Оглавление

Об авторе ... 27

О научном редакторе ... 29

Предисловие ... 30
- Структура книги ... 30
- Необходимое программное обеспечение ... 34
- Условные обозначения ... 34
- От издательства ... 35

Глава 1. Привет, C#! Здравствуй, .NET Core! ... 36
- Настройка среды разработки ... 37
  - Использование Visual Studio Code для разработки кросс-платформенных приложений ... 37
  - Использование GitHub Codespaces для разработки в облаке ... 38
  - Использование Visual Studio 2019 для разработки Windows-приложений ... 38
  - Использование Visual Studio на Mac для разработки мобильных приложений ... 39
  - Рекомендуемые инструменты и операционные системы ... 39
  - Кросс-платформенное развертывание ... 40
  - Знакомство с версиями Microsoft Visual Studio Code ... 40
  - Скачивание и установка среды Visual Studio Code ... 42
  - Установка расширений ... 43
- Знакомство с .NET ... 43
  - Обзор .NET Framework ... 43
  - Проекты Mono и Xamarin ... 44
  - Обзор .NET Core ... 45
  - Обзор .NET 5 и последующих версий .NET ... 45

Поддержка .NET Core ... 46
Версии .NET Runtime и .NET SDK ... 47
Удаление старых версий .NET ... 48
В чем особенность .NET Core ... 49
Обзор .NET Standard ... 50
Платформы .NET в изданиях этой книги ... 51
Знакомство с промежуточным языком ... 52
Сравнение технологий .NET ... 53
Разработка консольных приложений с использованием Visual Studio Code ... 53
Написание кода с помощью Visual Studio Code ... 53
Компиляция и запуск кода с использованием инструмента командной строки dotnet ... 56
Написание программ верхнего уровня ... 56
Скачивание кода решения из репозитория GitHub ... 57
Использование системы Git в Visual Studio Code ... 58
Клонирование репозитория с примерами из книги ... 58
Поиск справочной информации ... 59
Знакомство с Microsoft Docs ... 59
Получение справки для инструмента dotnet ... 59
Получение определений типов и их элементов ... 60
Ищем ответы на Stack Overflow ... 62
Поисковая система Google ... 62
Подписка на официальный блог .NET ... 63
Видео от Скотта Хансельмана ... 63
Практические задания ... 63
Упражнение 1.1. Проверочные вопросы ... 63
Упражнение 1.2. Практическое задание ... 64
Упражнение 1.3. Дополнительные ресурсы ... 64
Резюме ... 65

Глава 2. Говорим на языке C# ... 66
Введение в C# ... 66
Обзор версий языка и их функций ... 67
Версии компилятора C# ... 71
Включение версии компилятора на определенном языке ... 72
Основы языка C# ... 74
Грамматика языка C# ... 75
Терминология языка C# ... 76
Изменение цветовой схемы синтаксиса ... 77
Сравнение языков программирования с естественными языками ... 77

Работа с переменными ... 82
Присвоение переменным имен ... 82
Хранение текста ... 83
Хранение чисел ... 84
Хранение логических значений ... 90
Использование рабочих областей Visual Studio Code ... 90
Хранение объектов любого типа ... 91
Хранение данных динамического типа ... 93
Локальные переменные ... 93
Использование целевого типа выражения new для создания экземпляров объектов ... 95
Получение значений по умолчанию для типов ... 95
Хранение нескольких значений ... 96
Работа со значениями null ... 97
Создание значимого типа, допускающего значение null ... 97
Включение ссылочных типов, допускающих и не допускающих значение null ... 99
Объявление переменных и параметров, не допускающих значение null ... 100
Проверка на null ... 102
Дальнейшее изучение консольных приложений ... 103
Отображение вывода пользователю ... 103
Форматирующие строки ... 104
Получение пользовательского ввода ... 106
Импорт пространства имен ... 106
Упрощение работы с командной строкой ... 107
Получение клавиатурного ввода от пользователя ... 107
Чтение аргументов ... 108
Настройка параметров с помощью аргументов ... 110
Работа с платформами, не поддерживающими некоторые API ... 111
Практические задания ... 112
Упражнение 2.1. Проверочные вопросы ... 112
Упражнение 2.2. Практическое задание — числовые размеры и диапазоны ... 112
Упражнение 2.3. Дополнительные ресурсы ... 113
Резюме ... 114

Глава 3. Управление потоком исполнения и преобразование типов ... 115
Работа с переменными ... 115
Унарные операции ... 116
Арифметические бинарные операции ... 117
Операция присваивания ... 119

Логические операции ... 119
Условные логические операции ... 120
Побитовые операции и операции побитового сдвига ... 122
Прочие операции ... 123
Операторы выбора ... 123
Ветвление с помощью оператора if ... 124
Почему в операторах if необходимы фигурные скобки ... 125
Сопоставление шаблонов с помощью операторов if ... 125
Ветвление с помощью оператора switch ... 126
Сопоставление шаблонов с помощью оператора switch ... 127
Упрощение операторов switch с помощью выражений switch ... 129
Операторы цикла ... 130
Оператор while ... 130
Оператор do ... 131
Оператор for ... 132
Оператор foreach ... 132
Приведение и преобразование типов ... 133
Явное и неявное приведение типов ... 134
Использование типа System.Convert ... 136
Округление чисел ... 137
Преобразование значения любого типа в строку ... 138
Преобразование двоичного (бинарного) объекта в строку ... 139
Разбор строк для преобразования в числа или значения даты и времени ... 140
Обработка исключений при преобразовании типов ... 142
Проверка переполнения ... 145
Практические задания ... 148
Упражнение 3.1. Проверочные вопросы ... 148
Упражнение 3.2. Циклы и переполнение ... 149
Упражнение 3.3. Циклы и операторы ... 149
Упражнение 3.4. Обработка исключений ... 150
Упражнение 3.5. Проверка знания операций ... 150
Упражнение 3.6. Дополнительные ресурсы ... 151
Резюме ... 151

Глава 4. Разработка, отладка и тестирование функций ... 152
Написание функций в языке C# ... 152
Написание функции для таблицы умножения ... 153
Функции, возвращающие значение ... 155
Написание математических функций ... 157
Документирование функций с помощью XML-комментариев ... 161

Использование лямбда-выражений в реализациях функций ... 163
Отладка в процессе разработки ... 166
Преднамеренное добавление ошибок в код ... 166
Установка точек останова ... 167
Навигация с помощью панели средств отладки ... 169
Панели отладки ... 169
Пошаговое выполнение кода ... 170
Настройка точек останова ... 171
Регистрация событий во время разработки и выполнения проекта ... 173
Работа с типами Debug и Trace ... 173
Прослушиватель трассировки ... 174
Настройка прослушивателей трассировки ... 175
Переключение уровней трассировки ... 176
Модульное тестирование функций ... 179
Создание библиотеки классов, требующей тестирования ... 180
Разработка модульных тестов ... 181
Выполнение модульных тестов ... 182
Практические задания ... 183
Упражнение 4.1. Проверочные вопросы ... 183
Упражнение 4.2. Функции, отладка и модульное тестирование ... 184
Упражнение 4.3. Дополнительные ресурсы ... 184
Резюме ... 185

Глава 5. Создание пользовательских типов с помощью объектно-ориентированного программирования ... 186
Коротко об объектно-ориентированном программировании ... 186
Разработка библиотек классов ... 187
Создание библиотек классов ... 188
Определение классов ... 189
Создание экземпляров классов ... 190
Управление несколькими файлами ... 192
Работа с объектами ... 192
Хранение данных в полях ... 193
Определение полей ... 193
Модификаторы доступа ... 194
Установка и вывод значений полей ... 195
Хранение значения с помощью типа-перечисления ... 196
Хранение группы значений с помощью типа enum ... 197
Хранение нескольких значений с помощью коллекций ... 199
Создание статического поля ... 200

Создание константного поля ... 201
Создание поля только для чтения ... 202
Инициализация полей с помощью конструкторов ... 203
Установка значения поля с использованием литерала для значения по умолчанию ... 204
Запись и вызов методов ... 206
Возвращение значений из методов ... 206
Возвращение нескольких значений с помощью кортежей ... 207
Определение и передача параметров в методы ... 210
Перегрузка методов ... 211
Передача необязательных параметров и именованных аргументов ... 211
Управление передачей параметров ... 213
Ключевое слово ref ... 215
Разделение классов с помощью ключевого слова partial ... 215
Управление доступом с помощью свойств и индексаторов ... 216
Определение свойств только для чтения ... 216
Определение изменяемых свойств ... 217
Определение индексаторов ... 219
Сопоставление шаблонов с объектами ... 220
Создание и работа с библиотеками классов .NET 5 ... 220
Определение пассажиров при полете ... 221
Изменения сопоставления с шаблоном в C# 9 ... 222
Работа с записями ... 223
Свойства только для инициализации ... 223
Записи ... 224
Упрощение членов данных ... 225
Позиционирование записей ... 226
Практические задания ... 227
Упражнение 5.1. Проверочные вопросы ... 227
Упражнение 5.2. Дополнительные ресурсы ... 227
Резюме ... 228

Глава 6. Реализация интерфейсов и наследование классов ... 229
Настройка библиотеки классов и консольного приложения ... 229
Упрощение методов ... 231
Реализация функционала с помощью методов ... 232
Реализация функционала с помощью операций ... 234
Реализация функционала с помощью локальных функций ... 235
Вызов и обработка событий ... 236
Вызов методов с помощью делегатов ... 236

Определение и обработка делегатов ... 237
Определение и обработка событий ... 239
Реализация интерфейсов ... 240
Универсальные интерфейсы ... 240
Сравнение объектов при сортировке ... 241
Сравнение объектов с помощью отдельных классов ... 243
Определение интерфейсов с реализациями по умолчанию ... 245
Обеспечение безопасности многократного использования типов с помощью дженериков ... 247
Работа с типами-дженериками ... 248
Работа с методами-дженериками ... 250
Управление памятью с помощью ссылочных типов и типов значений ... 251
Работа со структурами ... 252
Освобождение неуправляемых ресурсов ... 253
Обеспечение вызова метода Dispose ... 256
Наследование классов ... 256
Расширение классов ... 257
Сокрытие членов класса ... 257
Переопределение членов ... 259
Предотвращение наследования и переопределения ... 260
Полиморфизм ... 260
Приведение в иерархиях наследования ... 262
Неявное приведение ... 262
Явное приведение ... 262
Обработка исключений приведения ... 263
Наследование и расширение типов .NET ... 264
Наследование исключений ... 265
Расширение типов при невозможности наследования ... 266
Практические задания ... 269
Упражнение 6.1. Проверочные вопросы ... 269
Упражнение 6.2. Создание иерархии наследования ... 269
Упражнение 6.3. Дополнительные ресурсы ... 270
Резюме ... 271

Глава 7. Описание и упаковка типов .NET ... 272
Введение в .NET 5 ... 272
.NET Core 1.0 ... 273
.NET Core 1.1 ... 274
.NET Core 2.0 ... 274
.NET Core 2.1 ... 274

.NET Core 2.2 ... 275
.NET Core 3.0 ... 275
.NET 5.0 ... 276
Повышение производительности с .NET Core 2.0 до .NET 5 ... 276
Использование компонентов .NET ... 277
Сборки, пакеты и пространства имен ... 277
Импорт пространства имен для использования типа ... 281
Связь ключевых слов языка C# с типами .NET ... 281
Создание кросс-платформенных библиотек классов при помощи .NET Standard ... 283
Создание библиотеки классов .NET Standard 2.0 ... 284
Публикация и развертывание ваших приложений ... 285
Разработка консольного приложения для публикации ... 286
Команды dotnet ... 287
Публикация автономного приложения ... 288
Публикация однофайлового приложения ... 289
Уменьшение размера приложений с помощью обрезки приложений ... 290
Декомпиляция сборок ... 291
Упаковка библиотек для распространения через NuGet ... 295
Ссылка на пакет NuGet ... 295
Упаковка библиотеки для NuGet ... 297
Тестирование пакета ... 299
Перенос приложений с .NET Framework на .NET 5 ... 301
Что означает перенос ... 301
Стоит ли переносить ... 302
Сравнение .NET Framework и .NET 5 ... 302
Анализатор переносимости .NET ... 303
Использование библиотек, не скомпилированных для .NET Standard ... 303
Практические задания ... 305
Упражнение 7.1. Проверочные вопросы ... 305
Упражнение 7.2. Дополнительные ресурсы ... 305
Резюме ... 306

Глава 8. Работа с распространенными типами .NET ... 307
Работа с числами ... 307
Большие целые числа ... 308
Комплексные числа ... 309
Работа с текстом ... 310
Извлечение длины строки ... 310
Извлечение символов строки ... 310

Разделение строк ... 311
Извлечение фрагмента строки ... 311
Поиск содержимого в строках ... 312
Конкатенация строк, форматирование и прочие члены типа string ... 313
Эффективное создание строк ... 314
Сопоставление шаблонов с использованием регулярных выражений ... 314
Проверка цифр, введенных как текст ... 315
Синтаксис регулярных выражений ... 316
Примеры регулярных выражений ... 317
Разбивка сложных строк, разделенных запятыми ... 318
Улучшение производительности регулярных выражений ... 319
Хранение данных с помощью коллекций ... 319
Общие свойства коллекций ... 320
Выбор коллекции ... 322
Работа со списками ... 324
Работа со словарями ... 325
Сортировка коллекций ... 326
Использование специализированных коллекций ... 327
Использование неизменяемых коллекций ... 327
Работа с интервалами, индексами и диапазонами ... 328
Управление памятью с помощью интервалов ... 328
Идентификация позиций с типом Index ... 329
Идентификация диапазонов с помощью типа Range ... 329
Использование индексов и диапазонов ... 330
Работа с сетевыми ресурсами ... 331
Работа с URI, DNS и IP-адресами ... 331
Проверка соединения с сервером ... 332
Работа с типами и атрибутами ... 334
Версии сборок ... 334
Чтение метаданных сборки ... 335
Создание пользовательских атрибутов ... 337
Еще немного об отражении ... 339
Работа с изображениями ... 340
Интернационализация кода ... 342
Обнаружение и изменение региональных настроек ... 342
Обработка часовых поясов ... 344
Практические задания ... 345
Упражнение 8.1. Проверочные вопросы ... 345
Упражнение 8.2. Регулярные выражения ... 345

Упражнение 8.3. Методы расширения 346
Упражнение 8.4. Дополнительные ресурсы 346
Резюме 347

Глава 9. Работа с файлами, потоками и сериализация 348
Управление файловой системой 348
Работа с различными платформами и файловыми системами 348
Управление дисками 350
Управление каталогами 351
Управление файлами 353
Управление путями 355
Извлечение информации о файле 356
Контроль работы с файлами 357
Чтение и запись с помощью потоков 358
Запись в текстовые потоки 360
Запись в XML-потоки 361
Освобождение файловых ресурсов 363
Сжатие потоков 365
Сжатие с помощью алгоритма Бротли 367
Высокопроизводительные потоки с использованием конвейеров 369
Асинхронные потоки 369
Кодирование и декодирование текста 369
Преобразование строк в последовательности байтов 370
Кодирование и декодирование текста в файлах 372
Сериализация графов объектов 373
XML-сериализация 373
Генерация компактного XML 376
XML-десериализация 377
JSON-сериализация 378
Высокопроизводительный процессор JSON 379
Практические задания 381
Упражнение 9.1. Проверочные вопросы 381
Упражнение 9.2. XML-сериализация 382
Упражнение 9.3. Дополнительные ресурсы 382
Резюме 383

Глава 10. Защита данных и приложений 384
Терминология безопасности 385
Ключи и их размеры 385
Векторы инициализации и размеры блоков 386

Соль ... 386
Генерация ключей и векторов инициализации ... 387
Шифрование и дешифрование данных ... 387
Симметричное шифрование с помощью алгоритма AES ... 388
Хеширование данных ... 393
Хеширование с помощью алгоритма SHA256 ... 393
Подписывание данных ... 396
Подписывание с помощью алгоритмов SHA256 и RSA ... 397
Генерация случайных чисел ... 400
Генерация случайных чисел для игр ... 400
Генерация случайных чисел для криптографии ... 401
Криптография: что нового ... 402
Аутентификация и авторизация пользователей ... 403
Реализация аутентификации и авторизации ... 405
Защита приложения ... 408
Практические задания ... 409
Упражнение 10.1. Проверочные вопросы ... 409
Упражнение 10.2. Защита данных с помощью шифрования и хеширования ... 410
Упражнение 10.3. Дешифрование данных ... 410
Упражнение 10.4. Дополнительные ресурсы ... 410
Резюме ... 411

Глава 11. Работа с базами данных с помощью Entity Framework Core ... 412
Современные базы данных ... 412
Введение в Entity Framework ... 413
Entity Framework Core ... 414
Использование образца реляционной базы данных ... 414
Создание образца базы данных Northwind для SQLite ... 416
Управление образцом базы данных Northwind в SQLiteStudio ... 417
Настройка EF Core ... 418
Выбор поставщика данных Entity Framework Core ... 418
Настройка инструмента dotnet-ef ... 419
Подключение к базе данных ... 420
Определение моделей EF Core ... 420
Соглашения EF Core ... 421
Атрибуты аннотаций Entity Framework Core ... 421
Entity Framework Core Fluent API ... 423
Заполнение таблиц базы данных ... 423
Создание модели Entity Framework Core ... 423

Создание моделей с использованием существующей базы данных 428
Запрос данных из моделей EF Core 433
Фильтрация включенных сущностей 434
Фильтрация и сортировка товаров 436
Получение сгенерированного SQL-кода 437
Логирование в EF Core 438
Теги запросов 442
Сопоставление с образцом с помощью оператора Like 442
Определение глобальных фильтров 443
Схемы загрузки данных при использовании EF Core 444
Жадная загрузка элементов 444
Использование ленивой загрузки 445
Явная загрузка элементов 446
Управление данными с помощью EF Core 449
Добавление элементов 449
Обновление элементов 450
Удаление элементов 451
Пулы соединений с базами данных 452
Транзакции 453
Определение явной транзакции 454
Практические задания 455
Упражнение 11.1. Проверочные вопросы 455
Упражнение 11.2. Экспорт данных с помощью различных форматов сериализации 456
Упражнение 11.3. Изучение документации EF Core 456
Резюме 456

Глава 12. Создание запросов и управление данными с помощью LINQ 457
Написание запросов LINQ 457
Расширение последовательностей с помощью класса Enumerable 458
Фильтрация элементов с помощью метода Where 459
Сортировка элементов 463
Фильтрация по типу 465
Работа с множествами с помощью LINQ 466
Использование LINQ с EF Core 468
Создание модели EF Core 468
Фильтрация и сортировка последовательностей 471
Проецирование последовательностей в новые типы 472
Объединение и группировка 473
Агрегирование последовательностей 477

Подслащение синтаксиса LINQ с помощью синтаксического сахара 478
Использование нескольких потоков и параллельного LINQ 480
Разработка приложения с помощью нескольких потоков....480
Создание собственных методов расширения LINQ....483
Работа с LINQ to XML....486
Генерация XML с помощью LINQ to XML....487
Чтение XML с помощью LINQ to XML....487
Практические задания 488
Упражнение 12.1. Проверочные вопросы 489
Упражнение 12.2. Создание запросов LINQ....489
Упражнение 12.3. Дополнительные ресурсы....490
Резюме 490

Глава 13. Улучшение производительности и масштабируемости с помощью многозадачности....491
Процессы, потоки и задачи....491
Мониторинг производительности и использования ресурсов 493
Оценка эффективности типов....493
Мониторинг производительности и использования памяти....494
Реализация класса Recorder 495
Асинхронное выполнение задач 499
Синхронное выполнение нескольких действий....499
Асинхронное выполнение нескольких действий с помощью задач 501
Ожидание выполнения задач....502
Задачи продолжения....503
Вложенные и дочерние задачи 505
Синхронизация доступа к общим ресурсам....506
Доступ к ресурсу из нескольких потоков 507
Применение к ресурсу взаимоисключающей блокировки....508
Оператор блокировки и избежание взаимоблокировок....509
Синхронизация событий 511
Выполнение атомарных операций 512
Использование других типов синхронизации 513
Ключевые слова async и await 513
Увеличение скорости отклика консольных приложений 514
Увеличение скорости отклика GUI-приложений....515
Улучшение масштабируемости клиент-серверных приложений....515
Распространенные типы, поддерживающие многозадачность 516
Ключевое слово await в блоках catch 516
Работа с асинхронными потоками 516

Практические задания 518
Упражнение 13.1. Проверочные вопросы 518
Упражнение 13.2. Дополнительные ресурсы 518
Резюме 519

Глава 14. Практическое применение C# и .NET 520
Модели приложений для C# и .NET 520
Разработка сайтов с помощью ASP.NET Core 521
Разработка сайтов с использованием системы управления веб-контентом (веб-содержимым) 521
Веб-приложения 522
Создание и использование веб-сервисов 523
Разработка интеллектуальных приложений 523
Нововведения ASP.NET Core 524
ASP.NET Core 1.0 524
ASP.NET Core 1.1 524
ASP.NET Core 2.0 524
ASP.NET Core 2.1 525
ASP.NET Core 2.2 525
ASP.NET Core 3.0 526
ASP.NET Core 3.1 526
Blazor WebAssembly 3.2 527
ASP.NET Core 5.0 527
SignalR 528
Blazor 529
JavaScript и другие технологии 530
Silverlight — использование C# и .NET через плагин 530
WebAssembly — база для Blazor 530
Blazor — на стороне сервера или клиента 531
Дополнительные материалы 531
Разработка кросс-платформенных мобильных и настольных Windows-приложений 532
Разработка настольных Windows-приложений с использованием устаревших технологий 533
Разработка модели данных объекта для Northwind 534
Разработка библиотеки классов для сущностных моделей Northwind 535
Создание моделей сущностей с использованием dotnet-ef 535
Улучшение сопоставления классов и таблиц вручную 537
Создание библиотеки классов для контекста базы данных Northwind 538
Резюме 541

Глава 15. Разработка сайтов с помощью ASP.NET Core Razor Pages 542
Веб-разработка 542
Протокол передачи гипертекста 542
Клиентская веб-разработка 546
Обзор ASP.NET Core 546
Классический ASP.NET против современного ASP.NET Core 548
Разработка проекта ASP.NET Core 548
Тестирование и защита сайта 550
Управление средой хостинга 554
Включение статических файлов и файлов по умолчанию 555
Технология Razor Pages 558
Включение Razor Pages 558
Определение Razor Pages 558
Использование общих макетов в Razor Pages 560
Использование файлов с выделенным кодом в Razor Pages 563
Использование Entity Framework Core совместно с ASP.NET Core 565
Настройка Entity Framework Core в виде сервиса 565
Управление данными с помощью страниц Razor 567
Применение библиотек классов Razor 569
Создание библиотеки классов Razor 569
Отключение компактных папок 569
Реализация функции сотрудников с помощью EF Core 571
Реализация частичного представления для отображения одного сотрудника 573
Использование библиотек классов Razor 574
Настройка служб и конвейера HTTP-запросов 575
Регистрация служб 576
Конфигурация конвейера HTTP-запросов 578
Простой проект сайта ASP.NET Core 582
Практические задания 583
Упражнение 15.1. Проверочные вопросы 583
Упражнение 15.2. Веб-приложение, управлямое данными 584
Упражнение 15.3. Создание веб-страниц для консольных приложений 584
Упражнение 15.4. Дополнительные ресурсы 584
Резюме 585

Глава 16. Разработка сайтов с использованием паттерна MVC 586
Настройка сайта ASP.NET Core MVC 586
Создание и изучение сайтов ASP.NET Core MVC 587
Обзор сайта ASP.NET Core MVC 590
Обзор базы данных ASP.NET Core Identity 592

Изучение сайта ASP.NET Core MVC 592
Запуск ASP.NET Core MVC 592
Маршрутизация по умолчанию 595
Контроллеры и действия 596
Соглашение о пути поиска представлений 597
Модульное тестирование MVC 598
Фильтры 598
Сущности и модели представлений 600
Представления 602
Добавление собственного функционала на сайт ASP.NET Core MVC 605
Определение пользовательских стилей 606
Настройка категории изображений 606
Синтаксис Razor 606
Определение типизированного представления 607
Тестирование измененной главной страницы 610
Передача параметров с помощью значения маршрута 611
Привязка моделей 613
Проверка модели 617
Методы класса-помощника для представления 620
Отправка запросов в базу данных и использование шаблонов отображения 621
Улучшение масштабируемости с помощью асинхронных задач 623
Использование других шаблонов проектов 625
Установка дополнительных шаблонов 626
Практические задания 627
Упражнение 16.1. Проверочные вопросы 627
Упражнение 16.2. Реализация MVC для страницы, содержащей сведения о категориях 628
Упражнение 16.3. Улучшение масштабируемости за счет понимания и реализации асинхронных методов действий 628
Упражнение 16.4. Дополнительные ресурсы 628
Резюме 629

Глава 17. Разработка сайтов с помощью системы управления контентом (CMS) 630
Преимущества CMS 630
Основные функции CMS 631
Возможности корпоративной CMS 632
Платформы CMS 632
Piranha CMS 633
Библиотеки с открытым исходным кодом и лицензирование 633
Создание веб-приложения с помощью Piranha CMS 634

Изучение сайта Piranha CMS 635
Редактирование содержимого сайта и страницы 636
Создание новой страницы верхнего уровня 641
Создание новой дочерней страницы 642
Обзор архива блога 643
Комментирование постов и страниц 644
Аутентификация и авторизация 646
Изучение конфигурации 648
Тестирование нового контента 649
Маршрутизация 650
Мультимедиа 652
Сервис приложения 653
Типы контента 654
Стандартные блоки 660
Типы компонентов и стандартных блоков 660
Определение компонентов, типов контента и шаблонов 662
Определение пользовательских шаблонов контента для типов контента 667
Настройка запуска и импорта из базы данных 670
Создание контента с использованием шаблона проекта 673
Тестирование сайта Northwind CMS 674
Загрузка изображений и создание корня каталога 674
Импорт контента (содержимого) страниц категорий и товаров 675
Управление контентом (содержимым) каталога 676
Хранение контента в системе Piranha 678
Практические задания 679
Упражнение 17.1. Проверочные вопросы 679
Упражнение 17.2. Определение типа блока для отображения видео с YouTube 680
Упражнение 17.3. Дополнительные ресурсы 680
Резюме 680

Глава 18. Разработка и использование веб-сервисов 681
Разработка веб-сервисов с помощью Web API в ASP.NET Core 681
Аббревиатуры, типичные для веб-сервисов 681
Разработка проекта Web API в ASP.NET Core 683
Функциональность веб-сервисов 686
Создание веб-сервиса для базы данных Northwind 687
Создание хранилищ данных для сущностей 689
Реализация контроллера Web API 692
Настройка хранилища данных клиентов и контроллера Web API 694

Спецификация деталей проблемы 698
Управление сериализацией XML 699
Документирование и тестирование веб-сервисов 700
Тестирование GET-запросов в браузерах 700
Тестирование HTTP-запросов с помощью расширения REST Client 702
Swagger 705
Тестирование запросов с помощью Swagger UI 707
Обращение к сервисам с помощью HTTP-клиентов 711
HttpClient 712
Настройка HTTP-клиентов с помощью HttpClientFactory 712
Получение списка клиентов в контроллере в формате JSON 713
Включение совместного использования ресурсов между источниками 716
Реализация расширенных функций 718
Мониторинг работоспособности — HealthCheck API 718
Реализация анализаторов и соглашений Open API 719
Обработка проходных отказов 720
Система маршрутизации на основе конечных точек 720
Настройки маршрутизации на основе конечных точек 721
Добавление HTTP-заголовков для безопасности 723
Защита веб-сервисов 725
Прочие коммуникационные технологии 725
Windows Communication Foundation (WCF) 725
gRPC 726
Практические задания 726
Упражнение 18.1. Проверочные вопросы 726
Упражнение 18.2. Создание и удаление клиентов с помощью HttpClient 727
Упражнение 18.3. Дополнительные ресурсы 727
Резюме 728

Глава 19. Разработка интеллектуальных приложений с помощью алгоритмов машинного обучения 729
Общие сведения о машинном обучении 729
Жизненный цикл систем машинного обучения 730
Наборы данных для обучения и тестирования 731
Задачи машинного обучения 732
Машинное обучение Microsoft Azure 733
Знакомство с ML.NET 734
Знакомство с Infer.NET 734
Обучающие конвейеры ML.NET 735

Концепции обучения моделей ... 736
Пропущенные значения и типы ключей ... 737
Характеристики и метки ... 737
Рекомендация товаров пользователю ... 738
Анализ проблем ... 738
Сбор и обработка данных ... 739
Разработка проекта сайта NorthwindML ... 740
Тестирование сайта с рекомендациями по использованию товара ... 753
Практические задания ... 755
Упражнение 19.1. Проверочные вопросы ... 755
Упражнение 19.2. Примеры ... 756
Упражнение 19.3. Дополнительные ресурсы ... 757
Резюме ... 757

Глава 20. Создание пользовательских веб-интерфейсов с помощью Blazor ... 759
Знакомство с Blazor ... 759
Модели хостинга Blazor ... 759
Компоненты Blazor ... 760
Blazor и Razor ... 761
Сравнение шаблонов проектов Blazor ... 761
Обзор шаблона проекта Blazor Server ... 762
CSS-изоляция ... 766
Запуск шаблона проекта Blazor Server ... 767
Обзор шаблона проекта Blazor WebAssembly ... 768
Сборка компонентов с помощью Blazor Server ... 771
Определение и тестирование простого компонента ... 771
Получение сущностей и их компонентов ... 773
Абстрагирование службы для компонента Blazor ... 776
Использование форм Blazor ... 778
Определение форм с помощью компонента EditForm ... 778
Создание и использование компонента формы клиента ... 779
Создание компонентов с использованием Blazor WebAssembly ... 784
Настройки сервера для Blazor WebAssembly ... 785
Настройка клиента для Blazor WebAssembly ... 788
Поддержка прогрессивных веб-приложений ... 792
Практические задания ... 794
Упражнение 20.1. Проверочные вопросы ... 794
Упражнение 20.2. Упражнения по созданию компонента ... 795
Упражнение 20.3. Дополнительные ресурсы ... 795
Резюме ... 796

Глава 21. Разработка кросс-платформенных мобильных приложений 797
Знакомство с XAML 798
Упрощение кода с помощью XAML 798
Выбор общих элементов управления 799
Расширения разметки 800
Знакомство с Xamarin и Xamarin.Forms 800
Xamarin.Forms в качестве расширения Xamarin 801
Мобильные стратегии 801
Доля рынка мобильных платформ 802
Дополнительная функциональность 802
Компоненты пользовательского интерфейса Xamarin.Forms 804
Разработка мобильных приложений с помощью Xamarin.Forms 805
Добавление Android SDK 806
Создание решения Xamarin.Forms 806
Создание модели объекта с двусторонней привязкой данных 808
Создание компонента для набора телефонных номеров 812
Создание представлений для списка клиентов и подробной информации о клиенте 815
Тестирование мобильного приложения в среде iOS 821
Взаимодействие мобильных приложений с веб-сервисами 824
Настройка веб-сервиса в целях разрешения небезопасных запросов 824
Настройка приложения для iOS в целях разрешения небезопасных подключений 825
Настройка приложения для Android в целях разрешения небезопасных подключений 826
Добавление NuGet-пакетов для потребления веб-сервиса 827
Получение данных о клиентах с помощью сервиса 827
Практические задания 829
Упражнение 21.1. Проверочные вопросы 830
Упражнение 21.2. Дополнительные ресурсы 830
Резюме 831

Послесловие 832

5 Создание пользовательских типов с помощью объектно-ориентированного программирования

Данная глава посвящена созданию пользовательских типов с помощью принципов *объектно-ориентированного программирования (ООП)*. Вы узнаете о различных категориях элементов, которые может иметь тип, в том числе о полях для хранения данных и методах для выполнения действий. Вы будете применять концепции ООП, такие как агрегирование и инкапсуляция. Вы изучите языковые функции, такие как поддержка синтаксиса кортежей и переменные `out`, литералы для значений по умолчанию и автоматически определяемые имена кортежей.

В этой главе:

- коротко об ООП;
- сборка библиотек классов;
- хранение данных в полях;
- запись и вызов методов;
- управление доступом с помощью свойств и индексаторов;
- сопоставление шаблонов с объектами;
- работа с записями.

Коротко об объектно-ориентированном программировании

Объект в реальном мире — это предмет, например автомобиль или человек. Объект в программировании часто представляет нечто в реальном мире, например товар или банковский счет, но может быть и чем-то более абстрактным.

В языке C# используются классы `class` (обычно) или структуры `struct` (редко) для определения каждого типа объекта. О разнице между классами и структурами вы узнаете в главе 6. Можно представить тип как шаблон объекта.

Ниже кратко описаны концепции объектно-ориентированного программирования.

- *Инкапсуляция* — комбинация данных и действий, связанных с объектом. К примеру, тип `BankAccount` может иметь такие данные, как `Balance` и `AccountName`, а также действия, такие как `Deposit` и `Withdraw`. При инкапсуляции часто возникает необходимость управлять тем, кто и что может получить доступ к этим действиям и данным, например ограничение доступа к внутреннему состоянию объекта или его изменению извне.
- *Композиция* — то, из чего состоит объект. К примеру, автомобиль состоит из разных частей, таких как четыре колеса, несколько сидений, двигатель и т. д.
- *Агрегирование* касается всего, что может быть объединено с объектом. Например, человек, не будучи частью автомобиля, может сидеть на водительском сиденье, а затем стать водителем. Два отдельных объекта объединены, чтобы сформировать новый компонент.
- *Наследование* — многократное использование кода с помощью подклассов, производных от базовых классов или суперклассов. Все функциональные возможности базового класса становятся доступными в производном классе. Например, базовый или суперкласс `Exception` имеет несколько членов, которые имеют одинаковую реализацию во всех исключениях. Подкласс же или производный класс `SqlException` наследует эти члены и имеет дополнительные, имеющие отношение только к тем случаям, когда возникает исключение базы данных SQL — например, свойство, содержащее информацию о подключении к базе данных.
- *Абстракция* — передача основной идеи объекта и игнорирование его деталей или особенностей. Язык C# имеет ключевое слово `abstract`, которое формализует концепцию. Если класс не явно абстрактный, то его можно описать как конкретный. Базовые классы часто абстрактны, например, суперкласс `Stream` — абстрактный, а его подклассы, такие как `FileStream` и `MemoryStream`, — конкретные. Абстракция — сложный баланс. Если вы сделаете класс слишком абстрактным, то большее количество классов сможет наследовать его, но количество возможностей для совместного использования уменьшится.
- *Полиморфизм* заключается в переопределении производным классом унаследованных методов для реализации собственного поведения.

Разработка библиотек классов

Сборки библиотек классов группируют типы в легко развертываемые модули (DLL-файлы). Не считая раздела, где вы изучали модульное тестирование, до сих пор вы создавали только консольные приложения, содержащие ваш код. Но чтобы он стал доступен для других проектов, его следует помещать в сборки библиотек классов, как это делают сотрудники корпорации Microsoft.

Создание библиотек классов

Первая задача — создать повторно используемую библиотеку классов .NET.

1. В существующей папке Code создайте папку Chapter05 с подпапкой PacktLibrary.
2. В программе Visual Studio Code выберите File ▸ Save Workspace As (Файл ▸ Сохранить рабочую область как), введите имя Chapter05, выберите папку Chapter05 и нажмите кнопку Save (Сохранить).
3. Выберите команду меню File ▸ Add Folder to Workspace (Файл ▸ Добавить папку в рабочую область), выберите папку PacktLibrary и нажмите кнопку Add (Добавить).
4. На панели TERMINAL (Терминал) введите следующую команду:

```
dotnet new classlib
```

5. Откройте файл PacktLibrary.csproj и обратите внимание, что по умолчанию библиотеки классов нацелены на .NET 5 и, следовательно, могут работать только с другими сборками, совместимыми с .NET 5, как показано ниже в коде:

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>

</Project>
```

6. Измените целевую платформу для поддержки .NET Standard 2.0, как показано ниже в коде:

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>netstandard2.0</TargetFramework>
  </PropertyGroup>

</Project>
```

7. Сохраните и закройте файл.
8. На панели TERMINAL (Терминал) скомпилируйте проект, используя следующую команду: `dotnet build`.

Чтобы использовать новейшие функции языка C# и платформы .NET, поместите типы в библиотеку классов .NET 5. Для поддержки устаревших платформ .NET, таких как .NET Core, .NET Framework и Xamarin, поместите типы, которые повторно можно использовать, в библиотеку классов .NET Standard 2.0.

Определение классов

Следующая задача — определить класс, который будет представлять человека.

1. На панели EXPLORER (Проводник) переименуйте файл `Class1.cs` в `Person.cs`.
2. Щелкните кнопкой мыши на файле `Person.cs`, чтобы открыть его, и измените имя класса на `Person`.
3. Измените название пространства имен на `Packt.Shared`.

Мы делаем это, поскольку важно поместить ваши классы в логически именованное пространство имен. Лучшее имя пространства имен будет специфичным для домена, например System.Numerics для типов, связанных с расширенными числовыми функциями, но в нашем случае мы создадим типы Person, BankAccount и WondersOfTheWorld, и у них нет общего домена.

Ваш файл класса теперь должен выглядеть следующим образом:

```
using System;

namespace Packt.Shared
{
  public class Person
  {
  }
}
```

Обратите внимание, что ключевое слово `public` языка C# указывается перед словом `class`. Это ключевое слово называется *модификатором доступа*, управляющим тем, как осуществляется доступ к данному классу.

Если вы явно не определили доступ к классу с помощью ключевого слова `public` (публичный), то он будет доступен только в определяющей его сборке. Это следствие того, что неявный модификатор доступа для класса считается `internal` (внутренний). Нам же нужно, чтобы класс был доступен за пределами сборки, поэтому необходимо ключевое слово `public`.

Члены

У этого типа еще нет членов, инкапсулированных в него. Скоро мы их создадим. Членами могут быть поля, методы или специализированные версии их обоих. Их описание представлено ниже.

- *Поля* используются для хранения данных. Существует три специализированных категории полей:
  - *константы* — данные в них никогда не меняются. Компилятор буквально копирует данные в любой код, который их читает;

  - *поля, доступные только для чтения,* — данные в таких полях не могут измениться после создания экземпляра класса, но могут быть рассчитаны или загружены из внешнего источника во время создания экземпляра;
  - *события* — данные ссылаются на один или несколько методов, вызываемых автоматически при возникновении определенной ситуации, например при нажатии кнопки. Тема событий будет рассмотрена в главе 6.
- *Методы* используются для выполнения операторов. Вы уже ознакомились с некоторыми примерами в главе 4. Существует четыре специализированных метода:
  - *конструкторы* выполняются, когда вы используете ключевое слово `new` для выделения памяти и создания экземпляра класса;
  - *свойства* выполняются, когда необходимо получить доступ к данным. Они обычно хранятся в поле, но могут храниться извне или рассчитываться во время выполнения. Использование свойств — предпочтительный способ инкапсуляции полей, если только не требуется выдать наружу адрес памяти поля;
  - *индексаторы* выполняются, когда необходимо получить доступ к данным с помощью синтаксиса массива `[]`;
  - *операции* выполняются, когда необходимо применить операции типа `+` и `/` для операндов вашего типа.

Создание экземпляров классов

В этом подразделе мы создадим *экземпляр* класса `Person` (данный процесс описывается как *инстанцирование* класса).

Ссылка на сборку

Прежде чем мы сможем создать экземпляр класса, нам нужно сослаться на сборку, которая его содержит.

1. Создайте подпапку PeopleApp в папке Chapter05.
2. В программе Visual Studio Code выберите File ▸ Add Folder to Workspace (Файл ▸ Добавить папку в рабочую область), выберите папку PeopleApp и нажмите кнопку Add (Добавить).
3. Выберите команду меню Terminal ▸ New Terminal (Терминал ▸ Новый терминал) и выберите пункт `PeopleApp`.
4. На панели TERMINAL (Терминал) введите следующую команду:

```
dotnet new console
```

5. На панели EXPLORER (Проводник) щелкните кнопкой мыши на файле `PeopleApp.csproj`.
6. Добавьте ссылку на проект в `PacktLibrary`, как показано ниже (выделено полужирным шрифтом):

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net5.0</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <ProjectReference Include="../PacktLibrary/PacktLibrary.csproj" />
  </ItemGroup>
</Project>
```

7. На панели TERMINAL (Терминал) введите команду для компиляции проекта `PeopleApp` и его зависимого проекта `PacktLibrary`, как показано в следующей команде:

```
dotnet build
```

8. Выберите `PeopleApp` в качестве активного проекта для OmniSharp.

Импорт пространства имен для использования типа

Теперь мы готовы написать операторы для работы с классом `Person`.

1. В программе Visual Studio Code в папке `PeopleApp` откройте проект `Program.cs`.
2. В начале файла `Program.cs` введите операторы для импорта пространства имен для нашего класса `People` и статически импортируйте класс `Console`, как показано ниже:

```
using Packt.Shared;
using static System.Console;
```

3. В методе `Main` введите операторы для:
   - создания экземпляра типа `Person`;
   - вывода экземпляра, используя его текстовое описание.

   Ключевое слово `new` выделяет память для объекта и инициализирует любые внутренние данные. Мы могли бы использовать `Person` вместо ключевого слова `var`, но применение последнего требует меньше ввода и по-прежнему понятно, как показано ниже:

```
var bob = new Person();
WriteLine(bob.ToString());
```

Вы можете спросить: «Почему у переменной bob имеется метод ToString? Класс Person пуст!» Не беспокойтесь, скоро вы все узнаете!

4. Запустите приложение, введя команду dotnet run на панели TERMINAL (Терминал), а затем проанализируйте результат:

```
Packt.Shared.Person
```

Управление несколькими файлами

Если требуется одновременная работа с несколькими файлами, то вы можете размещать их рядом друг с другом по мере их редактирования.

1. На панели EXPLORER (Проводник) разверните два проекта.
2. Откройте файлы Person.cs и Program.cs.
3. Нажав и удерживая кнопку мыши, перетащите вкладку окна редактирования для одного из ваших открытых файлов, чтобы расположить их так, чтобы вы могли одновременно видеть оба файла Person.cs и Program.cs.

Вы можете нажать кнопку Split Editor Right (Разделить редактор) или нажать сочетание клавиш Ctrl+\ или Cmd+\, чтобы разместить два окна файла друг рядом с другом.

Более подробно о работе с пользовательским интерфейсом Visual Studio Code вы можете прочитать на сайте: https://code.visualstudio.com/docs/getstarted/userinterface.

Работа с объектами

Хотя наш класс Person явно не наследуется ни от какого типа, все типы косвенно наследуются от специального типа System.Object. Реализация метода ToString в типе System.Object выдает полные имена пространства имен и типа.

Возвращаясь к исходному классу Person, мы могли бы явно сообщить компилятору, что Person наследуется от типа System.Object:

```
public class Person : System.Object
```

Когда класс Б *наследуется* от класса А, мы говорим, что А — *базовый класс* или суперкласс, а Б — *производный класс*. В нашем случае System.Object — базовый класс (суперкласс), а Person — производный.

Мы также можем использовать в C# псевдоним типа — ключевое слово object:

```
public class Person : object
```

Наследование System.Object

Сделаем наш класс явно наследуемым от `object`, а затем рассмотрим, какие члены имеют все объекты.

1. Измените свой класс `Person` для явного наследования от `object`.
2. Затем установите указатель мыши внутри ключевого слова `object` и нажмите клавишу F12 или щелкните правой кнопкой мыши на ключевом слове `object` и выберите Go to Definition (Перейти к определению).

Вы увидите определение типа `System.Object` и его членов. Вам не нужно разбираться во всем этом определении, но обратите внимание на метод `ToString`, показанный на рис. 5.1.

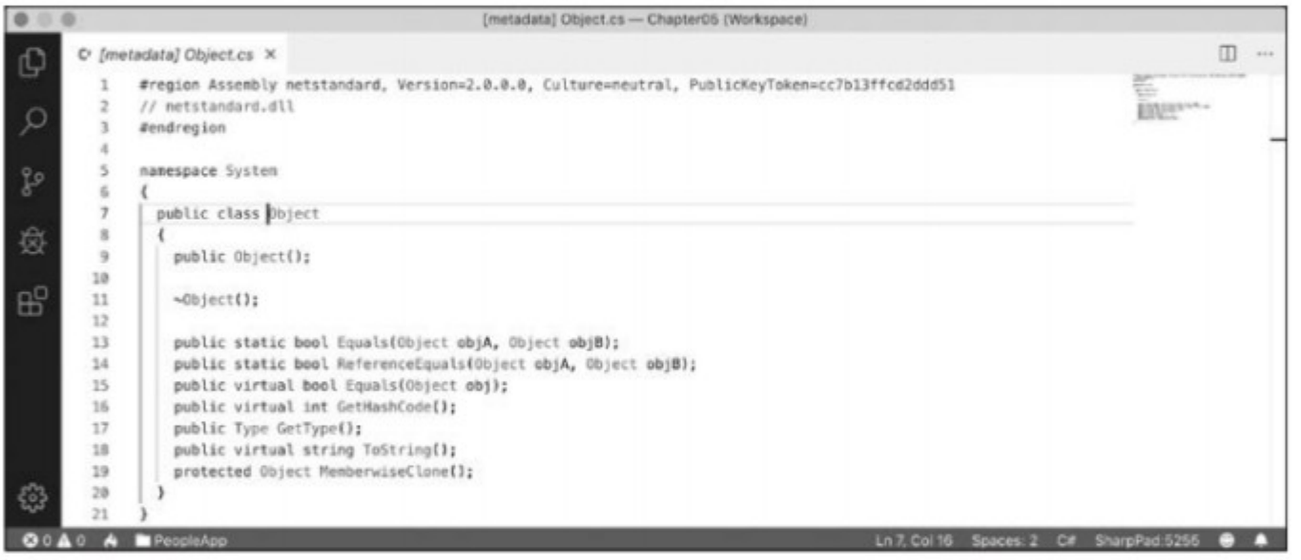

Рис. 5.1. Определение класса System.Object

Будьте уверены: другие программисты знают, что, если наследование не указано, класс наследуется от System.Object.

Хранение данных в полях

Теперь определим в классе несколько полей для хранения информации о человеке.

Определение полей

Допустим, мы решили, что информация о человеке включает имя и дату рождения. Мы инкапсулируем оба значения в классе `Person` и также сделаем поля общедоступными.

В классе `Person` напишите операторы для объявления двух общедоступных полей для хранения имени и даты рождения человека:

```
public class Person : object
{
  // поля
  public string Name;
  public DateTime DateOfBirth;
}
```

Вы можете использовать любые типы для полей, включая массивы и коллекции, такие как списки и словари. Они вам будут полезны при необходимости хранить несколько значений в одном именованном поле. В данном примере информация о человеке содержит только одно имя и одну дату рождения.

Модификаторы доступа

При реализации инкапсуляции важно выбрать, насколько видны элементы.

Обратите внимание: работая с классом, мы использовали ключевое слово `public` по отношению к созданным полям. Если бы мы этого не сделали, то поля были бы закрытыми, то есть доступными только в пределах класса.

Доступны четыре ключевых слова для модификаторов доступа, каждое из которых вы можете применить к члену класса, например к полю или методу, как показано в табл. 5.1.

Таблица 5.1. Модификаторы доступа

Модификатор доступа	Описание
private	Доступ ограничен содержащим типом. Используется по умолчанию
internal	Доступ ограничен содержащим типом и любым другим типом в текущей сборке
protected	Доступ ограничен содержащим типом или типами, производными от содержащего типа
public	Неограниченный доступ
internal protected	Доступ ограничен содержащим типом и любым другим типом в текущей сборке, а также типами, производными от содержащего класса. Аналогичен вымышленному модификатору доступа internal_or_protected (то есть дает доступ по принципу «internal ИЛИ protected»).
private protected	Доступ ограничен содержащим типом и любым другим типом, который наследуется от типа и находится в той же сборке. Аналогичен вымышленному модификатору доступа internal_and_protected (то есть дает доступ по принципу «internal И protected»). Эта комбинация доступна только для версии C# 7.2 или более поздних