# Введение

Первое издание этой книги появилось вместе с выходом второй бета-версии .NET 1.0 летом 2001 г. С тех пор автор с удовольствием и благодарностью наблюдал за тем, как эта работа пользовалась популярностью в прессе и, самое главное, среди читателей. Через несколько лет она даже была номинирована на премию Jolt Award (которую, увы, заполучить так и не удалось) и на премию Reference Excellence Award в категории лучшей книги по программированию за 2003 г. (которую удалось-таки выиграть).

С тех пор автор постоянно обновлял книгу, чтобы она не утрачивала актуальность, выпуская все новые и новые издания, в том числе и вышедшее ограниченным тиражом специальное издание, в котором был представлен материал по технологиям .NET 3.0 (Windows Presentation Foundation, Windows Communication Foundation и Windows Workflow Foundation), а также предложен краткий обзор нескольких готовящихся к выходу технологий, теперь известных под общим названием LINQ.

В текущем четвертом издании, которое вы сейчас держите в руках, учтены все серьезные изменения, что появились в .NET 3.5. Здесь автор не только добавил несколько совершенно новых глав, но и значительно расширил многие из глав, предлагаемых ранее.

Как и в предыдущих изданиях, в этом издании весь материал по языку программирования C# и библиотекам базовых классов .NET подается в очень дружественной и понятной 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 сегодня, а не глубокому изучению эзотерических деталей, которые на самом деле мало кому интересны.

## Автор и читатель — 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, J2EE, COM и т.д.) очень 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 локальных органов власти, а кто-то сотрудничает с НАСА или военными департаментами. Что касается автора настоящей книги, то он сам занимается разработкой детского образовательного ПО, различных многоуровневых систем и проектов в медицинской и финансовой сфере. А это значит, что код, который приходиться писать читателю на месте его работы, скорее всего, имеет мало чего общего с тем, с которым имеет дело автор.

Поэтому в настоящей книге специально не использовались примеры, отражающие специфику конкретной области производства или программирования. Концепции, связанные с C#, ООП, CLR и библиотеками базовых классов .NET, объясняются на общих, не привязанных ни к какой области действия примерах. Вместо того, чтобы приводить для каждого примера свою таблицу данных, вычислять выплаченные суммы и т.п., было отдано предпочтение теме, которая так или иначе близка всем, а именно — автомобилям. Остальное остается за читателем.

*Задача автора* состоит в том, что объяснить читателю, что собой представляет язык программирования C# и основные концепции платформы .NET настолько хорошо, насколько возможно, а также описать все возможные инструменты и стратегии, которые могут потребоваться читателю для продолжения его 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 читателю, не будут связаны с автомобилями и их дружественными именами, но именно в этом и состоит вся суть применения получаемых знаний на практике! Читатель может быть уверен, что после изучения представленных в этой книге концепций он сможет спокойно создавать решения .NET, удовлетворяющие требованиям конкретной среде программирования.

# Краткий обзор содержания

Эта книга логически поделена на восемь частей, в каждой из которых содержится ряд взаимосвязанных между собой глав. Тем, кому приходилось читать предыдущие издания данной книги, сразу же заметят массу отличий. Например, некоторые темы (наподобие конструкций C#, объектно-ориентированного программирования и разработки независящих от платформы приложений .NET) были расширены до нескольких глав. Более того, в этом издании еще добавилось несколько совершенно новых глав, посвященных специально функциональным возможностям, которые появились в .NET 3.0–3.5 (LINQ, WCF, WPF, WF и т.д.). Ниже приведено кратко описание содержимого каждой из частей и глав.

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

Задачей первой части этой книги является общее ознакомление читателя с природой платформы .NET и различными средствами разработки, которые можно применять 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 основы для всего остального излагаемого в данной книге материала. В начале рассказывается о традиционной разработке приложений Windows и недостатках, которые существовали в этой сфере ранее. Главной целью этой главы является ознакомление читателя с рядом ключевых компонентов .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 получит первое впечатление о том, что собой представляет язык программирования C# и как выглядит формат сборок .NET, а также узнает о независимой от платформы природе .NET (о которой еще более детально рассказывается в приложении Б).

### *Глава 2. Создание приложений на языке C#*

Задачей этой главы является ознакомление читателя с процессом компиляции файлов исходного кода на C# с помощью различных средств и методик. Прежде всего, читатель узнает о том, как использовать компилятор командной строки (`csc.exe`) и ответные файлы C#. В остальной части главы будут описаны различные редакторы кода и интегрированные среды разработки (IDE), а именно — TextPad, Notepad++,

SharpDevelop, Visual C# 2008 Express и Visual Studio 2008, а также дополнительные средства программирования, которые любой разработчик .NET должен всегда иметь под рукой.

# Часть II. Главные конструкции программирования на C#

Темы, представленные в этой части книги, довольно важны, поскольку применимы при разработке программ .NET любого типа (веб-приложений, настольных приложений с графическим пользовательским интерфейсом, библиотек кода, служб Windows и т.д.). Здесь читатель ознакомится с главными конструкциями языка C#, а также деталями объектно-ориентированного программирования (ООП). Также в этой части будет говориться о способах обработки исключений во время выполнения и деталях выполняемого в .NET автоматически процесса сборки мусора.

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

В этой главе начинается формальное изучение языка программирования C#. Здесь читатель узнает о роли, которую играет метод `Main()`, и многочисленных деталях работы с внутренними типами данных в .NET, в том числе и о том, как манипулировать текстовыми данными с помощью `System.String` и `System.Text.StringBuilder`. Кроме того, здесь будет показано, как в C# выглядят итерационные конструкции и конструкции принятия решений, операции сужения и расширения, а также применение ключевого слова `unchecked`.

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

В этой главе завершается изучение ключевых аспектов C#. Вначале читатель узнает о том, как создавать перегруженные методы в типах и определять параметры с использованием ключевых слов `out`, `ref` и `params`. Потом будет показано, как создавать и манипулировать массивами данных, как определять нулевые типы (операциями `?` и `??`) и чем отличаются *типы-значения* (в том числе перечисления и специальные конструкции) и *ссылочные типы*.

## *Глава 5. Определение инкапсулированных типов классов*

В этой главе начинается изучение концепций объектно-ориентированного программирования (ООП) на языке C#. Вначале рассказывается о базовых понятиях ООП (таких как инкапсуляция, наследование и полиморфизм), а затем о том, как создавать надежные типы классов за счет применения конструкторов, свойств, статических членов, констант и доступных только для чтения полей. Напоследок приводится краткий обзор процесса определения частичных типов и синтаксиса для документирования XML-кода, который можно использовать в C#.

## *Глава 6. Понятия наследования и полиморфизма*

Здесь читатель сможет поближе познакомиться с двумя такими важными средствами ОПП, как наследование и полиморфизм, которые позволяют создавать семейства взаимосвязанных типов классов. По ходу главы будет рассказано о роли виртуальных и абстрактных методов (а также абстрактных базовых классов) и о том, что собой представляет *полиморфный интерфейс*. И, наконец, напоследок будет описана роль самого главного базового класса в .NET — `System.Object`.

### Глава 7. Структурированная обработка исключений

Целью этой главы является описание того, как справляться с возникающими в коде во время выполнения аномалиями за счет применения методики структурированной обработки исключений. Здесь читатель узнает не только о том, какие ключевые слова доступны в C# для решения подобных проблем (`try`, `catch`, `throw` и `finally`), но и о том, чем исключения уровня приложения отличаются от системных исключений. Помимо этого будут описаны различные инструменты, которые предлагаются в Visual Studio 2008 и позволяют производить отладку исключений, которые по той или иной причине ускользнули от глаз разработчика.

### Глава 8. Время жизни объектов

В этой главе рассказывается о том, как CLR-среда управляет памятью с помощью сборщика мусора .NET. Здесь читатель узнает о роли корневых элементов приложений, поколений объектов и типа `System.GC`, а также о создании самоочищаемых объектов (посредством интерфейса `IDisposable`) и обеспечении выполнения процесса финализации (с помощью метода `Object.Finalize()`).

## Часть III. Дополнительные конструкции программирования на C#

В этой части читателю предоставляется возможность углубить свои знания языка C# за счет изучения ряда более совершенных (и очень важных) концепций. Здесь можно будет завершить ознакомление с системой типов .NET, изучив последние оставшиеся типы, а именно — типы делегатов и интерфейсов, а также узнать о роли, которую играют обобщения. Также описаны многочисленные новые функциональные возможности, появившиеся в C# 2008, и даны вводные сведения о технологии LINQ.

### Глава 9. Работа с интерфейсами

Материал, излагаемый в этой главе, предполагает наличие навыков по разработке объектов и посвящен программированию интерфейсов. Здесь читатель узнает, как определять типы, чтобы они поддерживали несколько видов поведения, как получать информацию об этих видах поведения во время выполнения и как выборочно скрывать некоторые из них за счет применения *явной реализации интерфейсов*. Помимо ознакомления с рядом предопределенных типов интерфейсов .NET, также будет показано, как использовать специальные интерфейсы и тем самым создавать нестандартную архитектуру событий.

### Глава 10. Коллекции и обобщения

Эта глава начинается с рассмотрения типов коллекций, предлагаемых в составе пространства имен `System.Collections`, которое поставляется в виде части платформы .NET, начиная с самой первой версии. После выхода .NET 2.0, однако, язык программирования C# стал поддерживать и так называемые *обобщения*. Как читатель сможет увидеть в этой главе, программирование с использованием обобщений позволяет значительно улучшить производительность приложений и повысить степень их безопасности в отношении типов. Здесь можно не только ознакомиться с различными обобщенными типами, которые предлагаются в составе пространства имен `System.Collections.Generic`, но и узнать о том, как создавать свои собственные обобщенные методы и типы (как с, так и без ограничений).

### Глава 11. Делегаты, события и лямбда-выражения

Эта глава посвящена рассмотрению типов *делегатов*. Попросту говоря, любой делегат в .NET представляет собой объект, который "указывает" на какие-то другие методы в приложении. За счет применения делегатов можно создавать системы, позволяющие многочисленным объектам взаимодействовать между собой в обоих направлениях. После изучения того, как применяются делегаты в .NET, читатель познакомится с ключевым словом `event`, которое используется в C# для упрощения манипулирования делегатами во время программирования. И, наконец, в завершение главы будет показана роль предлагаемой в C# 2008 лямбда-операции (=>) и описана связь между делегатами, анонимными методами и лямбда-выражениями.

### Глава 12. Индексаторы, операции и указатели

В этой главе читателю предлагается углубить знания языка программирования C# и ознакомиться с рядом усовершенствованных методик программирования. Здесь можно будет узнать о том, как перегружать операции и создавать специальные (явные неявные) процедуры преобразования для типов, а взаимодействовать с *индексаторами типов* и манипулировать указателями в стиле C за счет использования "небезопасного" контекста в коде.

### Глава 13. Средства языка C# 2008

С выходом версии .NET 3.5 язык C# был усовершенствован так, чтобы поддерживать огромное количество новых конструкций программирования, многие из которых служат для обеспечения доступа к API-интерфейсу LINQ (см. главу 14). В частности, в этой главе читатель сможет узнать о том, какую роль играет неявная типизация локальных переменных, частичные методы, автоматические свойства, методы расширений, анонимные типы и синтаксис инициализации объектов.

### Глава 14. Введение в LINQ

В этой главе начинается рассмотрение технологии LINQ, которую можно считать самой интригующей из тех, что предлагаются в .NET 3.5. Как будет показано в этой главе, LINQ позволяет создавать строго типизированные *выражения запросов*, применять их к целому ряду различных целевых объектов LINQ и тем самым манипулировать данными в самом широком смысле этого слова. Здесь читатель узнает об API-интерфейсе LINQ to Objects, который позволяет применять LINQ-выражения к контейнерам данных (массивам, коллекциям и специальным типам). Эта информация пригодится позже при рассмотрении способов применения LINQ-выражений к реляционным базам данных (с помощью LINQ to ADO) и XML-документам (посредством LINQ to XML) в главе 24.

## Часть IV. Программирование с использованием сборок .NET

Эта часть книги посвящена изучению деталей, касающихся формата сборок .NET. Здесь читатель узнает не только о том, как развертывать и конфигурировать библиотеки кода .NET, но и как вообще в .NET устроен двоичный образ внутри. В этой части рассказывается о роли в .NET атрибутов и о создании многопоточных приложений. В более поздних главах рассматриваются некоторые дополнительные и более сложные темы вроде контекста объектов, CIL-кода и динамических сборок.

### Глава 15. Сборки .NET

На самом высоком уровне "сборка" представляет собой термин, применяемый для описания любого управляемого двоичного файла `*.dll` или `*.exe`. Однако на самом деле

возможности сборок куда богаче. Здесь читатель узнает о том, в чем состоит разница между однофайловыми и многофайловыми сборками, и как создавать и развертывать сборки каждого из этих видов. Еще будет показано, как конфигурировать приватные и разделяемые сборки с помощью основанных на XML файлов `*.config` и специальных сборок политик издателя. Также можно будет ознакомиться с внутренним устройством глобального кэша сборок (Global Assembly Cache — CAG) и ролью утилиты .NET Framework Configuration.

### Глава 16. Рефлексия типов, позднее связывание и программирование с использованием атрибутов

В главе 16 продолжается изучение сборок .NET и рассказывается о том, как обнаруживать типы во время выполнения с использованием пространства имен `System.Reflection`. За счет применения типов из этого пространства имен можно создавать приложения, способные считывать метаданные сборки на лету. Кроме того, будет показано, как динамически загружать и создавать типы во время выполнения за счет использования методики *позднего связывания*, а также то, какую роль в .NET играют атрибуты (как стандартные, так и специальные). Чтобы можно было оценить пользу от всего этого на практике, в конце главы приводится пример построения расширяемого приложения Windows Forms.

### Глава 17. Процессы, домены приложений и контексты объектов

К этому моменту у читателя должно уже сложиться четкое представление о сборках, поэтому в данной главе осуществляется переход на следующий уровень и рассказывается о том, как создавать загружаемые исполняемые файлы .NET. Целью этой главы является иллюстрация взаимоотношений между процессами, доменами приложений и контекстными границами. Все эти темы обеспечивают надлежащую основу для изучения материала следующей главы, которая посвящена созданию многопоточных приложений.

### Глава 18. Создание многопоточных приложений

В этой главе речь идет о создании многопоточных приложений и иллюстрируется ряд приемов, которые можно использовать для написания кода, безопасного в отношении типов. В начале главы вкратце рассказывается о том, что собой представляет тип делегата в .NET, чтобы было легче разобраться во внутренней поддержке асинхронного вызова методов. Далее рассматриваются типы, предлагаемые в составе пространства имен `System.Threading`. Здесь читатель сможет ознакомиться с многочисленными типами (`Thread`, `ThreadStart` и т.д.), которые позволяют создавать дополнительные потоки выполнения. В завершении главы приводится описание такого важного типа, как `BackgroundWorker`, который значительно упрощает создание потоков внутри настольного пользовательского интерфейса.

### Глава 19. Язык CIL и роль динамических сборок

В последней главе этой части рассматриваются две взаимосвязанных темы. В частности, рассматривается синтаксис и семантика языка CIL гораздо более подробно, чем это было в предыдущих главах, а затем — роль пространства имен `System.Reflection.Emit`. С помощью типов из этого пространства имен можно создавать программное обеспечение, способное генерировать сборки .NET в памяти прямо во время выполнения. Формально сборки, которые определяются и выполняются в памяти, называются *динамическими сборками*.

# Часть V. Введение в библиотеки базовых классов .NET

К этому моменту читатель должен довольно неплохо разбираться в языке C# и деталях формата сборок .NET, потому в данной части предлагается изучить ряд наиболее часто применяемых служб из тех, что поставляются в составе библиотек базовых классов .NET, наподобие службы файлового ввода-вывода и службы получения доступа к базам данных через ADO.NET. Еще в этой части речь идет о создании распределенных приложений с помощью Windows Communication Foundation (WCF) и приложений с рабочими потоками, предусматривающими применение API-интерфейса Windows Workflow Foundation (WF).

## *Глава 20. Файловый ввод-вывод и изолированное хранилище*

Пространство имен `System.IO` позволяет взаимодействовать с имеющейся на машине структурой файлов и каталогов. В ходе этой главы читатель узнает о том, как программно создавать (и уничтожать) систему каталогов, а также перемещать данные в и из различных потоков (файловых, строковых, памяти и т.д.). Позже в этой главе можно будет ознакомиться с методикой изолированного хранения, которая позволяет сохранять данные каждого пользователя в безопасной "песочнице", какими бы ни были настройки безопасности на целевой машине. Для понимания определенных аспектов API-интерфейса `System.IO.IsolatedStorage`, вниманию читателя также будет представлен и краткий обзор системы безопасности доступа кода (Code Access Security — CAS).

## *Глава 21. Введение в сериализацию объектов*

В этой главе рассказывается о службах сериализации объектов, которые поставляются в составе платформы .NET. Попросту говоря, *сериализация* позволяет сохранять данные о состоянии объекта (или целого ряда взаимосвязанных объектов) в потоке для использования в более позднее время, а *десериализация*, соответственно (как и следовало ожидать), представляет собой процесс извлечения данных о состоянии объекта из потока в память для последующего использования в приложении. После изучения основных деталей читатель также узнает и о том, как настраивать процесс сериализации с помощью интерфейса `ISerializable` и ряда атрибутов .NET.

## *Глава 22. ADO.NET, часть I: подключенный уровень*

В этой первой из двух посвященных базам данных главам читатель познакомится с API-интерфейсом ADO.NET. В частности, в главе будет описана роль поставщиков данных .NET и способы взаимодействия с реляционной базой данных с помощью *подключенного уровня* ADO.NET, представленного объектами соединения, объектами команд, объектами транзакций и объектами чтения данных. Следует иметь в виду, что в этой главе приводится пример создания специальной базы данных и библиотеки доступа к данным, которые будут применяться не раз в остальных главах книги.

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

В этой главе продолжается рассмотрение способов работы с базами данных и рассказывается об *автономном уровне* ADO.NET. Здесь читатель узнает о том, какую роль играет тип `DataSet` и объекты адаптеров данных, а также, какие многочисленные средства предлагаются в Visual Studio для упрощения процесса создания управляемых данными приложений. В ходе главы также будет показано, как объекты `DataTable` можно связывать с элементами пользовательского интерфейса, подобными `GridView`, который предлагается в составе API-интерфейса Windows Forms.

### Глава 24. Программирование с использованием API-интерфейсов LINQ

В главе 14 давалось введение в модель программирования LINQ, в частности, в API-интерфейс LINQ to Objects. Здесь читателю предлагается углубить свои знания о технологии LINQ и изучить способы применения LINQ-запросов к реляционным базам данных, объектам `DataSet` и XML-документам. По ходу данной главы читатель также сможет узнать о том, какую роль играют объекты контекста данных, чем может быть полезна утилита `sqlmetal.exe` и какие аспекты в Visual Studio 2008 касаются LINQ.

### Глава 25. Введение в Windows Communication Foundation

В версии .NET 3.0 появился совершенно новый API-интерфейс Windows Communication Foundation (WCF), позволяющий создавать распределенные приложения симметричным образом, какими бы не были лежащие в их основе низкоуровневые детали. В данной главе читатель узнает, как выглядит процесс создания WCF-службы, хостов и клиентов, и увидит, что WCF-службы являются чрезвычайно гибкими, поскольку позволяют использовать для клиентов и хостов основанные на XML конфигурационные файлы и декларативным образом указывать в них необходимые адреса, привязки и контракты.

### Глава 26. Введение в Windows Workflow Foundation

Помимо WCF, в версии .NET 3.0 появился новый API-интерфейс Windows Workflow Foundation (WF), позволяющий определять, выполнять и осуществлять мониторинг *рабочих потоков* и тем самым моделировать сложные бизнес-процессы. В настоящей главе читатель узнает о предназначении Windows Workflow Foundation в общем, а также о роли действий, конструкторов рабочих потоков, механизма выполнения рабочих потоков и поддерживающих рабочие потоки библиотек кода.

## Часть VI. Настольные пользовательские интерфейсы

Довольно часто новички ошибочно полагают, что платформа .NET представляет собой каркас, предназначенный для создания только таких пользовательских интерфейсов, которые могут работать в веб-сети (чему виной, надо полагать, служит присутствие в ее названии слова ".NET", наталкивающего на мысль об Интернете и, следовательно, веб-программах). В .NET действительно предоставляется замечательная поддержка для создания веб-приложений, но также поставляется и два графических каркаса Windows Forms и Windows Presentation Foundation (WPF), которые позволяют создавать традиционные настольные пользовательские интерфейсы; их рассмотрению как раз посвящена эта часть.

### Глава 27. Программирование с использованием Windows Forms

Исходный набор инструментов для построения настольных пользовательских интерфейсов, который поставляется в составе платформы .NET, называется *Windows Forms*. В данной главе рассказано о роли этого каркаса при построении пользовательских интерфейсов и показано, как с его помощью создавать главные окна, диалоговые окна и системы меню. Также здесь описана роль, которую играет наследование форм, и продемонстрировано, как визуализировать двухмерные графические данные с помощью пространства имен `System.Drawing`. Для иллюстрации всех этих тем на конкретном примере в конце главы предложен конкретный процесс построения (не слишком сложного в плане функциональных возможностей) приложения для рисования.

### Глава 28. Введение в Windows Presentation Foundation и XAML

В версии .NET 3.0 был предложен совершенно новый набор инструментов для построения графических пользовательских интерфейсов под названием *Windows Presentation Foundation* (WPF). По сути, WPF позволяет создавать чрезвычайно интерактивные и богатые различными средствами интерфейсные части для настольных приложений (и, косвенно, для веб-приложений). В отличие от Windows Forms, этот сверхмощный каркас интегрирует ряд ключевых служб (наподобие двухмерной и трехмерной графики, эффектов анимации, форматированных документов и т.п.) в одну универсальную объектную модель. В данной главе читатель начнет изучать WPF и язык XAML (Extendable Application Markup Language — расширяемый язык разметки приложений). В частности, здесь будет показано, как создавать WPF-программы без использования XAML, с использованием только XAML и с применением комбинированного подхода. В завершении главы для примера описан процесс создания специального средства для просмотра XAML-данных, которое будет не раз встречаться в остальных главах, посвященных WPF.

### Глава 29. Программирование с использованием элементов управления WPF

В этой главе читатель научится работать с предлагаемыми в модели WPF элементами управления, а также изучит ряд связанных с ними аспектов, наподобие свойств зависимости и маршрутизируемых событий. Как не трудно догадаться, эта глава посвящена описанию способов работы с различными элементами управления WPF, однако, что более интересно, в ней объясняются способы использования диспетчеров компоновки, команд управления и поддерживаемой WPF модели привязки данных.

### Глава 30. Визуализация двухмерной графики, ресурсы и темы WPF

В этой заключительной главе данной части завершается обзор WPF, и рассматриваются три относительно не связанных между собой темы. Службы визуализации графики в WPF обычно требуют определения специальных ресурсов. С помощью этих ресурсов затем можно легко генерировать специальные анимационные эффекты, а посредством этих анимационных эффектов вместе с графикой и ресурсами создавать специальные темы для WPF-приложений. Для иллюстрации в конце главы приведен наглядный пример применения специальных графических тем во время выполнения.

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

Часть VII посвящена рассмотрению деталей построения веб-приложений с использованием API-интерфейса ASP.NET. Как здесь будет показано, ASP.NET был специально разработан для моделирования процесса создания настольных пользовательских интерфейсов наложением поверх стандартного объектно-ориентированного, управляемого событиями каркаса запросов и ответов HTTP.

### Глава 31. Построение веб-страниц ASP.NET

В этой главе начинается изучение основ разработки веб-приложений с помощью ASP.NET. Как читатель увидит, серверные сценарии теперь заменили самые настоящие объектно-ориентированные языки (C#, VB .NET и т.п.). В данной главе рассказывается о процессе создания веб-страниц ASP.NET, лежащей в основе модели программирования и некоторых других важных аспектах ASP.NET, наподобие выбираемого веб-сервера и применения файлов `Web.config`.

### Глава 32. Веб-элементы управления, темы и мастер-страницы ASP.NET

Если в предыдущей главе рассказывалось о создании объектов `Page` в ASP.NET, то в этой главе речь идет об элементах управления, которые предлагаются в составе внутреннего дерева элементов управления ASP.NET. Здесь читатель ознакомится с некоторыми наиболее важными для веб-приложений элементами управления ASP.NET, в том числе элементами управления для проверки достоверности, элементами управления для внутренней навигации по сайту и различными операциями для связывания данных. Вдобавок в главе читатель узнает о роли, которую играют *мастер-страницы*, и механизме применения тем в ASP.NET, которые являются своего рода серверным аналогом традиционных таблиц стилей.

### Глава 33. Управление состоянием в ASP.NET

В этой главе читателю предлагается расширить текущие знания об ASP.NET и изучить различные способы управления состоянием в .NET. Подобно ASP, ASP.NET позволяет легко создавать cookie-наборы, а также переменные уровня приложения и уровня сеанса. Однако в ASP.NET поддерживается еще одно средство для управления состоянием — кэш приложения. После изучения многочисленных способов работы с данными состояния в ASP.NET, читатель узнает также и о том, какую роль играет базовый класс `System.HttpApplication` (применяемый в файле `Global.asax`) и как динамически изменять поведение веб-приложения во время выполнения с помощью файла `Web.config`.

## Часть VIII. Приложения

В этой заключительной части книги рассматриваются две очень важных темы, которые не совсем вписывались в контекст основного материала и потому были представлены в форме приложений. Здесь читатель сможет завершить свое изучение языка C# и платформы .NET, узнав о том, как интегрировать в .NET-приложения унаследованный код, а также как выносить разработку .NET-приложений за рамки семейства операционных систем Windows.

### Приложение А. Возможность взаимодействия COM и .NET

Те, кому приходилось создавать программы для операционной системы Windows до перехода на .NET, наверняка знакомы с технологией COM. Хотя у COM и .NET нет ничего общего (кроме разве факта, что обе эти технологии являются продуктом Microsoft), в составе платформы .NET все равно поставляется отдельное пространство имен (`System.Runtime.InteropServices`), которое позволяет делать так, чтобы программы .NET могли работать с компонентами COM и наоборот. В данном приложении читатель сможет детально узнать о том, как обеспечить подобный уровень взаимодействия, поскольку эта тема играет важную роль, когда дело доходит до необходимости использования в новых .NET-приложениях унаследованного кода.

### Приложение Б. Независимая от платформы разработка .NET с помощью Mono

И, наконец, в приложении Б речь идет об использовании реализации платформы .NET с открытым исходным кодом под названием *Mono*. Эта реализация позволяет создавать такие многофункциональные приложения .NET, которые могут создаваться, развертываться и выполняться в среде самых различных операционных систем, наподобие Mac OS X, Solaris, AIX и многочисленных дистрибутивов Linux. Из-за того, что Mono по большей части эмулирует предлагаемую Microsoft платформу .NET, ее функциональные

возможности являются вполне очевидными. Поэтому в данном приложении основное внимание уделяется рассмотрению не возможностей Mono, а процесса инсталляции Mono, предлагаемых в составе Mono инструментов для разработки, а также исполняющей среде Mono.

# Исходный код примеров

Исходный код примеров, рассмотренных в книге, а также дополнительные главы, не вошедшие в печатное издание, доступны 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