Введение

Язык C# и связанную с ним среду .NET Framework можно без преувеличения назвать самой значительной из предлагаемых в настоящее время технологий для разработчиков. Среда .NET является такой средой, которая была создана для того, чтобы в ней можно было разрабатывать практически любое приложение для запуска в Windows, а C# является языком программирования, который был специально создан для использования в .NET Framework. Например, с применением C# и .NET Framework можно создавать динамические веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возможностями для работы в оперативном и автономном режимах. В настоящей книге рассматривается версия .NET Framework 4. Для тех, кто по-прежнему использует какую-то из предшествующих версий .NET Framework, некоторые разделы этой книги не подойдут. Мы стараемся всегда специально уведомлять об элементах, которые являются новыми и присутствуют только в .NET Framework 4.

Не стоит поддаваться заблуждению из-за наличия в названии Framework слова "NET" и думать, что данная среда предназначена только для создания приложений, ориентированных на Интернет. Слово "NET" здесь является лишь показателем того, что, по мнению Microsoft, *распределенные приложения*, в которых обработка распределяется между клиентом и сервером, являются шагом вперед. Однако важно понимать, что C# представляет собой язык, предназначенный не только для написания приложений, способных работать в Интернете и в сети. Он предоставляет средства для кодирования практически любого типа программного обеспечения или компонентов для платформы Windows. Язык C# и среда .NET привели к революционным изменениям в способе написания разработчиками программ и сделали программирование приложений для Windows гораздо более простым, чем когда-либо.

Так что же такого важного в .NET и C#?

Важность .NET и C#

Для понимания важности .NET не помешает вспомнить о природе многих технологий Windows, которые появились в последние примерно 18 лет. Хотя на первый взгляд все они могут выглядеть довольно разными, на самом деле все операционные системы Windows, начиная с Windows 3.1 (которая вышла в 1992 г.) и заканчивая Window 7 и Windows Server 2008 R2, в основе своей имеют один и тот же хорошо знакомый API-интерфейс Windows. По мере появления новых версий Windows в этот API-интерфейс добавлялось много новых функций, но это был скорее процесс совершенствования и расширения API-интерфейса, а не его замена.

То же самое можно сказать и о многих других технологий и каркасах, которые применялись для разработки программного обеспечения, ориентированного на Windows. Например, технология *COM* (Component Object Model – объектная модель компонентов) первоначально называлась технологией *OLE* (Object Linking and Embedding – связывание и внедрение объектов) и по большей части представляла собой средство для связывания различных типов документов Office, например, для размещения в документе Word таблицы Excel. После этого она эволюционировала в технологию COM, затем в *DCOM* (Distributed COM – распределенная объектная модель компонентов) и, наконец, в сложную технологию *COM+*, которая стала основой для обеспечения связи между всеми компонентами, а также реализации транзакций, служб обмена сообщениями и организации пула объектов.

Подобный подход с совершенствованием программного обеспечения был выбран Microsoft по вполне очевидной причине, а именно — для обеспечения обратной совместимости. За многие годы независимыми разработчиками была написана масса программ для Windows, и ОС Windows не была бы такой успешной, если бы при каждом добавлении новой технологии существующая кодовая база нарушалась.

Хотя наличие обратной совместимости является важным свойством технологий Windows и одной из сильных сторон платформы Windows, с ним также связан и один серьезный недостаток. При каждом совершенствовании какой-нибудь технологии и добавлении в нее новых возможностей, из-за необходимости в наличии обратной совместимости она получается немного сложнее, чем была раньше.

Из-за этого со временем стало ясно, что нужно что-то менять. В Microsoft не могли до бесконечности расширять одни и те же языки и средства для разработки, постоянно делая их все более и более сложными для удовлетворения конфликтующих между собой потребностей в поддержке новейшего оборудования и обеспечении обратной совместимости с тем, что было в ходу, когда Windows впервые стала популярной в начале 90-х годов прошлого века. Настал момент начать с чистого листа и создать простой, но при этом совершенный набор языков, сред и средств разработки, который бы позволял разработчиками легко писать современное программное обеспечение.

Для этого первоначально и были предназначены язык C# и среда .NET. Грубо говоря, .NET представляет собой платформу или API-интерфейс для программирования на платформе Windows. Вместе с .NET Framework язык C# был разработан с нуля специально для работы в .NET, а также для объединения всех достижений, которые были сделаны в средах разработки, и всех принципов объектно-ориентированного программирования, которые были выведены за последние 25 лет.

Прежде чем продолжить, следует отметить, что обратная совместимость при этом не утратилась. Существующие программы все равно будут продолжать работать, потому что в .NET предусмотрена возможность для работы с существующим программным обеспечением. В настоящее время связь между программными компонентами в Windows осуществляется практически полностью за счет использования технологии COM. С учетом этого в .NET Framework предлагается и возможность для создания оболочек вокруг существующих компонентов COM и тем самым позволения компонентам .NET взаимодействовать с ними.

Изучать язык C# для того, чтобы писать код для .NET, вовсе не обязательно. В Microsoft расширили возможности языка C++, а также внесли значительные изменения в Visual Basic, сделав его более мощным языком. Это позволяет писать код для среды .NET на любом из этих языков. Однако языки C++ и Visual Basic являются результатом их совершенствования на протяжении многих лет, а не созданными заново с учетом современных технологий.

В настоящей книге показано, как программировать на C#, а также описана работа самой архитектуры .NET. Здесь рассматриваются не только фундаментальные аспекты языка C#, но и приводятся примеры приложений, предусматривающих использование и многих других связанных технологий, в том числе доступа к базам данных, создания динамических веб-страниц, добавления усовершенствованной графики и доступа к каталогам.

Преимущества .NET

Пока что лишь в общем говорилось о том, насколько замечательной является технология .NET, но ничего конкретно о том, каким образом она облегчает жизнь разработчикам. Поэтому в настоящем разделе кратко перечисляются некоторые из наиболее совершенных функциональных возможностей .NET.

- **Объектно-ориентированное программирование**. И .NET Framework, и C# изначально основаны на принципах объектно-ориентированного программирования.
- **Хороший дизайн**. Поставляемая библиотека базовых классов построена полностью с нуля и является интуитивно понятной.

- **Независимость от языка**. В .NET код, написанный на любом языке — Visual Basic, C# или управляемом C++, — компилируется в код на промежуточном языке (*Intermediate Language* — IL). Это делает языки способными к взаимодействию в невиданной до сих пор мере.
- **Усовершенствованная поддержка для создания динамических веб-страниц**. Хотя в классической технологии ASP предлагалась довольно высокая степень гибкости, ее все равно не хватало из-за необходимости использования интерпретируемых сценарных языков, а отсутствие объектно-ориентированного дизайна часто приводило к получению довольно запутанного кода ASP. В .NET предлагается интегрированная поддержка для создания веб-страниц с помощью ASP.NET. В случае применения ASP.NET код создаваемых страниц поддается компиляции и может быть написан на любом поддерживающем .NET языке высокого уровня, например, C# или Visual Basic 2010. В новой версии .NET эта поддержка улучшилась еще больше, сделав возможным применение новейших технологий вроде Ajax и jQuery.
- **Эффективный доступ к данным**. Набор компонентов .NET, известный под общим названием ADO.NET, позволяет получать эффективный доступ к реляционным базам данных и многим другим источникам данных. Также предлагаются компоненты, позволяющие получать доступ к файловой системе и каталогам. В частности, в .NET встроена поддержка XML, позволяющая манипулировать данными, импортируемыми и экспортируемыми на платформы, отличные от Windows.
- **Разделение кода**. В .NET был полностью переделан способ разделения кода между приложениями за счет введения понятия *сборки* (assembly) вместо традиционных библиотек DLL. Сборки обладают формальными средствами для управления версиями и допускают одновременное существование рядом нескольких различных версий сборок.
- **Повышенная безопасность**. В каждой сборке может содержаться встроенная информация о безопасности, указывающая, какому конкретно пользователю или группе пользователей либо процессу разрешено вызывать методы из тех или иных классов. Это позволяет очень точно настраивать возможный способ применения развертываемых сборок.
- **Установка с нулевым воздействием**. Сборки бывают двух типов: разделяемые и приватные. Разделяемые сборки представляют собой обычные библиотеки, доступные всему программному обеспечению, а приватные сборки предназначены для использования только с определенными программами. Приватные сборки являются полностью самодостаточными, поэтому процесс их установки выглядит просто. Никакие записи в системный реестр не добавляются; все нужные файлы просто размещаются в соответствующей папке файловой системы.
- **Поддержка для создания веб-служб**. В .NET предлагается полностью интегрированная поддержка для разработки веб-служб таким же простым образом, как и приложений любых других типов.
- **Visual Studio 2010**. Вместе с .NET поставляется среда разработки Visual Studio 2010, которая способна одинаково хорошо справляться как с кодом на языке C++, C# и Visual Basic 2010, так и с кодом ASP.NET или XML. В Visual Studio 2010 интегрированы все наилучшие возможности сред конкретных языков из всех предыдущих версий этой замечательной IDE-среды.
- **C#**. Язык C# представляет собой мощный и популярный объектно-ориентированный язык, предназначенный специально для применения вместе с .NET.

Преимущества архитектуры .NET более подробно рассматриваются в главе 1.

Что нового в .NET Framework 4

Первая версия .NET Framework (1.0) была выпущена в 2002 г. и встречена с большим энтузиазмом. Версия .NET Framework 2.0 вышла в 2005 г. и получила статус серьезного выпуска. Версия .NET Framework 4 является еще одним серьезным выпуском данного продукта с множеством замечательных новых возможностей.

В каждом выпуске .NET Framework в Microsoft всегда старались делать так, чтобы изменений, нарушающих целостность предыдущего разработанного кода, было как можно меньше. Пока что им удавалось довольно успешно справляться с этой задачей.

В следующем разделе описаны некоторые изменения, появившиеся в C# 2010 и .NET Framework 4.

Динамическая типизация

В мире программирования наблюдается значительный рост применения динамических языков, таких как JavaScript, Python и Ruby. По этой причине в C# была добавлена возможность динамической типизации (dynamic typing). Знать статическим образом, какими объекты могут получаться в конце, не всегда возможно. Теперь вместо использования ключевого слова `object` и назначения этого типа всем сущностям можно предоставить возможность решать этот вопрос среде DLR (Dynamic Language Runtime – исполняющая среда динамического языка) непосредственно во время выполнения.

Динамические возможности C# обеспечивают лучшее взаимодействие. Появляется возможность взаимодействовать с различными динамическими языками и работать с DOM гораздо более простым образом. Кроме того, облегчается работа с API-интерфейсами COM для Microsoft Office.

Среда DLR входит в состав версии .NET Framework 4. Среда DLR построена на основе среды CLR (Common Language Runtime – общеязыковая исполняющая среда) для предоставления возможности связывать вместе все взаимодействие с динамическими языками.

Доступ к новой среде DLR в C# получается с помощью нового ключевого слова `dynamic`. Это ключевое слово служит флагом для компилятора; при каждой встрече с ним компилятор будет понимать, что речь идет о динамическом, а не обычном статическом вызове.

Необязательные и именованные параметры

Необязательные и именованные параметры уже некоторое время присутствовали в Visual Basic, но в C# до выхода версии .NET 4 доступны не были. Необязательные параметры позволяют предоставлять используемые по умолчанию значения для некоторых параметров методов, а также обеспечивать разновидность перегрузки потребителем при наличии даже одного метода для обработки всех вариантов. Например:

```
public void CreateUser(string firstname, string lastname,
  bool isAdmin, bool isTrialUser)
{
}
```

Если необходимо обеспечить возможность перегрузки этого метода и применение значений по умолчанию для двух объектов `bool`, можно было бы легко добавить еще несколько методов, подставляющих эти значения для потребителя, и затем сделать вызов главного метода для выполнения фактической работы. С помощью необязательных параметров теперь можно поступить следующим образом:

```
public void CreateUser(string firstname, string lastname,
  bool isAdmin = false, bool isTrialUser = true)
{
}
```

В этом фрагменте кода для параметров `firstname` и `lastname` значения по умолчанию не устанавливаются, а для `isAdmin` и `isTrailUser` — устанавливаются. Благодаря этому, потребитель кода теперь сможет делать такие вызовы:

```
myClass.CreateUser("Bill", "Evjen");
myClass.CreateUser("Bill", "Evjen", true);
myClass.CreateUser("Bill", "Evjen", true, false);
myClass.CreateUser("Bill", "Evjen", isTrailUser: false);
```

В последнем вызове используются именованные параметры, которые тоже являются нововведением C# 2010. Именованные параметры потенциально будут изменять способ написания кода. Они позволят делать код более удобным для чтения и понимания. Чтобы увидеть пример их применения в действии, возьмем метод `File.Copy()` из пространства имен `System.IO`. Обычно он создается примерно так:

```
File.Copy(@"C:\myTestFile.txt", @"C:\myOtherFile.txt", true);
```

В данном случае этот простой метод работает с тремя параметрами, но какие элементы на самом деле передаются методу `Copy()`? Не зная этот метод вдоль и поперек, сказать, что в нем происходит, просто взглянув на него, довольно трудно. Именованные параметры позволяют указывать в коде перед предоставляемым значением имя параметра, например:

```
File.Copy(sourceFileName: @"C:\myTestFile.txt",
  destFileName: @"C:\myOtherFile.txt", overwrite: true);
```

Теперь благодаря наличию именованных параметров стало легче читать и понимать, что происходит в данной строке кода. Применение именованных параметров никакого влияния на скомпилированный код не оказывает; оно имеет значение лишь при написании кода для приложения.

Ковариантность и контравариантность

Возможности ковариантности (covariance) и контравариантности (contravariance) предлагались и в предыдущих версиях .NET Framework, но в версии .NET Framework 4 они были расширены таким образом, чтобы лучше функционировать с обобщениями, делегатами и прочими элементами. В предыдущих версиях .NET контравариантность можно было использовать с объектами и массивами, но нельзя, например, с обобщенными интерфейсами. В .NET 4 это стало возможным.

Технология ASP.NET MVC

Технология ASP.NET MVC, ставшая последним серьезным добавлением в ASP.NET, вызвала большую шумиху в сообществе разработчиков. Она предоставляет средства для создания приложений ASP.NET с использованием шаблона “модель–представление–контроллер” (Model-View-Controller), которых давно ожидали многие разработчики. ASP.NET MVC обеспечивает разработчиков возможностями тестирования, гибкости и обслуживания создаваемых ими приложений. Важно понимать, что ASP.NET MVC не предназначена служить заменой хорошо известной технологии ASP.NET, а является просто другим способом построения приложений.

В данном выпуске ASP.NET приложения можно создавать с использованием этой новой модели. Технология ASP.NET MVC полностью встроена в .NET Framework и Visual Studio.

Для чего подходит C#

В определенном смысле C# по отношению к языкам программирования представляет собой то же самое, что .NET по отношению к среде Windows. Точно так же, как Microsoft

добавляла все больше и больше средств к Windows и Windows API в течение последних полутора десятилетий, так и Visual Basic 2010 и C++ подвергались постоянному расширению. Хотя Visual Basic и C++ в результате этого стали весьма мощными языками, оба они страдают от проблем, связанных с "тяжелым наследием" их эволюции.

В случае Visual Basic 6 и ранних версий сильная сторона языка определялась тем фактом, что его было легко понять, и он значительно облегчал решение многих задач программирования, скрывая от разработчиков детали Windows API и инфраструктуру компонентов COM. Отрицательная же сторона состояла в том, что Visual Basic никогда не был по-настоящему объектно-ориентированным, так что крупные приложения быстро теряли внутреннюю организацию, и их становилось трудно сопровождать. К тому же, поскольку синтаксис Visual Basic был унаследован от ранних версий BASIC (который, в свою очередь, был задуман скорее как простой для понимания начинающими программистами язык, нежели для написания крупных коммерческих приложений), на самом деле он не был приспособлен для построения хорошо структурированных объектно-ориентированных программ.

С другой стороны, корни C++ относятся к определению языка ANSI C++. Он не полностью совместим со стандартом ANSI (хотя и близок к нему) по той простой причине, что Microsoft впервые разработала свой компилятор C++ до того, как определение ANSI стало официальным. К сожалению, это привело к двум проблемам. Во-первых, корни ANSI C++ лежат в технологиях двадцатилетней давности, поэтому ему не хватает поддержки современных концепций (таких как строки Unicode и генерация XML-документации), и он содержит некоторые архаичные синтаксические структуры, предназначенные для компиляторов прошлых лет (такие как отделение объявления от определения функций-членов). Во-вторых, разработчики Microsoft одновременно пытаются развивать C++, чтобы он стал языком, предназначенным для выполнения высокопроизводительных задач под Windows, а потому вынуждены были добавить к нему огромное количество специфичных для Microsoft ключевых слов и разнообразных библиотек. В результате язык C++ для Windows стал чрезвычайно "запутанным". Попробуйте спросить разработчиков C++, сколько определений строковых типов они знают: `char*`, `LPTSTR`, `string`, `CString` (версия MFC), `CString` (версия WTL), `wchar_t*`, `OLECHAR*` и так далее и тому подобное...

Теперь появилась .NET — полностью новая среда, которая повлекла за собой появление новых расширений обоих языков. Microsoft добавила еще больше специфичных ключевых слов с C++, и провела полную реконструкцию Visual Basic, приведя его к текущей версии Visual Basic 2010 — языку, который унаследовал некоторый базовый синтаксис VB, но дизайн которого полностью отличается от того, к чему мы привыкли; он практически стал совершенно новым языком.

В упомянутом контексте в Microsoft решили предложить разработчикам альтернативу — язык, ориентированный специально на .NET, к тому же спроектированный с "чистого листа". Так и появился язык C#. Официально в Microsoft описывают C# как "простой, современный, объектно-ориентированный и безопасный к типам язык программирования, производный от C и C++". Большинство независимых обозревателей, вероятно, изменили бы это на "производный от языков C, C++ и Java". Такое описание является технически точным, но мало что говорит о красоте и элегантности языка. Синтаксически C# очень похож на C++ и Java, в том смысле, что многие ключевые слова являются теми же; кроме того, C# также разделяет с языками C++ и Java ту же блочную структуру с фигурными скобками для выделения блоков кода и точками с запятой для завершения операторов. Первое впечатление от фрагмента кода C# состоит в том, что он выглядит подобно C++ или Java. Но, несмотря на это внешнее сходство, C# изучить намного легче, чем C++, и по сложности освоения он примерно равен Java. Его дизайн в большей степени соответствует современным инструментам разработки, чем у обоих его предшественников, и он предлагает простоту в использовании, как Visual Basic, вместе с высокой производительностью и низкоуровневым доступом к памяти, которые характерны для C++, когда это необходимо.

Ниже перечислены возможности C#, которые следует отметить особо.

- Полная поддержка классов и объектно-ориентированного программирования, включая наследование реализации и интерфейсов, виртуальные функции и перегрузку операций.
- Согласованный и четко определенный набор базовых типов.
- Встроенная поддержка автоматической генерации XML-документации.
- Автоматическая очистка динамически распределяемой памяти.
- Средство маркировки классов и методов пользовательскими атрибутами. Это может быть полезно для документирования и может иметь некоторый эффект при компиляции (например, помеченные методы могут компилироваться только для отладочных сборок).
- Полная поддержка библиотеки базовых классов .NET наряду с легким доступом к Windows API (если вы действительно в этом нуждаетесь, что случается нечасто).
- Указатели и прямой доступ в память при необходимости доступны, но язык спроектирован так, что в большинстве случаев без них можно обойтись.
- Поддержка свойств и событий в стиле Visual Basic.
- Простым изменением опций компилятора можно собирать либо исполняемые программы, либо библиотеки компонентов .NET, которые могут быть вызваны из стороннего кода — так же, как это делается с элементами управления Active X (COM-компонентами).
- Возможность использования для написания динамических веб-страниц ASP.NET и веб-служб XML.

Бо́льшая часть перечисленного также касается Visual Basic 2010 и управляемого C++. Тот факт, что язык C# изначально спроектирован для работы с .NET, однако, означает, что он поддерживает средства .NET в более полной мере и предлагает в этом контексте более подходящий синтаксис, чем прочие языки. Хотя язык C# сам по себе очень похож на Java, есть несколько существенных преимуществ. В частности, язык Java не предназначен для работы в среде .NET.

Прежде чем завершить тему, следует отметить и ряд ограничений C#. Одна область, для которой данный язык не предназначен — это критичный ко времени или исключительно высокопроизводительный код — когда приходится заботиться о том, чтобы цикл занимал 1000 или 1050 тактов процессора, либо когда необходимо очищать ресурсы в течение считанных миллисекунд после того, как отпадает потребность в них. Вероятно, C++ продолжает оставаться самым непревзойденным из высокоуровневых языков в этой области. C# недостает некоторых ключевых средств для построения наиболее высокопроизводительных приложений, включая возможность спецификации встроенных функций и деструкторов, которые гарантированно запускаются в определенной точке кода. Однако пропорциональное отношение таких приложений к их общему числу чрезвычайно низко.

Что необходимо для написания и выполнения кода на C#

Версия .NET Framework 4 будет работать в среде Windows XP, Windows Server 2003, Windows 7 и новейшей версии Windows Server 2008 R2. Для того чтобы писать код с использованием .NET, понадобится установить комплект .NET 4 SDK.

Помимо этого, если только не планируется писать код на C# с помощью какого-то простого текстового редактора или другой среды для разработки, практически наверняка по-

надобится установить Visual Studio 2010. Для выполнения управляемого кода полный комплект SDK не нужен, но исполняющая среда .NET необходима. Вам может понадобиться распространять эту исполняющую среду вместе с разработанным кодом для тех клиентов, у которых она еще не установлена.

Как организована эта книга

Книга начинается с обзора общей архитектуры .NET в главе 1, чтобы дать вам представление о том, что понадобится для написания управляемого кода. Книга состоит из ряда частей, описывающих как язык C#, так и его применение в различных областях.

Часть I. Язык C#

В этой части предоставляются хорошие базовые сведения о самом языке C#. Наличие познаний в каком-то конкретном языке здесь не ожидается, но наличие опыта в программировании действительно предполагается. Сначала рассматривается базовый синтаксис и типы данных C#, а затем рассказывается об объектно-ориентированных возможностях C#, о которых необходимо знать, прежде чем переходить к изучению более сложных тем, связанных с программированием на C#.

Часть II. Visual Studio

В этой части рассматривается основная IDE-среда, которой пользуются разработчики приложений на C# во всем мире — Visual Studio 2010. В двух главах в этой части показано, как лучше всего применять эту среду для построения приложений на основе .NET Framework 4. Кроме того, здесь описаны способы развертывания проектов.

Часть III. Основы

В этой части рассказывается об основных принципах программирования в среде .NET. Рассматриваются вопросы обеспечения безопасности, организация потоков, локализация, транзакции, создание служб Windows, генерация собственных библиотек в виде сборок, а также многое другое.

Часть IV. Данные

В этой части показано, как получать доступ к базам данных с помощью ADO.NET и LINQ и взаимодействовать с каталогами и файлами. Здесь также подробно рассматривается предлагаемая в .NET и со стороны операционной системы Windows поддержка для XML, а также средства .NET, встроенные в SQL Server 2008.

Часть V. Презентация

В этой части сначала рассматривается создание классических приложений Windows, которые в .NET называются приложениями *Windows Forms*. Такие приложения являются версией "толстого" клиента и за счет применения .NET их можно создавать очень легко и быстро. Затем показано, как создавать приложения на основе Windows Presentation Foundation и Silverlight и писать компоненты, которые будут функционировать на веб-сайтах и обслуживать веб-страницы. И, наконец, здесь рассматривается огромное количество возможностей, которые предлагают технологии ASP.NET и ASP.NET MVC.

Часть VI. Коммуникации (на компакт-диске)

Эта часть целиком посвящена коммуникациям. Здесь описаны службы для независимых от платформы коммуникаций, реализуемые с помощью Windows Communication Foundation (WCF). Кроме того, рассматриваются способы обеспечения асинхронных коммуникаций в автономном режиме с использованием Message Queuing, а также применение Windows Workflow Foundation (WF), организация одноранговой взаимодействия и создание синдицируемых каналов.

Приложение A (на компакт-диске)

В приложении описана разработка приложений для Windows 7 и Windows Server 2008 R2.

Дополнительные главы (на компакт-диске)

Несмотря на большой объем, уместить в печатное издание все сведения о языке C# и его применении с другими технологиями .NET не удалось, поэтому десять дополнительных глав представлены в электронном виде на прилагаемом к книге компакт-диске. Эти главы посвящены различным темам, в том числе технологии GDI+, служащей для построения приложений с усовершенствованной графикой; технологии .NET Remoting, используемой для обеспечения связи между клиентами и серверами .NET; технологии Enterprise Services, которая применяется для создания служб, способных функционировать в фоновом режиме; и технологии Managed Add-In Framework (MAF). Также в этих дополнительных главах можно найти сведения о разработке VSTO и использованию LINQ to SQL.

Соглашения

Для извлечения читателем максимальной пользы из текста и получения им более четкого представления о происходящем, в настоящей книге используется ряд соглашений.

Во врезках с пиктограммой, содержащей знак предупреждения, дается важная информация, которую следует запомнить. Эта информация имеет непосредственное отношение к тексту, внутри которого находится.

Во врезах с пиктограммой, содержащей изображение карандаша, перечислены различные советы, подсказки, полезные приемы или сведения, предлагаемые в качестве дополнения к текущему обсуждению.

В тексте встречаются следующие стили.

- Новые термины и другие важные слова при первом упоминании выделяются *курсивом*.
- Сочетания клавиш представляются следующим образом: <Ctrl+A>.
- Имена файлов, URL-адреса и код внутри текста выделяется таким стилем: `persistence.properties`.
- Примеры кода представляются двумя разными способами:

```
В большинстве примеров кода используется моноширинный шрифт без выделения.
```

```
Полужирным начертанием выделяются фрагменты кода, которые являются особенно важными
в текущем контексте или которые изменились по сравнению с предыдущими примерами.
```

Исходный код

Работая с примерами, вы можете либо вводить их код вручную, либо использовать файлы исходного кода, находящиеся на прилагаемом к книге компакт-диске.

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@dialektika.com`
WWW: `http://www.dialektika.com`

Информация для писем из:

России: 127055, г. Москва, ул. Лесная, д. 43, стр. 1
Украины: 03150, Киев, а/я 152