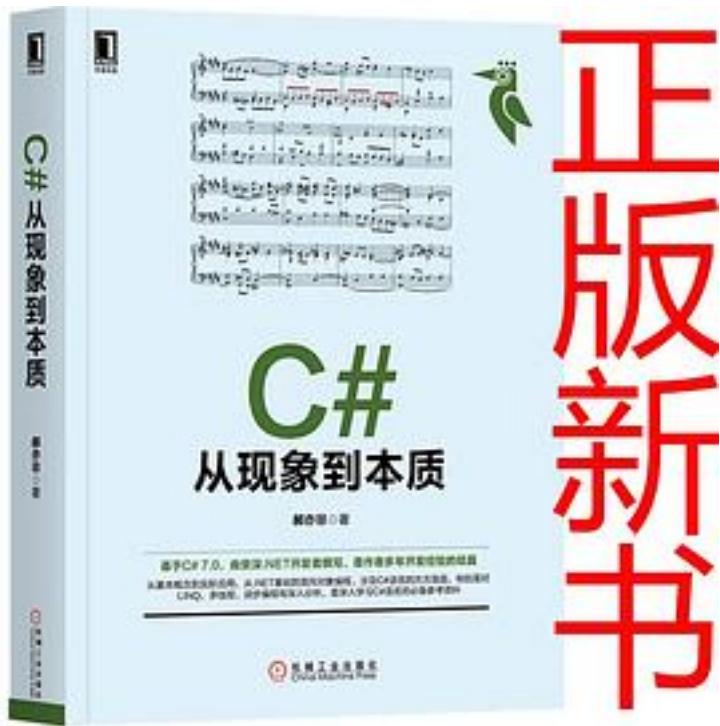


C#从现象到本质



正版新书

[C#从现象到本质 下载链接1](#)

著者:郝亦非

出版者:机械工业出版社

出版时间:2018-8-1

装帧:平装

isbn:9787111604402

本书详细介绍用C#语言进行程序开发需要掌握的知识和技术。全书由浅入深分三大部分,共21章,第一部分“基础知识”,包括.NET基础知识、C#类型基础、C#的面向对象技术、字符串、垃圾回收、异常处理;第二部分“C#特性”,包括委托和事件、泛型、反射、数据结构、LINQ的相关技术、动态语言运行时;第三部分“多线程和异步”,包括多线程的概念、多线程同步、异步编程理论与实例、任务并行库等。本书精选大量案例,循序渐进地讲解C#语言,内容丰富而翔实,并给出练习题,帮助读者更好地巩固所学知识,提升能力。前言和附录分别给出.NET程序员开发职位要求、技能等级、进阶之路,以及面试宝典,可帮助开发者新人快速进阶,找到适合自己的工作。

作者介绍:

郝亦非

.NET开发人员，拥有多年开发经验，长期负责后端系统的搭建、重构与维护，对C#语言理解深刻，在工作中积累了丰富的开发经验，乐于分享，在社区贡献了大量文章。

目录: 引言

第一部分 基础知识

第1章 .NET基础知识 2

1.1 .NET框架介绍 2

1.2 .NET框架发展史 3

1.3 .NET框架的主要成员 5

1.3.1 两步编译与跨平台 5

1.3.2 CLR 6

1.3.3 CLI 7

1.3.4 CTS和CLS 8

1.3.5 框架类库 (FCL) 8

1.3.6 基础类库 (BCL) 9

1.4 程序集 10

1.4.1 反向工程—使用ILSpy观察IL 10

1.4.2 程序集与托管模块 11

1.4.3 程序集的部署 18

1.5 .NET程序的编译: IL与JIT 20

1.5.1 什么是IL (CIL) 21

1.5.2 初识IL 21

1.5.3 System.Reflection.Emit 31

1.5.4 即时编译 (JIT) 33

1.5.5 运行时的验证 34

1.5.6 Visual Studio的编译模式与本地代码的优化 34

1.5.7 托管代码与非托管代码的互操作性 35

1.6 CLR启动与Hello World的运行 36

1.7 本章小结 37

1.8 思考题 37

第2章 C#类型基础 (上) 38

2.1 公共类型系统 38

2.2 堆与栈 40

2.2.1 堆 40

2.2.2 栈 41

2.3 引用类型的内存分配 42

2.3.1 字段的对齐 44

2.3.2 同步块索引 45

2.3.3 方法表指针和类型对象 45

2.3.4 静态字段和属性 46

2.4 使用WinDbg探查内存 46

2.4.1 WinDbg简易命令速查 47

2.4.2 使用WinDbg探查引用类型 47

2.4.3 引用类型的复制 52

2.5 值类型 53

2.5.1 基元类型 54

2.5.2 值类型的内存分配 54

2.5.3 值类型的构造函数 56

2.5.4 何时考虑使用值类型 57

2.5.5 值类型是密封的	58
2.5.6 值类型和引用类型的区别与联系	58
2.5.7 嵌套：值类型与引用类型	58
2.6 装箱和拆箱	59
2.6.1 装箱的过程	59
2.6.2 拆箱的过程	60
2.6.3 如何避免拆箱和装箱	60
2.7 本章小结	61
2.8 思考题	61
第3章 C#类型基础（下）	62
3.1 类型的非方法成员	62
3.1.1 常量	62
3.1.2 字段	63
3.1.3 无参属性	64
3.1.4 有参属性	66
3.1.5 属性的意义	67
3.2 类型的构造函数和析构函数	67
3.2.1 静态类	67
3.2.2 实例构造函数（引用类型）	68
3.2.3 实例构造函数（值类型）	69
3.2.4 静态构造函数	69
3.2.5 构造函数的执行顺序	70
3.2.6 析构函数	71
3.3 类型的普通方法成员	72
3.3.1 方法表	73
3.3.2 使用WinDbg探查方法表	73
3.3.3 方法槽表与方法描述表在JIT前后的变化	75
3.3.4 方法调用	77
3.3.5 方法参数的按值传递和按引用传递	80
3.4 类型转换	82
3.4.1 将一个对象转换为它的基类型	83
3.4.2 将一个对象转换为它的派生类型	84
3.4.3 基元类型的类型转换	84
3.4.4 自定义类型转换	84
3.5 System.Object类型的主要方法	85
3.6 本章小结	90
3.7 思考题	91
第4章 C#和面向对象	92
4.1 面向对象程序设计	92
4.2 继承	93
4.2.1 方法表与继承	94
4.2.2 再论Call与Callvirt	94
4.2.3 IL中修饰方法的关键字	95
4.2.4 方法的重载、重写和隐藏	96
4.2.5 值类型的方法调用	100
4.3 接口和多态	101
4.3.1 抽象类与Is A关系	102
4.3.2 接口与Has A关系	102
4.3.3 显式接口实现	103
4.3.4 显式接口实现与泛型接口	104
4.3.5 抽象类VS接口	105
4.3.6 接口不继承自Object	106
4.3.7 接口方法分派	106
4.4 面向对象编程五大原则（SOLID）	106

4.4.1 单一职责原则	107
4.4.2 开闭原则	107
4.4.3 里氏代换原则	111
4.4.4 接口隔离原则	111
4.4.5 依赖倒转原则	112
4.5 本章小结	112
4.6 思考题	113
第5章 字符串	114
5.1 字符	114
5.2 字符串的特性	115
5.2.1 字符串与普通的引用类型相比	116
5.2.2 IL中创建字符串	116
5.2.3 字符串的不变性	117
5.2.4 通过成员来证明不变性	118
5.2.5 为什么要这么设计	119
5.3 字符串驻留	119
5.4 字符串的相加	121
5.5 不变性只是针对托管代码	124
5.6 本章小结	125
5.7 思考题	125
第6章 垃圾回收	126
6.1 垃圾回收的概念	127
6.1.1 GC堆的构造	127
6.1.2 识别垃圾	128
6.1.3 压缩	136
6.1.4 C#的GC规则总结	136
6.2 垃圾回收策略	136
6.2.1 Dispose方法和IDisposable接口	136
6.2.2 析构函数（终结器）	137
6.2.3 如何回收托管资源	140
6.2.4 如何回收非托管资源	142
6.2.5 using关键字	143
6.2.6 总结：怎样实现垃圾回收策略	144
6.3 GC的工作模式	145
6.3.1 工作站模式	145
6.3.2 服务器模式	146
6.4 大对象	146
6.5 本章小结	147
6.6 思考题	147
第7章 异常与异常处理	148
7.1 C#的异常处理	149
7.1.1 try	149
7.1.2 catch	149
7.1.3 finally	150
7.1.4 结构化异常处理	152
7.1.5 throw和throw ex	153
7.2 IL中的异常处理机制	155
7.3 开发中的异常处理	159
7.3.1 提高程序的健壮性	159
7.3.2 使用.NET自带的日志类型	159
7.4 本章小结	163
7.5 思考题	163
第二部分 C#特性	
第8章 委托和事件	166

8.1 委托	166
8.1.1 委托初探	166
8.1.2 使用委托达到代码复用的目的	168
8.1.3 委托的协变和逆变	171
8.1.4 委托的本质	171
8.1.5 委托链与多路广播	174
8.1.6 委托的异步调用	176
8.2 事件	176
8.2.1 事件初探	177
8.2.2 事件的本质	180
8.2.3 Winform中的经典事件：单击按钮	182
8.2.4 Windows消息机制简介	183
8.2.5 观察者模式	185
8.3 本章小结	189
8.4 思考题	189
第9章 泛型	190
9.1 泛型方法	190
9.1.1 类型安全与代码爆炸	191
9.1.2 泛型类型的静态成员	193
9.1.3 泛型与继承	194
9.1.4 泛型约束	194
9.1.5 泛型委托	195
9.1.6 使用泛型委托达到代码复用的目的	196
9.2 可空类型	197
9.3 协变和逆变	200
9.3.1 可变性	200
9.3.2 通过反射调用泛型方法	207
9.4 本章小结	208
9.5 思考题	208
第10章 反射	209
10.1 初识反射	209
10.1.1 获得类型的基本信息	210
10.1.2 获得类型成员和方法调用	211
10.1.3 加载程序集（晚期绑定）	215
10.1.4 反射与泛型	216
10.2 反射的应用场景	218
10.3 反射的性能问题	218
10.3.1 方法反射调用有多慢	218
10.3.2 通过其他方法优化反射调用方式	220
10.3.3 反射优化的性能比较	225
10.4 反射的应用：一个简单的ORM	225
10.4.1 通过反射和特性建立表格和实体的联系	226
10.4.2 为实体增加主键	227
10.4.3 建立本地数据库	228
10.4.4 建立表格	228
10.4.5 删除表格	231
10.4.6 为表格插入数据	231
10.4.7 选择数据	233
10.4.8 多表联查怎么办	234
10.4.9 优化ORM：使用委托来获得值	235
10.4.10 完整的实现代码	235
10.4.11 小结	242
10.5 本章小结	242
10.6 思考题	243

第11章 C#的数据结构	244
11.1 IEnumerable	244
11.1.1 迭代器模式	244
11.1.2 什么是IEnumerable	245
11.1.3 实现一个继承IEnumerable的类型	245
11.1.4 yield的延迟执行特性	256
11.1.5 IEnumerable是一个可枚举序列，但不是容器	257
11.1.6 在迭代的过程中改变集合的状态	259
11.1.7 IEnumerable的缺点和总结	259
11.2 IEnumerable的派生类	260
11.2.1 Array	260
11.2.2 ArrayList	261
11.2.3 IDictionary	262
11.3 IEnumerable的派生类	272
11.3.1 IList	272
11.3.2 LinkedList	272
11.3.3 Queue	281
11.3.4 Stack	283
11.3.5 IDictionary	284
11.3.6 ISet	287
11.4 常用数据结构特征以及操作时间复杂度	288
11.5 如何选择数据结构	289
11.6 本章小结	289
11.7 思考题	290
第12章 LINQ的准备工作	291
12.1 匿名函数、捕获变量与闭包	291
12.1.1 匿名函数	291
12.1.2 演示捕获变量	293
12.1.3 闭包	300
12.2 LINQ的准备工作	301
12.2.1 自动实现的属性	302
12.2.2 隐式类型的局部变量	302
12.2.3 匿名类型	303
12.2.4 扩展方法	305
12.3 Lambda表达式和表达式树	306
12.3.1 从匿名函数到Lambda表达式	306
12.3.2 表达式简介	308
12.3.3 构建简单的表达式树	309
12.3.4 表达式树与反射	311
12.4 本章小结	315
12.5 思考题	315
第13章 LINQ to Object	316
13.1 LINQPad	316
13.2 Enumerable是什么	318
13.3 使用Northwind数据源演示查询操作	318
13.4 投影操作符与过滤操作符	318
13.4.1 Select整个表	318
13.4.2 Select部分列	319
13.4.3 Select结合Lambda表达式	320
13.4.4 使用where进行过滤	320
13.4.5 SelectMany	321
13.4.6 Distinct	324
13.5 排序操作符	325
13.6 分组操作符	325

- 13.7 通过Let声明局部变量 328
- 13.8 连接操作符 329
 - 13.8.1 使用join子句的内连接 330
 - 13.8.2 使用join into子句进行外连接 330
- 13.9 其他常用的操作符 331
- 13.10 延迟执行 333
- 13.11 查询表达式和方法语法 335
- 13.12 本章小结 336
- 13.13 思考题 336
- 第14章 LINQ to SQL 337
 - 14.1 IQueryable 337
 - 14.2 IQueryable与 IEnumerable的异同 338
 - 14.3 数据库操作 339
 - 14.3.1 弱类型实体集 339
 - 14.3.2 Entity Framework 340
 - 14.3.3 Repository模式 342
 - 14.4 使用LINQ to Entity Framework 342
 - 14.4.1 Database First 342
 - 14.4.2 Model First 346
 - 14.4.3 Code First 347
 - 14.5 表达式树转化为SQL 348
 - 14.5.1 准备工作 349
 - 14.5.2 实现IQueryable 351
 - 14.5.3 实现IQueryProvider 352
 - 14.5.4 测试IQueryable的运行流程 354
 - 14.5.5 表达式查看器 355
 - 14.5.6 第一步：解析Where表达式 355
 - 14.5.7 第二步：解析Where Lambda表达式 357
 - 14.5.8 表达式树转化为SQL的总结 360
 - 14.6 LINQ与EF的性能问题 361
 - 14.6.1 避免Select N+1 361
 - 14.6.2 避免重复枚举同一序列 362
 - 14.6.3 避免毫无必要地枚举整个序列 363
 - 14.6.4 Entity Framework的预热 364
 - 14.6.5 AsNoTracking方法 365
 - 14.6.6 简化传入字符串排序的代码 365
 - 14.7 LINQ to Rx 370
 - 14.7.1 事件流序列操作LINQ化 371
 - 14.7.2 事件的限流 372
 - 14.7.3 本质：推模型和拉模型 373
 - 14.8 本章小结 373
 - 14.9 思考题 374
- 第15章 动态语言运行时 375
 - 15.1 dynamic关键字简介 375
 - 15.1.1 动态类型简化晚期绑定 376
 - 15.1.2 动态类型简化COM互操作 378
 - 15.1.3 动态类型相比泛型更加灵活 379
 - 15.1.4 使用ExpandoObject创建动态类型 380
 - 15.1.5 动态类型的好处 381
 - 15.1.6 动态类型的限制 381
 - 15.2 动态类型的原理 382
 - 15.2.1 调用点 382
 - 15.2.2 使用C# object类型模拟动态类型 383
 - 15.2.3 建立调用点 384

15.2.4 初始化调用点	384
15.2.5 使用调用点的目标	385
15.2.6 DLR的缓存策略	386
15.3 本章小结	387
15.4 思考题	387
第16章 C# 6与C# 7的重要特性	388
16.1 C# 6的主要特性	388
16.1.1 自动属性的再次进化	388
16.1.2 简易函数表达式写法	390
16.1.3 字符串插值	392
16.1.4 使用static using调用静态类方法	392
16.1.5 判定null的简写操作符	393
16.1.6 异常过滤	394
16.1.7 nameof运算符	394
16.2 C# 7的主要特性	395
16.2.1 数字字面量	395
16.2.2 改进的out关键字	396
16.2.3 模式匹配	397
16.2.4 C# 7的值类型元组	399
16.2.5 解构	402
16.2.6 局部函数	405
16.2.7 引用返回值和引用局部变量	410
16.2.8 更多的表达式体成员	414
16.2.9 在更多的地方抛出异常	415
16.2.10 具有值类型的引用语义	415
16.3 C# 8前瞻	416
16.4 本章小结	417
16.5 思考题	417
第三部分 多线程和异步	
第17章 多线程概念	420
17.1 进程和线程	420
17.1.1 操作系统简单发展史	420
17.1.2 进程	421
17.1.3 进程调度	424
17.1.4 进程的上下文切换	424
17.1.5 线程	425
17.1.6 单核CPU多线程是否能够提高性能	428
17.2 .NET中的进程与应用程序域	428
17.2.1 .NET中的进程（托管进程）	428
17.2.2 进程间通信	428
17.2.3 使用剪贴板实现进程间通信	429
17.2.4 应用程序域	431
17.3 .NET中的线程（托管线程）与Thread类的基本使用	432
17.3.1 创建线程	433
17.3.2 线程命名	433
17.3.3 前台线程和后台线程	434
17.3.4 线程的状态	435
17.3.5 向线程传递数据	437
17.3.6 异常处理	438
17.4 线程池	439
17.4.1 线程池是如何管理线程的	439
17.4.2 线程池的线程调度策略	440
17.4.3 使用线程池：显式操作	441
17.4.4 使用线程池：异步委托	442

17.4.5 使用线程池：通过任务 442
17.5 线程局部存储区（TLS） 443
17.5.1 使用[ThreadStatic]特性 443
17.5.2 使用Thread类上的两个方法：GetData和SetData 444
17.5.3 使用ThreadLocal 445
17.6 本章小结 446
17.7 思考题 447
第18章 多线程同步 448
18.1 多线程同步的概念 448
18.2 锁：基元构造 455
18.3 用户模式构造 456
18.3.1 同步问题的提出 456
18.3.2 JIT优化、有序性和指令重排 457
18.3.3 C#的内存模型 459
18.3.4 可见性、有序性、内存栅栏与volatile关键字 460
18.3.5 避免使用volatile 464
18.3.6 互锁构造 465
18.3.7 Interlocked Anything模式 466
18.3.8 使用用户模式构造的例子 467
18.3.9 实现自旋锁 468
18.3.10 SpinLock 469
18.4 内核模式构造 470
18.4.1 通过WaitHandle操作内核对象 471
18.4.2 事件构造 472
18.4.3 信号量构造 477
18.4.4 使用信号量实现锁 478
18.4.5 互斥量构造 478
18.4.6 基元构造锁总结 479
18.5 锁：混合构造 480
18.5.1 一个简单的混合锁 481
18.5.2 优化DIY混合锁：使锁支持递归和自旋 482
18.5.3 最常用的锁：Monitor的工作原理 484
18.5.4 Monitor的递归调用 485
18.5.5 如何选择同步对象 486
18.6 .NET 4中新增的轻量级同步工具 488
18.6.1 ManualResetEventSlim和SemaphoreSlim类 489
18.6.2 ReaderWriterLockSlim类 489
18.7 这么多锁：总结 491
18.8 线程安全的集合类 491
18.9 本章小结 494
18.10 思考题 494
第19章 .NET 4之前的异步编程 495
19.1 基本概念 495
19.2 使用System.Threading进行异步编程 497
19.2.1 耗时任务 497
19.2.2 最简单的异步编程 497
19.2.3 获得执行结果 498
19.2.4 实现回调函数获得执行结果 499
19.2.5 线程统一取消模型 500
19.2.6 总结 504
19.3 基于委托的异步编程模型 505
19.3.1 APM的设计规范 505
19.3.2 获得异步委托的执行结果 506
19.3.3 System.IAsyncResult接口 507

- 19.3.4 通过回调的方式获得异步委托的执行结果 508
- 19.3.5 使用线程统一取消模型进行取消 509
- 19.3.6 捕获异步编程的异常 511
- 19.3.7 GUI的线程处理模型 512
- 19.3.8 GUI中使用委托进行异步—Control.Begin-Invoke原理 513
- 19.3.9 GUI中使用委托进行异步—WinForm程序示例 515
- 19.3.10 总结 517
- 19.4 基于事件的异步编程模式 518
 - 19.4.1 EAP的设计规范 518
 - 19.4.2 BackgroundWorker类简介 519
 - 19.4.3 GUI中使用EAP进行异步—WPF程序示例 521
 - 19.4.4 总结 525
- 19.5 我们需要这么多异步编程的方法吗 525
- 19.6 本章小结 525
- 19.7 思考题 526
- 第20章 任务并行库 527
 - 20.1 任务并行库 527
 - 20.2 使用任务进行异步编程 528
 - 20.2.1 新建任务 528
 - 20.2.2 任务的状态 529
 - 20.2.3 任务调度器 530
 - 20.2.4 任务工厂 531
 - 20.3 获得任务的执行结果 532
 - 20.3.1 等待任务完成 532
 - 20.3.2 任务连接作为回调函数 533
 - 20.3.3 使用过去的方式 539
 - 20.4 任务的异常处理 539
 - 20.4.1 父子任务中的异常处理（附加子任务） 541
 - 20.4.2 父子任务中的异常处理（分离子任务） 541
 - 20.5 任务的取消 544
 - 20.5.1 直接退出任务 544
 - 20.5.2 利用ThrowIfCancellation-Requested方法 545
 - 20.5.3 任务取消的异常处理 545
 - 20.5.4 任务异常处理：总结 548
 - 20.6 在WPF中使用任务 549
 - 20.7 并行编程与PLINQ 552
 - 20.8 Parallel类 553
 - 20.8.1 Parallel.Invoke与任务并行 554
 - 20.8.2 Parallel.For/ForEach与数据并行 555
 - 20.8.3 收集任务的执行结果 558
 - 20.8.4 指定最大并行度 558
 - 20.8.5 在Parallel类中捕捉异常 559
 - 20.8.6 使用ParallelOption取消并行任务 560
 - 20.8.7 使用ParallelLoopState退出For/ForEach循环 562
 - 20.8.8 Parallel方法的取消与异常处理总结 563
 - 20.9 PLINQ 564
 - 20.9.1 指定执行模式 566
 - 20.9.2 指定并行度 567
 - 20.9.3 PLINQ与排序 567
 - 20.9.4 PLINQ的数据分区策略 568
 - 20.9.5 PLINQ的输出端与结果收集策略 571
 - 20.9.6 使用Aggregate的并行版本进行计算 572
 - 20.9.7 取消PLINQ查询 572
 - 20.9.8 在PLINQ中捕捉异常 573

20.9.9 总结：是否需要使用PLINQ 574
20.10 Lazy和延迟初始化 574
20.11 本章小结 576
20.12 思考题 576
第21章 .NET 4.5异步编程实例 577
21.1 初识async/await 577
21.1.1 异步方法的执行流程 579
21.1.2 异步方法的异常处理 582
21.1.3 异步方法的取消 584
21.1.4 调用方法获得结果的几种方式 585
21.1.5 async和await总结 587
21.2 了解async/await的原理 588
21.2.1 示例程序 588
21.2.2 使用ILSpy反编译 589
21.2.3 状态机结构 589
21.2.4 骨架方法 594
21.2.5 异常处理 595
21.3 GUI：使用async和await关键字进行异步操作 595
21.3.1 异步操作的基本模型 596
21.3.2 使用WhenAll等待一组任务完成 597
21.3.3 使用WhenAny等待任意一个任务完成 598
21.3.4 任务完成时的处理 599
21.4 可等待模式 600
21.4.1 定义 600
21.4.2 await anything 601
21.5 本章小结 603
21.6 思考题 604
后记 605
附录 程序员面试流程概览 606
· · · · · (收起)

[C#从现象到本质 下载链接1](#)

标签

C

#.NET

编程

评论

是一本C#编程的好书。最起码前言里关于工作的总结就超过市面上大部分的书。
对C#中比较重要的概念进行了详细有深度的讲解，总结的很全面，深刻理解C#有很好的帮助。

[C#从现象到本质 下载链接1](#)

书评

[C#从现象到本质 下载链接1](#)