

C#高级编程(第11版)



[C#高级编程\(第11版\) 下载链接1](#)

著者:[美]克里斯琴·内格尔 (Christian Nagel) 著

出版者:清华大学出版社

出版时间:2019-3

装帧:平装

isbn:9787302522560

C# 7内幕指南，包括高级新特性

目前Visual Studio 2017提供了C# 7。发布为NuGet包的每个.NET Core部分都进行了更新。阅读这本专家级指南是经验丰富的程序员提高效率的更快捷方式。C# 7以更快的速度完成更多工作，没有人比Christian Nagel更适合传播在现实世界中极有价值的权威信息。本书论述清晰，内容完整详尽，为开发人员展示了如何将.NET引入非微软平台，如何操作这些平台上的工具，例如Docker、Gulp和NPM。

★为需要新工具的高级开发人员介绍了C# 7和.NET Core 2.0的扩展新特性

★揭示了Visual Studio

2017的新技巧和切合实际的提示，包括新的用户界面、新增的模板、编辑器的改进等

★论述了计划工作流的全新方式，使编码更快，诊断和调试更精确，测试更频繁，发布更自信

★为开发在Android、iOS、Windows、Linux、网络和云上运行的应用程序提供了循序渐进的指南

★掌握Visual Studio的高效率特性，以加速AI革新

作者介绍:

Christian Nagel是Visual Studio和开发技术方向的Microsoft MVP、软件架构师、资深开发人员，自2000年开始就使用.NET技术建立解决方案。他编著过多本畅销的.NET图书，经常在Ignite和Tech Days等国际会议上发言，是.NET用户组的主要支持者。他拥有微软认证培训师（MCT）称号，是通用Windows应用程序、ASP.NET Core和Microsoft Azure的专业开发人员。

目录: 第 I 部分 C# 语言

第1章 .NET 应用程序和工具 3

1.1 选择技术 3

1.2 回顾.NET 历史 4

1.2.1 C# 1.0——一种新语言 4

1.2.2 带有泛型的C# 2 和.NET 2 6

1.2.3 .NET 3.0——Windows Presentation Foundation 6

1.2.4 C# 3 和.NET 3.5——LINQ 6

1.2.5 C# 4 和.NET 4.0——dynamic 和TPL 7

1.2.6 C# 5 和异步编程 7

1.2.7 C# 6 和.NET Core 1.0 8

1.2.8 C# 7 和.NET Core 2.0 8

1.2.9 选择技术，继续前进 9

1.3 .NET 术语 10

1.3.1 .NET Framework 11

1.3.2 .NET Core 11

1.3.3 .NET Standard 11

1.3.4 NuGet 包 12

1.3.5 名称空间 12

1.3.6 公共语言运行库 13

1.3.7 Windows 运行库 13

1.4 用.NET Core CLI 编译 14

1.4.1 设置环境 14

1.4.2 创建应用程序 15

1.4.3 构建应用程序 16

1.4.4 运行应用程序 16

1.4.5 创建Web 应用程序 17

1.4.6 发布应用程序 17

1.5 使用Visual Studio 2017 19

1.6 应用程序类型和技术 24

1.6.1 数据访问 24

- 1.6.2 Windows 应用程序 24
- 1.6.3 Xamarin 24
- 1.6.4 Web 应用程序 25
- 1.6.5 Web API 25
- 1.6.6 WebHooks 和SignalR 25
- 1.6.7 Microsoft Azure 25
- 1.7 开发工具 26
 - 1.7.1 Visual Studio Community 27
 - 1.7.2 Visual Studio Professional 27
 - 1.7.3 Visual Studio Enterprise 27
 - 1.7.4 Visual Studio for Mac 27
 - 1.7.5 Visual Studio Code 27
- 1.8 小结 27
- 第2章 核心C# 29
 - 2.1 C#基础 29
 - 2.2 变量 31
 - 2.2.1 初始化变量 31
 - 2.2.2 类型推断 32
 - 2.2.3 变量的作用域 33
 - 2.2.4 常量 34
 - 2.3 预定义数据类型 35
 - 2.3.1 值类型和引用类型 35
 - 2.3.2 .NET 类型 36
 - 2.3.3 预定义的值类型 36
 - 2.3.4 预定义的引用类型 40
 - 2.4 程序流控制 42
 - 2.4.1 条件语句 42
 - 2.4.2 循环 44
 - 2.4.3 跳转语句 47
 - 2.5 名称空间 47
 - 2.5.1 using 语句 48
 - 2.5.2 名称空间的别名 49
 - 2.6 Main()方法 49
 - 2.7 使用注释 50
 - 2.7.1 源文件中的内部注释 50
 - 2.7.2 XML 文档 51
 - 2.8 C#预处理器指令 52
 - 2.8.1 #define 和#undef 52
 - 2.8.2 #if、#elif、#else 和#endif 52
 - 2.8.3 #warning 和 #error 53
 - 2.8.4 #region 和#endregion 53
 - 2.8.5 #line 53
 - 2.8.6 #pragma 54
 - 2.9 C#编程准则 54
 - 2.9.1 关于标识符的规则 54
 - 2.9.2 用法约定 55
 - 2.10 小结 58
- 第3章 对象和类型 59
 - 3.1 创建及使用类 60
 - 3.2 类和结构 60
 - 3.3 类 61
 - 3.3.1 字段 61
 - 3.3.2 只读字段 61
 - 3.3.3 属性 62

- 3.3.4 匿名类型 65
- 3.3.5 方法 66
- 3.3.6 构造函数 69
- 3.4 结构 73
 - 3.4.1 结构是值类型 74
 - 3.4.2 只读结构 75
 - 3.4.3 结构和继承 75
 - 3.4.4 结构的构造函数 75
 - 3.4.5 ref 结构 76
- 3.5 按值和按引用传递参数 76
 - 3.5.1 ref 参数 77
 - 3.5.2 out 参数 77
 - 3.5.3 in 参数 78
- 3.6 可空类型 79
- 3.7 枚举类型 79
- 3.8 部分类 81
- 3.9 扩展方法 83
- 3.10 Object 类 83
- 3.11 小结 84
- 第4章 继承 85
 - 4.1 面向对象 85
 - 4.2 继承的类型 85
 - 4.2.1 多重继承 86
 - 4.2.2 结构和类 86
 - 4.3 实现继承 86
 - 4.3.1 虚方法 87
 - 4.3.2 多态性 88
 - 4.3.3 隐藏方法 89
 - 4.3.4 调用方法的基类版本 90
 - 4.3.5 抽象类和抽象方法 90
 - 4.3.6 密封类和密封方法 91
 - 4.3.7 派生类的构造函数 91
 - 4.4 修饰符 93
 - 4.4.1 访问修饰符 93
 - 4.4.2 其他修饰符 94
 - 4.5 接口 94
 - 4.5.1 定义和实现接口 95
 - 4.5.2 派生的接口 97
 - 4.6 is 和 as 运算符 98
 - 4.7 小结 99
- 第5章 泛型 100
 - 5.1 泛型概述 100
 - 5.1.1 性能 101
 - 5.1.2 类型安全 102
 - 5.1.3 二进制代码的重用 102
 - 5.1.4 代码的扩展 102
 - 5.1.5 命名约定 102
 - 5.2 创建泛型类 103
 - 5.3 泛型类的功能 105
 - 5.3.1 默认值 106
 - 5.3.2 约束 106
 - 5.3.3 继承 108
 - 5.3.4 静态成员 108
 - 5.4 泛型接口 109

- 5.4.1 协变和抗变 109
- 5.4.2 泛型接口的协变 110
- 5.4.3 泛型接口的抗变 111
- 5.5 泛型结构 111
- 5.6 泛型方法 113
 - 5.6.1 泛型方法示例 113
 - 5.6.2 带约束的泛型方法 114
 - 5.6.3 带委托的泛型方法 115
 - 5.6.4 泛型方法规范 115
- 5.7 小结 116
- 第6章 运算符和类型强制转换 117
 - 6.1 运算符和类型转换 117
 - 6.2 运算符 118
 - 6.2.1 运算符的简化操作 119
 - 6.2.2 运算符的优先级和关联性 125
 - 6.3 使用二进制运算符 126
 - 6.3.1 位的移动 128
 - 6.3.2 有符号数和无符号数 128
 - 6.4 类型的安全性 129
 - 6.4.1 类型转换 130
 - 6.4.2 装箱和拆箱 132
 - 6.5 比较对象的相等性 133
 - 6.5.1 比较引用类型的相等性 133
 - 6.5.2 比较值类型的相等性 134
 - 6.6 运算符重载 135
 - 6.6.1 运算符的工作方式 135
 - 6.6.2 运算符重载的示例：Vector 结构 136
 - 6.6.3 比较运算符的重载 139
 - 6.6.4 可以重载的运算符 140
 - 6.7 实现自定义的索引运算符 141
 - 6.8 用户定义的类型强制转换 142
 - 6.8.1 实现用户定义的类型强制转换 143
 - 6.8.2 多重类型强制转换 147
 - 6.9 小结 150
- 第7章 数组 151
 - 7.1 相同类型的多个对象 151
 - 7.2 简单数组 152
 - 7.2.1 数组的声明 152
 - 7.2.2 数组的初始化 152
 - 7.2.3 访问数组元素 153
 - 7.2.4 使用引用类型 153
 - 7.3 多维数组 154
 - 7.4 锯齿数组 155
 - 7.5 Array 类 156
 - 7.5.1 创建数组 156
 - 7.5.2 复制数组 156
 - 7.5.3 排序 157
 - 7.6 数组作为参数 159
 - 7.7 数组协变 159
 - 7.8 枚举 160
 - 7.8.1 IEnumerator 接口 160
 - 7.8.2 foreach 语句 160
 - 7.8.3 yield 语句 161
 - 7.9 结构比较 164

- 7.10 Span 165
 - 7.10.1 创建切片 166
 - 7.10.2 使用Span 改变值 166
 - 7.10.3 只读的Span 167
- 7.11 数组池 167
 - 7.11.1 创建数组池 168
 - 7.11.2 从池中租用内存 168
 - 7.11.3 将内存返回给池 168
- 7.12 小结 169
- 第8章 委托、lambda 表达式和事件 170
 - 8.1 引用方法 170
 - 8.2 委托 170
 - 8.2.1 声明委托 171
 - 8.2.2 使用委托 172
 - 8.2.3 简单的委托示例 174
 - 8.2.4 Action<T>和Func<T>委托 175
 - 8.2.5 BubbleSorter 示例 176
 - 8.2.6 多播委托 177
 - 8.2.7 匿名方法 180
 - 8.3 lambda 表达式 181
 - 8.3.1 参数 181
 - 8.3.2 多行代码 181
 - 8.3.3 闭包 182
 - 8.4 事件 182
 - 8.4.1 事件发布程序 182
 - 8.4.2 事件侦听器 184
 - 8.5 小结 185
- 第9章 字符串和正则表达式 186
 - 9.1 System.String 类 187
 - 9.1.1 构建字符串 188
 - 9.1.2 StringBuilder 成员 190
 - 9.2 字符串格式 190
 - 9.2.1 字符串插值 191
 - 9.2.2 日期时间和数字的格式 192
 - 9.2.3 自定义字符串格式 193
 - 9.3 正则表达式 194
 - 9.3.1 正则表达式概述 194
 - 9.3.2 RegularExpressionsPlayground 示例 195
 - 9.3.3 显示结果 197
 - 9.3.4 匹配、组和捕获 198
 - 9.4 字符串和Span 200
 - 9.5 小结 201
- 第10章 集合 202
 - 10.1 概述 202
 - 10.2 集合接口和类型 203
 - 10.3 列表 203
 - 10.3.1 创建列表 204
 - 10.3.2 只读集合 210
 - 10.4 队列 210
 - 10.5 栈 213
 - 10.6 链表 214
 - 10.7 有序列表 217
 - 10.8 字典 219
 - 10.8.1 字典初始化器 219

- 10.8.2 键的类型 219
- 10.8.3 字典示例 220
- 10.8.4 Lookup 类 223
- 10.8.5 有序字典 223
- 10.9 集 224
- 10.10 性能 225
- 10.11 小结 227
- 第11 章 特殊的集合 228
 - 11.1 概述 228
 - 11.2 处理位 228
 - 11.2.1 BitArray 类 229
 - 11.2.2 BitVector32 结构 230
 - 11.3 可观察的集合 232
 - 11.4 不变的集合 233
 - 11.4.1 使用构建器和不变的集合 235
 - 11.4.2 不变集合类型和接口 235
 - 11.4.3 使用LINQ 和不变的数组 236
 - 11.5 并发集合 236
 - 11.5.1 创建管道 237
 - 11.5.2 使用BlockingCollection 239
 - 11.5.3 使用ConcurrentDictionary 240
 - 11.5.4 完成管道 241
 - 11.6 小结 242
- 第12 章 LINQ 243
 - 12.1 LINQ 概述 243
 - 12.1.1 列表和实体 244
 - 12.1.2 LINQ 查询 246
 - 12.1.3 扩展方法 246
 - 12.1.4 推迟查询的执行 248
 - 12.2 标准的查询操作符 249
 - 12.2.1 筛选 250
 - 12.2.2 用索引筛选 251
 - 12.2.3 类型筛选 252
 - 12.2.4 复合的from 子句 252
 - 12.2.5 排序 253
 - 12.2.6 分组 254
 - 12.2.7 LINQ 查询中的变量 255
 - 12.2.8 对嵌套的对象分组 255
 - 12.2.9 内连接 256
 - 12.2.10 左外连接 258
 - 12.2.11 组连接 260
 - 12.2.12 集合操作 262
 - 12.2.13 合并 263
 - 12.2.14 分区 264
 - 12.2.15 聚合操作符 264
 - 12.2.16 转换操作符 266
 - 12.2.17 生成操作符 267
 - 12.3 并行LINQ 267
 - 12.3.1 并行查询 268
 - 12.3.2 分区器 268
 - 12.3.3 取消 269
 - 12.4 表达式树 269
 - 12.5 LINQ 提供程序 271
 - 12.6 小结 272

- 第13 章 C#函数式编程 273
 - 13.1 概述 273
 - 13.1.1 避免状态突变 274
 - 13.1.2 函数作为第一个类 275
 - 13.2 表达式体的成员 275
 - 13.3 扩展方法 276
 - 13.4 using static 声明 277
 - 13.5 本地函数 278
 - 13.5.1 本地函数与yield 语句 279
 - 13.5.2 递归本地函数 281
 - 13.6 元组 282
 - 13.6.1 元组的声明和初始化 282
 - 13.6.2 元组解构 283
 - 13.6.3 元组的返回 283
 - 13.6.4 幕后的原理 284
 - 13.6.5 ValueTuple 与元组的兼容性 285
 - 13.6.6 推断出元组名称 285
 - 13.6.7 元组与链表 286
 - 13.6.8 元组和LINQ 286
 - 13.6.9 解构 287
 - 13.6.10 解构与扩展方法 288
 - 13.7 模式匹配 288
 - 13.7.1 模式匹配与is 运算符 288
 - 13.7.2 模式匹配与switch 语句 290
 - 13.7.3 模式匹配与泛型 291
 - 13.8 小结 291
- 第14 章 错误和异常 292
 - 14.1 简介 292
 - 14.2 异常类 293
 - 14.3 捕获异常 294
 - 14.3.1 异常和性能 296
 - 14.3.2 实现多个catch 块 296
 - 14.3.3 在其他代码中捕获异常 299
 - 14.3.4 System.Exception 属性 299
 - 14.3.5 异常过滤器 299
 - 14.3.6 重新抛出异常 300
 - 14.3.7 没有处理异常时发生的情况 303
 - 14.4 用户定义的异常类 303
 - 14.4.1 捕获用户定义的异常 304
 - 14.4.2 抛出用户定义的异常 305
 - 14.4.3 定义用户定义的异常类 307
 - 14.5 调用者信息 309
 - 14.6 小结 310
- 第15 章 异步编程 311
 - 15.1 异步编程的重要性 311
 - 15.2 异步编程的.NET 历史 312
 - 15.2.1 同步调用 312
 - 15.2.2 异步模式 313
 - 15.2.3 基于事件的异步模式 314
 - 15.2.4 基于任务的异步模式 314
 - 15.2.5 异步Main()方法 315
 - 15.3 异步编程的基础 315
 - 15.3.1 创建任务 316
 - 15.3.2 调用异步方法 316

- 15.3.3 使用Awaiter 317
- 15.3.4 延续任务 317
- 15.3.5 同步上下文 318
- 15.3.6 使用多个异步方法 318
- 15.3.7 使用ValueTasks 319
- 15.3.8 转换异步模式 320
- 15.4 错误处理 320
 - 15.4.1 异步方法的异常处理 321
 - 15.4.2 多个异步方法的异常处理 321
 - 15.4.3 使用AggregateException 信息 322
- 15.5 异步与Windows 应用程序 322
 - 15.5.1 配置await 323
 - 15.5.2 切换到UI 线程 324
 - 15.5.3 使用IAsyncOperation 325
 - 15.5.4 避免阻塞情况 325
- 15.6 小结 325
- 第16 章 反射、元数据和动态编程 326
 - 16.1 在运行期间检查代码和动态编程 326
 - 16.2 自定义特性 327
 - 16.2.1 编写自定义特性 327
 - 16.2.2 自定义特性示例：WhatsNewAttributes 330
 - 16.3 反射 331
 - 16.3.1 System.Type 类 332
 - 16.3.2 TypeView 示例 333
 - 16.3.3 Assembly 类 335
 - 16.3.4 完成WhatsNewAttributes 示例 336
 - 16.4 为反射使用动态语言扩展 339
 - 16.4.1 创建Calculator 库 339
 - 16.4.2 动态实例化类型 339
 - 16.4.3 用Reflection API 调用成员 340
 - 16.4.4 使用动态类型调用成员 340
 - 16.5 dynamic 类型 341
 - 16.6 DynamicObject 和ExpandoObject概述 344
 - 16.6.1 DynamicObject 344
 - 16.6.2 ExpandoObject 345
 - 16.7 小结 347
- 第17 章 托管和非托管内存 348
 - 17.1 内存 348
 - 17.2 后台内存管理 349
 - 17.2.1 值数据类型 349
 - 17.2.2 引用数据类型 350
 - 17.2.3 垃圾收集 352
 - 17.3 强引用和弱引用 354
 - 17.4 处理非托管的资源 354
 - 17.4.1 析构函数或终结器 355
 - 17.4.2 IDisposable 接口 356
 - 17.4.3 using 语句 356
 - 17.4.4 实现IDisposable 接口和析构函数 357
 - 17.4.5 IDisposable 和终结器的规则 358
 - 17.5 不安全的代码 358
 - 17.5.1 用指针直接访问内存 358
 - 17.5.2 指针示例：PointerPlayground 364
 - 17.5.3 使用指针优化性能 367
 - 17.6 引用的语义 369

- 17.6.1 传递ref 和返回ref 371
- 17.6.2 ref 和数组 371
- 17.7 Span<T> 373
 - 17.7.1 Span 引用托管堆 373
 - 17.7.2 Span 引用栈 373
 - 17.7.3 Span 引用本机堆 374
 - 17.7.4 Span 扩展方法 374
- 17.8 平台调用 375
- 17.9 小结 378
- 第18 章 Visual Studio 2017 379
 - 18.1 使用Visual Studio 2017 379
 - 18.1.1 Visual Studio 的版本 382
 - 18.1.2 Visual Studio 设置 382
 - 18.2 创建项目 383
 - 18.2.1 面向多个版本的.NET 384
 - 18.2.2 选择项目类型 385
 - 18.3 浏览并编写项目 388
 - 18.3.1 Solution Explorer 388
 - 18.3.2 使用代码编辑器 394
 - 18.3.3 学习和理解其他窗口 399
 - 18.3.4 排列窗口 402
 - 18.4 构建项目 402
 - 18.4.1 构建、编译和生成代码 403
 - 18.4.2 调试版本和发布版本 403
 - 18.4.3 选择配置 404
 - 18.4.4 编辑配置 404
 - 18.5 调试代码 406
 - 18.5.1 设置断点 407
 - 18.5.2 使用数据提示和调试器可视化工具 407
 - 18.5.3 Live Visual Tree 408
 - 18.5.4 监视和修改变量 409
 - 18.5.5 异常 409
 - 18.5.6 多线程 410
 - 18.6 重构工具 411
 - 18.7 诊断工具 412
 - 18.8 通过Docker 创建和使用容器 415
 - 18.8.1 Docker 简介 416
 - 18.8.2 在Docker 容器中运行ASP .NET Core 416
 - 18.8.3 创建Dockerfile 417
 - 18.8.4 使用Visual Studio 418
 - 18.9 小结 420
- 第II 部分 .NET Core 与Windows Runtime
- 第19 章 库、程序集、包和NuGet 423
 - 19.1 库的地狱 423
 - 19.2 程序集 425
 - 19.3 创建库 426
 - 19.3.1 .NET 标准 427
 - 19.3.2 创建.NET 标准库 428
 - 19.3.3 解决方案文件 428
 - 19.3.4 引用项目 429
 - 19.3.5 引用NuGet 包 429
 - 19.3.6 NuGet 的来源 430
 - 19.3.7 使用.NET Framework 库 431
 - 19.4 使用共享项目 433

- 19.5 创建NuGet 包 435
 - 19.5.1 NuGet 包和命令行 435
 - 19.5.2 支持多个平台 435
 - 19.5.3 NuGet 包与Visual Studio 436
- 19.6 小结 438
- 第20 章 依赖注入 439
 - 20.1 依赖注入的概念 439
 - 20.1.1 使用没有依赖注入的服务 440
 - 20.1.2 使用依赖注入 441
 - 20.2 使用.NET Core DI 容器 442
 - 20.3 服务的生命周期 443
 - 20.3.1 使用单例和临时服务 445
 - 20.3.2 使用Scoped 服务 446
 - 20.3.3 使用自定义工厂 448
 - 20.4 使用选项初始化服务 449
 - 20.5 使用配置文件 450
 - 20.6 创建平台独立性 452
 - 20.6.1 .NET 标准库 452
 - 20.6.2 WPF 应用程序 453
 - 20.6.3 UWP 应用程序 454
 - 20.6.4 Xamarin 应用程序 455
 - 20.7 使用其他DI 容器 456
 - 20.8 小结 457
- 第21 章 任务和并行编程 458
 - 21.1 概述 459
 - 21.2 Parallel 类 460
 - 21.2.1 使用Parallel.For()方法循环 460
 - 21.2.2 提前中断Parallel.For 462
 - 21.2.3 Parallel.For()方法的初始化 462
 - 21.2.4 使用Parallel.ForEach()方法循环 463
 - 21.2.5 通过Parallel.Invoke()方法调用多个方法 464
 - 21.3 任务 464
 - 21.3.1 启动任务 464
 - 21.3.2 Future——任务的结果 466
 - 21.3.3 连续的任务 467
 - 21.3.4 任务层次结构 468
 - 21.3.5 从方法中返回任务 468
 - 21.3.6 等待任务 468
 - 21.4 取消架构 470
 - 21.4.1 Parallel.For()方法的取消 470
 - 21.4.2 任务的取消 471
 - 21.5 数据流 472
 - 21.5.1 使用动作块 472
 - 21.5.2 源和目标数据块 473
 - 21.5.3 连接块 474
 - 21.6 Timer 类 475
 - 21.7 线程问题 477
 - 21.7.1 争用条件 477
 - 21.7.2 死锁 479
 - 21.8 lock 语句和线程安全 480
 - 21.9 Interlocked 类 483
 - 21.10 Monitor 类 484
 - 21.11 SpinLock 结构 485
 - 21.12 WaitHandle 基类 485

- 21.13 Mutex 类 485
- 21.14 Semaphore 类 486
- 21.15 Events 类 487
- 21.16 Barrier 类 490
- 21.17 ReaderWriterLockSlim 类 492
- 21.18 Lock 和await 494
- 21.19 小结 496
- 第22 章 文件和流 497
 - 22.1 概述 498
 - 22.2 管理文件系统 498
 - 22.2.1 检查驱动器信息 499
 - 22.2.2 使用Path 类 500
 - 22.2.3 创建文件和文件夹 500
 - 22.2.4 访问和修改文件属性 501
 - 22.2.5 使用File 执行读写操作 502
 - 22.3 枚举文件 503
 - 22.4 使用流 504
 - 22.4.1 使用文件流 505
 - 22.4.2 读取流 508
 - 22.4.3 写入流 508
 - 22.4.4 复制流 509
 - 22.4.5 随机访问流 509
 - 22.4.6 使用缓存的流 510
 - 22.5 使用读取器和写入器 511
 - 22.5.1 StreamReader 类 511
 - 22.5.2 StreamWriter 类 512
 - 22.5.3 读写二进制文件 512
 - 22.6 压缩文件 513
 - 22.6.1 使用压缩流 514
 - 22.6.2 使用Brotli 514
 - 22.6.3 压缩文件 515
 - 22.7 观察文件的更改 515
 - 22.8 使用内存映射的文件 516
 - 22.8.1 使用访问器创建内存映射文件 517
 - 22.8.2 使用流创建内存映射文件 518
 - 22.9 使用管道通信 520
 - 22.9.1 创建命名管道服务器 520
 - 22.9.2 创建命名管道客户端 521
 - 22.9.3 创建匿名管道 522
 - 22.10 通过Windows 运行库使用文件和流 523
 - 22.10.1 Windows App 编辑器 523
 - 22.10.2 把Windows Runtime 类型映射为.NET 类型 525
 - 22.11 小结 526
- 第23 章 网络 527
 - 23.1 概述 527
 - 23.2 HttpClient 类 528
 - 23.2.1 发出异步的Get 请求 528
 - 23.2.2 抛出异常 529
 - 23.2.3 传递标题 529
 - 23.2.4 访问内容 531
 - 23.2.5 用HttpMessageHandler 自定义请求 531
 - 23.2.6 使用SendAsync 创建HttpRequestMessage 532
 - 23.2.7 使用HttpClient和Windows Runtime 532
 - 23.3 使用WebListener 类 534

- 23.4 使用实用工具类 536
 - 23.4.1 URI 537
 - 23.4.2 IPAddress 538
 - 23.4.3 IPHostEntry 538
 - 23.4.4 Dns 539
- 23.5 使用TCP 540
 - 23.5.1 使用TCP 创建HTTP 客户程序 540
 - 23.5.2 创建TCP 侦听器 541
 - 23.5.3 创建TCP 客户端 547
 - 23.5.4 TCP 和UDP 550
- 23.6 使用UDP 550
 - 23.6.1 建立UDP 接收器 550
 - 23.6.2 创建UDP 发送器 551
 - 23.6.3 使用多播 553
- 23.7 使用套接字 554
 - 23.7.1 使用套接字创建侦听器 554
 - 23.7.2 使用NetworkStream 和套接字 556
 - 23.7.3 通过套接字使用读取器和写入器 557
 - 23.7.4 使用套接字实现接收器 557
- 23.8 小结 559
- 第24 章 安全性 560
 - 24.1 概述 560
 - 24.2 验证用户信息 561
 - 24.2.1 使用Windows 标识 561
 - 24.2.2 Windows Principal 562
 - 24.2.3 使用声称 562
 - 24.3 加密数据 564
 - 24.3.1 创建和验证签名 565
 - 24.3.2 实现安全的数据交换 567
 - 24.3.3 使用RSA 签名和散列 569
 - 24.4 保护数据 571
 - 24.4.1 实现数据保护 571
 - 24.4.2 用户机密 573
 - 24.5 资源的访问控制 575
 - 24.6 Web 安全性 577
 - 24.6.1 编码 577
 - 24.6.2 SQL 注入 579
 - 24.6.3 跨站点请求伪造 580
 - 24.7 小结 581
- 第25 章 ADO.NET 和事务 582
 - 25.1 ADO.NET 概述 582
 - 25.1.1 示例数据库 583
 - 25.1.2 NuGet 包和名称空间 583
 - 25.2 使用数据库连接 584
 - 25.2.1 管理连接字符串 585
 - 25.2.2 连接池 585
 - 25.2.3 连接信息 585
 - 25.3 命令 587
 - 25.3.1 ExecuteNonQuery() 方法 587
 - 25.3.2 ExecuteScalar() 方法 588
 - 25.3.3 ExecuteReader() 方法 589
 - 25.3.4 调用存储过程 590
 - 25.4 异步数据访问 591
 - 25.5 事务 592

- 25.6 事务和System.Transaction 595
 - 25.6.1 可提交的事务 597
 - 25.6.2 依赖事务 598
 - 25.6.3 环境事务 599
 - 25.6.4 嵌套作用域和环境事务 601
- 25.7 小结 602
- 第26 章 Entity Framework Core 604
 - 26.1 Entity Framework 简史 605
 - 26.2 EF Core 简介 606
 - 26.2.1 创建模型 607
 - 26.2.2 约定、注释和流利API 607
 - 26.2.3 创建上下文 608
 - 26.2.4 创建数据库 608
 - 26.2.5 删除数据库 609
 - 26.2.6 写入数据库 609
 - 26.2.7 读取数据库 610
 - 26.2.8 更新记录 610
 - 26.2.9 删除记录 611
 - 26.2.10 日志记录 611
 - 26.3 使用依赖注入 612
 - 26.4 创建模型 614
 - 26.4.1 创建关系 614
 - 26.4.2 数据注释 614
 - 26.4.3 流利API 615
 - 26.4.4 自包含类型的配置 616
 - 26.4.5 在数据库中搭建模型 617
 - 26.4.6 映射到字段 618
 - 26.4.7 阴影属性 619
 - 26.5 查询 621
 - 26.5.1 基本查询 621
 - 26.5.2 客户端和服务端求值 622
 - 26.5.3 原始SQL 查询 623
 - 26.5.4 已编译查询 624
 - 26.5.5 全局查询过滤器 624
 - 26.5.6 EF.Functions 625
 - 26.6 关系 625
 - 26.6.1 使用约定的关系 625
 - 26.6.2 显式加载相关数据 627
 - 26.6.3 即时加载相关数据 628
 - 26.6.4 使用注释的关系 628
 - 26.6.5 使用流利API 的关系 629
 - 26.6.6 根据约定的每个层次结构的表 630
 - 26.6.7 使用流利API 的每个层次结构中的表 632
 - 26.6.8 表的拆分 633
 - 26.6.9 拥有的实体 634
 - 26.7 保存数据 636
 - 26.7.1 用关系添加对象 636
 - 26.7.2 对象的跟踪 637
 - 26.7.3 更新对象 638
 - 26.7.4 更新未跟踪的对象 638
 - 26.7.5 批处理 639
 - 26.8 冲突的处理 640
 - 26.8.1 最后一个更改获胜 640
 - 26.8.2 第一个更改获胜 641

- 26.9 上下文池 644
- 26.10 使用事务 644
 - 26.10.1 使用隐式的事务 644
 - 26.10.2 创建显式的事务 646
- 26.11 迁移 647
 - 26.11.1 准备项目文件 647
 - 26.11.2 利用ASP.NET Core MVC 托管应用程序 648
 - 26.11.3 托管.NET Core 控制台应用程序 648
 - 26.11.4 创建迁移 649
 - 26.11.5 以编程方式应用迁移 651
 - 26.11.6 应用迁移的其他方法 652
- 26.12 小结 652
- 第27章 本地化 653
 - 27.1 全球市场 653
 - 27.2 System.Globalization 名称空间 654
 - 27.2.1 Unicode 问题 654
 - 27.2.2 区域性和区域 655
 - 27.2.3 使用区域性 658
 - 27.2.4 排序 663
 - 27.3 资源 664
 - 27.3.1 资源读取器和写入器 664
 - 27.3.2 使用资源文件生成器 665
 - 27.3.3 通过ResourceManager 使用资源文件 666
 - 27.3.4 System.Resources 名称空间 666
 - 27.4 使用ASP.NET Core 本地化 667
 - 27.4.1 注册本地化服务 667
 - 27.4.2 注入本地化服务 668
 - 27.4.3 区域性提供程序 668
 - 27.4.4 在ASP.NET Core 中使用资源 669
 - 27.4.5 使用控制器和视图进行本地化 670
 - 27.5 本地化UWP 674
 - 27.5.1 给UWP 使用资源 674
 - 27.5.2 使用多语言应用程序工具集进行本地化 675
 - 27.6 小结 677
- 第28章 测试 678
 - 28.1 概述 678
 - 28.2 使用MSTest 进行单元测试 679
 - 28.2.1 使用MSTest 创建单元测试 679
 - 28.2.2 运行单元测试 681
 - 28.2.3 使用MSTest 预期异常 682
 - 28.2.4 测试全部代码路径 683
 - 28.2.5 外部依赖 683
 - 28.3 使用xUnit 进行单元测试 685
 - 28.3.1 使用xUnit 和.NET Core 686
 - 28.3.2 创建Fact 属性 686
 - 28.3.3 创建Theory 特性 687
 - 28.3.4 使用Mocking 库 688
 - 28.4 实时单元测试 690
 - 28.5 使用EF Core 进行单元测试 692
 - 28.6 使用Windows 应用程序进行UI测试 693
 - 28.7 Web 集成、负载和性能测试 697
 - 28.7.1 ASP.NET Core 集成测试 697
 - 28.7.2 创建Web 测试 698
 - 28.7.3 运行Web 测试 700

- 28.8 小结 702
- 第29 章 跟踪、日志和分析 703
 - 29.1 诊断概述 703
 - 29.2 使用EventSource 跟踪 704
 - 29.2.1 EventSource 的简单用法 705
 - 29.2.2 跟踪工具 706
 - 29.2.3 派生自EventSource 707
 - 29.2.4 使用注释和EventSource 709
 - 29.2.5 创建事件清单模式 710
 - 29.2.6 使用活动ID 712
 - 29.3 创建自定义侦听器 714
 - 29.4 使用ILogger 接口编写日志 715
 - 29.4.1 配置提供程序 716
 - 29.4.2 使用作用域 717
 - 29.4.3 过滤 718
 - 29.4.4 配置日志记录 718
 - 29.4.5 使用没有依赖注入的ILogger 719
 - 29.5 使用Visual Studio App Center进行分析 720
 - 29.6 小结 722
- 第III 部分 Web 应用程序和服务
- 第30 章 ASP.NET Core 727
 - 30.1 概述 727
 - 30.2 Web 技术 728
 - 30.2.1 HTML 728
 - 30.2.2 CSS 729
 - 30.2.3 JavaScript 和TypeScript 729
 - 30.2.4 脚本库 729
 - 30.3 ASP.NET Web 项目 730
 - 30.3.1 启动 733
 - 30.3.2 示例应用程序 735
 - 30.4 添加客户端内容 736
 - 30.4.1 为客户端内容使用工具 737
 - 30.4.2 通过Bower 使用客户端库 738
 - 30.4.3 使用JavaScript 包管理器npm 739
 - 30.4.4 捆绑 739
 - 30.4.5 用webpack 打包 740
 - 30.5 请求和响应 741
 - 30.5.1 请求标题 742
 - 30.5.2 查询字符串 744
 - 30.5.3 编码 745
 - 30.5.4 表单数据 745
 - 30.5.5 cookie 746
 - 30.5.6 发送JSON 747
 - 30.6 依赖注入 747
 - 30.6.1 定义服务 748
 - 30.6.2 注册服务 748
 - 30.6.3 注入服务 748
 - 30.6.4 调用控制器 749
 - 30.7 简单的路由 749
 - 30.8 创建自定义的中间件 750
 - 30.9 会话状态 751
 - 30.10 用ASP.NET Core 配置 752
 - 30.10.1 读取配置 753
 - 30.10.2 修改配置提供程序 755

- 30.10.3 基于环境的不同配置 756
- 30.11 小结 757
- 第31 章 ASP.NET Core MVC 758
- 31.1 为ASP.NET Core MVC 建立服务 758
- 31.2 定义路由 760
 - 31.2.1 添加路由 760
 - 31.2.2 使用路由约束 761
- 31.3 创建控制器 761
 - 31.3.1 理解动作方法 762
 - 31.3.2 使用参数 762
 - 31.3.3 返回数据 762
 - 31.3.4 使用Controller 基类和POCO控制器 763
- 31.4 创建视图 765
 - 31.4.1 向视图传递数据 765
 - 31.4.2 Razor 语法 766
 - 31.4.3 创建强类型视图 766
 - 31.4.4 定义布局 767
 - 31.4.5 用部分视图定义内容 770
 - 31.4.6 使用视图组件 773
 - 31.4.7 在视图中使用依赖注入 774
 - 31.4.8 为多个视图导入名称空间 775
- 31.5 从客户端提交数据 775
 - 31.5.1 模型绑定器 777
 - 31.5.2 注解和验证 778
- 31.6 使用HTML Helper 779
 - 31.6.1 简单的Helper 779
 - 31.6.2 使用模型数据 779
 - 31.6.3 定义HTML 特性 780
 - 31.6.4 创建列表 780
 - 31.6.5 强类型化的Helper 781
 - 31.6.6 编辑器扩展 782
 - 31.6.7 实现模板 782
- 31.7 Tag Helper 783
 - 31.7.1 激活Tag Helper 783
 - 31.7.2 使用锚定Tag Helper 783
 - 31.7.3 使用Label Tag Helper 784
 - 31.7.4 使用Input Tag Helper 785
 - 31.7.5 使用表单进行验证 786
 - 31.7.6 environment Tag Helper 787
 - 31.7.7 创建自定义Tag Helper 788
 - 31.7.8 用Tag Helper 创建元素 790
- 31.8 实现动作过滤器 792
- 31.9 创建数据驱动的应用程序 793
 - 31.9.1 定义模型 794
 - 31.9.2 创建数据库 795
 - 31.9.3 创建服务 796
 - 31.9.4 创建控制器 798
 - 31.9.5 创建视图 800
- 31.10 实现身份验证和授权 803
 - 31.10.1 存储和检索用户信息 803
 - 31.10.2 启动身份系统 804
 - 31.10.3 执行用户注册 804
 - 31.10.4 设置用户登录 806
 - 31.10.5 验证用户的身份 807

31.10.6 使用Azure Active Directory 对用户进行身份验证 807

31.11 Razor 页面 812

31.11.1 创建一个Razor 页面项目 812

31.11.2 实现数据访问 813

31.11.3 使用内联代码 814

31.11.4 使用内联代码和页面模型 816

31.11.5 使用代码隐藏文件 817

31.11.6 页面参数 817

31.12 小结 818

第32 章 Web API 819

32.1 概述 819

32.2 创建服务 820

32.2.1 定义模型 821

32.2.2 创建服务 821

32.2.3 创建控制器 823

32.2.4 修改响应格式 824

32.2.5 REST 结果和状态码 825

32.3 创建异步服务 826

32.4 创建.NET 客户端 827

32.4.1 发送GET 请求 828

32.4.2 从服务中接收XML 832

32.4.3 发送POST 请求 833

32.4.4 发送PUT 请求 833

32.4.5 发送DELETE 请求 834

32.5 写入数据库 835

32.5.1 使用EF Core 835

32.5.2 创建数据访问服务 836

32.6 用OpenAPI 或Swagger 创建元数据 837

32.7 创建和使用OData服务 841

32.7.1 创建数据模型 842

32.7.2 创建数据库 843

32.7.3 OData 启动代码 844

32.7.4 创建OData 控制器 844

32.7.5 OData 查询 845

32.8 使用Azure Function 847

32.8.1 创建Azure Function 847

32.8.2 使用依赖注入容器 848

32.8.3 实现GET、POST 和PUT 请求 849

32.8.4 运行Azure Function 851

32.9 小结 852

第IV 部分 应用程序

第33 章 Windows 应用程序 855

33.1 Windows 应用程序简介 855

33.1.1 Windows 运行库 856

33.1.2 Hello, Windows 856

33.1.3 应用程序清单文件 857

33.1.4 应用程序启动 859

33.1.5 主页 859

33.2 XAML 861

33.2.1 XAML 标准 861

33.2.2 将元素映射到类 861

33.2.3 通过XAML 使用定制的.NET 类 862

33.2.4 将属性用作特性 863

33.2.5 将属性用作元素 863

- 33.2.6 依赖属性 864
- 33.2.7 创建依赖属性 864
- 33.2.8 值变更回调和事件 865
- 33.2.9 路由事件 866
- 33.2.10 附加属性 867
- 33.2.11 标记扩展 868
- 33.2.12 自定义标记扩展 869
- 33.2.13 条件XAML 870
- 33.3 控件 871
 - 33.3.1 框架派生的UI 元素 872
 - 33.3.2 控件派生的控件 875
 - 33.3.3 范围控件 881
 - 33.3.4 内容控件 882
 - 33.3.5 按钮 883
 - 33.3.6 项控件 884
 - 33.3.7 Flyout 控件 884
- 33.4 数据绑定 884
 - 33.4.1 用INotifyPropertyChanged 更改通知 885
 - 33.4.2 创建图书列表 886
 - 33.4.3 列表绑定 887
 - 33.4.4 把事件绑定到方法 887
 - 33.4.5 使用数据模板和数据模板选择器 888
 - 33.4.6 绑定简单对象 890
 - 33.4.7 值的转换 891
- 33.5 导航 892
 - 33.5.1 导航回最初的页面 892
 - 33.5.2 重写Page 类的导航 893
 - 33.5.3 在页面之间导航 894
 - 33.5.4 后退按钮 895
 - 33.5.5 Hub 896
 - 33.5.6 Pivot 898
 - 33.5.7 NavigationView 899
- 33.6 布局 902
 - 33.6.1 StackPanel 902
 - 33.6.2 Canvas 903
 - 33.6.3 Grid 903
 - 33.6.4 VariableSizedWrapGrid 904
 - 33.6.5 RelativePanel 906
 - 33.6.6 自适应触发器 906
 - 33.6.7 XAML 视图 909
 - 33.6.8 延迟加载 909
- 33.7 小结 910
- 第34 章 模式和XAML 应用程序 911
 - 34.1 使用 MWM 的原因 911
 - 34.2 定义 MWM 模式 912
 - 34.3 共享代码 913
 - 34.3.1 使用API 协定和通用Windows平台 913
 - 34.3.2 使用共享项目 915
 - 34.3.3 使用.NET 标准库 916
 - 34.4 示例解决方案 917
 - 34.5 模型 918
 - 34.5.1 实现变更通知 918
 - 34.5.2 使用Repository 模式 919
 - 34.6 服务 920

- 34.7 视图模型 921
 - 34.7.1 使用IEditableObject 923
 - 34.7.2 视图模型的具体实现 924
 - 34.7.3 命令 925
 - 34.7.4 服务、ViewModel 和依赖注入 926
- 34.8 视图 927
 - 34.8.1 从视图模型中打开对话框 930
 - 34.8.2 页面之间的导航 931
 - 34.8.3 自适应用户界面 933
 - 34.8.4 显示进度信息 935
 - 34.8.5 使用列表项中的操作 936
- 34.9 使用事件传递消息 938
- 34.10 使用框架 939
- 34.11 小结 940
- 第35 章 样式化Windows 应用程序 941
 - 35.1 样式设置 941
 - 35.2 形状 942
 - 35.3 几何图形 944
 - 35.3.1 使用段的几何图形 944
 - 35.3.2 使用PathMarkup 的几何图形 945
 - 35.4 变换 945
 - 35.4.1 缩放 945
 - 35.4.2 平移 946
 - 35.4.3 旋转 946
 - 35.4.4 倾斜 946
 - 35.4.5 组合变换和复合变换 946
 - 35.4.6 使用矩阵的变换 947
 - 35.5 画笔 947
 - 35.5.1 SolidColorBrush 947
 - 35.5.2 LinearGradientBrush 947
 - 35.5.3 ImageBrush 948
 - 35.5.4 AcrylicBrush 948
 - 35.5.5 RevealBrush 949
 - 35.6 样式和资源 949
 - 35.6.1 样式 949
 - 35.6.2 资源 951
 - 35.6.3 从代码中访问资源 952
 - 35.6.4 资源字典 952
 - 35.6.5 主题资源 953
 - 35.7 模板 954
 - 35.7.1 控件模板 955
 - 35.7.2 数据模板 958
 - 35.7.3 样式化ListView 959
 - 35.7.4 ListView 项的数据模板 960
 - 35.7.5 项容器的样式 960
 - 35.7.6 项面板 961
 - 35.7.7 列表视图的控件模板 961
 - 35.8 动画 962
 - 35.8.1 时间轴 962
 - 35.8.2 缓动函数 964
 - 35.8.3 关键帧动画 968
 - 35.8.4 过渡 969
 - 35.9 可视化状态管理器 971
 - 35.9.1 用控件模板预定义状态 972

- 35.9.2 定义自定义状态 973
- 35.9.3 设置自定义的状态 973
- 35.10 小结 974
- 第36 章 高级Windows 应用程序 975
- 36.1 概述 975
- 36.2 应用程序的生命周期 976
 - 36.2.1 应用程序的执行状态 976
 - 36.2.2 在页面之间导航 976
- 36.3 导航状态 978
 - 36.3.1 暂停应用程序 979
 - 36.3.2 激活暂停的应用程序 980
 - 36.3.3 测试暂停 980
 - 36.3.4 页面状态 981
- 36.4 共享数据 983
 - 36.4.1 共享源 983
 - 36.4.2 共享目标 986
- 36.5 应用程序服务 991
 - 36.5.1 创建模型 991
 - 36.5.2 为应用程序服务连接创建后台任务 992
 - 36.5.3 注册应用程序服务 993
 - 36.5.4 调用应用程序服务 994
- 36.6 高级的编译绑定 996
 - 36.6.1 已编译数据绑定的生命周期 996
 - 36.6.2 绑定到方法上 997
 - 36.6.3 用x:Bind 分阶段 998
- 36.7 使用文本 1002
 - 36.7.1 使用字体 1002
 - 36.7.2 内联和块元素 1004
 - 36.7.3 使用溢出区域 1005
- 36.8 上墨 1008
- 36.9 自动建议 1011
- 36.10 小结 1013
- 第37 章 Xamarin.Forms 1015
- 37.1 Xamarin 开发入门 1015
 - 37.1.1 用Android 架构Xamarin 1016
 - 37.1.2 用iOS 架构Xamarin 1016
 - 37.1.3 Xamarin.Forms 1017
- 37.2 Xamarin 开发工具 1018
 - 37.2.1 Android 1018
 - 37.2.2 iOS 1019
 - 37.2.3 Visual Studio 2017 1019
 - 37.2.4 Visual Studio for Mac 1019
 - 37.2.5 Visual Studio App Center 1020
- 37.3 Android 基础 1020
 - 37.3.1 活动 1021
 - 37.3.2 资源 1022
 - 37.3.3 显示列表 1022
 - 37.3.4 显示消息 1024
- 37.4 iOS 基础 1025
 - 37.4.1 iOS 应用程序结构 1025
 - 37.4.2 故事板 1026
 - 37.4.3 控制器 1028
 - 37.4.4 显示消息 1028
- 37.5 Xamarin. Forms 应用程序 1029

- 37.5.1 托管Xamarin 的Windows应用程序 1029
- 37.5.2 托管Xamarin 的Android 1030
- 37.5.3 托管Xamarin 的iOS 1031
- 37.5.4 共享的项目 1031
- 37.6 使用公共库 1032
- 37.7 控件层次结构 1032
- 37.8 页面 1033
- 37.9 导航 1034
- 37.10 布局 1035
- 37.11 视图 1037
- 37.12 数据绑定 1037
- 37.13 命令 1038
- 37.14 ListView 和ViewCell 1038
- 37.15 小结 1039
- 附赠章节电子版(请扫描封底二维码获取)
- 第1章 Composition 1
 - BC1.1 概述 1
 - BC1.2 Composition 库的体系结构 2
 - BC1.2.1 使用特性的Composition 3
 - BC1.2.2 基于约定的部件注册 8
 - BC1.3 定义协定 10
 - BC1.4 导出部件 13
 - BC1.4.1 创建部件 13
 - BC1.4.2 使用部件的部件 17
 - BC1.4.3 导出元数据 17
 - BC1.4.4 使用元数据进行惰性加载 19
 - BC1.5 导入部件 19
 - BC1.5.1 导入连接 22
 - BC1.5.2 部件的惰性加载 23
 - BC1.5.3 读取元数据 23
 - BC1.6 小结 25
- 第2章 XML 和JSON 26
 - BC2.1 数据格式 26
 - BC2.1.1 XML 27
 - BC2.1.2 .NET 支持的XML 标准 28
 - BC2.1.3 在框架中使用XML 28
 - BC2.1.4 JSON 29
 - BC2.2 读写流格式的XML 30
 - BC2.2.1 使用XmlReader 类读取XML 31
 - BC2.2.2 使用XmlWriter 类 33
 - BC2.3 在.NET 中使用DOM 34
 - BC2.3.1 使用XmlDocument 类读取 35
 - BC2.3.2 遍历层次结构 35
 - BC2.3.3 使用XmlDocument 插入节点 36
 - BC2.4 使用XPathNavigator 类 37
 - BC2.4.1 XPathDocument 类 37
 - BC2.4.2 XPathNavigator 类 37
 - BC2.4.3 XPathNodeIterator 类 38
 - BC2.4.4 使用XPath 导航XML 38
 - BC2.4.5 使用XPath 评估 39
 - BC2.4.6 用XPath 修改XML 39
 - BC2.5 在XML 中序列化对象 40
 - BC2.5.1 序列化简单对象 40
 - BC2.5.2 序列化一个对象树 42

- BC2.5.3 没有特性的序列化 44
- BC2.6 LINQ to XML 46
 - BC2.6.1 XDocument 对象 46
 - BC2.6.2 XElement 对象 47
 - BC2.6.3 XNamespace 对象 47
 - BC2.6.4 XComment 对象 49
 - BC2.6.5 XAttribute 对象 49
 - BC2.6.6 使用LINQ 查询XML 文档 50
 - BC2.6.7 查询动态的XML 文档 50
 - BC2.6.8 转换为对象 52
 - BC2.6.9 转换为XML 52
- BC2.7 JSON 53
 - BC2.7.1 创建JSON 53
 - BC2.7.2 转换对象 54
 - BC2.7.3 序列化对象 55
 - BC2.7.4 遍历JSON 节点 55
- BC2.8 小结 56
- 第3 章 WebHooks 和SignalR 57
 - BC3.1 概述 57
 - BC3.2 WebSockets 58
 - BC3.2.1 WebSockets 服务器 58
 - BC3.2.2 WebSockets 客户端 60
 - BC3.3 使用SignalR 的简单聊天程序 62
 - BC3.3.1 创建集线器 62
 - BC3.3.2 用HTML 和JavaScript 创建客户端 63
 - BC3.3.3 创建SignalR .NET 客户端 65
 - BC3.4 分组连接 68
 - BC3.4.1 用分组扩展集线器 68
 - BC3.4.2 用分组扩展Windows 客户端 69
 - BC3.5 WebHooks 的体系结构 71
 - BC3.6 创建Dropbox 和GitHub 接收器 72
 - BC3.6.1 创建Web 应用程序 73
 - BC3.6.2 为Dropbox 和GitHub 配置WebHooks 73
 - BC3.6.3 实现处理程序 73
 - BC3.6.4 用Dropbox 和GitHub 配置应用程序 76
 - BC3.6.5 运行应用程序 77
 - BC3.7 小结 77
- 第4 章 机器人和认知服务 79
 - BC4.1 机器人的定义 79
 - BC4.2 创建对话框机器人 80
 - BC4.2.1 配置状态服务 81
 - BC4.2.2 接收机器人消息 82
 - BC4.2.3 定义对话框 83
 - BC4.2.4 使用PromptDialog 85
 - BC4.3 为对话框使用Form Flow 88
 - BC4.4 创建英雄卡 89
 - BC4.5 机器人和LUIS 91
 - BC4.5.1 定义意图和话语 92
 - BC4.5.2 访问LUIS 中的建议 95
 - BC4.5.3 使用带有活动检查的Form Flow 96
 - BC4.6 小结 96
- 第5 章 Windows 应用程序的更多特性 97
 - BC5.1 概述 97
 - BC5.2 相机 98

BC5.3 Geolocation 和MapControl 99
BC5.3.1 使用MapControl 99
BC5.3.2 使用Geolocator 定位信息 102
BC5.3.3 街景地图 103
BC5.3.4 继续请求位置信息 104
BC5.4 传感器 105
BC5.4.1 光线 106
BC5.4.2 罗盘 107
BC5.4.3 加速计 107
BC5.4.4 倾斜计 108
BC5.4.5 陀螺仪 109
BC5.4.6 方向 109
BC5.4.7 Rolling Marble 示例 110
BC5.5 小结 112
• • • • • ([收起](#))

[C#高级编程\(第11版\)_下载链接1](#)

标签

C

#C

#.NET

软件开发

编程

計算機

Programming

评论

第一轮，刷完第一篇

虽然这本书在业内久负盛名，但是我觉得机翻的书还不如不出版。

怎么说呢，这本书不适合新手，更像一本工具字典，面面俱到，到到浅尝辄止，不晓得是原书就垃圾还是翻译垃圾，书上的很多代码无法复现，特别是.net core以后的部分，简直就是操蛋，把代码贴出来也不注明引用的哪些类库，到windows应用部分可以说简直就是敷衍了，代码很难复现不说，甚至下载到源码都是错误的，购买谨慎呐

[C#高级编程\(第11版\) 下载链接1](#)

书评

可以说是不错的入门书。虽然写着高级，但是也就入门级最大的问题在于中文翻译，很多地方读得莫名其妙。直到前几天找到了原版的PDF，看过原文后才恍然大悟。早期看得就忘了，只记得seal方法那里翻译有误。这几天看得翻译有问题的地方如下： 1.P123 IEnumerator接口的方法...

由于.net整个框架体系实在是太庞大了，所以光靠一本书来写，这完全是不可能的。这可能也是我对于这本书的唯一的遗憾吧。
不过，如果说作为对.net框架的一个整体的通览的话，这本书确是最合适不过的了。

我学C#的第一本书，因为那个时候刚入门编程，很多概念不懂，所以学得很困难，但是坚持下来了，里面的内容页基本都学会了。从C#到CLR从winform到WPF，从asp.net webform到asp.net MVC到Silverlight。从ado.net到linq到entity framework。看这本书可以大致了解.net的整个框架，但...

终于拿到书了！翻了几下挺高兴的。我自己喜欢学编程，就买来想好好再学下。说实话

，书是随便买的，因为那么多也一下子看不出哪本好哪本不太好。才看前两章还行，再往后翻就觉得吃力了。还好在网上找到一个专门教编程课程的猎豹网校，在那里试听了一下，觉得有老师教和带，看着...

都是些基础的东西，讲的比较全，但是比较宽泛，没有什么深入的探讨，对于初学者来说是一本不错的入门书籍。最好是先阅读《C#入门经典(第4版)》，这样形成一个循序渐进的过程
都是些基础的东西，讲的比较全，但是比较宽泛，没有什么深入的探讨，对于初学者来说是一本不错的入...

很厚的书,讲的什么都有,可谓面面俱到,但什么都没深入.初学者无法抓主语言精髓.对于非初学者又无参考价值.推荐买书,买那些只将一个领域的书,那样的书往往收获很多.最讨厌这种凑字数,毫无技术含量还打微软旗号的书,您好意思么?

因为做项目比较紧，这本书读了近四个月，差不多读完了，感觉还行。c#是博大精深的，指望一本书根本不行，不管它有多厚，这本书只是提供了c#的整体框架，很多地方讲的不细，只是一笔带过。但是知道了整体框架，以后真的遇到的实际项目中的问题，就可以做到有的放矢，具体的查找...

学.net时看的第一本书，面比较广，但是每点都不太深入，给我最深的印象是翻译的太晦涩，不太适合初学者。

打算面面俱到，但又不深入讲每一个细节，比入门书深，比MSDN潜。最关键的是除了罗列语言细节，没有什么吸引我的地方。作为字典还是不错的，作为学习，我认为不够系统不够深入。

最大的特点就是，太厚了……我看的第几版忘记了，不过每一版都是加上些新版本.net framework的新技术，所以很全。
如果是初学，要想从头看到尾，这是一个十分艰巨的任务，我倒宁愿把新概念英语第四册背一遍。老师说这个书是用来查的，查锤子呀，后面每一章的知识都要前面的...

9.5成新。低于半价转让，有意联系：13811786703。
北京的朋友可以约个地点，见面交易！

刚开始几章还可以，但是看到中间以后，感觉这本书的翻译人员纯粹是在骗钱，有些地方的翻译，只是比用有道字典的翻译稍微强一些。而且感觉很多地方的翻译，是由根本不懂技术的人来翻译的，完全的词不达意。本来这本书的原著英文版是非常经典的，但是被这帮混饭吃的翻译给弄成垃...

很多地方，要对照着英文原版才能看懂，翻译的完全词不达意。可惜了原版的经典。在哪找一帮学生来凑数啊。从一些细节上可以看出，很多章节，翻译者本身并不懂。完全是一个技术外行来翻译这本技术书啊。
很多地方，要对照着英文原版才能看懂，翻译的完全词不达意。可惜了原版的经...

Comprehensive, advanced coverage of C# 5.0 and .NET 4.5.1 Whether you're a C# guru or transitioning from C/C++, staying up to date is critical to your success. Professional C# 5.0 and .NET 4.5.1 is your go-to guide for navigating the programming environm...

真的体会不出高级在哪。入门书。但是，太厚。以至于几百页都是电子版。不适合阅读。但当工具书又太浅了，没有多少参考意义。这还短？ 这还短？ 这还短？ 这还短？ 这还短？ 这还短？ 这还短？ 这还短？ 这还短？ ...

[C#高级编程\(第11版\) 下载链接1](#)