Введение

Первое издание настоящей книги появилось вместе с выпуском Microsoft второй бета-версии .NET 1.0 (летом 2001 г.) и с тех пор постоянно переиздавалось в том или ином виде. С того самого времени автор с чрезвычайным удовольствием и благодарностью наблюдал за тем, как данная работа продолжала пользоваться популярностью в прессе и, самое главное, среди читателей. Через некоторое время, в 2002 г., она даже была номинирована на премию Jolt Award (которую, увы, получить так и не удалось, но книга попала в число финалистов), а в 2003 г. — еще и на премию Referenceware Excellence Award, где стала лучшей книгой года по программированию.

Даже еще более важно то, что автору стали приходить электронные письма от читателей со всего мира. Общаться с множеством людей и узнавать, что данная книга как-то помогла им в карьере, просто замечательно. В связи с этим, хотелось бы отметить, что настоящая книга с каждым разом становится все лучше именно благодаря читателям, которые присылают различные предложения по улучшению, указывают на допущенные в тексте опечатки и обращают внимание на прочие промахи.

Автор был просто ошеломлен, узнав, что эта книга использовалась и продолжает использоваться на занятиях в колледжах и университетах и является обязательной для прочтения на многих предвыпускных и выпускных курсах в области вычислительной техники.

Автор благодарит прессу, читателей, преподавателей и всех остальных и желает им успешного программирования!

С самого первого выпуска настоящей книги автор прилагал все усилия и обновлял книгу так, чтобы она отражала текущие возможности каждой выходившей версии платформы .NET. В настоящем издании материал был полностью пересмотрен и расширен с целью охвата новых средств языка C# 2010 и платформы .NET 4.0. Вы найдете информацию по таким новым компонентам, как Dynamic Language Runtime (DLR), Task Parallel Library (TPL), Parallel LINQ (PLINQ) и ADO.NET Entity Framework (EF). Кроме того, в книге описан ряд менее значительных (но очень полезных) обновлений, наподобие именованных и необязательных аргументов в C# 2010, типа класса `Lazy<T>` и т.д.

Помимо описания новых компонентов и возможностей, в книге по-прежнему предоставляется весь необходимый базовый материал по языку C# в целом, основам объектно-ориентированного программирования (ООП), конфигурированию сборок, получению доступа к базам данных (через ADO.NET), а также процессу построения настольных приложений с графическим пользовательским интерфейсом, веб-приложений и распределенных систем (и многим другим темам).

Как и в предыдущих изданиях, в этом издании весь материал по языку программирования C# и библиотекам базовых классов .NET подается в дружественной и понятной читателю манере. Автор никогда не понимал, зачем другие технические авторы стараются писать свои книги так, чтобы те больше напоминали сложный научный труд, а не легкое для восприятия пособие. В новом издании основное внимание уделяется предоставлению информации, которой необходимо владеть для того, чтобы разрабатывать программные решения прямо сегодня, а не глубокому изучению малоинтересных эзотерических деталей.

Автор и читатели — одна команда

Авторам книг по технологиям приходится писать для очень требовательной группы людей. Всем известно, что детали разработки программных решений с помощью любой платформы (.NET, Java и COM) очень сложны и сильно зависят от отдела, компании, клиентской базы и поставленной задачи. Кто-то работает в сфере электронных публикаций, кто-то занимается разработкой систем для правительства и региональных органов власти, а кто-то сотрудничает с НАСА или военными департаментами. Что касается автора настоящей книги, то сам он занимается разработкой детского образовательного ПО (возможно, вам приходилось слышать об Oregon Trail или Amazon Trail), а также различных многоуровневых систем и проектов в медицинской и финансовой сфере. А это значит, что код, который придется писать читателю, скорее всего, будет иметь мало чего общего с тем, с которым приходится иметь дело автору.

Поэтому в настоящей книге автор специально старался избегать приведения примеров, свойственных только какой-то конкретной области производства или программирования. Из-за этого все концепции, связанные с C#, ООП, CLR и библиотеками базовых классов .NET, объясняются на общих примерах. В частности, здесь везде применяется одна и та же тема, которая так или иначе близка каждому — автомобили. Остальное остается за читателем.

Задача автора состоит в том, чтобы максимально доступно объяснить читателю язык программирования C# и основные концепции платформы .NET настолько хорошо, а также описать все возможные инструменты и стратегии, которые могут потребоваться для продолжения обучения по прочтении данной книги.

Задача читателя состоит в том, чтобы усвоить всю эту информацию и научиться применять ее на практике при разработке своих программных решений. Конечно, скорее всего, проекты, которые понадобится выполнять в будущем, не будут связаны с автомобилями и их дружественными именами, но именно в этом и состоит вся суть применения получаемых знаний на практике! После изучения представленных в этой книге концепций можно будет спокойно создавать решения .NET, удовлетворяющие требованиям любой среды программирования.

Краткий обзор содержания

Эта книга логически разделена на восемь частей, в каждой из которых содержится ряд взаимосвязанных между собой глав. Те, кто читал предыдущие издания данной книги, сразу же отметят ряд отличий. Например, новые средства языка C# больше не описываются в отдельной главе. Вместо этого они рассматриваются в тех главах, в которых их появление является вполне естественным. Кроме того, по просьбе читателей был значительно расширен материал по технологии Windows Presentation Foundation (WPF). Ниже приведено краткое описание содержимого каждой из частей и глав настоящей книги.

Часть I. Общие сведения о языке C# и платформе .NET

Назначением первой части этой книги является общее ознакомление читателя с природой платформы .NET и различными средствами разработки, которые могут применяться при построении приложений .NET (многие из которых распространяются с открытым исходным кодом), а также некоторыми основными концепциями языка программирования C# и системы типов .NET.

Глава 1. Философия .NET

Первая глава выступает в роли своего рода основы для изучения всего остального излагаемого в данной книге материала. В начале в ней рассказывается о традиционной разработке приложений Windows и недостатках, которые существовали в этой сфере ранее. Главной целью данной главы является ознакомление читателя с набором ключевых составляющих .NET: общеязыковой исполняющей средой (Common Language Runtime — CLR), общей системой типов (Common Type System — CTS), общеязыковой спецификацией (Common Language Specification — CLRS) и библиотеками базовых классов. Здесь читатель сможет получить первоначальное впечатление о том, что собой представляет язык программирования C#, и том, как выглядит формат сборок .NET, а также узнать о независимой от платформы природе .NET (о которой более детально рассказывается в приложении Б).

Глава 2. Создание приложений на языке C#

Целью этой главы является ознакомление читателя с процессом компиляции файлов исходного кода на C# с применением различных средств и методик. Будет показано, как использовать компилятор командной строки C# (`csc.exe`) и файлы ответов, а также различные редакторы кода и интегрированные среды разработки, в том числе Notepad++, SharpDevelop, Visual C# 2010 Express и Visual Studio 2010. Кроме того, вы узнаете о том, как устанавливать на машину разработки локальную копию документации .NET Framework 4.0 SDK.

Часть II. Главные конструкции программирования на C#

Темы, представленные в этой части книги, довольно важны, поскольку подходят для разработки приложений .NET любого типа (т.е. веб-приложений, настольных приложений с графическим пользовательским интерфейсом, библиотек кода и служб Windows). Здесь читатель ознакомится с основными конструкциями языка C# и некоторыми деталями объектно-ориентированного программирования (ООП), обработкой исключений на этапе выполнения, а также автоматическим процессом сборки мусора в .NET.

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

В этой главе начинается формальное изучение языка программирования C#. Здесь читатель узнает о роли метода `Main()` и многочисленных деталях работы с внутренними типами данных в .NET, в том числе — о манипуляциях текстовыми данными с помощью `System.String` и `System.Text.StringBuilder`. Кроме того, будут описаны итерационные конструкции и конструкции принятия решений, операции сужения и расширения, а также ключевое слово `unchecked`.

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

В этой главе завершается рассмотрение ключевых аспектов C#. Будет показано, как создавать перегруженные методы в типах и определять параметры с использованием ключевых слов `out`, `ref` и `params`. Также рассматриваются появившиеся в C# 2010 концепции именованных аргументов и необязательных параметры. Кроме того, будут описаны создание и манипулирование массивами данных, определение нулевых типов (с помощью операций `?` и `??`) и отличия *типов значения* (включающих перечисления и специальные структуры) от *ссылочных типов*.

Глава 5. Определение инкапсулированных типов классов

В этой главе начинается рассмотрение концепций объектно-ориентированного программирования (ООП) в языке C#. Вначале объясняются базовые понятия ООП (та-

кие как инкапсуляция, наследование и полиморфизм). Затем показано, как создавать надежные типы классов с применением конструкторов, свойств, статических членов, констант и доступных только для чтения полей. Наконец, рассматриваются частичные определения типов, синтаксис инициализации объектов и автоматические свойства.

Глава 6. Понятия наследования и полиморфизма

Здесь читатель сможет ознакомиться с двумя такими основополагающими концепциями ООП, как наследование и полиморфизм, которые позволяют создавать семейства взаимосвязанных типов классов. Будет описана роль виртуальных и абстрактных методов (а также абстрактных базовых классов) и *полиморфных интерфейсов*. И, наконец, в главе рассматривается роль одного из главных базовых классов в .NET — `System.Object`.

Глава 7. Структурированная обработка исключений

В этой главе рассматривается решение проблемы аномалий, возникающих в коде во время выполнения, за счет применения методики структурированной обработки исключений. Здесь описаны ключевые слова, предусмотренные для этого в C# (`try`, `catch`, `throw` и `finally`), а также отличия между исключениями уровня приложения и уровня системы. Кроме того, рассматриваются различные инструменты, предлагаемые в Visual Studio 2010, которые предназначены для проведения отладки исключений.

Глава 8. Время жизни объектов

В этой главе рассказывается об управлении памятью CLR-средой с использованием сборщика мусора .NET. Будет описана роль корневых элементов приложений, поколений объектов и типа `System.GC`. Будет показано, как создавать самоочищаемые объекты (с применением интерфейса `IDisposable`) и обеспечивать процесс финализации (с помощью метода `System.Object.Finalize()`). Вы узнаете о появившемся в .NET 4.0 классе `Lazy<T>`, который позволяет определять данные так, чтобы они не размещались в памяти до тех пор, пока вызывающая сторона не запросит их. Этот класс позволяет не загромождать память такими объектами, которые пока не требуются.

Часть III. Дополнительные конструкции программирования на C#

В этой части читателю предоставляется возможность углубить знания языка C# за счет изучения других более сложных (но очень важных) концепций. Здесь завершается ознакомление с системой типов .NET описанием типов делегатов и интерфейсов. Кроме того, описана роль обобщений и дано краткое введение в язык LINQ (Language Integrated Query). Также рассматриваются некоторые более сложные средства C# (такие как методы расширения, частичные методы и приемы манипулирования указателями).

Глава 9. Работа с интерфейсами

Материал этой главы предполагает наличие понимания концепций объектно-ориентированной разработки и посвящен программированию с использованием интерфейсов. Здесь будет показано, как определять классы и структуры, поддерживающие множество поведений, как обнаруживать эти поведения во время выполнения и как выборочно скрывать какие-то из них за счет *явной реализации интерфейсов*. Помимо создания специальных интерфейсов, рассматриваются вопросы реализации стандартных интерфейсов из состава .NET и их применения для построения объектов, которые могут сортироваться, копироваться, перечисляться и сравниваться.

Глава 10. Обобщения

Эта глава посвящена обобщениям. Программирование с использованием обобщений позволяет создавать типы и члены типов, содержащие заполнители, которые заполняются вызывающим кодом. В целом, обобщения позволяют значительно улучшить производительность приложений и безопасность в отношении типов. В главе рассматриваются типы обобщений из пространства имен `System.Collections.Generic`, а также показано, как создавать собственные обобщенные методы и типы (с ограничениями и без).

Глава 11. Делегаты, события и лямбда-выражения

Благодаря этой главе, станет понятно, что собой представляет тип *делегата*. Любой делегат в .NET представляет собой объект, который *указывает* на другие методы в приложении. С помощью делегатов можно создавать системы, позволяющие многочисленным объектам взаимодействовать между собой в обоих направлениях. После изучения способов применения делегатов в .NET, будет показано, как применять в C# ключевое слово `event`, которое упрощает манипулирование делегатами. Кроме того, рассматривается роль лямбда-операции (`=>`) и связь между делегатами, анонимными методами и лямбда-выражениями.

Глава 12. Расширенные средства языка C#

В этой главе описаны расширенные средства языка C#, в том числе перегрузка операций, создание специальных процедур преобразования (явного и неявного) для типов, построение и взаимодействие с *индексаторами типов*, работа с *расширяющими методами*, *анонимными типами*, *частичными методами*, а также указателями C# с использованием в коде контекста `unsafe`.

Глава 13. LINQ to Object

В этой главе начинается рассмотрение LINQ (Language Integrated Query — язык интегрированных запросов). Эта технология позволяет создавать строго типизированные *выражения запросов*, применять их к ряду различных целевых объектов LINQ и тем самым манипулировать *данными* в самом широком смысле этого слова. Глава посвящена API-интерфейсу LINQ to Objects, который позволяет применять LINQ-выражения к контейнерам данных (т.е. массивам, коллекциям и специальным типам). Эта информация будет полезна позже при рассмотрении других дополнительных API-интерфейсов, таких как LINQ to XML, LINQ to DataSet, PLINQ и LINQ to Entities.

Часть IV. Программирование с использованием сборок .NET

Эта часть книги посвящена деталям формата сборок .NET. Здесь вы узнаете не только о способах развертывания и конфигурирования библиотек кода .NET, но также о внутреннем устройстве двоичного образа .NET. Будет описана роль атрибутов .NET и определения информации о типе во время выполнения. Кроме того, рассматривается роль среды DLR (Dynamic Language Runtime — исполняющая среда динамического языка) в .NET 4.0 и ключевого слова `dynamic` в C# 2010. Наконец, объясняется, что собой представляют контексты объектов, как устроен CIL-код и как создавать сборки в памяти.

Глава 14. Конфигурирование сборок .NET

На самом высоком уровне термин *сборка* применяется для описания любого двоичного файла `*.dll` или `*.exe`, который создается с помощью компилятора .NET. В действительности возможности сборок намного шире. В этой главе будет показано, чем отличаются однофайловые и многофайловые сборки, как создавать и развертывать сборки обеих разновидностей, как делать сборки приватными и разделяемыми с использовани-

ем XML-файлов `*.config` и специальных сборок политик издателя. Кроме того, в главе описан глобальный кэш сборок (Global Assembly Cache — CAG), а также изменения CAG в версии .NET 4.0.

Глава 15. Рефлексия типов, позднее связывание и программирование с использованием атрибутов

В главе 15 продолжается изучение сборок .NET. В ней показано, как обнаруживать типы во время выполнения с использованием пространства имен `System.Reflection`. Типы из этого пространства имен позволяют создавать приложения, способные считывать метаданные сборки на лету. Кроме того, в главе рассматривается динамическая загрузка и создание типов во время выполнения с помощью *позднего связывания*, а также роль атрибутов .NET (стандартных и специальных). Для закрепления материала в конце главы приводится пример построения расширяемого приложения Windows Forms.

Глава 16. Процессы, домены приложений и контексты объектов

В этой главе рассказывается о создании загруженных исполняемых файлов .NET. Целью главы является иллюстрация отношений между процессами, доменами приложений и контекстными границами. Все эти темы подготавливают базу для изучения процесса создания многопоточных приложений в главе 19.

Глава 17. Язык CIL и роль динамических сборок

В этой главе подробно рассматривается синтаксис и семантика языка CIL, а также роль пространства имен `System.Reflection.Emit`, с помощью типов из которого можно создавать программное обеспечение, способное генерировать сборки .NET в памяти во время выполнения. Формально сборки, которые определяются и выполняются в памяти, называются *динамическими сборками*. Их не следует путать с динамическими типами, которые являются темой главы 18.

Глава 18. Динамические типы и исполняющая среда динамического языка

В .NET 4.0 появился новый компонент исполняющей среды .NET, который называется *исполняющей средой динамического языка* (Dynamic Language Runtime — DLR). С помощью DLR в .NET и ключевого слова `dynamic` в C# 2010 можно определять данные, которые в действительности разрешаются во время выполнения. Использование этих средств значительно упрощает решение ряда очень сложных задач по программированию приложений .NET. В главе рассматриваются некоторые практические способы применения динамических данных, включая более гладкое использование API-интерфейсы рефлексии .NET, а также упрощенное взаимодействие с унаследованными библиотеками COM.

Часть V. Введение в библиотеки базовых классов .NET

В этой части рассматривается ряд наиболее часто применяемых служб, поставляемых в составе библиотек базовых классов .NET, включая создание многопоточных приложений, файловый ввод-вывод и доступ к базам данных с помощью ADO.NET. Здесь также показано, как создавать распределенные приложения с помощью Windows Communication Foundation (WCF) и приложения с рабочими потоками, которые используют API-интерфейсы Windows Workflow Foundation (WF) и LINQ to XML.

Глава 19. Многопоточность и параллельное программирование

Эта глава посвящена созданию многопоточных приложений. В ней демонстрируется ряд приемов, которые можно применять для написания кода, безопасного в отноше-

нии потоков. В начале главы кратко напоминается о том, что собой представляет тип делегата в .NET для упрощения понимания предусмотренной в нем внутренней поддержки для асинхронного вызова методов. Затем рассматриваются типы пространства имен `System.Threading` и новый API-интерфейс TPL (Task Parallel Library — библиотека параллельных задач), появившийся в .NET 4.0. С применением этого API-интерфейса можно создавать .NET-приложения, распределяющие рабочую нагрузку среди доступных ЦП в исключительно простой манере. В главе также описан API-интерфейс PINQ (Parallel LINQ), который позволяет создавать масштабируемые LINQ-запросы.

Глава 20. Файловый ввод-вывод и сериализация объектов

Пространство имен `System.IO` позволяет взаимодействовать существующей структурой файлов и каталогов. В главе будет показано, как программно создавать (и удалять) систему каталогов и перемещать данные в различные потоки (файловые, строковые и находящиеся в памяти). Кроме того, рассматриваются службы .NET, предназначенные для сериализации объектов. *Сериализация* представляет собой процесс, который позволяет сохранять данные о состоянии объекта (или набора взаимосвязанных объектов) в потоке для использования в более позднее время, а *десериализация* — процесс извлечения данных о состоянии объекта из потока в память для последующего использования в приложении. В главе описана настройка процесса сериализации с применением интерфейса `ISerializable` и набора атрибутов .NET.

Глава 21. ADO.NET, часть I: подключенный уровень

В этой первой из трех посвященных базам данных главам дано введение в API-интерфейс ADO.NET. Рассматривается роль поставщиков данных .NET и взаимодействие с реляционной базой данных с применением так называемого *подключенного уровня* ADO.NET, который представлен объектами подключения, объектами команд, объектами транзакций и объектами чтения данных. В этой главе также приведен пример создания специальной базы данных и первой версии специальной библиотеки доступа к данным (`AutoLotDAL.dll`), неоднократно применяемой в остальных примерах книги.

Глава 23. ADO.NET, часть II: автономный уровень

В этой главе продолжается описание способов работы с базами данных и рассказывается об *автономном уровне* ADO.NET. Рассматривается роль типа `DataSet` и объектов адаптеров данных, а также многочисленных средств Visual Studio 2010, которые способны упростить процесс создания приложений, управляемых данными. Будет показано, как связывать объекты `DataTable` с элементами пользовательского интерфейса, а также как применять запросы LINQ к находящимся в памяти объектам `DataSet` с использованием API-интерфейса LINQ to DataSet.

Глава 23. ADO.NET, часть III: Entity Framework

В этой главе завершается изучение ADO.NET и рассматривается роль технологии Entity Framework (EF), которая позволяет создавать код доступа к данным с использованием строго типизированных классов, напрямую отображающихся на бизнес-модель. Здесь будут описаны роли таких входящих в состав EF компонентов, как службы объектов EF, клиент сущностей и контекст объектов. Будет показано устройство файла `*.edmx`, а также взаимодействие с реляционными базами данных с применением API-интерфейса LINQ to Entities. Кроме того, в главе создается последняя версия специальной библиотеки доступа к данным (`AutoLotDAL.dll`), которая будет использоваться в нескольких последних главах книги.

Глава 24. Введение в LINQ to XML

В главе 14 были даны общие сведения о модели программирования LINQ и об API-интерфейсе LINQ to Objects. В этой главе читателю предлагается углубить свои знания о технологии LINQ и научиться применять запросы LINQ к XML-документам. Сначала будут рассмотрены сложности, которые существовали в .NET первоначально в области манипулирования XML-данными, на примере применения типов из сборки `System.Xml.dll`. Затем будет показано, как создавать XML-документы в памяти, обеспечивать их сохранение на жестком диске и перемещаться по их содержимому с использованием модели программирования LINQ (LINQ to XML).

Глава 25. Введение в Windows Communication Foundation

В этой главе читатель узнает об API-интерфейсе Windows Communication Foundation (WCF), который позволяет создавать распределенные приложения симметричным образом, какими бы не были лежащие в их основе низкоуровневые детали. Будет показано, как создавать службы, хосты и клиентские приложения WCF. Службы WCF являются чрезвычайно гибкими, поскольку позволяют использовать для клиентов и хостов конфигурационные файлы на основе XML, в которых декларативно задаются необходимые адреса, привязки и контракты. Кроме того, рассматриваются полезные сокращения, которые появились в .NET 4.0.

Глава 26. Введение в Windows Workflow Foundation 4.0

API-интерфейс Windows Workflow Foundation (WF) вызывает больше всего путаницы у разработчиков-новичков. В версии .NET 4.0 первоначальный вариант API-интерфейса WF (появившийся в .NET 3.0) полностью переделан. В этой главе описана роль приложений, поддерживающих рабочие потоки, и способы моделирования бизнес-процессов с применением API-интерфейса WF 4.0. Рассматривается библиотека действий, поставляемая в составе WF 4.0, а также показано, как создавать специальные действия.

Часть VI. Построение настольных пользовательских приложений с помощью WPF

В .NET 3.0 был предложен замечательный API-интерфейс под названием Windows Presentation Foundation (WPF). Он быстро стал заменой модели программирования настольных приложений Windows Forms. WPF позволяет создавать настольные приложения с векторной графикой, интерактивной анимацией и операциями привязки данных с использованием декларативной грамматики разметки XAML. Более того, архитектура элементов управления WPF позволяет легко изменять внешний вид и поведение любого элемента управления с помощью правильно оформленного XAML-кода. В настоящем издании модели программирования WPF посвящено целых пять глав.

Глава 27. Введение в Windows Presentation Foundation и XAML

Технология WPF позволяет создавать чрезвычайно интерактивные и многофункциональные интерфейсы для настольных приложений (и косвенно для веб-приложений). В отличие от Windows Forms, в WPF множество ключевых служб (наподобие двухмерной и трехмерной графики, анимации, форматированных документов и т.п.) интегрируется в одну универсальную объектную модель. В главе предлагается введение в WPF и язык XAML (Extendable Application Markup Language — расширяемый язык разметки приложений). Будет показано, как создавать WPF-приложения без использования только кода без XAML, с использованием одного лишь XAML и с применением обоих подходов вместе, а также приведен пример создания специального XAML-редактора, который пригодиться при изучении остальных глав, посвященных WPF.

Глава 28. Программирование с использованием элементов управления WPF

В этой главе читатель научится работать с предлагаемыми в WPF элементами управления и диспетчерами компоновки. Будет показано, как создавать системы меню, разделители окон, панели инструментов и строки состояния. Также в главе рассматриваются API-интерфейсы (и связанные с ними элементы управления), входящие в состав WPF — Documents API, Ink API и модель привязки данных. В главе приводится начальное описание IDE-среды Expression Blend, которая значительно упрощает процесс создания многофункциональных пользовательских интерфейсов для приложений WPF.

Глава 29. Службы визуализации графики WPF

В API-интерфейсе WPF интенсивно используется графика, в связи с чем WPF предоставляет три пути визуализации графических данных — фигуры, рисунки и визуальные объекты. В главе подробно описаны все эти пути. Кроме того, рассматривается набор важных графических примитивов (таких как кисти, перья и трансформации), а применение Expression Blend для создания графики. Также показано, как выполнять операции проверки попадания (hit-testing) в отношении графических данных.

Глава 30. Ресурсы, анимация и стили WPF

В этой главе освещены три важных (и связанных между собой) темы, которые позволят углубить знания API-интерфейса Windows Presentation Foundation. В первую очередь рассказывается о роли *логических ресурсов*. Система логических (также называемых *объектными*) ресурсов позволяет назначать наиболее часто используемым в WPF-приложении объектам имена и затем ссылаться на них. Кроме того, будет показано, как определять, выполнять и управлять анимационной последовательностью. И, наконец, в главе рассматривается роль стилей в WPF. Подобно тому, как для веб-страниц могут применяться таблицы стилей CSS и механизм тем ASP.NET, в приложениях WPF для набора элементов управления может быть определен общий вид и поведение.

Глава 31. Шаблоны элементов управления WPF и пользовательские элементы управления

В этой главе завершается изучение модели программирования WPF и демонстрируется процесс создания специализированных элементов управления. Сначала рассматриваются две важных темы, связанные с созданием любого специального элемента — *свойства зависимости* и *маршрутизируемые события*. Затем описывается роль *шаблонов по умолчанию* и способы их просмотра в коде во время выполнения. Наконец, рассказывается о том, как создавать специальные классы `UserControl` с помощью Visual Studio 2010 и Expression Blend, в том числе и с применением .NET 4.0 Visual State Manager (VSM).

Часть VII. Построение веб-приложений с использованием ASP.NET

Эта часть посвящена деталям построения веб-приложений с применением API-интерфейса ASP.NET. Данный интерфейс был разработан Microsoft специально для предоставления возможности моделировать процесс создания настольных пользовательских интерфейсов путем наложения стандартного объектно-ориентированной, управляемой событиями платформы поверх стандартных запросов и ответов HTTP.

Глава 32. Построение веб-страниц ASP.NET

В этой главе начинается изучение процесса разработки веб-приложений с помощью ASP.NET. Как будет показано, вместо кода серверных сценариев теперь применяются самые настоящие объектно-ориентированные языки (наподобие C# и VB.NET). В главе

рассматривается типовой процесс создания веб-страницы ASP.NET, лежащая в основе модель программирования и другие важные аспекты ASP.NET, вроде того, как выбирать веб-сервер и работать с файлами `Web.config`.

Глава 33. Веб-элементы управления, мастер-страницы и темы ASP.NET

В отличие от предыдущей главы, посвященной созданию объектов `Page` из ASP.NET, в данной главе рассказывается об элементах управления, которые заполняют внутреннее дерево элементов ASP.NET. Здесь описаны основные веб-элементы управления ASP.NET, включая элементы управления проверкой достоверности, элементы управления навигацией по сайту и различные операции привязки данных. Кроме того, рассматривается роль *мастер-страниц* и механизма применения тем ASP.NET, который является серверным аналогом традиционных таблиц стилей.

Глава 34. Управление состоянием в ASP.NET

В этой главе рассматриваются разнообразные способы управления состоянием в .NET. Как и в классическом ASP, в ASP.NET можно создавать cookie-наборы и переменные уровня приложения и уровня сеанса. Кроме того, в ASP.NET есть еще одно средство для управления состоянием, которое называется кэшем приложения. Вдобавок в главе рассказывается о роли базового класса `HttpApplication` и демонстрируется динамическое переключение поведения веб-приложения с помощью файла `Web.config`.

Часть VIII. Приложения

В этой заключительной части книги рассматриваются две темы, которые не совсем вписывались в контекст основного материала. В частности, здесь кратко рассматривается более ранняя платформа Windows Forms для построения графических интерфейсов настольных приложений, а также использование платформы Mono для создания приложений .NET, функционирующих под управлением операционных систем, отличных от Microsoft Windows.

Приложение А. Программирование с помощью Windows Forms

Исходный набор инструментов для построения настольных пользовательских интерфейсов, который поставляется в рамках платформы .NET с самого начала, называется *Windows Forms*. В этом приложении описана роль этого каркаса и показано, как с его помощью создавать главные окна, диалоговые окна и системы меню. Кроме того, здесь рассматриваются вопросы наследования форм и визуализации двухмерной графики с помощью пространства имен `System.Drawing`. В конце приложения приводится пример создания программы для рисования (средней сложности), иллюстрирующий практическое применение всех описанных концепций.

Приложение Б. Независимая от платформы разработка .NET-приложений с помощью Mono

Приложение Б посвящено использованию распространяемой с открытым исходным кодом реализации платформы .NET под названием *Mono*. Она позволяет разрабатывать многофункциональные приложения .NET, которые можно создавать, развертывать и выполнять под управлением самых разных операционных систем, включая Mac OS X, Solaris, AIX и многочисленные дистрибутивы Linux. Так как Mono в основном эмулирует платформу .NET от Microsoft, ее функциональные возможности вполне очевидны. Поэтому в приложении основное внимание уделено не возможностям платформы Mono, а процессу ее установки, предлагаемым в ее составе инструментам для разработки, а также используемому механизму исполняющей среды.

Исходный код примеров

Исходный код всех рассматриваемых в настоящей книге примеров доступен для загрузки на сайте издательства по адресу:

`http://www.williamspublishing.com`

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. 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 интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

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

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

Наши координаты:

E-mail: `info@williamspublishing.com`

WWW: `http://www.williamspublishing.com`

Информация для писем из:

России: 127055, г. Москва, ул. Лесная, д. 43, стр. 1

Украины: 03150, Киев, а/я 152