framework Core • 436	
Understanding Database First and Code First • 437	
Performance improvements in EF Core 7 • 437	
Creating a console app for working with EF Core • 438	
Using a sample relational database • 438	
Using SQLite • 439	
<i>Setting up SQLite for Windows • 439</i>	
<i>Setting up SQLite for macOS • 440</i>	
<i>Setting up SQLite for other OSes • 440</i>	
Creating the Northwind sample database for SQLite • 440	

<i>If you are using Visual Studio 2022</i>	• 441
Managing the Northwind sample database with SQLiteStudio	• 441
Using the lightweight ADO.NET provider for SQLite	• 443
Using SQL Server for Windows	• 443
Setting up EF Core 443
Choosing an EF Core database provider	• 444
Connecting to a database	• 444
Defining the Northwind database context class	• 444
Defining EF Core models 446
Using EF Core conventions to define the model	• 446
Using EF Core annotation attributes to define the model	• 446
Using the EF Core Fluent API to define the model	• 448
<i>Understanding data seeding with the Fluent API</i>	• 448
Building EF Core models for the Northwind tables	• 448
<i>Defining the Category and Product entity classes</i>	• 449
Adding tables to the Northwind database context class	• 451
Setting up the dotnet-ef tool	• 452
Scaffolding models using an existing database	• 453
Customizing the reverse engineering templates	• 457
Configuring preconvention models	• 457
Querying EF Core models 457
Filtering included entities	• 460
Filtering and sorting products	• 462
Getting the generated SQL	• 464
Logging EF Core	• 465
<i>Filtering logs by provider-specific values</i>	• 466
<i>Logging with query tags</i>	• 467
Pattern matching with Like	• 467
Generating a random number in queries	• 468
Defining global filters	• 469
Loading patterns with EF Core 470
Eager loading entities using the Include extension method	• 470
Enabling lazy loading	• 471
Explicit loading entities using the Load method	• 472
Modifying data with EF Core 474
Inserting entities	• 475
Updating entities	• 477

Deleting entities • 479	
More efficient updates and deletes • 480	
Pooling database contexts • 484	
Working with transactions	484
Controlling transactions using isolation levels • 484	
Defining an explicit transaction • 485	
Defining Code First EF Core models	486
Practicing and exploring	486
Exercise 10.1 – Test your knowledge • 486	
Exercise 10.2 – Practice exporting data using different serialization formats • 487	
Exercise 10.3 – Explore topics • 487	
Exercise 10.4 – Explore NoSQL databases • 487	
Summary	487
 Chapter 11: Querying and Manipulating Data Using LINQ	 489
Why LINQ?	489
Comparing imperative and declarative language features • 489	
Writing LINQ expressions	490
What makes LINQ? • 490	
Building LINQ expressions with the Enumerable class • 491	
Understanding deferred execution • 493	
Filtering entities with Where • 494	
Targeting a named method • 496	
Simplifying the code by removing the explicit delegate instantiation • 497	
Targeting a lambda expression • 497	
Sorting entities • 497	
<i>Sorting by a single property using OrderBy • 498</i>	
<i>Sorting by a subsequent property using ThenBy • 498</i>	
Sorting by the item itself • 499	
Declaring a query using var or a specified type • 499	
Filtering by type • 499	
Working with sets and bags using LINQ • 501	
Using LINQ with EF Core	503
Building an EF Core model • 503	
<i>Using Visual Studio 2022 with SQLite databases • 506</i>	
Filtering and sorting sequences • 506	
Projecting sequences into new types • 508	

Joining and grouping sequences • 510	
<i>Joining sequences • 510</i>	
<i>Group-joining sequences • 512</i>	
Aggregating sequences • 513	
Be careful with Count! • 515	
Paging with LINQ • 517	
Sweetening LINQ syntax with syntactic sugar	521
Using multiple threads with parallel LINQ	522
Creating your own LINQ extension methods	522
Trying the chainable extension method • 524	
<i>Trying the mode and median methods • 525</i>	
Working with LINQ to XML	526
Generating XML using LINQ to XML • 526	
Reading XML using LINQ to XML • 527	
Practicing and exploring	528
Exercise 11.1 – Test your knowledge • 528	
Exercise 11.2 – Practice querying with LINQ • 529	
Exercise 11.3 – Explore topics • 529	
Summary	529
 Chapter 12: Introducing Web Development Using ASP.NET Core	 531
Understanding ASP.NET Core	531
Classic ASP.NET versus modern ASP.NET Core • 532	
Building websites using ASP.NET Core • 533	
<i>Building websites using a content management system • 533</i>	
<i>Building web applications using SPA frameworks • 534</i>	
Building web and other services • 535	
New features in ASP.NET Core	535
ASP.NET Core 1.0 • 535	
ASP.NET Core 1.1 • 535	
ASP.NET Core 2.0 • 535	
ASP.NET Core 2.1 • 535	
ASP.NET Core 2.2 • 536	
ASP.NET Core 3.0 • 536	
ASP.NET Core 3.1 • 537	
Blazor WebAssembly 3.2 • 537	
ASP.NET Core 5.0 • 537	

ASP.NET Core 6.0 • 537	
ASP.NET Core 7.0 • 538	
Structuring projects	538
Structuring projects in a solution or workspace • 538	
Building an entity model for use in the rest of the book	539
Creating a class library for entity models using SQLite • 540	
<i>Improving the class-to-table mapping • 541</i>	
<i>Creating a class library for a Northwind database context • 545</i>	
Creating a class library for entity models using SQL Server • 549	
Testing the class libraries • 552	
Understanding web development	553
Understanding Hypertext Transfer Protocol • 553	
<i>Understanding the components of a URL • 554</i>	
<i>Assigning port numbers for projects in this book • 555</i>	
Using Google Chrome to make HTTP requests • 555	
Understanding client-side web development technologies • 557	
Practicing and exploring	558
Exercise 12.1 – Test your knowledge • 558	
Exercise 12.2 – Know your webabbreviations • 558	
Exercise 12.3 – Explore topics • 559	
Summary	559
 Chapter 13: Building Websites Using ASP.NET Core Razor Pages	 561
Exploring ASP.NET Core	561
Creating an empty ASP.NET Core project • 561	
Testing and securing the website • 563	
<i>Enabling stronger security and redirecting to a secure connection • 567</i>	
Controlling the hosting environment • 568	
Enabling a website to serve static content • 570	
<i>Creating a folder for static files and a web page • 570</i>	
<i>Enabling static and default files • 571</i>	
Exploring ASP.NET Core Razor Pages	572
Enabling Razor Pages • 572	
Adding code to a Razor Page • 573	
Using shared layouts with Razor Pages • 574	
Using code-behind files with Razor Pages • 576	

Using Entity Framework Core with ASP.NET Core	578
Configuring Entity Framework Core as a service • 579	
Manipulating data using Razor Pages • 581	
<i>Enabling a model to insert entities • 581</i>	
<i>Defining a form to insert a new supplier • 582</i>	
Injecting a dependency service into a Razor Page • 583	
Using Razor class libraries	583
Disabling compact folders for Visual Studio Code • 584	
Creating a Razor class library • 584	
Implementing a partial view to show a single employee • 587	
Using and testing a Razor class library • 588	
Configuring services and the HTTP request pipeline	589
Understanding endpoint routing • 589	
Configuring endpoint routing • 589	
Reviewing the endpoint routing configuration in our project • 590	
Setting up the HTTP pipeline • 592	
Summarizing key middleware extension methods • 593	
Visualizing the HTTP pipeline • 594	
Implementing an anonymous inline delegate as middleware • 594	
Enabling request decompression support • 596	
Enabling HTTP/3 support	597
Practicing and exploring	599
Exercise 13.1 – Test your knowledge • 599	
Exercise 13.2 – Practice building a data-driven web page • 599	
Exercise 13.3 – Practice building web pages for console apps • 599	
Exercise 13.4 – Explore topics • 600	
Summary	600
 Chapter 14: Building Websites Using the Model-View-Controller Pattern	 601
Setting up an ASP.NET Core MVC website	601
Creating an ASP.NET Core MVC website • 602	
Creating the authentication database for SQL Server LocalDB • 603	
Exploring the default ASP.NET Core MVC website • 605	
Starting the MVC website project • 606	
Exploring visitor registration • 607	
Reviewing an MVC website project structure • 608	
Reviewing the ASP.NET Core Identity database • 610	

Exploring an ASP.NET Core MVC website	610
ASP.NET Core MVC initialization • 610	
The default MVC route • 613	
Controllers and actions • 614	
<i>The ControllerBase class • 614</i>	
<i>The Controller class • 614</i>	
<i>The responsibilities of a controller • 615</i>	
The view search path convention • 617	
Logging using the dependency service • 618	
Entity and view models • 619	
Views • 622	
Understanding how cache busting with Tag Helpers works • 624	
Customizing an ASP.NET Core MVC website	624
Defining a custom style • 625	
Setting up the category images • 625	
Razor syntax and expressions • 625	
Defining a typed view • 626	
Passing parameters using a route value • 629	
Model binders in more detail • 631	
<i>Disambiguating action methods • 633</i>	
<i>Passing a route parameter • 634</i>	
<i>Passing a form parameter • 635</i>	
Validating the model • 635	
Defining views with HTML Helper methods • 638	
Defining views with Tag Helpers • 638	
Cross-functional filters • 639	
<i>Using a filter to secure an action method • 639</i>	
<i>Enabling role management and creating a role programmatically • 640</i>	
<i>Using a filter to define a custom route • 643</i>	
<i>Using a filter to cache a response • 644</i>	
Using output caching • 645	
<i>Output caching endpoints • 645</i>	
<i>Output caching MVC views • 646</i>	
<i>Varying cached data by query string • 648</i>	
Querying a database and using display templates	650
Improving scalability using asynchronous tasks	653
Making controller action methods asynchronous • 653	

Practicing and exploring	654
Exercise 14.1 – Test your knowledge • 654	
Exercise 14.2 – Practice implementing MVC by implementing a category detail page • 655	
Exercise 14.3 – Practice improving scalability by understanding and implementing async action methods • 655	
Exercise 14.4 – Practice unit testing MVC controllers • 655	
Exercise 14.5 – Explore topics • 655	
Summary	656
 Chapter 15: Building and Consuming Web Services	 657
Building web services using the ASP.NET Core Web API	657
Understanding web service acronyms • 657	
Understanding HTTP requests and responses for Web APIs • 658	
Creating an ASP.NET Core Web API project • 660	
Reviewing the web service’s functionality • 663	
Creating a web service for the Northwind database • 665	
Creating data repositories for entities • 667	
Implementing a Web API controller • 670	
<i>Understanding action method return types • 671</i>	
Configuring the customer repository and Web API controller • 672	
Specifying problem details • 676	
Controlling XML serialization • 677	
Documenting and testing web services	678
Testing GET requests using a browser • 678	
Testing HTTP requests with the REST Client extension • 679	
<i>Making GET requests using REST Client • 679</i>	
<i>Making other requests using REST Client • 680</i>	
Understanding Swagger • 682	
Testing requests with Swagger UI • 683	
Enabling HTTP logging • 686	
Support for logging additional request headers in W3CLogger • 688	
Consuming web services using HTTP clients	688
Understanding HttpClient • 689	
Configuring HTTP clients using HttpClientFactory • 689	
Getting customers as JSON in the controller • 690	
Starting multiple projects • 692	
<i>If you are using Visual Studio 2022 • 692</i>	

<i>If you are using Visual Studio Code</i> • 693	
Starting the web service and MVC client projects • 694	
Implementing advanced features for web services	695
Building web services using Minimal APIs	695
Testing the minimal weather service • 698	
Adding weather forecasts to the Northwind website home page • 698	
Practicing and exploring	700
Exercise 15.1 – Test your knowledge • 700	
Exercise 15.2 – Practice creating and deleting customers with HttpClient • 701	
Exercise 15.3 – Explore topics • 701	
Summary	701
 Chapter 16: Building User Interfaces Using Blazor	 703
Understanding Blazor	703
JavaScript and friends • 703	
Silverlight – C# and .NET using a plugin • 704	
WebAssembly – a target for Blazor • 704	
Blazor hosting models • 704	
Blazor components • 705	
What is the difference between Blazor and Razor? • 706	
Comparing Blazor project templates	706
Reviewing the Blazor Server project template • 706	
<i>CSS and JavaScript isolation</i> • 712	
Blazor routing to page components • 713	
<i>How to define a routable page component</i> • 713	
<i>How to navigate Blazor routes</i> • 713	
<i>How to pass route parameters</i> • 713	
<i>Understanding base component classes</i> • 714	
<i>How to use the navigation link component with routes</i> • 715	
Running the Blazor Server project template • 716	
Reviewing the Blazor WebAssembly project template • 717	
<i>Deployment choices for Blazor WebAssembly apps</i> • 718	
<i>Differences between Blazor Server and Blazor WebAssembly projects</i> • 718	
<i>Similarities between Blazor Server and Blazor WebAssembly projects</i> • 722	
Building components using Blazor Server	722
Defining and testing a simple Blazor Server component • 722	
Making the component a routable page component • 723	

Getting entities into a component • 724	
Abstracting a service for a Blazor component • 727	
Defining forms using the EditForm component • 730	
Building a shared customer detail component • 731	
Building customer create, edit, and delete components • 732	
Testing the customer components • 735	
Building components using Blazor WebAssembly	735
Configuring the server for Blazor WebAssembly • 736	
Configuring the client for Blazor WebAssembly • 739	
Testing the Blazor WebAssembly components and service • 742	
Improving Blazor WebAssembly apps	744
Practicing and exploring	744
Exercise 16.1 – Test your knowledge • 744	
Exercise 16.2 – Practice by creating a times table component • 744	
Exercise 16.3 – Practice by creating a country navigation item • 744	
Exercise 16.4 – Explore topics • 745	
Summary	746
 Chapter 17: Epilogue	 747
The next steps on your C# and .NET learning journey	747
Polishing your skills with design guidelines • 747	
A companion book to continue your learning journey • 748	
Other books to take your learning further • 748	
The eighth edition coming November 2023	749
Good luck!	749
 Index	 751
