Введение

Если мы опишем язык C# и связанную с ним среду .NET Framework как наиболее важную технологию для разработчиков за многие годы, это не будет преувеличением. .NET предназначена стать новой средой, внутри которой вы можете разработать почти любое приложение, работающее под Windows, в то время как C# — новый язык программирования, который был спроектирован специально для работы с .NET. Используя C#, вы можете, например, написать динамическую Web-страницу, Web-службу XML, компонент распределенного приложения, компонент доступа к базе данных, классическое настольное приложение Windows или даже приложение интеллектуального клиента, оснащенного онлайновыми и автономными возможностями. Эта книга посвящена версии .NET Framework 3.5. Если вы продолжаете кодировать, используя версии 1.0, 1.1, 2.0 или даже 3.0, то некоторые разделы этой книги вам не помогут. Мы специально старались отмечать элементы, которые появились только в .NET Framework 3.5.

Пусть вас не вводит в заблуждение марка ".NET". Часть "NET" в названии должна подчеркнуть ориентацию Microsoft на распределенные приложения, в которых обработка распределена между клиентом и сервером, но C# — это язык не только для написания приложений Internet или просто сетевых приложений. Он предлагает средства кодирования почти любого типа программного обеспечения или компонентов, которые вам придется писать для платформы Windows. C# и .NET призваны революционно изменить способ написания программ, намного облегчая задачу программирования для Windows, по сравнению с тем, что было когда-либо ранее.

Это довольно амбициозное утверждение требует обоснования. В конце концов, все мы знаем, насколько быстро меняются компьютерные технологии. Каждый год Microsoft выпускает новое программное обеспечение, инструменты программирования или версии Windows, сопровождаемые громогласными утверждениями об их невероятной ценности для разработчиков. Так в чем же отличие .NET и C#?

Значение .NET и C#

Чтобы понять значение .NET, полезно вспомнить природу многих технологий Windows, которые появились за последние 10 лет или около того. Хотя все они на первый взгляд могут показаться довольно разными, все операционные системы Windows, начиная с Windows 3.1 (выпущенной в 1992 г.) и вплоть до Windows Server 2008, имели в основе своей один и тот же программный интерфейс Windows API. По мере появления новых версий Windows к этому API было добавлено огромное число новых функций, но все это в рамках процесса эволюции и расширения API вместо его замены.

То же можно сказать о многих технологиях и каркасах, которые мы использовали для разработки программного обеспечения под Windows. Например, *COM* (Component Object Model — объектная модель компонентов) изначально появилась под именем *OLE* (Object Linking and Embedding — связывание и внедрение объектов). На то время она была, главным образом, просто средством связывания разнотипных документов Office, так что вы, например, могли поместить маленькую электронную таблицу

Excel в документ Word. Отсюда эта технология эволюционировала до COM, *DCOM* (Distributed COM — распределенный COM), и, наконец, *COM+* — изощренной технологии, формирующей основы взаимодействия почти всех компонентов, вместе с реализацией транзакций, служб обмена сообщениями и организации спула объектов.

Microsoft выбрала этот эволюционный подход к программному обеспечению по очевидной причине — заботясь об обратной совместимости. За многие годы независимыми разработчиками для Windows был разработан гигантский объем программного обеспечения, и Windows не достигла бы такого успеха, если бы всякий раз, когда Microsoft представляла новую технологию, существующая кодовая база переставала бы работать!

В то время как обратная совместимость всегда была важнейшим свойством технологий Windows и одной из сильных сторон платформы Windows, она также имела большой недостаток. Всякий раз, когда технология развивалась и добавлялись новые средства, это приводило к усложнению по сравнению с тем, что было раньше.

Стало ясно, что нужно что-то менять. Microsoft не могла до бесконечности расширять существующие средства разработки и языки, постоянно все более усложняя их, чтобы удовлетворить противоречивые требования поддержки нового оборудования и обратной совместимости с тем, что было во времена, когда Windows впервые стала популярной — в начале 90-х прошлого века. Наступил момент, когда нужно было начать с чистого листа, имея простой, но достаточный набор языков, сред и средств разработки, которые позволили бы разработчиками легко писать полезное и работоспособное программное обеспечение.

Таким "началом с нуля" стали C# и .NET. Грубо говоря, .NET — это каркас, API-интерфейс для программирования на платформе Windows. Вместе с .NET Framework, C# представляет собой язык, спроектированный "с нуля", предназначенный для работы с .NET и впитавший в себя все достижения прогресса сред разработки и понимание принципов объектно-ориентированного программирования, которое выкристаллизовалось за последние 20 лет.

Прежде чем продолжить, следует пояснить, что обратная совместимость вовсе не была утеряна. Существующие программы продолжают работать, и каркас .NET спроектирован так, что способен работать с этим существующим программным обеспечением. В настоящее время коммуникации между программными компонентами Windows почти целиком строятся с использованием COM. Принимая это во внимание, .NET предоставляет оболочки вокруг существующих компонентов COM, так что компоненты .NET могут свободно общаться с ними.

Верно и то, что вы не обязаны изучать C#, чтобы писать код для .NET. Microsoft предлагает расширенный C++, еще один новый язык, называемый J#, а также вносит существенные изменения в Visual Basic, превратив его в более мощный язык — Visual Basic .NET, чтобы позволить на любом из этих языков разрабатывать приложения, ориентированные на среду .NET. Однако все эти языки скорее представляют собой результат эволюции в течение ряда лет, и не разработаны изначально с учетом требований современных технологий.

Эта книга научит вас программировать на C#, одновременно закладывая необходимый фундамент знаний о работе архитектуры .NET. Мы не только раскроем основы языка C#, но также представим примеры приложений, использующих широкий диапазон взаимосвязанных технологий, включая доступ к базам данных, динамические Web-страницы, расширенную графику и доступ к каталогам. Единственное требование — чтобы вы были знакомы хотя бы с одним из других высокоуровневых языков программирования, используемых в Windows — C++, Visual Basic или J++.

Преимущества .NET

Мы уже говорили в общих чертах о том, насколько замечателен .NET, но пока не сказали ничего о том, как он облегчает жизнь вам как разработчику. В этом разделе мы кратко опишем некоторые усовершенствованные средства .NET.

- **Объектно-ориентированное программирование**. И среда .NET Framework, и C# изначально полностью базировались на объектно-ориентированных принципах.
- **Хороший дизайн**. Библиотека базовых классов, которая спроектирована "с нуля", исключительно интуитивно понятным образом.
- **Независимость от языка**. Благодаря .NET, код всех языков, то есть Visual Basic .NET, C#, J# и управляемого C++, компилируется в общий язык промежуточного уровня — *Intermediate Language*. Это значит, что все эти языки обладают возможностями взаимодействия, как никогда ранее.
- **Лучшая поддержка динамических Web-страниц**. Хотя ASP предлагал высокую степень гибкости, он также был и неэффективен из-за своих интерпретируемых сценарных языков, а недостаток объектно-ориентированного дизайна часто приводил к запутанному коду ASP. .NET предлагает интегрированную поддержку Web-страниц с применением новой технологии — ASP.NET. В ASP.NET код ваших страниц компилируется и может быть написан на языке высокого уровня, поддерживающего .NET — таком как C#, J# или Visual Basic 2008.
- **Эффективный доступ к данным**. Набор компонентов .NET, известный под общим названием ADO.NET, предоставляет эффективный доступ к реляционным базам данных и широкому разнообразию других источников данных. Также имеются компоненты, предоставляющие доступ к файловой системе и каталогам. В частности, в .NET встроена поддержка XML, что позволяет манипулировать данными, которые могут быть импортированы их других, не-Windows платформ, и экспортированы на них.
- **Разделение кода**. Среда .NET полностью изменила способ разделения кода между приложениями, введя концепцию *сборки* (assembly), которая заменила традиционные библиотеки DLL. Сборки имеют форматные средства для указания версий, и одновременно в системе могут существовать разные версии одних и тех же сборок.
- **Повышенная безопасность**. Каждая сборка также может содержать встроенную информацию безопасности, которая в точности описывает, кому и каким категориям пользователей или процессов какие методы каких классов разрешено вызывать. Это обеспечивает очень высокую степень контроля за тем, как могут использоваться сборки, которые вы поставляете.
- **Инсталляция с нулевым воздействием**. Существует два типа сборок: разделяемые и приватные. Разделяемые сборки — это обычные библиотеки, доступные всему программному обеспечению, в то время как приватные сборки предназначены для использования совершенно определенными программами. Приватные сборки полностью самодостаточны, поэтому процесс инсталляции прост. Нет никаких элементов реестра, нужные файлы просто помещаются в соответствующую папку файловой системы.
- **Поддержка Web-служб**. .NET предлагает полностью интегрированную поддержку разработки Web-служб — все так же просто, как и создание приложений любого другого типа.

- **Visual Studio 2008**. .NET поставляется со средой разработки Visual Studio 2008, которая одинаково хорошо справляется с языками C++, C#, J# и Visual Basic 2008, а также с ASP.NET. Visual Studio 2008 интегрирует в себе все лучшие средства соответствующих специфичных языковых сред Visual Studio .NET 2002/2003/2005 и Visual Studio 6.
- **C#**. C# представляет собой новый объектно-ориентированный язык, предназначенный для применения с .NET.

Более подробно преимущества .NET рассматриваются в главе 1.

Что нового в .NET Framework 3.5

Первая версия .NET Framework (1.0) была выпущена в 2002 г. и встречена с большим энтузиазмом. Версия .NET Framework 2.0 появилась в 2005 г. и рассматривалась как главная версия Framework. Версия .NET Framework 3.5, хотя и не столь объемная, как 2.0, рассматривается как еще более главная версия продукта, включающая множество выдающихся новых средств.

С каждым новым выпуском .NET Framework в Microsoft всегда старались обеспечить минимум изменений, которые могли бы нарушить целостность ранее разработанного кода. Пока с этой задачей справляться удавалось.

> Удостоверьтесь, что у вас есть установочный сервер для полноценного тестирования перехода ваших приложений на .NET Framework 3.5, а не пытайтесь просто обновить "живое" приложение.

В следующем разделе детализируются некоторые изменения, которые были внесены в C# 2008, .NET Framework 3.5, а также новые дополнения к Visual Studio 2008 — среде разработки для .NET Framework 3.5.

Неявно типизированные переменные

Используя C# 2008, вы можете объявлять переменные и позволять компилятору определять их тип неявно. Вы обнаружите, что LINQ использует эту способность для работы с создаваемыми запросами. Чтобы воспользоваться этой новой возможностью, применяйте ключевое слово `var`:

```
var x = 5;
```

В результате задания этого оператора компилятор в действительности использует значение 5, чтобы определить тип, который должна иметь переменная `x`. В данном случае это означает, что оператор станет таким, как следовало ожидать:

```
int x = 5;
```

Автоматически реализуемые свойства

Обычная задача объявления свойств в C# 2008 облегчена. До появления этой версии вы должны были объявлять свойства следующим образом:

```
private int _myItem;
public int MyItem
{
  get {
    return myItem
  }
```

```
  set {
    myItem = value;
  }
}
```

Теперь вы можете поручить компилятору выполнение этой работы за вас. Вместо постоянного повторения в коде приведенной выше структуры вы можете воспользоваться сокращением автоматически реализуемых свойств:

```
public int MyProperty { get; set; }
```

Использование этого синтаксиса даст тот же результат, что и предыдущий более длинный пример. Компилятор выполнит операцию преобразования этой краткой формы в правильный формат. Это повышает читабельность вашего кода и ускоряет его разработку.

Инициализаторы объектов и коллекций

C# 2008 теперь позволяет присваивать значения свойствам объектов в момент их инициализации. Например, предположим, что в коде есть следующий объект:

```
public class MyStructure
{
  public int MyProperty1 { get; set; }
  public int MyProperty2 { get; set; }
}
```

Применяя C# 2008, вы можете создать экземпляр объекта `MyStructure` следующим образом:

```
MyStructure myStructure = new MyStructure() { MyProperty1 = 5,
  MyProperty2 = 10 };
```

То же средство позволяет объявлять множество элементов коллекции в один прием:

```
List<int> myInts = new List<int>() { 5, 10, 15, 20, 25 };
```

В этом случае все числа добавляются в коллекцию `myInts`, как если бы для каждого из них был вызван метод `Add()`.

Встроенная поддержка ASP.NET AJAX

Хотя вы могли строить страницы ASP.NET AJAX и в .NET Framework 2.0, это требовало дополнительной инсталляции. Теперь поддержка ASP.NET AJAX встроена в ASP.NET 3.5 и Visual Studio 2008.

В настоящее время каждая страница, которую вы строите с применением ASP.NET и .NET Framework 3.5, является Ajax-ориентированной (всю конфигурацию Ajax вы можете видеть в файле `Web.config`). Также в наборе инструментов ASP.NET вы найдете некоторые новые элементы управления, которые позволяют добавлять возможности Ajax к существующим Web-сайтам. Подробнее об ASP.NET AJAX читайте в главе 39.

Каркас языка интегрированных запросов .NET (LINQ)

Одним из самый совершенных и наиболее ожидаемых средств новой версии стал язык LINQ, предоставляющий возможность легкого доступа к лежащим в основе данным. Microsoft представила LINQ как легковесный фасад, обеспечивающий строго типизированный интерфейс к лежащим в основе хранилищам данных. LINQ предлагает средства для разработчиков оставаться в привычной среде кодирования, которую они постоянно используют, при этом получая доступ к данным как к объектам в IDE, Intellisense и даже отладчике.

Используя LINQ, вы можете выполнять запросы к объектам, наборам данных, базе данных SQL Server, XML и многим другим источникам. Замечательно то, что независимо от лежащего в основе источника данных, получение информации осуществляется в одной и той же манере, потому что LINQ обеспечивает структурированный способ опроса данных.

Пример получения псевдо-XML-документа и извлечения всех имен заказчиков из файла XML представлен ниже.

```
XDocument xdoc = XDocument.Load(@"C:\Customers.xml");
var query = from people in xdoc.Descendants("CustomerName")
            select people.Value;
Console.WriteLine("{0} клиентов найдено", query.Count());
Console.WriteLine();
foreach (var item in query)
{
   Console.WriteLine(item);
}
```

В главах 11, 27 и 29 освещены различные аспекты LINQ.

Многоцелевые возможности Visual Studio

Во многих случаях разработчикам .NET теперь приходится работать с множеством приложений .NET, ориентированных на разные версии .NET — 2.0, 3.0, а теперь и 3.5. Было бы неэффективно держать несколько версий Visual Studio на компьютере разработчика, чтобы работать с разными версиями .NET Framework.

По этой причине в последней версии Visual Studio 2008 поддерживается возможность указания целевой версии каркаса, на которую вы ориентируетесь. Теперь при создании новых приложений вы можете указывать версию .NET Framework, под управлением которой они должны работать: 2.0, 3.0 или 3.5.

Поддержка новейших типов приложений

Не так много времени прошло с момента выхода версии .NET Framework 3.0, принесшей с собой некоторые замечательные новые возможности. Так, например, в эту версию была включена возможность построения нового типа приложений с применением Windows Presentation Foundation (WPF), а также приложений и библиотек на основе Windows Communication Foundation (WCF) и Windows Workflow Foundation (WF).

В выпуске Visual Studio 2008 вы обнаружите возможности для построения всех этих приложений — теперь они доступны в виде типов проектов с новыми элементами управления и мастерами и возможностями 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 последнего десятилетия, так и Visual Basic 2005 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 нико-

гда 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, создав Visual Basic .NET и Visual Basic 2005 — язык, 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 Microsoft решила предложить разработчикам альтернативу — язык, ориентированный специально на .NET, к тому же спроектированный с чистого листа. В результате появился Visual C# 2005. Официально Microsoft описывает C# как "простой, современный, объектно-ориентированный и безопасный к типам язык программирования, унаследованный от C и C++". Большинство независимых обозревателей, вероятно, изменили бы это на "унаследованный от 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 — те 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 у Visual Basic, вместе с высокой производительностью и низкоуровневым доступом к памяти, характерные 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 Web-страниц ASP.NET и Web-служб XML.

Бо́льшая часть перечисленного также касается Visual Basic 2005 и управляемого C++. Тот факт, что 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 по времени или исключительно высокопроизводительный код — когда приходится заботиться о том, чтобы цикл занимал 1000 или 1050 тактов процессора, когда необходимо очищать ресурсы в течение 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 специфицировать встроенные функции и деструкторы, 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 C#

.NET Framework 3.5 работает под управлением Windows XP, Windows Server 2003, Windows Vista и новейшей версии Windows Server 2008. Чтобы писать код с использованием .NET, необходимо инсталлировать .NET 3.5 SDK.

Кроме того, если только вы не намерены писать код C# в простом текстовом редакторе или некоторой среде разработки от независимых поставщиков, почти наверняка вы пожелаете использовать Visual Studio 2008. Полный комплект SDK не нужен для выполнения управляемого кода, а исполняющая система .NET — необходима. Вам может понадобиться распространять эту исполняющую систему вместе с вашим кодом для тех клиентов, у которых она еще не инсталлирована.

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 нескольких разделов, описывающих как язык C#, так и его применение в различных областях.

Часть I. Язык C#

Эта часть закладывает хороший фундамент знаний о самом языке C#. Она не предполагает знания какого-либо конкретного языка, хотя предполагается, что вы — опытный программист. Мы начнем с обзора базового синтаксиса C# и его типов данных, а затем обсудим объектно-ориентированные возможности C#, после чего можно будет перейти к более сложным темам программирования на C#.

Часть II. Visual Studio

Эта часть посвящена главной IDE-среде, применяемой разработчиками на C# во всем мире — Visual Studio 2008. В двух главах этой части будут показаны лучшие способы использования этого инструмента для построения приложений на базе .NET Framework 2.0/3.0/3.5, а также рассматриваются вопросы развертывания проектов.

Часть III. Библиотеки базовых классов

В этой части рассматриваются принципы программирования в среде .NET. В частности, описываются концепции безопасности, многопоточность, локализация, транзакции, построение Windows-служб и генерация собственных библиотек в виде сборок.

Часть IV. Данные

Здесь рассматривается доступ к базам данных средствами ADO.NET и LINQ, а также взаимодействие с каталогами и файлами. Кроме того, подробно раскрывается поддержка .NET технологии XML на стороне операционной системы Windows, а также средства .NET, встроенные в SQL Server 2008. В пределах обширной темы LINQ особое внимание уделено LINQ to SQL и LINQ to XML.

Часть V. Презентации

Эта часть сосредоточена на построении классических приложений Windows, которые в .NET называются Windows Forms. Windows Forms представляют собой версии приложений “толстого” клиента, а применение .NET для их построения — простой и быстрый способ решения этой задачи. В дополнение к рассмотрению Windows Forms рассматривается GDI+ — технология, используемая для построения приложений, работающих с расширенной графикой. В этой части также раскрываются компоненты, работающие на Web-сайтах, доставляющие Web-страницы. Сюда входит огромный объем новых средств, предлагаемых ASP.NET 3.5. Кроме того, в этой же части рассматриваются вопросы построения приложений на базе Windows Presentation Foundation и VSTO.

Часть VI. Коммуникации

Эта часть полностью посвящена коммуникациям. Здесь описываются Web-службы, применяемые для коммуникаций, независящих от платформы, .NET Remoting — для коммуникаций между клиентами и серверами .NET, Enterprise Services (службы уровня предприятия) — для служб, работающих в фоновом режиме, и коммуникации DCOM. На примере системы асинхронных сообщений показаны коммуникации в отключенном режиме. В этой части также рассматриваются вопросы использования Windows Communication Foundation и Windows Workflow Foundation.

Часть VII. Приложения

Эта часть включает три приложения, посвященные построению программ с учетом новых средств и новых препятствий, присутствующих в Windows Vista. Также здесь рассматривается технология ADO.NET Entities и ее применение в приложениях C#.

Соглашения

В этой книге используется множество стилей текста, которые призваны помочь отличать разнородную информацию. Ниже представлены примеры используемых стилей и объяснение их назначения.

- *Важные слова* выделены курсивом.
- Клавиши, которые вы нажимаете на клавиатуре, представлены как <Ctrl>, <Enter> и так далее.

Код появляется в книге в разных видах. Если слово, о котором говорится в тексте, является частью исходного кода, например, в обсуждении конструкции `if...else`, оно представлено моноширинным шрифтом. Если это блок кода, который вы можете ввести как программу и запустить, то он представлен следующим образом:

```
public static void Main()
{
  AFunc(1,2,"abc");
}
    // Если мы не достигли конца, вернуть true, в противном случае
    // установить позицию как недействительную и вернуть false.
    pos++;
    if (pos < 4)
      return true;
    else {
      pos = -1;
      return false;
    }
```

Советы, подсказки и сопровождающая информация выделены курсивом.

> Важные части информации взяты в рамку.

Синтаксис применения методов, свойств и тому подобного представлен в следующем формате:

```
Regsvcs BookDistributor.dll [COM+AppName] [TypeLibrary.tbl]
```

Здесь курсивом выделены объектные ссылки, переменные или значения параметров, которые должны быть заменены; квадратные скобки указывают на необязательные параметры.

Исходный код и приложения

Работая с примерами этой книги, вы можете либо вводить весь 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 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@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