PART 4  C# 4: PLAYING NICELY WITH OTHERS. .......................... 363

  13 • Minor changes to simplify code 365
  14 • Dynamic binding in a static language 401
  15 • Letting your code speak more clearly with Code Contracts 452
  16 • Whither now? 490
PART 1  PREPARING FOR THE JOURNEY.

1  The changing face of C# development 3

1.1 Starting with a simple data type  4
The Product type in C#  4  •  Strongly typed collections in C#  2  6  •  Automatically implemented properties in C#  3  7
Named arguments in C#  4  7

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  15

1.4 Introducing LINQ  16
Query expressions and in-process queries  16  •  Querying XML  17-LINQtoSQL  18
1.5 COM and dynamic typing 19

Simplifying COM interoperability 19 * Interoperating with a dynamic language 20

1.6 Dissecting the .NET platform 21

C#, the language 22 * Runtime 22 * Framework libraries 22

1.7 Making your code super awesome 23

Presenting full programs as snippets 23 * Didactic code isn't production code 24 * Your new best friend: the language specification 25

1.8 Summary 26

Core foundations: building on C# 1 27

2.1 Delegates 28

A recipe for simple delegates 28 * Combining and removing delegates 33 * A brief diversion into events 34 • Summary of delegates 35

2.2 Type system characteristics 36

C#'s place in the world of type systems 36 * When is C# 1 's type system not rich enough? 39 * Summary of type system characteristics 42

2.3 Value types and reference types 42

Values and references in the real world 43 * Value and reference type fundamentals 43 * Dispelling myths 45 * Boxing and unboxing 47 * Summary of value types and reference types 48

2.4 Beyond C# 1: new features on a solid base 48

Features related to delegates 49 * Features related to the type system 51 * Features related to value types 53

2.5 Summary 54

PART 2 C# 2: SOLVING THE ISSUES OF C# 155

3 Parameterized typing with genetics 57

3.1 Why generics are necessary 58

3.2 Simple generics for everyday use 60

Learning by example: a generic dictionary 60 • Generic types and type parameters 62 • Generic methods and reading generic declarations 65
3.3 Beyond the basics 68
   Type constraints 69 * Type inference for type arguments of generic methods 74 * Implementing generics 75

3.4 Advanced generics 81
   Static fields and static constructors 81 * How the JIT compiler handles generics 83 * Generic iteration 85 * Reflection and generics 88

3.5 Limitations of generics in C# and other languages 91 -
   Lack of generic variance 92 * Lack of operator constraints or a "numeric" constraint 97* Lack of generic properties, indexers, and other member types 98 * Comparison with C++ templates 99
   Comparison with Java generics 100

3.6 Summary 101

4 Saying nothing with nullable types 103
4.1 What do you do when you just don't have a value? 104
   Why value type variables can't be null 104 * Patterns for representing null values in C# 1 105

   Introducing Nullable<T> 107 * Boxing Nullable<T> and unboxing 110* Equality of Nullable<T> instances 111
   Support from the nongeneric Nullable class 111

4.3 C# 2’s syntactic sugar for nullable types 112
   The ? modifier 113 * Assigning and comparing with null 114
   Nullable conversions and operators 116* Nullable logic 119
   Using the as operator with nullable types 120* The null coalescing operator 121

4.4 Novel uses of nullable types 124
   Trying an operation without using output parameters 124
   Painless comparisons with the null coalescing operator 126

4.5 Summary 129

5 Fast-tracked delegates 130
5.1 Saying goodbye to awkward delegate syntax 131
5.2 Method group conversions 133
5.3 Covariance and contravariance 134
   Contravariance for delegate parameters 135* Covariance of delegate return types 136 * A small risk of incompatibility 138
5.4 Inline delegate actions with anonymous methods 138
Starting simply: acting on a parameter 139 * Returning values from anonymous methods 141 * Ignoring delegate parameters 143

5.5 Capturing variables in anonymous methods 144
Defining closures and different types of variables 145
Examining the behavior of captured variables 146 * What's the point of captured variables ? 147 * The extended lifetime of captured variables 148 * Local variable instantiations 149
Mixtures of shared and distinct variables 151 • Captured variable guidelines and summary 153

5.6 Summary 154

Implementing iterators the easy way 156
6.1 C# 1: the pain of handwritten iterators 157
6.2 C# 2: simple iterators with yield statements 160
Introducing iterator blocks and yield return 160 * Visualizing an iterator's workflow 162 * Advanced iterator execution flow 164
Quirks in the implementation 167

6.3 Real-life iterator examples 169
Iterating over the dates in a timetable 169 * Iterating over lines in a file 170 * Filtering items lazily using an iterator block and a predicate 173

6.4 Pseudo-synchronous code with the Concurrency and Coordination Runtime 175

6.5 Summary 177

w Concluding C# 2: the final features 179

* 7.1 Partial types 180
Creating a type with multiple files 181 * Uses of partial types 183 * Partial methods—C# 3 only! 184

7.2 Static classes 186
7.3 Separate getter/setter property access 189
7.4 Namespace aliases 190
Qualifying namespace aliases 191 * The global namespace alias 192 * Extern aliases 192

7.5 Pragma directives 194
Warning pragmas 194 * Checksum pragmas 195

7.6 Fixed-size buffers in unsafe code 196
PART 3 C# 3: REVOLUTIONIZING HOW WE CODE

Cutting fluff with a smart compiler

8.1 Automatically implemented properties
8.2 Implicit typing of local variables
8.3 Simplified initialization
8.4 Implicitly typed arrays
8.5 Anonymous types
8.6 Summary

Lambda expressions and expression trees

9.1 Lambda expressions as delegates
9.2 Simple examples using List<T> and events
9.3 Expression trees
10 Extension methods 256

10.1 Life before extension methods 257

10.2 Extension method syntax 259

10.3 Extension methods in .NET 3.5 265

10.4 Usage ideas and guidelines 273

11 Query expressions and LINQ to Objects 279

11.1 Introducing LINQ 280

11.2 Simple beginnings: selecting elements 285

11.3 Filtering and ordering a sequence 294

11.4 Let clauses and transparent identifiers 298
11.5 Joins 301

Inner joins using join clauses 301 * Group joins with join ... into clauses 305 * Cross joins and flattening sequences using multiple from clauses 308

11.6 Groupings and continuations 311

Grouping with the group ... by clause 311 * Query continuations 314

11.7 Choosing between query expressions and dot notation 317

Operations that require dot notation 317 • Query expressions where dot notation may be simpler 318 * Where query expressions shine 319

11.8 Summary 320

12 LINQ beyond collections 321

12.1 Querying a database with LINQ to SQL 322

Getting started: the database and model 323 * Initial queries 325 * Queries involving joins 327

12.2 Translations using IQueryable and IQueryProvider 329

Introducing IQueryable<T> and related interfaces 330 * Faking it: interface implementations to log calls 331 • Gluing expressions together: the Queryable extension methods 334 * The fake query provider in action 335 * Wrapping up IQueryable 337

12.3 LINQ-friendly APIs and LINQ to XML 337

Core types in LINQ to XML 338 • Declarative construction 340 * Queries on single nodes 342 * Flattened query operators 344 * Working in harmony with LINQ 345

12.4 Replacing LINQ to Objects with Parallel LINQ 346

Plotting the Mandelbrot set with a single thread 346 * Introducing ParallelEnumerable, ParallelQuery, and AsParallel 347 * Tweaking parallel queries 349

12.5 Inverting the query model with LINQ to Rx 350

IObservable<T> and IObserver<T> 351 * Starting simply (again) 353 * Querying observables 354 * What’s the point? 356

12.6 Extending LINQ to Objects 357

Design and implementation guidelines 357 * Sample extension: selecting a random element 359

12.7 Summary 360
PART 4  C# 4: PLAYING NICELY WITH OTHERS  363

13 Minor changes to simplify code  365
  13.1 Optional parameters and named arguments  366
      * Optional parameters  366
      * Named arguments  372
    Putting the two together  376
  13.2 Improvements for COM interoperability  380
      * The horrors of automating Word before C# 4  380
      * The revenge of optional parameters and named arguments  381
      * When is a ref parameter not a ref parameter?  382
      * Calling named indexers  383
      * Linking primary interop assemblies  385
  13.3 Generic variance for interfaces and delegates  387
      * Types of variance: covariance and contravariance  387
      * Using variance in interfaces  389
      * Using variance in delegates  392
      * Complex situations  392
      * Restrictions and notes  394
  13.4 Teeny tiny changes to locking and field-like events  398
      * Robust locking  398
      * Changes to field-like events  399
  13.5 Summary  400

14 Dynamic binding in a static language  401
      * What is dynamic typing?  403
      * When is dynamic typing useful, and why?  404
      * How does C# 4 provide dynamic typing?  405
  14.2 The five-minute guide to dynamic  406
  14.3 Examples of dynamic typing  408
      * COM in general, and Microsoft Office in particular  408
      * Dynamic languages such as IronPython  410
      * Dynamic typing in purely managed code  415
  14.4 Looking behind the scenes  421
      * Introducing the Dynamic Language Runtime  421
      * DLR core concepts  422
      * How the C# compiler handles dynamic  426
      * The C# compiler gets even smarter  430
      * Restrictions on dynamic code  432
  14.5 Implementing dynamic behavior  435
      * Using ExpandoObject  435
      * Using DynamicObject  440
      * Implementing IDynamicMetaObjectProvider  446
  14.6 Summary  450
Letting your code speak more clearly with Code Contracts 452

15.1 Life before Code Contracts 454
15.2 Introducing Code Contracts 455
   * Preconditions 456
   * Postconditions 458
   * Invariants 459
   * Assertions and assumptions 461
   * Legacy contracts 462
15.3 Rewriting binaries with ccrewrite and ccrefgen 464
   * Simple rewriting 464
   * Contract inheritance 466
   * Contract reference assemblies 469
   * Failure behavior 470
15.4 Static checking 472
   * Getting started with static checking 473
   * Implicit obligations 475
   * Selective checking 478
15.5 Documenting contracts with ccdocgen 480
15.6 Practical contracts 482
   * Philosophy: what's in a contract? 483
   * How do I get started? 484
   * Options, options everywhere 485
15.7 Summary 488

Whither now? 490

16.1 C#—mixing tradition and modernity 490
16.2 Computerscience and.NET 491
16.3 The world of computing 492
16.4 Farewell 493

appendix A LINQ standard query operators 495
appendix B Generic collections in .NET 508
appendix C Version summaries 521
index 529