

Table of Contents

About the Authors	xlv
About the Technical Reviewers	xlvii
Acknowledgments	xlix
Introduction	ii
■Part I: Introducing C# and .NET 5	1
■Chapter 1: Introducing C# and .NET (Core) 5	3
Exploring Some Key Benefits of the .NET Core Platform.....	4
Understanding the .NET Core Support Lifecycle.....	4
Previewing the Building Blocks of the .NET Core Platform (.NET Runtime, CTS, and CLS)	5
The Role of the Base Class Libraries	6
What C# Brings to the Table.....	6
Managed vs. Unmanaged Code	10
Using Additional .NET Core–Aware Programming Languages.....	10
Getting an Overview of .NET Assemblies	11
The Role of the Common Intermediate Language	11
Benefits of CIL	14
Compiling CIL to Platform-Specific Instructions.....	15
The Role of .NET Core Type Metadata.....	15
The Role of the Assembly Manifest	16
Understanding the Common Type System.....	17
CTS Class Types.....	17
CTS Interface Types	18

CTS Structure Types	18
CTS Enumeration Types	19
CTS Delegate Types	19
CTS Type Members	19
Intrinsic CTS Data Types	20
Understanding the Common Language Specification	21
Ensuring CLS Compliance.....	22
Understanding the .NET Core Runtime.....	22
Distinguishing Between Assembly, Namespace, and Type.....	23
Accessing a Namespace Programmatically	24
Referencing External Assemblies.....	25
Exploring an Assembly Using ildasm.exe	25
Summary.....	26
■Chapter 2: Building C# Applications.....	27
Installing .NET 5	27
Understanding the .NET 5 Version Numbering Scheme.....	27
Confirming the .NET 5 Install	28
Building .NET Core Applications with Visual Studio.....	29
Installing Visual Studio 2019 (Windows).....	29
Taking Visual Studio 2019 for a Test-Drive	31
Building .NET Core Applications with Visual Studio Code.....	41
Taking Visual Studio Code for a Test-Drive	41
Finding the .NET Core and C# Documentation	45
Summary.....	45
■Part II: Core C# Programming.....	47
■Chapter 3: Core C# Programming Constructs, Part 1	49
Breaking Down a Simple C# Program	49
Using Variations of the Main() Method (Updated 7.1)	51
Using Top-Level Statements (New 9.0).....	52
Specifying an Application Error Code (Updated 9.0).....	53

Processing Command-Line Arguments (Updated 9.0).....	55
Specifying Command-Line Arguments with Visual Studio.....	57
An Interesting Aside: Some Additional Members of the System.Environment Class.....	58
Using the System.Console Class	59
Performing Basic Input and Output (I/O) with the Console Class	60
Formatting Console Output.....	61
Formatting Numerical Data	62
Formatting Numerical Data Beyond Console Applications	63
Working with System Data Types and Corresponding C# Keywords.....	63
Understanding Variable Declaration and Initialization	65
Using Intrinsic Data Types and the new Operator (Updated 9.0)	67
Understanding the Data Type Class Hierarchy.....	67
Understanding the Members of Numerical Data Types	69
Understanding the Members of System.Boolean	70
Understanding the Members of System.Char.....	70
Parsing Values from String Data.....	71
Using TryParse to Parse Values from String Data	71
Using System.DateTime and System.TimeSpan.....	72
Working with the System.Numerics Namespace	72
Using Digit Separators (New 7.0).....	74
Using Binary Literals (New 7.0/7.2).....	74
Working with String Data	75
Performing Basic String Manipulation.....	75
Performing String Concatenation	76
Using Escape Characters.....	77
Performing String Interpolation	78
Defining Verbatim Strings (Updated 8.0)	79
Working with Strings and Equality	79
Strings Are Immutable.....	81
Using the System.Text.StringBuilder Type	83

Narrowing and Widening Data Type Conversions	84
Using the checked Keyword	86
Setting Project-wide Overflow Checking.....	88
Setting Project-wide Overflow Checking (Visual Studio).....	88
Using the unchecked Keyword	89
Understanding Implicitly Typed Local Variables	89
Declaring Numerics Implicitly.....	91
Understanding Restrictions on Implicitly Typed Variables	91
Implicit Typed Data Is Strongly Typed Data.....	92
Understanding the Usefulness of Implicitly Typed Local Variables.....	93
Working with C# Iteration Constructs	94
Using the for Loop	94
Using the foreach Loop.....	95
Using Implicit Typing Within foreach Constructs	95
Using the while and do/while Looping Constructs	96
A Quick Discussion About Scope	96
Working with Decision Constructs and the Relational/Equality Operators	97
Using the if/else Statement	97
Using Equality and Relational Operators	98
Using if/else with Pattern Matching (New 7.0).....	99
Making Pattern Matching Improvements (New 9.0)	99
Using the Conditional Operator (Updated 7.2, 9.0)	100
Using Logical Operators.....	102
Using the switch Statement	102
Performing switch Statement Pattern Matching (New 7.0, Updated 9.0).....	105
Using switch Expressions (New 8.0).....	108
Summary	109
■ Chapter 4: Core C# Programming Constructs, Part 2	111
Understanding C# Arrays	111
Looking at the C# Array Initialization Syntax	112
Understanding Implicitly Typed Local Arrays	113

Defining an Array of Objects	114
Working with Multidimensional Arrays	114
Using Arrays As Arguments or Return Values	116
Using the System.Array Base Class	117
Using Indices and Ranges (New 8.0)	118
Understanding Methods	120
Understanding Expression-Bodied Members	121
Understanding Local Functions (New 7.0, Updated 9.0)	121
Understanding Static Local Functions (New 8.0)	122
Understanding Method Parameters	123
Understanding Method Parameter Modifiers	123
Understanding the Default Parameter-Passing Behavior	124
Using the out Modifier (Updated 7.0)	125
Using the ref Modifier	127
Using the in Modifier (New 7.2)	128
Using the params Modifier	129
Defining Optional Parameters	130
Using Named Arguments (Updated 7.2)	131
Understanding Method Overloading	133
Understanding the enum Type	135
Controlling the Underlying Storage for an enum	137
Declaring enum Variables	137
Using the System.Enum Type	138
Dynamically Discovering an enum's Name-Value Pairs	139
Using Enums, Flags, and Bitwise Operations	140
Understanding the Structure (aka Value Type)	142
Creating Structure Variables	144
Using Read-Only Structs (New 7.2)	145
Using Read-Only Members (New 8.0)	145
Using ref Structs (New 7.2)	146
Using Disposable ref Structs (New 8.0)	147

Understanding Value Types and Reference Types	147
Using Value Types, Reference Types, and the Assignment Operator.....	148
Using Value Types Containing Reference Types.....	150
Passing Reference Types by Value.....	152
Passing Reference Types by Reference.....	153
Final Details Regarding Value Types and Reference Types.....	154
Understanding C# Nullable Types.....	155
Using Nullable Value Types.....	156
Using Nullable Reference Types (New 8.0).....	157
Operating on Nullable Types.....	159
Understanding Tuples (New/Updated 7.0).....	161
Getting Started with Tuples.....	162
Using Inferred Variable Names (Updated 7.1).....	163
Understanding Tuple Equality/Inequality (New 7.3).....	163
Understanding Tuples as Method Return Values.....	164
Understanding Discards with Tuples.....	164
Understanding Tuple Pattern Matching switch Expressions (New 8.0).....	165
Deconstructing Tuples.....	165
Summary.....	167
■Part III: Object Oriented Programming with C#.....	169
■Chapter 5: Understanding Encapsulation.....	171
Introducing the C# Class Type.....	171
Allocating Objects with the new Keyword.....	173
Understanding Constructors.....	174
Understanding the Role of the Default Constructor.....	174
Defining Custom Constructors.....	175
Understanding the Default Constructor Revisited.....	177
Understanding the Role of the this Keyword.....	178
Chaining Constructor Calls Using this.....	179
Observing Constructor Flow.....	182
Revisiting Optional Arguments.....	183

Understanding the static Keyword 184

- Defining Static Field Data 185
- Defining Static Methods 187
- Defining Static Constructors 188
- Defining Static Classes 190
- Importing Static Members via the C# using Keyword 191

Defining the Pillars of OOP 192

- Understanding the Role of Encapsulation 192
- Understanding the Role of Inheritance 193
- Understanding the Role of Polymorphism 194

Understanding C# Access Modifiers (Updated 7.2) 196

- Using the Default Access Modifiers 196
- Using Access Modifiers and Nested Types 197

Understanding the First Pillar: C#'s Encapsulation Services 197

- Encapsulation Using Traditional Accessors and Mutators 198
- Encapsulation Using Properties 201
- Using Properties Within a Class Definition 204
- Properties Read-Only Properties 205
- Properties Write-Only Properties 206
- Mixing Private and Public Get/Set Methods on Properties 206
- Revisiting the static Keyword: Defining Static Properties 207
- Pattern Matching with Property Patterns (New 8.0) 207

Understanding Automatic Properties 209

- Interacting with Automatic Properties 210
- Properties Automatic Properties and Default Values 211
- Initializing Automatic Properties 213

Understanding Object Initialization 213

- Looking at the Object Initialization Syntax 214
- Using init-Only Setters (New 9.0) 215
- Calling Custom Constructors with Initialization Syntax 216
- Initializing Data with Initialization Syntax 217

Working with Constant and Read-Only Field Data	218
Understanding Constant Field Data	219
Understanding Read-Only Fields	220
Understanding Static Read-Only Fields	221
Understanding Partial Classes	221
Using Records (New 9.0).....	222
Understanding Equality with Record Types	225
Summary.....	227
■Chapter 6: Understanding Inheritance and Polymorphism.....	229
Understanding the Basic Mechanics of Inheritance.....	229
Specifying the Parent Class of an Existing Class.....	230
Regarding Multiple Base Classes	232
Using the sealed Keyword	232
Revisiting Visual Studio Class Diagrams	233
Understanding the Second Pillar of OOP: The Details of Inheritance	235
Calling Base Class Constructors with the base Keyword	236
Keeping Family Secrets: The protected Keyword	238
Adding a sealed Class	239
Understanding Inheritance with Record Types (New 9.0).....	240
Programming for Containment/Delegation.....	243
Understanding Nested Type Definitions.....	244
Understanding the Third Pillar of OOP: C#'s Polymorphic Support.....	246
Using the virtual and override Keywords.....	247
Overriding Virtual Members with Visual Studio/Visual Studio Code	249
Sealing Virtual Members.....	249
Understanding Abstract Classes.....	250
Understanding the Polymorphic Interface	252
Understanding Member Shadowing	255
Understanding Base Class/Derived Class Casting Rules.....	257
Using the C# as Keyword.....	259
Using the C# is Keyword (Updated 7.0, 9.0)	260

Revisiting Pattern Matching (New 7.0) 262

Understanding the Super Parent Class: System.Object..... 263

 Overriding System.Object.ToString()..... 266

 Overriding System.Object.Equals() 266

 Overriding System.Object.GetHashCode() 267

 Testing Your Modified Person Class..... 268

 Using the Static Members of System.Object 269

Summary..... 270

■Chapter 7: Understanding Structured Exception Handling 271

 Ode to Errors, Bugs, and Exceptions 271

 The Role of .NET Exception Handling 272

 The Building Blocks of .NET Exception Handling..... 273

 The System.Exception Base Class..... 273

 The Simplest Possible Example 274

 Throwing a General Exception..... 277

 Catching Exceptions 278

 Throw As Expression (New 7.0)..... 280

 Configuring the State of an Exception..... 280

 The TargetSite Property 280

 The StackTrace Property..... 281

 The HelpLink Property 281

 The Data Property..... 282

 System-Level Exceptions (System.SystemException)..... 284

 Application-Level Exceptions (System.ApplicationException)..... 285

 Building Custom Exceptions, Take 1 285

 Building Custom Exceptions, Take 2 287

 Building Custom Exceptions, Take 3 288

 Processing Multiple Exceptions 289

 General catch Statements 292

 Rethrowing Exceptions..... 292

 Inner Exceptions 293

The finally Block	294
Exception Filters	294
Debugging Unhandled Exceptions Using Visual Studio	295
Summary	296
■ Chapter 8: Working with Interfaces.....	297
Understanding Interface Types	297
Interface Types vs. Abstract Base Classes.....	298
Defining Custom Interfaces	301
Implementing an Interface	302
Invoking Interface Members at the Object Level.....	304
Obtaining Interface References: The as Keyword	305
Obtaining Interface References: The is Keyword (Updated 7.0)	306
Default Implementations (New 8.0).....	306
Static Constructors and Members (New 8.0)	308
Interfaces as Parameters	308
Interfaces as Return Values.....	310
Arrays of Interface Types.....	311
Implementing Interfaces Using Automatically	312
Explicit Interface Implementation	314
Designing Interface Hierarchies	317
Interface Hierarchies with Default Implementations (New 8.0).....	318
<i>Multiple Inheritance with Interface Types</i>	320
The IEnumerable and IEnumerator Interfaces.....	322
Building Iterator Methods with the yield Keyword.....	325
Building a Named Iterator.....	327
The ICloneable Interface.....	329
A More Elaborate Cloning Example.....	331
The IComparable Interface	334
Specifying Multiple Sort Orders with IComparer	337
Custom Properties and Custom Sort Types.....	338
Summary.....	339

■Chapter 9: Understanding Object Lifetime.....	341
Classes, Objects, and References	341
The Basics of Object Lifetime.....	343
The CIL of new	343
Setting Object References to null	345
Determining If an Object Is Live	345
Understanding Object Generations.....	347
Ephemeral Generations and Segments	348
Garbage Collection Types	349
Background Garbage Collection	349
The System.GC Type	349
Forcing a Garbage Collection.....	351
Building Finalizable Objects	354
Overriding System.Object.Finalize().....	355
Detailing the Finalization Process	357
Building Disposable Objects.....	357
Reusing the C# using Keyword.....	359
Using Declarations (New 8.0)	361
Building Finalizable and Disposable Types.....	362
A Formalized Disposal Pattern	362
Understanding Lazy Object Instantiation.....	364
Customizing the Creation of the Lazy Data.....	367
Summary	368
■Part IV: Advanced C# Programming	369
■Chapter 10: Collections and Generics.....	371
The Motivation for Collection Classes	371
The System.Collections Namespace	373
A Survey of System.Collections.Specialized Namespace.....	374

The Problems of Nongeneric Collections	375
The Issue of Performance.....	376
The Issue of Type Safety.....	379
A First Look at Generic CollectionsT.....	383
The Role of Generic Type Parameters	384
Specifying Type Parameters for Generic Classes/Structures	385
Specifying Type Parameters for Generic Members.....	386
Specifying Type Parameters for Generic Interfaces.....	387
The System.Collections.Generic Namespace.....	388
Understanding Collection Initialization Syntax	389
Working with the List<T> Class	391
Working with the Stack<T> Class.....	393
Working with the Queue<T> Class.....	394
Working with the SortedSet<T> Class.....	395
Working with the Dictionary<TKey, TValue> Class.....	397
The System.Collections.ObjectModel Namespace	398
Working with ObservableCollection<T>.....	398
Creating Custom Generic Methods.....	400
Inference of Type Parameters.....	402
Creating Custom Generic Structures and Classes.....	403
Default Value Expressions with Generics	404
Default Literal Expressions (New 7.1).....	405
Pattern Matching with Generics (New 7.1).....	406
Constraining Type Parameters.....	406
Examples of Using the where Keyword.....	407
The Lack of Operator Constraints.....	408
Summary.....	409
■Chapter 11: Advanced C# Language Features	411
Understanding Indexer Methods	411
Indexing Data Using String Values.....	413
Overloading Indexer Methods.....	415

Indexers with Multiple Dimensions	415
Indexer Definitions on Interface Types.....	416
Understanding Operator Overloading	417
Overloading Binary Operators.....	418
What of the += and -= Operators?	420
Overloading Unary Operators.....	420
Overloading Equality Operators	421
Overloading Comparison Operators.....	422
Final Thoughts Regarding Operator Overloading.....	423
Understanding Custom Type Conversions	423
Recall: Numerical Conversions	423
Recall: Conversions Among Related Class Types.....	424
Creating Custom Conversion Routines	425
Additional Explicit Conversions for the Square Type.....	427
Defining Implicit Conversion Routines.....	428
Understanding Extension Methods.....	429
Defining Extension Methods.....	430
Invoking Extension Methods.....	431
Importing Extension Methods.....	432
Extending Types Implementing Specific Interfaces	432
Extension Method GetEnumerator Support (New 9.0)	434
Understanding Anonymous Types	436
Defining an Anonymous Type.....	436
The Internal Representation of Anonymous Types.....	437
The Implementation of ToString() and GetHashCode()	439
The Semantics of Equality for Anonymous Types	439
Anonymous Types Containing Anonymous Types	441
Working with Pointer Types.....	442
The unsafe Keyword.....	443
Working with the * and & Operators.....	445
An Unsafe (and Safe) Swap Function	446

Field Access via Pointers (the -> Operator).....	447
The stackalloc Keyword.....	448
Pinning a Type via the fixed Keyword.....	448
The sizeof Keyword.....	449
Summary.....	450
■Chapter 12: Delegates, Events, and Lambda Expressions	451
Understanding the Delegate Type.....	452
Defining a Delegate Type in C#.....	452
The System.MulticastDelegate and System.Delegate Base Classes.....	454
The Simplest Possible Delegate Example	456
Investigating a Delegate Object.....	458
Sending Object State Notifications Using Delegates.....	459
Enabling Multicasting.....	462
Removing Targets from a Delegate's Invocation List.....	464
Method Group Conversion Syntax	465
Understanding Generic Delegates.....	466
The Generic Action<> and Func<> Delegates.....	467
Understanding C# Events	469
The C# event Keyword.....	471
Events Under the Hood.....	472
Listening to Incoming Events.....	473
Simplifying Event Registration Using Visual Studio.....	474
Creating Custom Event Arguments.....	476
The Generic EventHandler<T> Delegate.....	478
Understanding C# Anonymous Methods	478
Accessing Local Variables	480
Using static with Anonymous Methods (New 9.0).....	481
Discards with Anonymous Methods (New 9.0).....	482
Understanding Lambda Expressions.....	482
Dissecting a Lambda Expression.....	485
Processing Arguments Within Multiple Statements.....	486

Lambda Expressions with Multiple (or Zero) Parameters.....	487
Using static with Lambda Expressions (New 9.0).....	488
Discards with Lambda Expressions (New 9.0)	489
Retrofitting the CarEvents Example Using Lambda Expressions.....	489
Lambdas and Expression-Bodied Members (Updated 7.0).....	490
Summary.....	491
■Chapter 13: LINQ to Objects.....	493
LINQ-Specific Programming Constructs.....	493
Implicit Typing of Local Variables	494
Object and Collection Initialization Syntax.....	495
Lambda Expressions.....	495
Extension Methods	496
Anonymous Types.....	497
Understanding the Role of LINQ	497
LINQ Expressions Are Strongly Typed	498
The Core LINQ Assemblies.....	498
Applying LINQ Queries to Primitive Arrays	498
Once Again, Using Extension Methods	500
Once Again, Without LINQ	501
Reflecting Over a LINQ Result Set	501
LINQ and Implicitly Typed Local Variables	503
LINQ and Extension Methods.....	504
The Role of Deferred Execution	505
The Role of Immediate Execution.....	506
Returning the Result of a LINQ Query.....	508
Returning LINQ Results via Immediate Execution	509
Applying LINQ Queries to Collection Objects.....	510
Accessing Contained Subobjects	510
Applying LINQ Queries to Nongeneric Collections	511
Filtering Data Using OfType<T>()	512

Investigating the C# LINQ Query Operators	513
Basic Selection Syntax	514
Obtaining Subsets of Data	515
Projecting New Data Types	516
Projecting to Different Data Types	517
Obtaining Counts Using Enumerable	518
Reversing Result Sets.....	518
Sorting Expressions	518
LINQ As a Better Venn Diagramming Tool	519
Removing Duplicates.....	521
LINQ Aggregation Operations.....	521
The Internal Representation of LINQ Query Statements.....	522
Building Query Expressions with Query Operators (Revisited)	523
Building Query Expressions Using the Enumerable Type and Lambda Expressions.....	523
Building Query Expressions Using the Enumerable Type and Anonymous Methods	525
Building Query Expressions Using the Enumerable Type and Raw Delegates.....	525
Summary.....	527
■Chapter 14: Processes, AppDomains, and Load Contexts	529
The Role of a Windows Process	529
The Role of Threads.....	530
Interacting with Processes Using .NET Core	531
Enumerating Running Processes.....	533
Investigating a Specific Process.....	534
Investigating a Process's Thread Set.....	535
Investigating a Process's Module Set.....	536
Starting and Stopping Processes Programmatically	538
Controlling Process Startup Using the ProcessStartInfo Class.....	539
Leveraging OS Verbs with ProcessStartInfo	540

Understanding .NET Application Domains 541

 The System.AppDomain Class 542

 Interacting with the Default Application Domain 542

 Enumerating Loaded Assemblies 543

Assembly Isolation with Application Load Contexts 544

Summarizing Processes, AppDomains, and Load Contexts..... 547

Summary..... 548

■Chapter 15: Multithreaded, Parallel, and Async Programming 549

The Process/AppDomain/Context/Thread Relationship..... 549

 The Problem of Concurrency 550

 The Role of Thread Synchronization 551

The System.Threading Namespace..... 551

The System.Threading.Thread Class 552

 Obtaining Statistics About the Current Thread of Execution 553

 The Name Property 554

 The Priority Property 554

Manually Creating Secondary Threads..... 555

 Working with the ThreadStart Delegate 555

 Working with the ParameterizedThreadStart Delegate 557

 The AutoResetEvent Class 558

 Foreground Threads and Background Threads 559

The Issue of Concurrency 560

 Synchronization Using the C# lock Keyword 562

 Synchronization Using the System.Threading.Monitor Type 564

 Synchronization Using the System.Threading.Interlocked Type 565

Programming with Timer Callbacks 566

 Using a Stand-Alone Discard (New 7.0)..... 567

Understanding the ThreadPool 568

Parallel Programming Using the Task Parallel Library 569

 The System.Threading.Tasks Namespace 569

 The Role of the Parallel Class 569

Data Parallelism with the Parallel Class	570
Accessing UI Elements on Secondary Threads.....	574
The Task Class	575
Handling Cancellation Request.....	575
Task Parallelism Using the Parallel Class	577
Parallel LINQ Queries (PLINQ).....	580
Opting in to a PLINQ Query	581
Cancelling a PLINQ Query	581
Async Calls with the async/await.....	582
A First Look at the C# async and await Keywords (Updated 7.1, 9.0)	582
SynchronizationContext and async/await.....	584
The Role of ConfigureAwait	585
Naming Conventions for Asynchronous Methods	585
Void Async Methods.....	586
Async Methods with Multiple Awaits.....	587
Calling Async Methods from Non-async Methods	588
Await in catch and finally Blocks.....	589
Generalized Async Return Types (New 7.0)	589
Local Functions (New 7.0)	590
Cancelling async/await Operations	591
Asynchronous Streams (New 8.0)	593
Wrapping Up async and await.....	594
Summary.....	594
■Part V: Programming with .NET Core Assemblies	595
■Chapter 16: Building and Configuring Class Libraries.....	597
Defining Custom Namespaces	597
Resolving Name Clashes with Fully Qualified Names	599
Resolving Name Clashes with Aliases	600
Creating Nested Namespaces	601
Change the Root Namespace of Visual Studio.....	602

The Role of .NET Core Assemblies..... 603

- Assemblies Promote Code Reuse..... 603
- Assemblies Establish a Type Boundary 604
- Assemblies Are Versionable Units 604
- Assemblies Are Self-Describing 604

Understanding the Format of a .NET Core Assembly..... 604

- Installing the C++ Profiling Tools 605
- The Operating System (Windows) File Header 606
- The CLR File Header 606
- CIL Code, Type Metadata, and the Assembly Manifest..... 607
- Optional Assembly Resources 608

Class Libraries vs. Console Applications 608

.NET Standard vs. .NET Core Class Libraries..... 608

Configuring Applications 609

Building and Consuming a .NET Core Class Library 610

- Exploring the Manifest..... 613
- Exploring the CIL..... 615
- Exploring the Type Metadata 615
- Building a C# Client Application..... 616
- Building a Visual Basic Client Application..... 618
- Cross-Language Inheritance in Action..... 618
- Exposing internal Types to Other Assemblies..... 619

NuGet and .NET Core 620

- Packaging Assemblies with NuGet 620
- Referencing NuGet Packages 622

Publishing Console Applications (Updated .NET 5)..... 623

- Publishing Framework-Dependent Applications..... 623
- Publishing Self-Contained Applications..... 624

How .NET Core Locates Assemblies 625

Summary 626

■ Chapter 17: Type Reflection, Late Binding, and Attribute-Based Programming..... 627

- The Necessity of Type Metadata..... 627**
 - Viewing (Partial) Metadata for the EngineStateEnum Enumeration 628
 - Viewing (Partial) Metadata for the Car Type 629
 - Examining a TypeRef 630
 - Documenting the Defining Assembly..... 631
 - Documenting Referenced Assemblies..... 631
 - Documenting String Literals 632
- Understanding Reflection..... 632**
 - The System.Type Class..... 633
 - Obtaining a Type Reference Using System.Object.GetType() 634
 - Obtaining a Type Reference Using typeof()..... 634
 - Obtaining a Type Reference Using System.Type.GetType() 635
- Building a Custom Metadata Viewer 635**
 - Reflecting on Methods..... 636
 - Reflecting on Fields and Properties..... 636
 - Reflecting on Implemented Interfaces..... 637
 - Displaying Various Odds and Ends 637
 - Adding the Top-Level Statements..... 638
 - Reflecting on Static Types 639
 - Reflecting on Generic Types 640
 - Reflecting on Method Parameters and Return Values 640
- Dynamically Loading Assemblies 641**
- Reflecting on Framework Assemblies..... 643**
- Understanding Late Binding 645**
 - The System.Activator Class 645
 - Invoking Methods with No Parameters..... 647
 - Invoking Methods with Parameters..... 647

Understanding the Role of .NET Attributes 648

- Attribute Consumers..... 649
- Applying Attributes in C# 649
- C# Attribute Shorthand Notation..... 650
- Specifying Constructor Parameters for Attributes 651
- The Obsolete Attribute in Action 651

Building Custom Attributes..... 652

- Applying Custom Attributes 653
- Named Property Syntax..... 654
- Restricting Attribute Usage..... 654

Assembly-Level Attributes 655

- Using the Project File for Assembly Attributes 656

Reflecting on Attributes Using Early Binding..... 657

Reflecting on Attributes Using Late Binding..... 658

Putting Reflection, Late Binding, and Custom Attributes in Perspective 659

Building an Extendable Application 660

- Building the Multiproject ExtendableApp Solution 661
- Building CommonSnappableTypes.dll 664
- Building the C# Snap-In..... 664
- Building the Visual Basic Snap-In..... 665
- Adding the Code for the ExtendableApp..... 665

Summary..... 667

■Chapter 18: Dynamic Types and the Dynamic Language Runtime 669

- The Role of the C# dynamic Keyword**..... 669
 - Calling Members on Dynamically Declared Data..... 671
 - The Scope of the dynamic Keyword 673
 - Limitations of the dynamic Keyword 673
 - Practical Uses of the dynamic Keyword 674
- The Role of the Dynamic Language Runtime** 674
 - The Role of Expression Trees..... 675
 - Dynamic Runtime Lookup of Expression Trees..... 675

Simplifying Late-Bound Calls Using Dynamic Types	676
Leveraging the dynamic Keyword to Pass Arguments	677
Simplifying COM Interoperability Using Dynamic Data (Windows Only).....	679
The Role of Primary Interop Assemblies.....	680
Embedding Interop Metadata	680
Common COM Interop Pain Points.....	682
COM Interop Using C# Dynamic Data	682
Summary.....	685
■ Chapter 19: Understanding CIL and the Role of Dynamic Assemblies.....	687
Motivation's for Learning the Grammar of CIL	687
Examining CIL Directives, Attributes, and Opcodes	688
The Role of CIL Directives.....	689
The Role of CIL Attributes	689
The Role of CIL Opcodes.....	689
The CIL Opcode/CIL Mnemonic Distinction.....	690
Pushing and Popping: The Stack-Based Nature of CIL.....	691
Understanding Round-Trip Engineering	692
The Role of CIL Code Labels	695
Interacting with CIL: Modifying an *.il File.....	695
Compiling CIL Code.....	696
Understanding CIL Directives and Attributes	697
Specifying Externally Referenced Assemblies in CIL.....	697
Defining the Current Assembly in CIL	697
Defining Namespaces in CIL.....	698
Defining Class Types in CIL.....	699
Defining and Implementing Interfaces in CIL.....	700
Defining Structures in CIL.....	701
Defining Enums in CIL.....	701
Defining Generics in CIL	701
Compiling the CILTypes.il File	702

.NET Base Class Library, C#, and CIL Data Type Mappings.....	702
Defining Type Members in CIL.....	703
Defining Field Data in CIL	703
Defining Type Constructors in CIL.....	704
Defining Properties in CIL.....	705
Defining Member Parameters.....	705
Examining CIL Opcodes.....	706
The .maxstack Directive	708
Declaring Local Variables in CIL	708
Mapping Parameters to Local Variables in CIL	709
The Hidden this Reference	710
Representing Iteration Constructs in CIL	711
The Final Word on CIL.....	711
Understanding Dynamic Assemblies.....	712
Exploring the System.Reflection.Emit Namespace.....	712
The Role of the System.Reflection.Emit.ILGenerator.....	713
Emitting a Dynamic Assembly	714
Emitting the Assembly and Module Set.....	716
The Role of the ModuleBuilder Type	717
Emitting the HelloClass Type and the String Member Variable.....	718
Emitting the Constructors.....	718
Emitting the SayHello() Method	719
Using the Dynamically Generated Assembly	719
Summary.....	720
■Part VI: File Handling, Object Serialization, and Data Access.....	721
■Chapter 20: File I/O and Object Serialization.....	723
Exploring the System.IO Namespace	723
The Directory(Info) and File(Info) Types.....	724
The Abstract FileSystemInfo Base Class	725

Working with the DirectoryInfo Type	725
Enumerating Files with the DirectoryInfo Type	727
Creating Subdirectories with the DirectoryInfo Type	728
Working with the Directory Type	729
Working with the DriveInfo Class Type	730
Working with the FileInfo Class	731
The FileInfo.Create() Method	732
The FileInfo.Open() Method	733
The FileInfo.OpenRead() and FileInfo.OpenWrite() Methods	734
The FileInfo.OpenText() Method	735
The FileInfo.CreateText() and FileInfo.AppendText() Methods	735
Working with the File Type	736
Additional File-centric Members	736
The Abstract Stream Class	737
Working with FileStreams	738
Working with StreamWriters and StreamReaders	740
Writing to a Text File	740
Reading from a Text File	741
Directly Creating StreamWriter/StreamReader Types	742
Working with StringWriters and StringReaders	743
Working with BinaryWriters and BinaryReaders	744
Watching Files Programmatically	746
Understanding Object Serialization	748
The Role of Object Graphs	748
Creating the Samples Types and Top-Level Statements	749
Serializing and Deserialization with the XmlSerializer	752
Serializing and Deserialization with System.Text.Json	755
Summary	762

Chapter 21: Data Access with ADO.NET	763
ADO.NET vs. ADO	763
Understanding ADO.NET Data Providers.....	764
ADO.NET Data Providers	765
The Types of the System.Data Namespace	766
The Role of the IDbConnection Interface	767
The Role of the IDbTransaction Interface.....	767
The Role of the IDbCommand Interface.....	767
The Role of the IDbDataParameter and IDataParameter Interfaces	768
The Role of the IDbDataAdapter and IDataAdapter Interfaces.....	769
The Role of the IDataReader and IDataRecord Interfaces	769
Abstracting Data Providers Using Interfaces.....	770
Setting Up SQL Server and Azure Data Studio	773
Installing SQL Server	773
Installing a SQL Server IDE.....	775
Connecting to SQL Server.....	775
Restoring the AutoLot Database Backup.....	779
Copying the Backup File to Your Container.....	779
Restoring the Database with SSMS.....	779
Restoring the Database with Azure Data Studio.....	781
Creating the AutoLot Database.....	782
Creating the Database	783
Creating the Tables	783
Creating the Table Relationships	785
Creating the GetPetName() Stored Procedure	787
Adding Test Records	787
The ADO.NET Data Provider Factory Model	789
A Complete Data Provider Factory Example	790
A Potential Drawback with the Data Provider Factory Model.....	794

Diving Deeper into Connections, Commands, and DataReaders	795
Working with Connection Objects.....	796
Working with Command Objects	798
Working with Data Readers	799
Working with Create, Update, and Delete Queries.....	801
Create the Car and CarViewModel Classes.....	802
Adding the InventoryDal Class.....	803
Adding the Deletion Logic.....	807
Adding the Update Logic.....	808
Working with Parameterized Command Objects	808
Executing a Stored Procedure	812
Creating a Console-Based Client Application	813
Understanding Database Transactions	814
Key Members of an ADO.NET Transaction Object	815
Adding a Transaction Method to InventoryDal.....	816
Testing Your Database Transaction	818
Executing Bulk Copies with ADO.NET	819
Exploring the SqlBulkCopy Class	819
Creating a Custom Data Reader	819
Executing the Bulk Copy	823
Testing the Bulk Copy	824
Summary.....	825
■Part VII: Entity Framework Core	827
■Chapter 22: Introducing Entity Framework Core.....	829
Object-Relational Mappers.....	830
Understanding the Role of the Entity Framework Core	830
The Building Blocks of the Entity Framework	831
The DbContext Class.....	831
The DbSet<T> Class.....	836
The ChangeTracker	838
Entities.....	839

Query Execution	859
Mixed Client-Server Evaluation	860
Tracking vs. NoTracking Queries	860
Notable EF Core Features.....	860
Handling Database-Generated Values	861
Concurrency Checking.....	861
Connection Resiliency	863
Related Data	863
Global Query Filters	866
Raw SQL Queries with LINQ.....	868
Batching of Statements	869
Owned Entity Types	870
Database Function Mapping.....	872
The EF Core Global Tool CLI Commands	872
The Migrations Commands.....	874
The Database Commands.....	877
The DbContext Commands	878
Summary.....	880
■ Chapter 23: Build a Data Access Layer with Entity Framework Core.....	881
Code First or Database First.....	881
Create the AutoLot.Dal and AutoLot.Models Projects	882
Scaffold the DbContext and Entities.....	883
Switch to Code First.....	883
Create the DbContext Design-Time Factory	884
Create the Initial Migration	884
Applying the Migration	885
Update the Model	886
The Entities.....	886
The ApplicationDbContext.....	895
Create the Migration and Update the Database.....	901

Add the Database View and Stored Procedure.....	902
Add the MigrationHelpers Class	902
Update and Apply the Migration	903
Add the ViewModel.....	904
Add the ViewModel	904
Add the ViewModel to the ApplicationDbContext.....	905
Adding Repositories	905
Adding the IRepo Base Interface	906
Adding the BaseRepo	906
Entity-Specific Repo Interfaces	910
Implement the Entity-Specific Repositories	912
Programmatic Database and Migration Handling	917
Drop, Create, and Clean the Database	917
Data Initialization.....	919
Create the Sample Data.....	919
Load the Sample Data	920
Setting Up the Test-Drives.....	922
Create the Project.....	922
Configure the Project.....	923
Create the Test Helper	923
Add the BaseTest Class	924
Add the EnsureAutoLotDatabase Test Fixture Class	926
Add the Integration Test Classes	927
Executing the Tests.....	930
Querying the Database.....	930
Entity State	930
LINQ Queries.....	930
SQL Queries with LINQ.....	949
Aggregate Methods	950
Any() and All()	951
Getting Data from Stored Procedures.....	952

Creating Records..... 953

- Entity State 954
- Add a Single Record 954
- Add a Single Record Using Attach 955
- Add Multiple Records at Once 955
- Identity Column Considerations When Adding Records 956
- Adding an Object Graph..... 956

Updating Records..... 957

- Entity State 958
- Update Tracked Entities 958
- Update Nontracked Entities 959
- Concurrency Checking..... 960

Deleting Records 961

- Entity State 961
- Delete Tracked Records 961
- Delete Nontracked Entities 962
- Catch Cascade Delete Failures 962
- Concurrency Checking..... 963

Summary..... 963

■Part VIII: Windows Client Development 965

■Chapter 24: Introducing Windows Presentation Foundation and XAML..... 967

- The Motivation Behind WPF**..... 967
 - Unifying Diverse APIs..... 968
 - Providing a Separation of Concerns via XAML..... 968
 - Providing an Optimized Rendering Model 969
 - Simplifying Complex UI Programming 969
- Investigating the WPF Assemblies**..... 970
 - The Role of the Application Class 971
 - Constructing an Application Class 972
 - Enumerating the Windows Collection 973
 - The Role of the Window Class 973

Understanding the Syntax of WPF XAML	977
Introducing Kaxaml.....	977
XAML XML Namespaces and XAML “Keywords”.....	979
Controlling Class and Member Variable Visibility.....	982
XAML Elements, XAML Attributes, and Type Converters	982
Understanding XAML Property-Element Syntax	983
Understanding XAML Attached Properties.....	984
Understanding XAML Markup Extensions.....	985
Building WPF Applications Using Visual Studio	987
The WPF Project Templates	988
The Toolbox and XAML Designer/Editor	988
Setting Properties Using the Properties Window.....	989
Handling Events Using the Properties Window.....	991
Handling Events in the XAML Editor	992
The Document Outline Window.....	993
Enable or Disable the XAML Debugger.....	994
Examining the App.xaml File	995
Mapping the Window XAML Markup to C# Code	996
The Role of BAML	998
Solving the Mystery of Main().....	998
Interacting with Application-Level Data.....	999
Handling the Closing of a Window Object.....	1000
Intercepting Mouse Events.....	1001
Intercepting Keyboard Events.....	1002
Summary.....	1003
■Chapter 25: WPF Controls, Layouts, Events, and Data Binding.....	1005
A Survey of the Core WPF Controls	1005
The WPF Ink Controls.....	1006
The WPF Document Controls	1006
WPF Common Dialog Boxes.....	1007

A Brief Review of the Visual Studio WPF Designer 1007

- Working with WPF Controls Using Visual Studio 1007
- Working with the Document Outline Editor 1008

Controlling Content Layout Using Panels 1009

- Positioning Content Within Canvas Panels 1010
- Positioning Content Within WrapPanel Panels 1012
- Positioning Content Within StackPanel Panels 1013
- Positioning Content Within Grid Panels 1015
- Grids with GridSplitter Types 1017
- Positioning Content Within DockPanel Panels 1018
- Enabling Scrolling for Panel Types 1018
- Configuring Panels Using the Visual Studio Designers 1019

Building a Window’s Frame Using Nested Panels 1023

- Building the Menu System 1024
- Building Menus Visually 1025
- Building the Toolbar 1025
- Building the Status Bar 1026
- Finalizing the UI Design 1026
- Implementing the MouseEnter/MouseLeave Event Handlers 1027
- Implementing the Spell-Checking Logic 1028

Understanding WPF Commands 1028

- The Intrinsic Command Objects 1029
- Connecting Commands to the Command Property 1030
- Connecting Commands to Arbitrary Actions 1030
- Working with the Open and Save Commands 1031

Understanding Routed Events 1033

- The Role of Routed Bubbling Events 1035
- Continuing or Halting Bubbling 1036
- The Role of Routed Tunneling Events 1036

A Deeper Look at WPF APIs and Controls 1038

- Working with the TabControl 1038

Building the Ink API Tab	1039
Designing the Toolbar	1039
The RadioButton Control	1040
Add the Save, Load, and Delete Buttons	1040
Add the InkCanvas Control	1041
Preview the Window	1041
Handling Events for the Ink API Tab	1041
Add Controls to the Toolbox	1042
The InkCanvas Control	1043
The ComboBox Control	1045
Saving, Loading, and Clearing InkCanvas Data	1047
Introducing the WPF Data-Binding Model	1048
Building the Data Binding Tab	1048
Establishing Data Bindings	1049
The DataContext Property	1049
Formatting the Bound Data	1050
Data Conversion Using IValueConverter	1050
Establishing Data Bindings in Code	1052
Building the DataGrid Tab	1053
Understanding the Role of Dependency Properties	1055
Examining an Existing Dependency Property	1057
Important Notes Regarding CLR Property Wrappers	1059
Building a Custom Dependency Property	1060
Adding a Data Validation Routine	1063
Responding to the Property Change	1063
Summary	1064
■ Chapter 26: WPF Graphics Rendering Services	1065
Understanding WPF's Graphical Rendering Services	1065
WPF Graphical Rendering Options	1066
Rendering Graphical Data Using Shapes	1067
Adding Rectangles, Ellipses, and Lines to a Canvas	1068

Removing Rectangles, Ellipses, and Lines from a Canvas..... 1071

Working with Polylines and Polygons..... 1072

Working with Paths..... 1072

WPF Brushes and Pens 1075

 Configuring Brushes Using Visual Studio..... 1076

 Configuring Brushes in Code 1079

 Configuring Pens 1080

Applying Graphical Transformations..... 1081

 A First Look at Transformations 1082

 Transforming Your Canvas Data..... 1083

Working with the Visual Studio Transform Editor 1085

 Building the Initial Layout..... 1085

 Applying Transformations at Design Time 1087

 Transforming the Canvas in Code..... 1088

Rendering Graphical Data Using Drawings and Geometries 1089

 Building a DrawingBrush Using Geometries..... 1089

 Painting with the DrawingBrush..... 1090

 Containing Drawing Types in a DrawingImage 1091

Working with Vector Images..... 1092

 Converting a Sample Vector Graphic File into XAML 1092

 Importing the Graphical Data into a WPF Project..... 1093

 Interacting with the Sign 1094

Rendering Graphical Data Using the Visual Layer 1094

 The Visual Base Class and Derived Child Classes 1094

 A First Look at Using the DrawingVisual Class 1095

 Rendering Visual Data to a Custom Layout Manager..... 1097

 Responding to Hit-Test Operations 1099

Summary 1101

■ Chapter 27: WPF Resources, Animations, Styles, and Templates	1103
Understanding the WPF Resource System	1103
Working with Binary Resources.....	1104
Working with Object (Logical) Resources	1108
The Role of the Resources Property	1108
Defining Window-wide Resources.....	1108
The {StaticResource} Markup Extension.....	1112
The {DynamicResource} Markup Extension.....	1112
Application-Level Resources	1113
Defining Merged Resource Dictionaries	1114
Defining a Resource-Only Assembly.....	1115
Understanding WPF's Animation Services	1116
The Role of the Animation Class Types.....	1116
The To, From, and By Properties	1117
The Role of the Timeline Base Class.....	1117
Authoring an Animation in C# Code	1118
Controlling the Pace of an Animation	1119
Reversing and Looping an Animation	1120
Authoring Animations in XAML	1121
The Role of Storyboards	1122
The Role of Event Triggers.....	1122
Animation Using Discrete Key Frames.....	1123
Understanding the Role of WPF Styles	1124
Defining and Applying a Style.....	1124
Overriding Style Settings.....	1125
The Effect of TargetType on Styles.....	1125
Subclassing Existing Styles	1127
Defining Styles with Triggers.....	1128
Defining Styles with Multiple Triggers.....	1128
Animated Styles.....	1129
Assigning Styles Programmatically	1129

Logical Trees, Visual Trees, and Default Templates 1131

- Programmatically Inspecting a Logical Tree 1132
- Programmatically Inspecting a Visual Tree 1133
- Programmatically Inspecting a Control's Default Template 1135

Building a Control Template with the Trigger Framework 1137

- Templates as Resources 1138
- Incorporating Visual Cues Using Triggers 1139
- The Role of the {TemplateBinding} Markup Extension 1140
- The Role of ContentPresenter 1141
- Incorporating Templates into Styles 1141

Summary 1142

■Chapter 28: WPF Notifications, Validations, Commands, and MVVM..... 1143

- Introducing Model-View-ViewModel 1143**
 - The Model 1143
 - The View 1144
 - The View Model 1144
 - Anemic Models or Anemic View Models 1144
- The WPF Binding Notification System 1145**
 - Observable Models and Collections 1145
 - Adding Bindings and Data 1147
 - Programmatically Changing the Vehicle Data 1147
 - Observable Models 1148
 - Observable Collections 1150
 - Wrapping Up Notifications and Observables 1152
- WPF Validations 1152**
 - Updating the Sample for the Validation Examples 1153
 - The Validation Class 1153
 - Validation Options 1153
 - Leverage Data Annotations with WPF 1163
 - Customizing the ErrorTemplate 1165
 - Wrapping Up Validations 1167

Creating Custom Commands	1167
Implementing the ICommand Interface	1168
Adding the ChangeColorCommand.....	1168
Creating the CommandBase Class	1170
Adding the AddCarCommand Class	1171
RelayCommands.....	1172
Wrapping Up Commands	1175
Migrate Code and Data to a View Model	1175
Moving the MainWindow.xaml.cs Code	1175
Updating the MainWindow Code and Markup	1176
Updating the Control Markup.....	1176
Wrapping Up View Models.....	1177
Updating AutoLot.Dal for MVVM	1177
Summary	1177
■Part IX: ASP.NET Core	1179
■Chapter 29: Introducing ASP.NET Core	1181
A Quick Look Back	1181
Introducing the MVC Pattern.....	1181
ASP.NET Core and the MVC Pattern	1182
ASP.NET Core and .NET Core	1182
One Framework, Many Uses.....	1182
ASP.NET Core Features from MVC/Web API	1183
Convention over Configuration	1183
Controllers and Actions.....	1184
Model Binding.....	1186
Model Validation	1190
Routing	1191
Filters.....	1197
What's New in ASP.NET Core	1198
Built-in Dependency Injection.....	1198
Environmental Awareness	1199

Application Configuration	1200
Deploying ASP.NET Core Applications	1201
Lightweight and Modular HTTP Request Pipeline.....	1201
Create and Configure the Solution	1202
Using Visual Studio	1202
Using the Command Line.....	1205
Running ASP.NET Core Applications	1206
Configuring the Launch Settings	1207
Using Visual Studio	1208
Using the Command Line or Visual Studio Code Terminal Window	1208
Using Visual Studio Code (VS Code).....	1209
Debugging ASP.NET Core Applications.....	1209
Update the AutoLot.Api Ports.....	1210
Create and Configure the WebHost	1210
The Program.cs File.....	1211
The Startup.cs File.....	1211
Logging.....	1219
Summary.....	1234
■Chapter 30: RESTful Services with ASP.NET Core	1235
Introducing ASP.NET Core RESTful Services.....	1235
Controller Actions with RESTful Services.....	1236
Formatted JSON Response Results.....	1236
The ApiController Attribute	1239
Update the Swagger/OpenAPI Settings	1241
Update the Swagger Calls in the Startup Class.....	1241
Add the XML Documentation File	1243
Add XML Comments to SwaggerGen.....	1245
Additional Documentation Options for API Endpoints.....	1246
Building API Action Methods.....	1248
The Constructor	1249
The Get Methods.....	1250

The UpdateOne Method	1251
The AddOne Method	1252
The DeleteOne Method	1253
The CarsController.....	1254
The Remaining Controllers.....	1255
Exception Filters.....	1257
Create the CustomExceptionFilter	1258
Test the Exception Filter	1260
Add Cross-Origin Requests Support.....	1260
Create a CORS Policy.....	1260
Add the CORS Policy to the HTTP Pipeline Handling.....	1261
Summary.....	1261
■Chapter 31: MVC Applications with ASP.NET Core.....	1263
Introducing the “V” in ASP.NET Core.....	1263
ViewResults and Action Methods	1263
The Razor View Engine and Razor Syntax	1266
Views	1269
Layouts	1273
Partial Views.....	1274
Update the Layout and Partial.....	1274
Sending Data to Views.....	1276
Tag Helpers	1277
Enabling Tag Helpers	1280
The Form Tag Helper.....	1281
The Form Action Tag Helper.....	1282
The Anchor Tag Helper.....	1282
The Input Tag Helper.....	1283
The TextArea Tag Helper	1284
The Select Tag Helper.....	1284
The Validation Tag Helpers.....	1285
The Environment Tag Helper.....	1286

The Link Tag Helper 1286

The Script Tag Helper 1287

The Image Tag Helper 1288

Custom Tag Helpers..... 1288

 Set the Foundation 1288

 Create the Base Class..... 1289

 The Item Details Tag Helper 1290

 The Item Delete Tag Helper..... 1291

 The Item Edit Tag Helper..... 1292

 The Item Create Tag Helper 1292

 The Item List Tag Helper 1293

 Making Custom Tag Helpers Visible..... 1294

HTML Helpers 1294

 The DisplayFor HTML Helper 1294

 The DisplayForModel HTML Helper..... 1295

 The EditorFor and EditorForModel HTML Helpers..... 1295

Managing Client-Side Libraries 1295

 Install Library Manager As a .NET Core Global Tool 1295

 Add Client-Side Libraries to AutoLot.Mvc 1295

Finish the CarsController and Cars Views 1299

 The CarsController 1299

 The Car List Partial View..... 1300

 The Index View..... 1302

 The ByMake View 1303

 The Details View 1304

 The Create View..... 1305

 The Edit View 1307

 The Delete View 1310

View Components..... 1311

 The Server-Side Code..... 1312

 Build the Partial View..... 1313

Invoking View Components.....	1314
Invoking View Components As Custom Tag Helpers	1314
Updating the Menu	1314
Bundling and Minification	1315
Bundling	1315
Minification.....	1315
The WebOptimizer Solution	1315
The Options Pattern in ASP.NET Core	1317
Add the Dealer Information.....	1317
Create the Service Wrapper	1319
Update the Application Configuration	1319
Create the ServiceSettings Class	1320
The API Service Wrapper	1320
Configure the Services	1324
Build the API CarsController	1325
The GetMakes Helper Method	1326
The GetOne Car Method.....	1326
The Public Action Methods	1326
Update the View Component.....	1328
Run AutoLot.Mvc and AutoLot.Api Together.....	1329
Using Visual Studio.....	1329
Using the Command Line.....	1330
Summary.....	1330
Index.....	1331

Introduction

We're a Team That Includes You

Technology authors write for a demanding group of people (for the best of possible reasons). You know that building software solutions using any platform or language is extremely complicated and is specific to your department, company, client base, and subject matter. Perhaps you work in the electronic publishing industry, develop systems for the state or local government, or work at NASA or a branch of the military. Collectively, we have worked in a variety of industries, including developing children's educational software (Oregon Trail/Amazon Trail), various enterprise systems, and projects within the medical and financial industries. The chances are almost 100 percent that the code you write at your place of employment has little to do with the code we have authored over the years.

Therefore, in this book, we have deliberately chosen to avoid creating demonstrations that tie the example code to a specific industry or vein of programming. Given this, we explain C#, OOP, the .NET Runtime, and the .NET Core base class libraries using industry-agnostic examples. Rather than having every example fill a grid with data, calculate payroll, or some other domain-specific task, we stick to subject matter we can all relate to: automobiles (with some geometric structures and employee payroll systems thrown in for good measure). And that's where you come in.

Our job is to explain the C# programming language and the core aspects of the .NET 5 platform the best we possibly can. As well, we will do everything we can to equip you with the tools and strategies you need to continue your studies at this book's conclusion.

Your job is to take this information and apply it to your specific programming assignments. We obviously understand that your projects most likely don't revolve around automobiles with friendly pet names (Zippy the BMW or a Yugo named Clunker, among others), but that's what applied knowledge is all about!

Rest assured, once you understand the topics and concepts presented within this text, you will be in a perfect position to build .NET 5 solutions that map to your own unique programming environment.

An Overview of This Book

Pro C# 9 with .NET 5 is logically divided into eight distinct parts, each of which contains a number of related chapters. Here is a part-by-part and chapter-by-chapter breakdown of the text.

Part I: Introducing C# and .NET 5

The purpose of Part I is to acclimate you to the nature of the .NET 5 platform and various development tools used during the construction of .NET 5 applications.

Chapter 1: Introducing C# and .NET (Core) 5

This first chapter functions as the backbone for the remainder of the text. The primary goal of this chapter is to acquaint you with a number of .NET Core–centric building blocks, such as the Common Language Runtime (CLR), Common Type System (CTS), Common Language Specification (CLS), and Base Class Libraries (BCL). Here, you will take an initial look at the C# programming language, namespaces, and the .NET 5 assembly format.

Chapter 2: Building C# Applications

The goal of this chapter is to introduce you to the process of compiling C# source code files. After installing the .NET 5 SDK and runtimes, you will learn about the completely free (and fully functional) Visual Studio Community edition as well as the extremely popular (and also free) Visual Studio Code. You learn how to create, run, and debug .NET 5 C# applications using both Visual Studio and Visual Studio Code.

Part II: Core C# Programming

The topics presented in this part of the book are quite important because you will use them regardless of which type of .NET 5 software you intend to develop (e.g., web applications, desktop GUI applications, code libraries, services, etc.). Here, you will learn about the fundamental data types of .NET 5, work with text manipulation, and learn the role of various C# parameter modifiers (including optional and named arguments).

Chapter 3: Core C# Programming Constructs, Part 1

This chapter begins your formal investigation of the C# programming language. Here, you will learn about the role of the `Main()` method, top-level statements (new in C# 9.0), and numerous details regarding the intrinsic data types of the .NET 5 platform and variable declaration. You will work with and manipulate textual data using `System.String` and `System.Text.StringBuilder`. You will also examine iteration and decision constructs, pattern matching, narrowing and widening operations, and the `unchecked` keyword.

Chapter 4: Core C# Programming Constructs, Part 2

This chapter completes your examination of the core aspects of C#, beginning with creating and manipulating arrays of data. Next, you examine how to construct overloaded type methods and define parameters using the `out`, `ref`, and `params` keywords. You will also learn about the `enum` type, structures, and nullable data types, and you will understand the distinction between value types and reference types. Finally, you will learn about tuples, a new feature in C# 7 and updated in C# 8.

Part III: Object-Oriented Programming with C#

In this part, you will come to understand the core constructs of the C# language, including the details of object-oriented programming. This part will also examine how to process runtime exceptions and will dive into the details of working with strongly typed interfaces. Finally, you will learn about object lifetime and garbage collection.

Chapter 5: Understanding Encapsulation

This chapter begins your examination of object-oriented programming (OOP) using the C# programming language. After you are introduced to the pillars of OOP (encapsulation, inheritance, and polymorphism), the remainder of this chapter will show you how to build robust class types using constructors, properties, static members, constants, and read-only fields. You will also learn about partial type definitions, object initialization syntax, and automatic properties, and the chapter will wrap up with an examination of record types, new in C# 9.0.

Chapter 6: Understanding Inheritance and Polymorphism

Here, you will examine the remaining pillars of OOP (inheritance and polymorphism), which allow you to build families of related class types. As you do this, you will examine the role of virtual methods, abstract methods (and abstract base classes), and the nature of the polymorphic interface. Then you will explore pattern matching with the `is` keyword, and finally, this chapter will explain the role of the ultimate base class of the .NET Core platform, `System.Object`.

Chapter 7: Understanding Structured Exception Handling

The point of this chapter is to discuss how to handle runtime anomalies in your code base through the use of structured exception handling. Not only will you learn about the C# keywords that allow you to handle such problems (`try`, `catch`, `throw`, `when`, and `finally`), but you will also come to understand the distinction between application-level and system-level exceptions. In addition, this chapter will show you how to set Visual Studio on break on all exceptions to debug the exceptions that escape your notice.

Chapter 8: Working with Interfaces

The material in this chapter builds upon your understanding of object-based development by covering the topic of interface-based programming. Here, you will learn how to define classes and structures that support multiple behaviors, how to discover these behaviors at runtime, and how to selectively hide particular behaviors using explicit interface implementation. In addition to creating a number of custom interfaces, you will also learn how to implement standard interfaces found within the .NET Core platform. You will use these to build objects that can be sorted, copied, enumerated, and compared.

Chapter 9: Understanding Object Lifetime

The final chapter of this part examines how the CLR manages memory using the .NET Core garbage collector. Here, you will come to understand the role of application roots, object generations, and the `System.GC` type. Once you understand the basics, you will examine the topics of disposable objects (using the `IDisposable` interface) and the finalization process (using the `System.Object.Finalize()` method). This chapter will also investigate the `Lazy<T>` class, which allows you to define data that will not be allocated until requested by a caller. As you will see, this feature can be helpful when you want to ensure you do not clutter the heap with objects that are not actually required by your programs.

Part IV: Advanced C# Programming

This part of the book will deepen your understanding of the C# language by walking you through a number of more advanced (but important) concepts. Here, you will complete your examination of the .NET Core type system by investigating collections and generics. You will also examine a number of more advanced features of C# (e.g., extension methods, operator overloading, anonymous types, and pointer manipulation). You will then examine delegates and lambda expressions, take a first look at Language Integrated Query, and finish the section with two chapters that focus on processes and multithreaded/async programming.

Chapter 10: Collections and Generics

This chapter explores the topic of *generics*. As you will see, generic programming gives you a way to create types and type members, which contain various *placeholders* that can be specified by the caller. In a nutshell, generics greatly enhance application performance and type safety. Not only will you explore various generic types within the `System.Collections.Generic` namespace, but you will also learn how to build your own generic methods and types (with and without constraints).

Chapter 11: Advanced C# Language Features

This chapter deepens your understanding of the C# programming language by introducing you to a number of advanced programming techniques. Here, you will learn how to overload operators and create custom conversion routines (both implicit and explicit) for your types. You will also learn how to build and interact with type indexers, as well as work with extension methods, anonymous types, partial methods, and C# pointers using an unsafe code context.

Chapter 12: Delegates, Events, and Lambda Expressions

The purpose of this chapter is to demystify the delegate type. Simply put, a .NET Core delegate is an object that points to other methods in your application. Using this type, you can build systems that allow multiple objects to engage in a two-way conversation. After you have examined the use of .NET Core delegates, you will then be introduced to the C# event keyword, which you can use to simplify the manipulation of raw delegate programming. You will wrap up this chapter by investigating the role of the C# lambda operator (`=>`) and exploring the connection between delegates, anonymous methods, and lambda expressions.

Chapter 13: LINQ to Objects

This chapter begins your examination of Language Integrated Query (LINQ). LINQ allows you to build strongly typed query expressions that can be applied to a number of LINQ targets to manipulate data in the broadest sense of the word. Here, you will learn about LINQ to Objects, which allows you to apply LINQ expressions to containers of data (e.g., arrays, collections, and custom types). This information will serve you well as you encounter a number of additional LINQ APIs throughout the remainder of this book.

Chapter 14: Processes, AppDomains, and Load Contexts

Now that you have a solid understanding of assemblies, this chapter dives deeper into the composition of a loaded .NET Core executable. The goal of this chapter is to illustrate the relationship between processes, application domains, and contextual boundaries. These topics provide the proper foundation for Chapter 15, where you will examine the construction of multithreaded applications.

Chapter 15: Multithreaded, Parallel, and Async Programming

This chapter examines how to build multithreaded applications and illustrates a number of techniques you can use to author thread-safe code. The chapter opens by revisiting the .NET delegate type to ensure explaining a delegate's intrinsic support for asynchronous method invocations. Next, you will investigate the types within the System.Threading namespace. The next section covers the Task Parallel Library (TPL). Using the TPL, .NET developers can build applications that distribute their workload across all available CPUs in a wickedly simple manner. At this point, you will also learn about the role of Parallel LINQ, which provides a way to create LINQ queries that scale across multiple machine cores. The remainder of the chapter covers creating nonblocking calls using the `async/await` keywords, introduced in C# 5; local functions and generalized async return types, both new in C# 7; and asynchronous streams, introduced in C# 8.

Part V: Programming with .NET Core Assemblies

Part V dives into the details of the .NET Core assembly format. Not only will you learn how to deploy and configure .NET Core code libraries, but you will also come to understand the internal composition of a .NET Core binary image. This section explains the role of .NET Core attributes and the role of resolving type information at runtime and the role of the Dynamic Language Runtime (DLR) and the C# `dynamic` keyword. The final chapter covers the syntax of Common Intermediate Language (CIL) and the role of dynamic assemblies.

Chapter 16: Building and Configuring Class Libraries

At a high level, *assembly* is the term used to describe a binary file created with a .NET Core compiler. However, the true story of .NET Core assemblies is far richer than that. Here, you will learn how to build and deploy assemblies and learn the difference between class libraries and console applications and the difference between .NET Core and .NET Standard class libraries. The final section covers the new options available in .NET 5, such as single file executables and ready-to-run publishing.

Chapter 17: Type Reflection, Late Binding, and Attribute-Based Programming

This chapter continues your examination of .NET Core assemblies by checking out the process of runtime type discovery using the System.Reflection namespace. Using the types of this namespace, you can build applications that can read an assembly's metadata on the fly. You will also learn how to load and create types at runtime dynamically using late binding. The final topic of this chapter will explore the role of .NET Core attributes (both standard and custom). To illustrate the usefulness of each of these topics, the chapter shows you how to construct an extendable application complete with snap-ins.

Chapter 18: Dynamic Types and the Dynamic Language Runtime

.NET 4.0 introduced a new aspect of the .NET runtime environment called the *Dynamic Language Runtime*. Using the DLR and the C# `dynamic` keyword, you can define data that is not truly resolved until runtime. Using these features simplifies some complex .NET Core programming tasks dramatically. In this chapter, you will learn some practical uses of dynamic data, including how to leverage the .NET Core reflection APIs in a streamlined manner, as well as how to communicate with legacy COM libraries with a minimum of fuss and bother.

Chapter 19: Understanding CIL and the Role of Dynamic Assemblies

The goal of the final chapter in this section is twofold. The first part examines the syntax and semantics of CIL in much greater detail than in previous chapters. The remainder of this chapter will cover the role of the `System.Reflection.Emit` namespace. You can use these types to build software that can generate .NET Core assemblies in memory at runtime. Formally speaking, assemblies defined and executed in memory are termed *dynamic assemblies*.

Part VI: File Handling, Object Serialization, and Data Access

By this point in the text, you have a solid handle on the C# language and the details of the .NET Core assembly format. Part VI leverages your newfound knowledge by exploring a number of commonly used services found within the base class libraries, including file I/O, object serialization, and database access using ADO.NET.

Chapter 20: File I/O and Object Serialization

The `System.IO` namespace allows you to interact with a machine's file and directory structure. Over the course of this chapter, you will learn how to create (and destroy) a directory system programmatically. You will also learn how to move data into and out of various streams (e.g., file based, string based, and memory based). The latter part of this chapter will examine the XML and JSON object serialization services of the .NET Core platform. Simply put, serialization allows you to persist the public state of an object (or a set of related objects) into a stream for later use. Deserialization (as you might expect) is the process of plucking an object from the stream into memory for consumption by your application.

Chapter 21: Data Access with ADO.NET

This chapter covers database access using ADO.NET, the database API for .NET Core applications. Specifically, this chapter will introduce you to the role of .NET data providers and how to communicate with a relational database using ADO.NET, which is represented by connection objects, command objects, transaction objects, and data reader objects. This chapter also begins the creation of the AutoLot database, which will be enhanced in Chapters 22 and 23.

Part VII: Entity Framework Core

By this point in the text, you have a solid handle on the C# language and the details of the .NET Core assembly format. Part VI leverages your newfound knowledge by exploring a number of commonly used services found within the base class libraries, including file I/O, database access using ADO.NET, and database access using Entity Framework Core.

Chapter 22: Introducing Entity Framework Core

This chapter introduces Entity Framework (EF) Core. EF Core is an object-relational mapping (ORM) framework built on top of ADO.NET. EF Core provides a way to author data access code using strongly typed classes that directly map to your business model. Here, you will come to understand the building blocks of EF Core, including `DbContext`, entities, the specialized collection class `DbSet<T>`, and the `DbChangeTracker`. Next, you will learn about query execution, tracking versus nontracking queries, and some of the more notable features of EF Core. The final piece of theory in this chapter is the EF Core global tool for the .NET Core command-line interface (CLI).

Chapter 23: Build a Data Access Layer with Entity Framework Core

This chapter builds the AutoLot data access layer. It begins with scaffolding the AutoLot database from Chapter 21 into a derived `DbContext` and entity classes. Then the project and database are updated to move to a code first approach. The entities are updated to their final version, and a migration is created and executed to update the database to match the entities. The final database change is to create a migration for the stored procedure from Chapter 21 and a new database view. Repositories are added for code encapsulation, and a data initialization process is created. The final part of the chapter test drives the data access layer using `xUnit` for automated integration tests.

Part VIII: Windows Presentation Foundation

The initial desktop GUI API supported by the .NET platform was termed Windows Forms. While this API is still fully supported, .NET 3.0 introduced programmers to an API called Windows Presentation Foundation (WPF). Unlike Windows Forms, this framework integrates a number of key services, including data binding, 2D and 3D graphics, animations, and rich documents, into a single, unified object model. This is all accomplished using a declarative markup grammar called Extensible Application Markup Language (XAML). Furthermore, the WPF control architecture provides a trivial way to restyle the look and feel of a typical control radically using little more than some well-formed XAML.

Chapter 24: Introducing Windows Presentation Foundation and XAML

In this chapter, you will begin by examining the motivation behind the creation of WPF (when there was already a desktop development framework in .NET). Then, you will learn about the syntax of XAML and, finally, take a look at the Visual Studio support for building WPF applications.

Chapter 25: WPF Controls, Layouts, Events, and Data Binding

This chapter will expose you to the process of using intrinsic WPF controls and layout managers. For example, you will learn to build menu systems, splitter windows, toolbars, and status bars. This chapter will also introduce you to a number of WPF APIs (and their related controls), including the Ink API, commands, routed events, the data-binding model, and dependency properties.

Chapter 26: WPF Graphics Rendering Services

WPF is a graphically intensive API; given this fact, WPF provides three ways to render graphics: shapes, drawings and geometrics, and visuals. In this chapter, you will evaluate each option and learn about a number of important graphics primitives (e.g., brushes, pens, and transformations) along the way. This chapter will also examine ways to incorporate vector images into your WPF graphics, as well as how to perform hit-testing operations against graphical data.

Chapter 27: WPF Resources, Animations, Styles, and Templates

This chapter will introduce you to three important (and interrelated) topics that will deepen your understanding of the Windows Presentation Foundation API. The first order of business is to learn the role of logical resources. As you will see, the logical resource (also termed an *object resource*) system provides a way for you to name and refer to commonly used objects within a WPF application. Next, you will learn how to define, execute, and control an animation sequence. Despite what you might be thinking, however, WPF

animations are not limited to the confines of video games or multimedia applications. You will wrap up the chapter by learning about the role of WPF styles. Similar to a web page that uses CSS or the ASP.NET theme engine, a WPF application can define a common look and feel for a set of controls.

Chapter 28: WPF Notifications, Validations, Commands, and MVVM

This chapter begins by examining three core WPF framework capabilities: notifications, validations, and commands. In the notifications section, you will learn about observable models and collections and how they keep your application data and UI in sync. Next, you will dig deeper into commands, building custom commands to encapsulate your code. In the validations section, you will learn how to use the several validation mechanisms available in WPF applications. The chapter closes with an examination of the Model-View-ViewModel (MVVM) pattern and ends by creating an application that demonstrates the MVVM pattern in action.

Part IX: ASP.NET Core

Part VIII is devoted to an examination of constructing web applications using ASP.NET Core. ASP.NET Core can be used to build web applications and RESTful services.

Chapter 29: Introducing ASP.NET Core

This chapter introduces ASP.NET Core and the Model-View-Controller (MVC) pattern. The next section lists (and explains) features brought forward into ASP.NET Core from classic ASP.NET MVC/WebAPI. These include controllers and actions, model binding, routing, and filters. The next section covers new features introduced in ASP.NET Core, including built-in dependency injection, cloud-ready, environmentally aware configuration system, deployment patterns, and the HTTP request pipeline. The rest of the chapter creates the two ASP.NET Core projects that will be finished in the next two chapters, shows the options to run ASP.NET Core applications, and begins the configuration process of the two ASP.NET Core projects.

Chapter 30: RESTful Services with ASP.NET Core

This chapter finishes the ASP.NET Core RESTful service application. The chapter begins with a demonstration of the different mechanisms for returning JSON results to a client and the built-in support for service applications provided by the `ApiController` attribute. Swagger/OpenAPI is added next to provide a platform for testing and documenting your service. The rest of the chapter builds the controllers for the application, as well as creating the exception filter.

Chapter 31: MVC Applications with ASP.NET Core

This is the final chapter covering ASP.NET Core and finishes the MVC-based web application. The chapter starts with a deep look in views and the Razor View Engine, including layouts and partials. Next, tag helpers (another new feature in ASP.NET Core) are explored, followed by managing client-side libraries and bundling/minification of those libraries. Next, the `CarsController` and its views are built, along with the custom tag helpers. A view component is added for the data-driven menu, and the options pattern is explored. Finally, an HTTP client service wrapper is created, and `CarsController` updated to use the ASP.NET Core service instead of the `AutoLot` data access layer.