Введение

Авторы и читатели — одна команда

Авторам книг по технологиям приходится писать для очень требовательной группы людей (по вполне понятным причинам). Вам известно, что построение программных решений с применением любой платформы или языка исключительно сложно и специфично для отдела, компании, клиентской базы и поставленной задачи. Возможно, вы работаете в индустрии электронных публикаций, разрабатываете системы для правительства или местных органов власти либо сотрудничаете с NASA или какой-то военной отраслью. Вместе мы трудимся в нескольких отраслях, включая разработку обучающего ПО для детей (Oregon Trail/Amazon Trail), разнообразных производственных систем и проектов в медицинской и финансовой сферах. Написанный вами код на месте вашего трудоустройства почти на 100% будет иметь мало общего с кодом, который мы создавали на протяжении многих лет.

По указанной причине в книге мы намеренно решили избегать демонстрации примеров кода, свойственного какой-то конкретной отрасли или направлению программирования. Таким образом, мы объясняем язык C#, объектно-ориентированное программирование, среду CLR и библиотеки базовых классов .NET с использованием примеров, не привязанных к отрасли. Вместо того чтобы заставлять каждый пример наполнять сетку данными, подчитывать фонд заработной платы или делать еще что-нибудь подобное, мы придерживаемся темы, близкой каждому из нас: автомобили (с добавлением умеренного количества геометрических структур и систем расчета заработной платы для сотрудников). И вот тут наступает ваш черед.

Наша работа заключается в как можно лучшем объяснении языка программирования C# и основных аспектов платформы .NET. Мы также будем делать все возможное для того, чтобы снарядить вас инструментами и стратегиями, которые необходимы для продолжения обучения после завершения работы с данной книгой.

Ваша работа предусматривает усвоение этой информации и ее применение к решению своих задач программирования. Мы полностью отдаем себе отчет, что ваши проекты с высокой вероятностью не будут связаны с автомобилями и их дружественными именами, но именно в том и состоит суть прикладных знаний.

Мы уверены, что после освоения тем и концепций, представленных в настоящей книге, вы сможете успешно строить решения .NET, которые соответствуют вашей конкретной среде программирования.

Краткий обзор книги

Книга логически разделена на девять частей, каждая из которых содержит несколько связанных друг с другом глав. Ниже приведено краткое содержание частей и глав.

Часть I. Введение в C# и платформу .NET

Эта часть книги предназначена для ознакомления с природой платформы .NET и различными инструментами разработки, которые используются во время построения приложений .NET.

Глава 1. Философия .NET

Первая глава выступает в качестве основы для всего остального материала. Ее главная цель в том, чтобы представить вам набор строительных блоков .NET, таких как общеязыковая исполняющая среда (Common Language Runtime), общая система типов (Common Type System), общеязыковая спецификация (Common Language Specification) и библиотеки базовых классов. Здесь вы впервые взглянете на язык программирования C# и формат сборок .NET. Кроме того, будет исследована независимая от платформы природа .NET.

Глава 2. Создание приложений на языке C#

Целью этой главы является введение в процесс компиляции файлов исходного кода C#. Здесь вы узнаете о совершенно бесплатной (но полнофункциональной) редакции Visual Studio Community, которая главным образом задействована в книге, а также о редакциях Professional и Enterprise продукта Visual Studio 2017. Вы научитесь конфигурировать машину разработки с применением нового процесса установки Visual Studio, основанного на рабочих нагрузках, включать в своих проектах средства C# 7.1 и устанавливать все важные инфраструктуры .NET 4.7 и .NET Core 2.0.

Часть II. Основы программирования на C#

Темы, представленные в этой части книги, очень важны, потому что они связаны с разработкой программного обеспечения .NET любого типа (например, веб-приложений, настольных приложений с графическим пользовательским интерфейсом, библиотек кода или служб Windows). Здесь вы узнаете о фундаментальных типах данных .NET, освоите манипулирование текстом и ознакомитесь с ролью модификаторов параметров C# (включая необязательные и именованные аргументы).

Глава 3. Главные конструкции программирования на C#: часть I

В этой главе начинается формальное исследование языка программирования C#. Здесь вы узнаете о роли метода `Main()` и многочисленных деталях, касающихся внутренних типов данных платформы .NET и объявления переменных, а также научитесь работать и манипулировать текстовыми данными, используя типы `System.String` и `System.Text.StringBuilder`. Кроме того, вы исследуете итерационные конструкции и конструкции принятия решений, сужающие и расширяющие операции и ключевое слово `unchecked`.

Глава 4. Главные конструкции программирования на C#: часть II

В этой главе завершается исследование ключевых аспектов C#, начиная с создания и манипулирования массивами данных. Затем вы узнаете, как конструировать перегруженные методы типов и определять параметры с применением ключевых слов `out`, `ref` и `params`. Также вы изучите тип перечисления, структуры и типы, допускающие `null`, плюс уясните отличие между типами значений и ссылочными типами. Наконец, вы освоите кортежи — новое средство версии C# 7.

Часть III. Объектно-ориентированное программирование на C#

В этой части вы изучите основные конструкции языка C#, включая детали объектно-ориентированного программирования. Здесь вы также научитесь обрабатывать исключения времени выполнения и взаимодействовать со строго типизированными интерфейсами.

Глава 5. Инкапсуляция

В этой главе начинается рассмотрение концепций объектно-ориентированного программирования (ООП) на языке C#. После представления главных принципов ООП (инкапсуляции, наследования и полиморфизма) будет показано, как строить надежные типы классов с использованием конструкторов, свойств, статических членов, констант и полей только для чтения. Глава завершается исследованием частичных определений типов, синтаксиса инициализации объектов и автоматических свойств.

Глава 6. Наследование и полиморфизм

Здесь вы ознакомитесь с оставшимися главными принципами ООП (наследованием и полиморфизмом), которые позволяют создавать семейства связанных типов классов. Вы узнаете о роли виртуальных и абстрактных методов (и абстрактных базовых классов), а также о природе полиморфных интерфейсов. Затем вы исследуете сопоставление с образцом — нововведение C# 7. Наконец, в главе будет объясняться роль главного базового класса платформы .NET — `System.Object`.

Глава 7. Структурированная обработка исключений

В этой главе обсуждаются способы обработки в кодовой базе аномалий, возникающих во время выполнения, за счет использования структурированной обработки исключений. Вы узнаете не только о ключевых словах C#, которые дают возможность решать такие задачи (`try`, `catch`, `throw`, `when` и `finally`), но и о разнице между исключениями уровня приложения и уровня системы. Вдобавок в главе будут исследованы разнообразные инструменты внутри Visual Studio, которые позволяют отлаживать исключения, оставшиеся без внимания.

Глава 8. Работа с интерфейсами

Материал этой главы опирается на ваше понимание объектно-ориентированной разработки и посвящен программированию на основе интерфейсов. Здесь вы узнаете, каким образом определять классы и структуры, поддерживающие несколько линий поведения, обнаруживать такие линии поведения во время выполнения и выборочно скрывать какие-то из них с применением явной реализации интерфейсов. В дополнение к созданию специальных интерфейсов вы научитесь реализовывать стандартные интерфейсы, доступные внутри платформы .NET, и использовать их для построения объектов, которые могут сортироваться, копироваться, перечисляться и сравниваться.

Часть IV. Дополнительные конструкции программирования на C#

В этой части книги вы углубите знания языка C# за счет исследования нескольких более сложных (и важных) концепций. Здесь вы завершите ознакомление с системой типов .NET, изучив интерфейсы и делегаты. Вы также узнаете о роли обобщений, кратко взглянете на язык LINQ (Language Integrated Query) и освоите ряд более сложных средств C# (например, методы расширения, частичные методы, манипулирование указателями и жизненный цикл объектов).

Глава 9. Коллекции и обобщения

В этой главе исследуется тема *обобщений*. Вы увидите, что обобщенное программирование предлагает способ создания типов и членов типов, которые содержат заполнители, указываемые вызывающим кодом. По существу обобщения значительно улучшают производительность приложений и безопасность в отношении типов. Здесь не только описаны разнообразные обобщенные типы из пространства имен `System.Collections.Generic`, но также показано, каким образом строить собственные обобщенные методы и типы (с ограничениями и без).

Глава 10. Делегаты, события и лямбда-выражения

Целью этой главы является прояснение типа делегата. Выражаясь просто, делегат .NET представляет собой объект, который указывает на определенные методы в приложении. С помощью делегатов можно создавать системы, которые позволяют многочисленным объектам участвовать в двухстороннем взаимодействии. После исследования способов применения делегатов .NET вы ознакомитесь с ключевым словом `event` языка C#, которое упрощает манипулирование низкоуровневыми делегатами в коде. В завершение вы узнаете о роли лямбда-операции C# (`=>`), а также о связи между делегатами, анонимными методами и лямбда-выражениями.

Глава 11. Расширенные средства языка C#

В этой главе вы сможете углубить понимание языка C# за счет исследования нескольких расширенных приемов программирования. Здесь вы узнаете, как перегружать операции и создавать специальные процедуры преобразования (явного и неявного) для типов. Вы также научитесь строить и взаимодействовать с индексаторами типов и работать с расширяющими методами, анонимными типами, частичными методами и указателями C#, используя контекст небезопасного кода.

Глава 12. LINQ to Objects

В этой главе начинается исследование языка интегрированных запросов (LINQ). Язык LINQ дает возможность строить строго типизированные выражения запросов, которые могут применяться к многочисленным целевым объектам LINQ для манипулирования данными в самом широком смысле этого слова. Здесь вы изучите API-интерфейс LINQ to Objects, который позволяет применять выражения LINQ к контейнерам данных (например, массивам, коллекциям и специальным типам). Приведенная в главе информация будет полезна позже в книге при рассмотрении других API-интерфейсов.

Глава 13. Время существования объектов

В финальной главе этой части исследуется управление памятью средой CLR с использованием сборщика мусора .NET. Вы узнаете о роли корневых элементов приложения, поколений объектов и типа `System.GC`. После представления основ будут рассматриваться темы освобождаемых объектов (реализующих интерфейс `IDisposable`) и процесса финализации (с применением метода `System.Object.Finalize()`). В главе также описан класс `Lazy<T>`, позволяющий определять данные, которые не будут размещаться в памяти вплоть до поступления запроса со стороны вызывающего кода. Вы увидите, что такая возможность очень полезна, когда нежелательно загромождать кучу объектами, которые в действительности программе не нужны.

Часть V. Программирование с использованием сборок .NET

Эта часть книги посвящена деталям формата сборок .NET. Здесь вы узнаете не только о том, как развертывать и конфигурировать библиотеки кода .NET, но также о внутреннем устройстве двоичного образа .NET. Будет описана роль атрибутов .NET и распознавания информации о типе во время выполнения. Кроме того, объясняется роль исполняющей среды динамического языка (Dynamic Language Runtime — DLR) и ключевого слова `dynamic` языка C#. Наконец, рассматриваются более сложные темы, касающиеся сборок, такие как домены приложений, синтаксис языка CIL и построение сборок в памяти.

Глава 14. Построение и конфигурирование библиотек классов

На самом высоком уровне термин "сборка" применяется для описания двоичного файла `*.dll` или `*.exe`, созданного с помощью компилятора .NET. Однако в действительности понятие сборки намного шире. Здесь будет показано, чем отличаются однофайловые и многофайловые сборки, как создавать и развертывать сборки обеих разновидностей. Также объясняется, каким образом конфигурировать закрытые и разделяемые сборки с помощью XML-файлов `*.config` и специальных сборок политик издателя. По ходу дела раскрывается внутренняя структура глобального кеша сборок (Global Assembly Cache — GAC).

Глава 15. Рефлексия типов, позднее связывание и программирование на основе атрибутов

В этой главе продолжается исследование сборок .NET. Здесь будет показано, как обнаруживать типы во время выполнения с использованием пространства имен `System.Reflection`. Посредством типов из упомянутого пространства имен можно строить приложения, способные считывать метаданные сборки на лету. Вы также узнаете, как загружать и создавать типы динамически во время выполнения с применением позднего связывания. Напоследок в главе обсуждается роль атрибутов .NET (стандартных и специальных). Для закрепления материала в главе демонстрируется построение расширяемого консольного приложения.

Глава 16. Динамические типы и среда DLR

В версии .NET 4.0 появился новый аспект исполняющей среды .NET, который называется *исполняющей средой динамического языка* (DLR). Используя DLR и ключевое слово `dynamic` языка C#, можно определять данные, которые в действительности не будут распознаваться вплоть до времени выполнения. Такие средства существенно упрощают решение ряда сложных задач программирования для .NET. В этой главе вы ознакомитесь со сценариями применения динамических данных, включая использование API-интерфейсов рефлексии .NET и взаимодействие с унаследованными библиотеками COM с минимальными усилиями.

Глава 17. Процессы, домены приложений и объектные контексты

Базируясь на хорошем понимании вами сборок, в этой главе подробно раскрывается внутреннее устройство загруженной исполняемой сборки .NET. Целью главы является иллюстрация отношений между процессами, доменами приложений и контекстными границами. Упомянутые темы формируют основу для главы 19, где будет исследоваться конструирование многопоточных приложений.

Глава 18. Язык CIL и роль динамических сборок

Последняя глава в этой части преследует двойную цель. В первой половине главы рассматривается синтаксис и семантика языка CIL, а во второй — роль пространства имен `System.Reflection.Emit`. Типы из указанного пространства имен можно применять для построения программного обеспечения, которое способно генерировать сборки .NET в памяти во время выполнения. Формально сборки, которые определяются и выполняются в памяти, называются *динамическими сборками*.

Часть VI. Введение в библиотеки базовых классов .NET

К настоящему моменту вы уже должны хорошо ориентироваться в языке C# и в подробностях формата сборок .NET. В данной части книги ваши знания расширяются исследованием нескольких часто используемых служб, которые можно обнаружить внутри библиотек базовых классов .NET, включая создание многопоточных приложений, файловый ввод-вывод и доступ к базам данных посредством ADO.NET. Здесь также раскрывается процесс создания распределенных приложений с применением Windows Communication Foundation (WCF) и API-интерфейс LINQ to XML.

Глава 19. Многопоточное, параллельное и асинхронное программирование

Эта глава посвящена построению многопоточных приложений. В ней демонстрируются приемы, которые можно использовать для написания кода, безопасного к потокам. Глава начинается с краткого напоминания о том, что собой представляет тип делегата .NET, и объяснения внутренней поддержки делегата для асинхронного вызова методов. Затем рассматриваются типы из пространства имен `System.Threading` и библиотека параллельных задач (Task Parallel Library — TPL). С применением TPL разработчики .NET могут строить приложения, которые распределяют рабочую нагрузку по всем доступным процессорам в исключительно простой манере. В главе также раскрыта роль API-интерфейса Parallel LINQ, который предлагает способ создания запросов LINQ, масштабируемых среди множества процессорных ядер. В завершение главы исследуется создание неблокирующих вызовов с использованием ключевых слов `async/await`, введенных в версии C# 5, а также локальных функций и обобщенных возвращаемых типов `async`, появившихся в версии C# 7.

Глава 20. Файловый ввод-вывод и сериализация объектов

Пространство имен `System.IO` позволяет взаимодействовать со структурой файлов и каталогов машины. В этой главе вы узнаете, как программно создавать (и удалять) систему каталогов. Вы также научитесь перемещать данные между различными потоками (например, файловыми, строковыми и находящимися в памяти). Кроме того, в главе рассматриваются службы сериализации объектов платформы .NET. Сериализация позволяет сохранять состояние объекта (или набора связанных объектов) в потоке для последующего использования. Десериализация представляет собой процесс извлечения объекта из потока в память с целью потребления внутри приложения. После описания основ вы освоите настройку процесса сериализации с применением интерфейса `ISerializable` и набора атрибутов .NET.

Глава 21. Доступ к данным с помощью ADO.NET

В этой первой из двух глав, посвященных работе с базами данных с использованием полной платформы .NET Framework, представлено введение в API-интерфейс доступа к базам данных платформы .NET, который называется ADO.NET. В частности, здесь рас-

сматривается роль поставщиков данных .NET и взаимодействие с реляционной базой данных с применением подключенного уровня ADO.NET, который представлен объектами подключений, объектами команд, объектами транзакций и объектами чтения данных.

Глава 22. Введение в Entity Framework 6

В этой главе изучение взаимодействия с базами данных завершается рассмотрением роли Entity Framework (EF) 6. Инфраструктура EF представляет собой систему объектно-реляционного отображения (object-relational mapping — ORM), которая предлагает способ написания кода доступа к данным с использованием строго типизированных классов, напрямую отображаемых на бизнес-модель. Здесь вы узнаете о роли класса `DbContext` из EF, применении аннотаций данных и интерфейса Fluent API для формирования базы данных, реализации хранилищ для инкапсуляции общего кода, транзакциях, миграциях, проверке параллелизма и перехвате команд. Вдобавок вы научитесь взаимодействовать с реляционными базами данных с помощью LINQ to Entities. В главе также создается специальная библиотека доступа к данным (`AutoLotDAL.dll`), которая будет использоваться в нескольких оставшихся главах книги.

Глава 23. Введение в Windows Communication Foundation

До этого места в книге все примеры приложений запускались на единственном компьютере. В настоящей главе вы ознакомитесь с API-интерфейсом Windows Communication Foundation (WCF), который позволяет создавать распределенные приложения в симметричной манере независимо от лежащих в их основе низкоуровневых деталей. Здесь будет объясняться конструкция служб, хостов и клиентов WCF, также применение конфигурационных файлов на основе XML для декларативного указания адресов, привязок и контрактов.

Часть VII. Windows Presentation Foundation

Первоначальный API-интерфейс для построения графических пользовательских интерфейсов настольных приложений, поддерживаемый платформой .NET, назывался Windows Forms. Хотя он по-прежнему доступен в полной платформе .NET Framework, в версии .NET 3.0 программистам был предложен замечательный API-интерфейс под названием Windows Presentation Foundation (WPF). В отличие от Windows Forms эта высокопроизводительная инфраструктура для построения пользовательских интерфейсов объединяет несколько основных служб, включая привязку данных, двумерную и трехмерную графику, анимацию и форматированные документы, в единую унифицированную модель. Все это достигается с использованием декларативной грамматики разметки, которая называется расширяемым языком разметки приложений (Extendable Application Markup Language — XAML). Более того, архитектура элементов управления WPF предлагает легкий способ радикального изменения внешнего вида и поведения типового элемента управления с применением всего лишь правильно оформленной разметки XAML.

Глава 24. Введение в Windows Presentation Foundation и XAML

Эта глава начинается с исследования мотивации создания WPF (с учетом того, что в .NET уже существовала инфраструктура для разработки графических пользовательских интерфейсов настольных приложений). Затем вы узнаете о синтаксисе XAML и ознакомитесь с поддержкой построения приложений WPF в Visual Studio.

Глава 25. Элементы управления, компоновки, события и привязка данных в WPF

В этой главе будет показано, как работать с элементами управления и диспетчерами компоновки, предлагаемыми WPF. Вы узнаете, каким образом создавать системы меню, окна с разделителями, панели инструментов и строки состояния. Также в главе рассматриваются API-интерфейсы (и связанные с ними элементы управления), входящие в состав WPF, в том числе Ink API, команды, маршрутизируемые события, модель привязки данных и свойства зависимости.

Глава 26. Службы визуализации графики WPF

Инфраструктура WPF является API-интерфейсом, интенсивно использующим графику, и с учетом этого WPF предоставляет три подхода к визуализации графических данных: фигуры, рисунки и геометрические объекты, а также визуальные объекты. В настоящей главе вы ознакомитесь с каждым подходом и попутно изучите несколько важных графических примитивов (например, кисти, перья и трансформации). Кроме того, вы узнаете, как встраивать векторные изображения в графику WPF, а также выполнять операции проверки попадания в отношении графических данных.

Глава 27. Ресурсы, анимация, стили и шаблоны WPF

В этой главе освещены важные (и взаимосвязанные) темы, которые позволят углубить знания API-интерфейса WPF. Первым делом вы изучите роль логических ресурсов. Система логических ресурсов (также называемых *объектными ресурсами*) предлагает способ именования и ссылки на часто используемые объекты внутри приложения WPF. Затем вы узнаете, каким образом определять, выполнять и управлять анимационной последовательностью. Вы увидите, что применение анимации WPF не ограничивается видеоиграми или мультимедиа-приложениями. В завершение главы вы ознакомитесь с ролью стилей WPF. Подобно веб-странице, использующей CSS или механизм тем ASP.NET, приложение WPF может определять общий вид и поведение для целого набора элементов управления.

Глава 28. Уведомления, проверка достоверности, команды и MVVM

Эта глава начинается с исследования трех основных возможностей инфраструктуры WPF: уведомлений, проверки достоверности и команд. В разделе, в котором рассматриваются уведомления, вы узнаете о наблюдаемых моделях и коллекциях, а также о том, как они поддерживают данные приложения и пользовательский интерфейс в синхронизированном состоянии. Затем вы научитесь создавать специальные команды для инкапсуляции кода. В разделе, посвященном проверке достоверности, вы ознакомитесь с несколькими механизмами проверки достоверности, которые доступны для применения в приложениях WPF. Глава завершается исследованием паттерна “модель-представление-модель представления” (Model View ViewModel — MVVM) и созданием приложения, демонстрирующего паттерн MVVM в действии.

Часть VIII. ASP.NET

Эта часть посвящена построению веб-приложений с использованием API-интерфейса ASP.NET. Легковесная инфраструктура ASP.NET MVC задействует паттерн “модель-представление-контроллер” (Model-View-Controller) и предназначена для создания веб-приложений. Версия ASP.NET Web API 2.2 основана (и похожа) на ASP.NET MVC и является инфраструктурой для разработки веб-служб REST.

Глава 29. Введение в ASP.NET MVC

В этой главе рассматривается ASP.NET MVC. Инфраструктура ASP.NET MVC основана на паттерне MVC, после освоения которого вы приступите к построению приложения MVC. Вы узнаете о шаблонах, маршрутизации, контроллерах, действиях и представлениях. Затем будет создано приложение ASP.NET MVC с применением уровня доступа к данным, разработанного в главе 22.

Глава 30. Введение в ASP.NET Web API

В этой главе вы построите службу REST, используя ASP.NET Web API 2.2. Служба будет поддерживать операции создания, чтения, обновления и удаления (create, read, update, delete — CRUD) в отношении складских данных с применением уровня доступа к данным, созданного в главе 22. Наконец, вы обновите приложение ASP.NET MVC для использования службы REST в качестве уровня доступа к данным.

Часть IX. .NET Core

Эта часть посвящена .NET Core — межплатформенной версии .NET. После общего ознакомления с .NET Core, мотивацией, а также отличиями между .NET Core и полной платформой .NET Framework уровень доступа к данным `AutoLotDAL` будет создан заново уже с применением Entity Framework Core. В последних двух главах части рассматривается построение веб-приложений и служб REST с помощью инфраструктуры ASP.NET Core.

Глава 31. Философия .NET Core

В этой главе предлагается введение в .NET Core — кардинально новую межплатформенную версию .NET. Вы узнаете о целях .NET Core, различных ее частях (вроде CoreCLR и CoreFX) и поддержке жизненного цикла .NET Core. После установки (и проверки) .NET Core будет приведено сравнение .NET Core с полной платформой .NET Framework.

Глава 32. Введение в Entity Framework Core

В этой главе рассматривается версия инфраструктуры Entity Framework в .NET Core. Хотя многие концепции EF сохранили свою актуальность, между EF 6 и EF Core существует несколько значительных и важных отличий. Глава начинается со сравнения инфраструктур EF 6 и EF Core и продолжается созданием библиотеки `AutoLotDAL_Core2` — версии EF Core уровня доступа к данным, построенного в главе 22. Обновленный уровень доступа к данным будет использоваться в оставшихся главах книги.

Глава 33. Введение в веб-приложения ASP.NET Core

Это первая из двух глав, посвященных ASP.NET Core, в которой строятся веб-приложения в стиле MVC. В начале главы с помощью нового шаблона ASP.NET Core Web Application (Веб-приложение ASP.NET Core) будет создано приложение `AutoLotMVC_Core2`. Затем рассматриваются нововведения ASP.NET Core (в сравнении с ASP.NET MVC 5), включая поддержку внедрения зависимостей, новую систему конфигурации, осведомленность о среде, вспомогательные функции дескрипторов и компоненты представлений. В конце главы будет построена версия ASP.NET Core приложения `AutoLotMVC` (из главы 29) с применением `AutoLotDAL_Core2` в качестве уровня доступа к данным.

Глава 34. Введение в приложения служб ASP.NET Core

Данная глава завершает исследование ASP.NET Core построением службы REST с использованием ASP.NET Core. Вместо того чтобы быть отдельными (хотя похожими) инфраструктурами, службы и веб-приложения в ASP.NET Core применяют одну и ту же кодовую базу, в итоге совместно используя MVC и Web API. Подобно службе, созданной в главе 30, служба `AutoLotAPI_Core2` поддерживает все операции CRUD для складских данных, применяя уровень доступа к данным `AutoLotDAL_Core2` из главы 32. Наконец, веб-приложение ASP.NET Core будет обновлено с целью использования новой службы вместо прямого обращения к `AutoLotDAL_Core2`.

Загружаемые приложения

В дополнение к печатным материалам хранилище GitHub содержит исходный код примеров, рассмотренных в книге (доступный по адресу `www.apress.com/9781484230176`), и дополнительные приложения в формате PDF (версии на русском языке доступны для загрузки на веб-сайте издательства). В них раскрывается несколько дополнительных API-интерфейсов платформы .NET, которые вы можете счесть удобными для своего рода занятий. В частности, вы найдете следующие материалы:

- Приложение А. Наборы данных, таблицы данных и адаптеры данных ADO.NET
- Приложение Б. Введение в LINQ to XML
- Приложение В. Введение в ASP.NET Web Forms
- Приложение Г. Веб-элементы управления, мастер-страницы и темы ASP.NET
- Приложение Д. Управление состоянием в ASP.NET

Исходный код примеров

Исходный код всех примеров, представленных в книге, доступен в открытом хранилище GitHub (`https://github.com/apress/pro-csharp-7`). Проекты организованы по главам.

Имейте в виду, что почти во всех главах книги присутствуют врезки, помеченные как “Исходный код”. Они служат визуальной подсказкой о том, что вы можете загрузить код обсуждаемого примера в IDE-среду Visual Studio для дальнейшего исследования и модификации.

Исходный код. Здесь дается ссылка на определенный каталог в хранилище GitHub.

Чтобы открыть решение в Visual Studio, выберите пункт меню File⇨Open⇨Project/Solution (Файл⇨Открыть⇨Проект/решение) и перейдите к интересующему файлу `*.sln` в соответствующем каталоге.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: `info@dialektika.com`
WWW: `http://www.dialektika.com`