

C# in Depth

THIRD EDITION

JON SKEET



MANNING
SHELTER ISLAND

brief contents

PART 1	PREPARING FOR THE JOURNEY.....	1
1	■ The changing face of C# development	3
2	■ Core foundations: building on C# 1	29
PART 2	C# 2: SOLVING THE ISSUES OF C# 1	57
3	■ Parameterized typing with generics	59
4	■ Saying nothing with nullable types	105
5	■ Fast-tracked delegates	133
6	■ Implementing iterators the easy way	159
7	■ Concluding C# 2: the final features	182
PART 3	C# 3: REVOLUTIONIZING DATA ACCESS	205
8	■ Cutting fluff with a smart compiler	207
9	■ Lambda expressions and expression trees	232
10	■ Extension methods	262
11	■ Query expressions and LINQ to Objects	285
12	■ LINQ beyond collections	328

PART 4	C# 4: PLAYING NICELY WITH OTHERS	369
13	■ Minor changes to simplify code	371
14	■ Dynamic binding in a static language	409
PART 5	C# 5: ASYNCHRONY MADE SIMPLE	461
15	■ Asynchrony with async/await	463
16	■ C# 5 bonus features and closing thoughts	519

contents

foreword xix
preface xxi
acknowledgments xxii
about this book xxiv
about the author xxix
about the cover illustration xxx

PART 1 PREPARING FOR THE JOURNEY.....1

1 The changing face of C# development 3

- 1.1 Starting with a simple data type 4
 - The Product type in C# 1 5* ■ *Strongly typed collections in C# 2 6*
 - Automatically implemented properties in C# 3 7* ■ *Named arguments in C# 4 8*
- 1.2 Sorting and filtering 9
 - Sorting products by name 9* ■ *Querying collections 12*
- 1.3 Handling an absence of data 14
 - Representing an unknown price 14* ■ *Optional parameters and default values 16*
- 1.4 Introducing LINQ 16
 - Query expressions and in-process queries 17* ■ *Querying XML 18* ■ *LINQ to SQL 19*

- 1.5 COM and dynamic typing 20
 - Simplifying COM interoperability* 20
 - *Interoperating with a dynamic language* 21
- 1.6 Writing asynchronous code without the heartache 22
- 1.7 Dissecting the .NET platform 23
 - C#, the language* 24
 - *Runtime* 24
 - *Framework libraries* 24
- 1.8 Making your code super awesome 25
 - Presenting full programs as snippets* 25
 - *Didactic code isn't production code* 26
 - *Your new best friend: the language specification* 27
- 1.9 Summary 28

2 Core foundations: building on C# 1 29

- 2.1 Delegates 30
 - A recipe for simple delegates* 30
 - *Combining and removing delegates* 35
 - *A brief diversion into events* 36
 - *Summary of delegates* 37
- 2.2 Type system characteristics 38
 - C#'s place in the world of type systems* 38
 - *When is C# 1's type system not rich enough?* 41
 - *Summary of type system characteristics* 44
- 2.3 Value types and reference types 44
 - Values and references in the real world* 45
 - *Value and reference type fundamentals* 46
 - *Dispelling myths* 47
 - *Boxing and unboxing* 49
 - *Summary of value types and reference types* 50
- 2.4 Beyond C# 1: new features on a solid base 51
 - Features related to delegates* 51
 - *Features related to the type system* 53
 - *Features related to value types* 55
- 2.5 Summary 56

PART 2 C# 2: SOLVING THE ISSUES OF C# 157

3 Parameterized typing with generics 59

- 3.1 Why generics are necessary 60
- 3.2 Simple generics for everyday use 62
 - Learning by example: a generic dictionary* 62
 - *Generic types and type parameters* 64
 - *Generic methods and reading generic declarations* 67

- 3.3 Beyond the basics 70
 - Type constraints* 71 ■ *Type inference for type arguments of generic methods* 76 ■ *Implementing generics* 77
- 3.4 Advanced generics 83
 - Static fields and static constructors* 84 ■ *How the JIT compiler handles generics* 85 ■ *Generic iteration* 87 ■ *Reflection and generics* 90
- 3.5 Limitations of generics in C# and other languages 94
 - Lack of generic variance* 94 ■ *Lack of operator constraints or a “numeric” constraint* 99 ■ *Lack of generic properties, indexers, and other member types* 101 ■ *Comparison with C++ templates* 101 ■ *Comparison with Java generics* 103
- 3.6 Summary 104

4 *Saying nothing with nullable types* 105

- 4.1 What do you do when you just don't have a value? 106
 - Why value type variables can't be null* 106
 - Patterns for representing null values in C# 1* 107
- 4.2 System.Nullable<T> and System.Nullable 109
 - Introducing Nullable<T>* 109 ■ *Boxing Nullable<T> and unboxing* 112 ■ *Equality of Nullable<T> instances* 113
 - Support from the nongeneric Nullable class* 114
- 4.3 C# 2's syntactic sugar for nullable types 114
 - The ? modifier* 115 ■ *Assigning and comparing with null* 116
 - Nullable conversions and operators* 118 ■ *Nullable logic* 121
 - Using the as operator with nullable types* 123 ■ *The null coalescing operator* 123
- 4.4 Novel uses of nullable types 126
 - Trying an operation without using output parameters* 127
 - Painless comparisons with the null coalescing operator* 129
- 4.5 Summary 131

5 *Fast-tracked delegates* 133

- 5.1 Saying goodbye to awkward delegate syntax 134
- 5.2 Method group conversions 136
- 5.3 Covariance and contravariance 137
 - Contravariance for delegate parameters* 138 ■ *Covariance of delegate return types* 139 ■ *A small risk of incompatibility* 141

- 5.4 Inline delegate actions with anonymous methods 142
 - Starting simply: acting on a parameter* 142
 - *Returning values from anonymous methods* 145
 - *Ignoring delegate parameters* 146
- 5.5 Capturing variables in anonymous methods 148
 - Defining closures and different types of variables* 148
 - Examining the behavior of captured variables* 149
 - *What's the point of captured variables?* 151
 - *The extended lifetime of captured variables* 152
 - *Local variable instantiations* 153
 - Mixtures of shared and distinct variables* 155
 - *Captured variable guidelines and summary* 156
- 5.6 Summary 158

6 *Implementing iterators the easy way* 159

- 6.1 C# 1: The pain of handwritten iterators 160
- 6.2 C# 2: Simple iterators with yield statements 163
 - Introducing iterator blocks and yield return* 163
 - *Visualizing an iterator's workflow* 165
 - *Advanced iterator execution flow* 167
 - *Quirks in the implementation* 170
- 6.3 Real-life iterator examples 172
 - Iterating over the dates in a timetable* 172
 - *Iterating over lines in a file* 173
 - *Filtering items lazily using an iterator block and a predicate* 176
- 6.4 Pseudo-synchronous code with the Concurrency and Coordination Runtime 178
- 6.5 Summary 180

7 *Concluding C# 2: the final features* 182

- 7.1 Partial types 183
 - Creating a type with multiple files* 184
 - *Uses of partial types* 186
 - *Partial methods—C# 3 only!* 188
- 7.2 Static classes 190
- 7.3 Separate getter/setter property access 192
- 7.4 Namespace aliases 193
 - Qualifying namespace aliases* 194
 - *The global namespace alias* 195
 - *Extern aliases* 196

- 7.5 Pragma directives 197
 - Warning pragmas* 197 ■ *Checksum pragmas* 198
- 7.6 Fixed-size buffers in unsafe code 199
- 7.7 Exposing internal members to selected assemblies 201
 - Friend assemblies in the simple case* 201 ■ *Why use InternalsVisibleTo?* 202 ■ *InternalsVisibleTo and signed assemblies* 203
- 7.8 Summary 204

PART 3 C# 3: REVOLUTIONIZING DATA ACCESS.....205

8 *Cutting fluff with a smart compiler* 207

- 8.1 Automatically implemented properties 208
- 8.2 Implicit typing of local variables 211
 - Using var to declare a local variable* 211 ■ *Restrictions on implicit typing* 213 ■ *Pros and cons of implicit typing* 214
 - Recommendations* 215
- 8.3 Simplified initialization 216
 - Defining some sample types* 216 ■ *Setting simple properties* 217
 - Setting properties on embedded objects* 219 ■ *Collection initializers* 220 ■ *Uses of initialization features* 223
- 8.4 Implicitly typed arrays 224
- 8.5 Anonymous types 225
 - First encounters of the anonymous kind* 225 ■ *Members of anonymous types* 227 ■ *Projection initializers* 228 ■ *What's the point?* 229
- 8.6 Summary 231

9 *Lambda expressions and expression trees* 232

- 9.1 Lambda expressions as delegates 234
 - Preliminaries: Introducing the Func<...> delegate types* 234
 - First transformation to a lambda expression* 235 ■ *Using a single expression as the body* 236 ■ *Implicitly typed parameter lists* 236
 - Shortcut for a single parameter* 237
- 9.2 Simple examples using List<T> and events 238
 - Filtering, sorting, and actions on lists* 238 ■ *Logging in an event handler* 240

- 9.3 Expression trees 241
 - Building expression trees programmatically* 242
 - *Compiling expression trees into delegates* 243
 - *Converting C# lambda expressions to expression trees* 244
 - *Expression trees at the heart of LINQ* 248
 - *Expression trees beyond LINQ* 249
- 9.4 Changes to type inference and overload resolution 251
 - Reasons for change: streamlining generic method calls* 252
 - Inferred return types of anonymous functions* 253
 - *Two-phase type inference* 254
 - *Picking the right overloaded method* 258
 - Wrapping up type inference and overload resolution* 260
- 9.5 Summary 260

10 Extension methods 262

- 10.1 Life before extension methods 263
- 10.2 Extension method syntax 265
 - Declaring extension methods* 265
 - *Calling extension methods* 267
 - *Extension method discovery* 268
 - *Calling a method on a null reference* 269
- 10.3 Extension methods in .NET 3.5 271
 - First steps with Enumerable* 271
 - *Filtering with Where and chaining method calls together* 273
 - *Interlude: haven't we seen the Where method before?* 275
 - *Projections using the Select method and anonymous types* 276
 - *Sorting using the OrderBy method* 277
 - *Business examples involving chaining* 278
- 10.4 Usage ideas and guidelines 280
 - "Extending the world" and making interfaces richer* 280
 - *Fluent interfaces* 280
 - *Using extension methods sensibly* 282
- 10.5 Summary 284

11 Query expressions and LINQ to Objects 285

- 11.1 Introducing LINQ 286
 - Fundamental concepts in LINQ* 286
 - *Defining the sample data model* 291
- 11.2 Simple beginnings: selecting elements 292
 - Starting with a source and ending with a selection* 293
 - *Compiler translations as the basis of query expressions* 293
 - *Range variables and nontrivial projections* 296
 - *Cast, OfType, and explicitly typed range variables* 298

- 11.3 Filtering and ordering a sequence 300
 - Filtering using a where clause* 300
 - *Degenerate query expressions* 301
 - *Ordering using an orderby clause* 302
- 11.4 Let clauses and transparent identifiers 304
 - Introducing an intermediate computation with let* 305
 - Transparent identifiers* 306
- 11.5 Joins 307
 - Inner joins using join clauses* 307
 - *Group joins with join...into clauses* 311
 - *Cross joins and flattening sequences using multiple from clauses* 314
- 11.6 Groupings and continuations 318
 - Grouping with the group...by clause* 318
 - *Query continuations* 321
- 11.7 Choosing between query expressions and dot notation 324
 - Operations that require dot notation* 324
 - *Query expressions where dot notation may be simpler* 325
 - *Where query expressions shine* 325
- 11.8 Summary 326

12 LINQ beyond collections 328

- 12.1 Querying a database with LINQ to SQL 329
 - Getting started: the database and model* 330
 - *Initial queries* 332
 - *Queries involving joins* 334
- 12.2 Translations using IQueryable and IQueryProvider 336
 - Introducing IQueryable<T> and related interfaces* 337
 - *Faking it: interface implementations to log calls* 338
 - *Gluing expressions together: the Queryable extension methods* 341
 - *The fake query provider in action* 342
 - *Wrapping up IQueryable* 344
- 12.3 LINQ-friendly APIs and LINQ to XML 344
 - Core types in LINQ to XML* 345
 - *Declarative construction* 347
 - Queries on single nodes* 349
 - *Flattened query operators* 351
 - Working in harmony with LINQ* 352
- 12.4 Replacing LINQ to Objects with Parallel LINQ 353
 - Plotting the Mandelbrot set with a single thread* 353
 - *Introducing ParallelEnumerable, ParallelQuery, and AsParallel* 354
 - Twinking parallel queries* 356

- 12.5 Inverting the query model with LINQ to Rx 357
 - IObservable<T> and IObservable<T>* 358 ■ *Starting simply (again)* 360 ■ *Querying observables* 360 ■ *What's the point?* 363
- 12.6 Extending LINQ to Objects 364
 - Design and implementation guidelines* 364 ■ *Sample extension: selecting a random element* 365
- 12.7 Summary 367

PART 4 C# 4: PLAYING NICELY WITH OTHERS 369

13 *Minor changes to simplify code* 371

- 13.1 Optional parameters and named arguments 372
 - Optional parameters* 372 ■ *Named arguments* 378 ■ *Putting the two together* 382
- 13.2 Improvements for COM interoperability 387
 - The horrors of automating Word before C# 4* 387 ■ *The revenge of optional parameters and named arguments* 388 ■ *When is a ref parameter not a ref parameter?* 389 ■ *Calling named indexers* 390 ■ *Linking primary interop assemblies* 391
- 13.3 Generic variance for interfaces and delegates 394
 - Types of variance: covariance and contravariance* 394 ■ *Using variance in interfaces* 396 ■ *Using variance in delegates* 399
 - Complex situations* 399 ■ *Restrictions and notes* 401
- 13.4 Teeny tiny changes to locking and field-like events 405
 - Robust locking* 405 ■ *Changes to field-like events* 406
- 13.5 Summary 407

14 *Dynamic binding in a static language* 409

- 14.1 What? When? Why? How? 411
 - What is dynamic typing?* 411 ■ *When is dynamic typing useful, and why?* 412 ■ *How does C# 4 provide dynamic typing?* 413
- 14.2 The five-minute guide to dynamic 414
- 14.3 Examples of dynamic typing 416
 - COM in general, and Microsoft Office in particular* 417 ■ *Dynamic languages such as IronPython* 419
 - Dynamic typing in purely managed code* 423

- 14.4 Looking behind the scenes 429
 - Introducing the Dynamic Language Runtime* 429
 - *DLR core concepts* 431
 - *How the C# compiler handles dynamic* 434
 - The C# compiler gets even smarter* 438
 - *Restrictions on dynamic code* 441
- 14.5 Implementing dynamic behavior 444
 - Using ExpandableObject* 444
 - *Using DynamicObject* 448
 - Implementing IDynamicMetaObjectProvider* 455
- 14.6 Summary 459

PART 5 C# 5: ASYNCHRONY MADE SIMPLE461

15 *Asynchrony with async/await* 463

- 15.1 Introducing asynchronous functions 465
 - First encounters of the asynchronous kind* 465
 - *Breaking down the first example* 467
- 15.2 Thinking about asynchrony 468
 - Fundamentals of asynchronous execution* 468
 - *Modeling asynchronous methods* 471
- 15.3 Syntax and semantics 472
 - Declaring an async method* 472
 - *Return types from async methods* 473
 - *The awaitable pattern* 474
 - *The flow of await expressions* 477
 - *Returning from an async method* 481
 - Exceptions* 482
- 15.4 Asynchronous anonymous functions 490
- 15.5 Implementation details: compiler transformation 492
 - Overview of the generated code* 493
 - *Structure of the skeleton method* 495
 - *Structure of the state machine* 497
 - *One entry point to rule them all* 498
 - *Control around await expressions* 500
 - *Keeping track of a stack* 501
 - *Finding out more* 503
- 15.6 Using async/await effectively 503
 - The task-based asynchronous pattern* 504
 - *Composing async operations* 507
 - *Unit testing asynchronous code* 511
 - The awaitable pattern redux* 515
 - *Asynchronous operations in WinRT* 516
- 15.7 Summary 517

16	<i>C# 5 bonus features and closing thoughts</i>	519
16.1	Changes to captured variables in foreach loops	520
16.2	Caller information attributes	520
	<i>Basic behavior</i>	521
	<i>Logging</i>	522
	<i>Implementing</i>	
	<i>INotifyPropertyChanged</i>	523
	<i>Using caller information attributes</i>	
	<i>without .NET 4.5</i>	524
16.3	Closing thoughts	525
<i>appendix A</i>	<i>LINQ standard query operators</i>	527
<i>appendix B</i>	<i>Generic collections in .NET</i>	540
<i>appendix C</i>	<i>Version summaries</i>	554
	<i>index</i>	563