Введение

Много лет тому назад (примерно в 2001 г.) я получил возможность написать книгу по предстоящей технологии Microsoft, которая на то время называлась NGWS (Next Generation Windows Software — программное обеспечение Windows нового поколения). Когда я начал исследовать исходный код, предоставленный Microsoft, то отметил многократные ссылки на язык программирования COOL (Common Object Oriented Language — общий объектно-ориентированный язык).

Пока я работал над своей первой рукописью книги *C# and the .NET Platform*, пользуясь предварительной альфа-сборкой (разумеется, без какой-либо документации), NGWS в конечном счете была переименована в платформу Microsoft .NET. И, как вы наверняка догадались, язык COOL нам сегодня известен под названием C#.

Первое издание этой книги вышло одновременно с версией .NET 1.0, бета 2. С тех пор я переписывал текст для учета многочисленных обновлений языка программирования C#, а также бурного появления новых API-интерфейсов в каждом новом выпуске платформы .NET.

С годами эта книга была очень хорошо принята прессой (финалист премии JOLT и книга года по программированию ReferenceWare), читателями и различными университетскими программами по вычислительной технике и разработке программного обеспечения.

Было просто замечательно общаться с читателями и преподавателями по всему миру. Спасибо вам всем за предложения, комментарии и (естественно) критику. Возможно, я не в состоянии ответить на каждое почтовое сообщение, но будьте уверены, что все они принимаются во внимание.

Автор и читатели — одна команда

Авторам книг по технологиям приходится писать для очень требовательной группы людей (я не могу не знать это, т.к. я один из них). Всем известно, что разработка программных решений с помощью любой платформы или языка очень сложна и сильно зависит 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 NASA или военными отраслями. Что касается меня, то я занимаюсь разработкой обучающего программного обеспечения для детей (Oregon Trail/Amazon Trail), различных многоуровневых систем и проектов в медицинской и финансовой сфере. Это значит, что код, который придется писать вам, скорее всего, будет иметь мало общего с кодом, с которым приходится иметь дело мне.

Таким образом, в этой книге я специально стараюсь избегать создания демонстраций, свойственных только конкретной отрасли или направлению программирования. Учитывая это, язык C#, объектно-ориентированное программирование, среда CLR и библиотеки базовых классов .NET объясняются на примерах, не привязанных к отрасли. В частности, здесь везде применяется одна и та же тема, так или иначе, близкая каждому — автомобили.

Моя работа, как автора, состоит в том, чтобы максимально доступно объяснить вам язык программирования C# и основные концепции платформы .NET. Кроме того, я также буду делать все возможное для снабжения вас инструментами и стратегиями, которые могут потребоваться для продолжения обучения по прочтении настоящей книги.

Ваша работа, как читателя, заключается в том, чтобы усвоить всю эту информацию и научиться применять ее на практике при разработке своих программных решений. Конечно, скорее всего, проекты, которые понадобится вам выполнять в будущем, не будут связаны с автомобилями и их дружественными именами, но именно в этом и состоит вся суть прикладных знаний.

После изучения представленных в этой книге тем и концепций вы сможете успешно строить решения .NET, удовлетворяющие требованиям конкретной среды программирования.

Краткий обзор книги

Эта книга логически разделена на восемь частей, каждая из которых содержит несколько взаимосвязанных между собой глав. Ниже приведено краткое содержание каждой из частей и глав настоящей книги.

Часть I. Введение в C# и платформу .NET

Часть I этой книги предназначена для ознакомления с природой платформы .NET и различными инструментами разработки (многие из которых распространяются с открытым исходным кодом), которые используются при построении приложений .NET.

Глава 1. Философия .NET

Первая глава выступает в качестве основы для всего остального материала. Главная ее цель заключается в том, чтобы ознакомить вас с набором строительных блоков .NET, таких как общеязыковая исполняющая среда (Common Language Runtime — CLR), общая система типов (Common Type System — CTS), общеязыковая спецификация (Common Language Specification — CLS) и библиотеки базовых классов. Здесь вы получите первоначальное представление о языке программирования C# и формате сборок .NET. Также вы узнаете о роли платформы .NET в рамках операционной системы Windows 8 и поймете разницу между приложением Windows 8 и приложением .NET.

Глава 2. Создание приложений на языке C#

Целью этой главы является ознакомление с процессом компиляции файлов исходного кода C# с применением различных средств и приемов. В начале главы будет показано, как использовать компилятор командной строки (`csc.exe`) и файлы ответов. Затем вы узнаете о многочисленных редакторах кода и интегрированных средах разработки, включая Notepad++, SharpDevelop, Visual C# Express и Visual Studio. Кроме того, будет показано, как установить на машине разработки локальную копию документации .NET Framework 4.5 SDK.

Часть II. Основы программирования на C#

Темы, представленные в этой части книги, исключительно важны, поскольку подходят для разработки приложений .NET любого типа (веб-приложений, настольных приложений с графическим пользовательским интерфейсом, библиотек кода или служб Windows). Здесь вы ознакомитесь с фундаментальными типами данных .NET, научитесь манипулировать текстом и узнаете о роли разнообразных модификаторов параметров C# (включая необязательные и именованные аргументы).

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

В этой главе начинается формальное изучение языка программирования C#. Здесь вы узнаете о роли метода `Main()` и многочисленных деталях, касающихся внутренних типов данных .NET, включая манипулирование текстовыми данными с использованием `System.String` и `System.Text.StringBuilder`. Кроме того, будут описаны итерационные конструкции и конструкции принятия решений, сужающие и расширяющие операции, 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 типов и определять параметры с использованием ключевых слов `out`, `ref` и `params`. Также рассматриваются два средства C#: *именованные* и *необязательные параметры*. Кроме того, будет описано создание и манипулирование массивами данных, определение типов, допускающих `null` (и операций ? и ??), и показаны отличия между *типами значений* (включающими перечисления и специальные структуры) и *ссылочными типами*.

Часть III. Объектно-ориентированное программирование на C#

В этой части вы освоите ключевые конструкции языка C#, в том числе *объектно-ориентированное программирование* (ООП). Здесь также будет показано, как обрабатывать исключения времени выполнения, и каким образом работать со строго типизированными интерфейсами.

Глава 5. Инкапсуляция

В этой главе начинается рассмотрение концепций объектно-ориентированного программирования на языке C#. После введения в основные принципы ООП (инкапсуляция, наследование и полиморфизм) будет показано, как создавать надежные типы классов с применением конструкторов, свойств, статических членов, констант и полей, предназначенных только для чтения. Глава завершается объяснением частичных определений типов, синтаксиса инициализации объектов и автоматических свойств.

Глава 6. Понятие наследования и полиморфизма

Здесь вы ознакомитесь с оставшимися двумя основными принципами ООП — наследованием и полиморфизмом, — которые позволяют строить семейства связанных типов классов. Вы узнаете о роли виртуальных и абстрактных методов (а также абстрактных базовых классов) и о природе полиморфных интерфейсов. И, наконец, в главе будет рассмотрена роль главного базового класса платформы .NET — `System.Object`.

Глава 7. Структурированная обработка исключений

В этой главе рассматривается решение проблемы аномалий, возникающих в коде во время выполнения, за счет применения *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, будут представлены различные инструменты в рамках Visual Studio, которые предназначены для отладки исключений, упущенных из виду.

Глава 8. Работа с интерфейсами

Материал этой главы предполагает наличие понимания концепций объектно-ориентированной разработки и посвящен *программированию на основе интерфейсов*. Здесь будет показано, как определять классы и структуры, поддерживающие множество по-

ведений, как обнаруживать эти поведения во время выполнения и как выборочно скрывать какие-то из них за счет явной реализации интерфейсов. В дополнение к созданию специальных интерфейсов, рассматриваются вопросы реализации стандартных интерфейсов из состава .NET и их применения для построения объектов, которые могут сортироваться, копироваться, перечисляться и сравниваться.

Часть IV. Дополнительные конструкции программирования на C#

В этой части книги вы получите возможность углубить знания языка C# за счет изучения других более сложных (но очень важных) концепций. Здесь завершается ознакомление с системой типов .NET описанием интерфейсов и делегатов. Кроме того, будет рассмотрена роль обобщений, дано краткое введение в язык LINQ (Language Integrated Query) и представлены некоторые более сложные функциональные возможности C# (такие как методы расширения, частичные методы и манипулирование указателями).

Глава 9. Коллекции и обобщения

Эта глава посвящена *обобщениям*. Вы увидите, что 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 не только рассматриваются различные обобщенные типы из пространства имен `System.Collections.Generic`, но также показано, как строить собственные обобщенные методы и типы (с ограничениями и без).

Глава 10. Делегаты, события и лямбда-выражения

В этой главе вы узнаете, что собой представляет тип *делегата*. Делегат .NET — это объект, который *указывает* на другие методы в приложении. С помощью делегатов можно создавать системы, позволяющие многочисленным объектам взаимодействовать между собой двухсторонним образом. После изучения способов применения делегатов в .NET будет показано, как использовать ключевое слово `event` в C#, которое упрощает манипулирование делегатами. Кроме того, рассматривается роль лямбда-операции (`=>`) в C# и связь между делегатами, анонимными методами и лямбда-выражениями.

Глава 11. Расширенные средства языка C#

В этой главе описаны расширенные средства языка C#, в том числе перегрузка операций и создание специальных процедур преобразования (явных и неявных) для типов. Кроме того, вы узнаете, как строить и взаимодействовать с *индексаторами типов* и работать с *расширяющими методами*, *анонимными типами*, *частичными методами* и указателями C#, используя контекст кода `unsafe`.

Глава 12. LINQ to Objects

В этой главе начинается рассмотрение языка интегрированных запросов LINQ (Language Integrated Query). Язык LINQ позволяет создавать строго типизированные *выражения запросов*, которые могут применяться к многочисленным целевым объектам LINQ для манипулирования *данными* в самом широком смысле этого слова. Глава посвящена API-интерфейсу LINQ to Objects, который позволяет применять выражения LINQ к контейнерам данных (например, 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.

Глава 13. Время жизни объектов

В финальной главе этой части объясняется, как среда CLR управляет памятью, используя сборщик мусора .NET. Вы узнаете о роли корневых элементов приложения, поколениях объектов и типе `System.GC`. После представления основ рассматриваются темы *освобождаемых объектов* (реализующих интерфейс `IDisposable`) и процесса финализации (с помощью метода `System.Object.Finalize()`). В главе также описан класс `Lazy<T>`, позволяющий определять данные, которые не будут размещаться вплоть до поступления запроса от вызывающего кода. Вы увидите, что эта возможность очень полезна, когда нежелательно загромождать кучу объектами, которые в текущий момент программе не нужны.

Часть V. 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) и ключевого слова `dynamic` в C#. Наконец, объясняются 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* используется для описания любого двоичного файла `*.dll` или `*.exe`, созданного с помощью компилятора .NET. Однако в действительности понятие сборки намного шире. В этой главе будет показано, чем отличаются однофайловые и многофайловые сборки, как создавать и развертывать сборки обеих разновидностей, как делать сборки закрытыми и разделяемыми с помощью XML-файлов `*.config` и специальных сборок политик издателя. Кроме того, в главе описана внутренняя структура глобального кеша сборок (Global Assembly Cache — CAG).

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

В главе 15 продолжается изучение сборок .NET. Здесь будет показано, как обнаруживать типы во время выполнения с использованием пространства имен `System.Reflection`. Типы из этого пространства имен позволяют строить приложения, способные считывать метаданные сборки на лету. Кроме того, в главе рассматривается динамическая загрузка и создание типов во время выполнения с применением *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. Динамические типы и среда DLR

В версии .NET 4.0 появился новый аспект исполняющей среды .NET, который называется *исполняющей средой динамического языка*. Используя DLR и ключевое слово `dynamic`, введенное в C# 2010, можно определять данные, которые не будут распознаваться вплоть до времени выполнения. Такие возможности существенно упрощают решение некоторых очень сложных задач программирования для .NET. В этой главе вы ознакомитесь с рядом практических применений 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

В этой главе подробно рассказывается о внутреннем устройстве загруженной исполняемой сборки .NET. Цель главы заключается в иллюстрации отношений между процессами, доменами приложений и контекстными границами. Все эти темы подготавливают базу для изучения процесса создания многопоточных приложений в главе 19.

Глава 18. Язык CIL и роль динамических сборок

Цель этой финальной главы в данной части двояка. В первой половине главы рассматривается синтаксис и семантика языка CIL, а во второй — роль пространства имен `System.Reflection.Emit`. Типы из этого пространства имен можно использовать 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 этому моменту вы уже должны хорошо знать основы языка C# и формат сборок .NET. Эта часть расширяет ваши знания исследованием ряда часто используемых 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 показано, как создавать распределенные приложения с применением Windows Communication Foundation (WCF) и приложения с рабочими потоками, которые используют API-интерфейс Windows Workflow Foundation (WF), а также описан API-интерфейс LINQ to XML.

Глава 19. Многопоточное, параллельное и асинхронное программирование

Эта глава посвящена построению многопоточных приложений. В ней демонстрируется ряд приемов, которые можно применять для написания кода, безопасного в отношении потоков. Глава начинается с краткого напоминания о том, что собой представляет тип делегата в .NET, и объяснения внутренней поддержки делегата для асинхронного вызова методов. Затем рассматриваются типы в пространстве имен `System.Threading`, а также *библиотека параллельных задач* (Task Parallel Library — TPL). С применением TPL разработчики .NET могут строить приложения, которые распределяют рабочую нагрузку по всем доступным центральным процессорам в исключительно простой манере. В главе также описана роль API-интерфейса PINQ (Parallel LINQ), который предлагает способ создания запросов LINQ, масштабируемых среди множества процессорных ядер. В завершение главы рассматриваются некоторые новые ключевые слова C#, появившиеся в .NET 4.5, которые интегрируют асинхронные вызовы методов непосредственно в язык.

Глава 20. Файловый ввод-вывод и сериализация объектов

Пространство имен `System.IO` позволяет взаимодействовать с существующей структурой файлов и каталогов. В этой главе будет показано, как программно создавать (и удалять) систему каталогов. Вы также узнаете, каким образом перемещать данные в и из разнообразных потоков (файловых, строковых и находящихся в памяти). Кроме того, в главе рассматриваются службы сериализации объектов платформы .NET. *Сериализация* позволяет сохранить состояние объекта (или набора связанных объектов) в потоке для последующего использования. *Десериализация* — это процесс извлечения объекта из потока в память для потребления в приложении. После описания основ вы узнаете, как настраивать процесс 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 представлено введение в API-интерфейс доступа к базам данных платформы .NET, который называется ADO.NET. В частности, рассматривается роль поставщиков данных .NET и взаимодействие с реляционной базой данных с использованием *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 главах книги.

Глава 22. ADO.NET, часть II: автономный уровень

В этой главе изучение способов работы с базами данных продолжается рассмотрением *автономного уровня* ADO.NET. Здесь вы узнаете о роли типа `DataSet` и объектов адаптеров данных. Кроме того, вы ознакомитесь с многочисленными инструментами Visual Studio, которые могут существенно упростить создание приложений, управляемых данными. Будет показано, как привязывать объекты `DataTable` к элементам пользовательского интерфейса, и как применять запросы LINQ к находящимся в памяти объектам `DataSet`, используя API-интерфейс LINQ to DataSet.

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

В этой главе изучение ADO.NET завершается рассмотрением роли инфраструктуры Entity Framework (EF), которая позволяет создавать код доступа к данным с использованием строго типизированных классов, напрямую отображающихся на бизнес-модель. Здесь будут описаны роли службы объектов EF, клиента сущностей и контекста объектов, а также показано устройство файла `*.edmx`. Кроме того, вы узнаете, как взаимодействовать с реляционными базами данных с применением LINQ to Entities. В главе также создается финальная версия специальной библиотеки доступа к данным (`AutoLotDAL.dll`), которая будет использоваться в нескольких оставшихся главах книги.

Глава 24. Введение в LINQ to XML

В главе 14 была представлена модель программирования LINQ — в частности, LINQ to Objects. В этой главе вы углубите свои знания языка LINQ, научившись применять запросы LINQ к XML-документам. Первым делом, будут описаны сложности с манипулированием XML-данными, которые существовали в .NET изначально, когда использовались типы из сборки `System.Xml.dll`. Затем будет показано, как создавать XML-документы в памяти, сохранять их на жестком диске и перемещаться по их содержимому посредством модели программирования LINQ (LINQ to XML).

Глава 25. Введение в Windows Communication Foundation

До этого места в книге все примеры приложений запускались на единственном компьютере. В этой главе вы узнаете об API-интерфейсе *Windows Communication Foundation* (WCF), который позволяет создавать распределенные приложения в симметричной манере, независимо от лежащих в их основе низкоуровневых деталей. Будет показано, как конструировать службы, хосты и клиенты WCF. Как вы увидите, службы WCF являются чрезвычайно гибкими, поскольку предоставляют клиентам и хостам возможность использования конфигурационных файлов на основе XML, в которых декларативно указываются адреса, привязки и контракты.

Глава 26. Введение в Windows Workflow Foundation

В этой главе вы узнаете о роли приложений, поддерживающих рабочие потоки, и ознакомитесь со способами моделирования бизнес-процессов с применением API-интерфейса WF, введенного в версии .NET 4.0. Кроме того, будет описана библиотека действий WF и показано, как создавать специальные действия, которые используют специальную библиотеку доступа к данным, созданную ранее в книге.

Часть VII. Windows Presentation Foundation

В .NET 3.0 программистам был предложен замечательный API-интерфейс под названием *Windows Presentation Foundation* (WPF). Он быстро стал заменой модели программирования настольных приложений Windows Forms. В сущности, WPF позволяет строить настольные приложения с векторной графикой, интерактивной анимацией и операциями привязки данных, используя декларативную грамматику разметки *XAML*. Более того, архитектура элементов управления WPF позволяет легко изменять внешний вид и поведение любого элемента управления с помощью правильно оформленной разметки XAML.

Глава 27. Введение в Windows Presentation Foundation и XAML

Инфраструктура WPF позволяет создавать исключительно интерактивные и многофункциональные пользовательские интерфейсы для настольных приложений (и косвенно для веб-приложений). В отличие от Windows Forms, в WPF множество ключевых служб (вроде двухмерной и трехмерной графики, анимации, форматированных документов и т.п.) интегрируются в единую унифицированную объектную модель. В этой главе предлагается введение в WPF и язык XAML (Extendable Application Markup Language — расширяемый язык разметки приложений). Вы узнаете, как создавать WPF-приложения вообще без XAML, с использованием только XAML и с применением комбинации обоих подходов. В завершение главы рассматривается пример построения специального редактора XAML, который будет использоваться в остальных главах, посвященных WPF.

Глава 28. Программирование с использованием элементов управления WPF

В этой главе будет показано, как работать с предлагаемыми WPF элементами управления и диспетчерами компоновки. Вы узнаете, как 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 — Documents API, Ink API и модель привязки данных.

Глава 29. Службы визуализации графики WPF

В API-интерфейсе WPF интенсивно используется графика, и с учетом этого WPF предоставляет три подхода к визуализации графических данных: *фигуры*, *рисунки и геометрии* и *визуальные объекты*. В этой главе вы ознакомитесь с каждым подходом и изучите несколько важных графических примитивов (таких как кисти, перья и трансформации). Также вы узнаете, как выполнять операции проверки попадания 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 вы узнаете о роли *логических ресурсов*. Система логических ресурсов (также называемых *объектными ресурсами*) предлагает способ именования и ссылки на часто исполь-

зуемые объекты внутри WPF-приложения. Затем вы научитесь определять, выполнять и управлять *анимационной* последовательностью. Вы увидите, что применение анимации WPF не ограничивается видеоиграми или мультимедиа-приложениями. И, наконец, 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 набора элементов управления.

Глава 31. Свойства зависимости, маршрутизируемые события и шаблоны

Эта глава начинается с рассмотрения двух важных тем, связанных с созданием специальных элементов управления: *свойства зависимости* и *маршрутизируемые события*. Затем описывается роль *стандартного шаблона* и способы его программного просмотра во время выполнения. В завершение главы объясняется, как строить специальные шаблоны.

Часть VIII. ASP.NET Web Forms

Эта часть посвящена построению веб-приложений с использованием API-интерфейса ASP.NET. Технология ASP.NET предназначена для моделирования процесса создания настольных пользовательских интерфейсов путем наложения управляемой событиями, объектно-ориентированной инфраструктуры поверх стандартного запроса/ответа HTTP.

Глава 32. Введение в ASP.NET Web Forms

В этой главе начинается изучение процесса разработки веб-приложений с помощью ASP.NET. Вы увидите, что код сценариев серверной стороны теперь заменен кодом на настоящих объектно-ориентированных языках программирования (например, C# и VB.NET). Здесь рассматривается конструирование веб-страницы ASP.NET, лежащая в основе модель программирования и другие ключевые аспекты ASP.NET, такие как выбор веб-сервера и работа с файлами `web.config`.

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

В то время как предыдущая глава была посвящена созданию объектов `Page` из ASP.NET, в этой главе рассказывается об элементах управления, которые наполняют внутреннее дерево элементов управления. Здесь описаны основные *веб-элементы управления*, включая элементы управления проверкой достоверности, элементы управления навигацией по сайту и различные операции привязки данных. Кроме того, рассматривается роль *мастер-страниц* и *механизма тем* ASP.NET, который является альтернативой серверной стороны традиционным таблицам стилей.

Глава 34. Управление состоянием в ASP.NET

Эта глава дополняет ваши знания ASP.NET описанием разнообразных способов управления состоянием в .NET. Как и в классическом ASP, в ASP.NET можно создавать cookie-наборы, а также переменные уровня приложения и уровня сеанса. Кроме того, в ASP.NET имеется еще один прием управления состоянием: *кеш приложения*. После исследования технологий управления состоянием с помощью ASP.NET рассматривается роль базового класса `HttpApplication` и демонстрируется динамическое переключение поведения веб-приложения с помощью файла `web.config`.

Загружаемые приложения

На сайте издательства для загрузки доступны два дополнительных приложения. Первое из них посвящено основам API-интерфейса Windows Forms, который используется в нескольких примерах построения пользовательского интерфейса, рассмотренных

в книге. Во втором приложении рассматривается независимая от платформы природа .NET на примере платформы Mono.

Загружаемое приложение А. Программирование с помощью Windows Forms

Первоначальный набор инструментов для построения настольных графических пользовательских интерфейсов, который поставлялся в рамках платформы .NET, называется *Windows Forms*. В этом приложении описана роль этой инфраструктуры пользовательских приложений и показано, как с ее 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.

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

Это приложение посвящено использованию распространяемой с открытым исходным кодом реализации платформы .NET под названием *Mono*. Платформу Mono можно применять для построения многофункциональных приложений .NET, которые допускается создавать, развертывать и выполнять под управлением разнообразных операционных систем, включая Mac OS X, Solaris и многочисленные дистрибутивы Linux. Учитывая, что Mono по большому счету сравнимо с платформой .NET от Microsoft, вы уже знаете большинство из того, что она предлагает. Поэтому в данном приложении основное внимание уделяется процессу установки Mono, инструментам разработки Mono, а также механизму исполняющей среды Mono.

Исходный код примеров

Исходный код всех рассматриваемых в настоящей книге примеров доступен для загрузки на веб-сайте издательства.

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 веб-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