Введение

Visual C# .NET (C#) изучить относительно легко любому, кто знаком с другим объектно-ориентированным языком. Даже человек, знакомый с Visual Basic 6.0, которому нужен объектно-ориентированный язык, обнаружит, что понять C# нетрудно. Однако, несмотря на то, что C# вместе с .NET Framework предоставляют легкий путь создания простых приложений, все же вам понадобится немало знаний и понимания того, как их правильно применять для создания сложных, надежных и устойчивых к сбоям приложений C#. В этой книге я научу вас всему, что необходимо знать, и расскажу, как наилучшим образом применять знания для того, чтобы быстро достичь уровня настоящего специалиста в C#.

Идиомы и шаблоны проектирования незаменимы для повышения уровня мастерства разработчика и применения его на практике, так что я покажу вам, как использовать многие из них для создания приложений — эффективных, надежных, устойчивых к сбоям и безопасных в отношении исключений. Хотя многие из этих шаблонов знакомы программистам на C++ и Java, все же некоторые уникальны для .NET и его общеязыковой исполняющей среды (Common Language Runtime — CLR). В последующих главах будет показано, как применять эти обязательные идиомы и приемы проектирования для гладкой интеграции ваших приложений C# с исполняющей системой .NET, причем основное внимание сосредоточивается на новых средствах C# 3.0.

Шаблоны проектирования документируют передовой опыт в проектировании приложений, который накоплен множеством программистов в течение длительного времени. Фактически .NET Framework сам по себе реализует многие хорошо известные шаблоны проектирования. К тому же последние три версии .NET Framework и две версии C# высветили многие новые идиомы и передовые приемы. Вы будете знакомиться с ними на протяжении всей книги. К тому же важно отметить, что этот бесценный набор технологий постоянно эволюционирует.

С появлением C# 3.0 вы получили возможность легко включать приемы функционального программирования, используя лямбда-выражения, расширяющие методы и язык интегрированных запросов (Language Integrated Query — LINQ). Лямбда-выражения облегчают объявление и создание экземпляров делегатов функций в одном месте. Вдобавок к лямбда-выражениям появилась возможность создавать функционалы — функции, принимающие в качестве аргументов функции и обычно возвращающие другие функции. Несмотря на то что вы могли реализовать технику программирования функционалов на C# (хотя и с некоторыми трудностями), новые средства языка C# 3.0 обеспечили среду, в которой программирование функционалов может органично переплетаться с обычным стилем императивного программирования C#. LINQ позволяет выразить операции запроса данных (которые обычно по природе своей являются функционалами), используя естественный для языка синтаксис. Однажды увидев, как работает LINQ, вы поймете, что можете

сделать много больше, чем простой запрос данных; вы можете использовать его для реализации сложных функциональным программ.

.NET и CLR предоставляют уникальную и стабильную межплатформенную исполняющую среду. C# — только один из языков, ориентированных на эту мощную исполняющую систему. Вы обнаружите, что многие из приемов, описанных в этой книге, также применимы к любому языку, ориентированному на исполняющую систему .NET. Тем из вас, у кого есть серьезный опыт работы на C++, и кто знаком с такими концепциями, как канонические формы C++, безопасность исключений, RAII (Resource Acquisition Is Initialization — захват ресурсов является инициализацией), а также корректность констант, книга объяснит, как применить все эти концепции в C#. Если вы — программист на Java или Visual Basic, который потратил годы на разработку собственного набора приемов, и хотите знать, как эффективно применить их в C#, вы найдете здесь ответ и на этот вопрос.

Как вы вскоре убедитесь, для того, чтобы стать экспертом в C#, не нужно тратить годы на приобретение опыта методом проб и ошибок. Вам просто нужно изучить правильные вещи и правильные способы того, как их следует делать. Именно потому я написал эту книгу для вас.

Как организована эта книга

Я предполагаю, что вы уже имеете практический опыт работы с некоторым объектно-ориентированным языком, таким как C++, Java или Visual Basic .NET. Поскольку C# унаследовал свой синтаксис от C++ и Java, я не стану тратить много времени на описание синтаксиса C#, за исключением тех моментов, в которых он отличается от C++ или Java. Если вы уже немного знаете C#, то можете лишь пролистать или вообще пропустить главы с 1 по 3.

Глава 1, “Обзор C#”, даст вам первоначальное представление о том, как выглядит простое приложение C#, и предложит описание некоторых базовых отличий среды программирования C# от среды “родного” C++.

Глава 2, “C# и CLR”, разовьет тему главы 1 и кратко ознакомит с управляемой средой, внутри которой выполняется приложение C#. Я расскажу о сборках — базовых строительных блоках приложений, в которые компилируются файлы кода C#. Вдобавок вы увидите, как метаданные делают сборки самодостаточными.

Глава 3, “Обзор синтаксиса C#”, предоставит описание синтаксиса языка C#. Я продемонстрирую две фундаментальных группы типов CLR: типы значений и ссылочные типы. Также я опишу пространства имен и то, как вы можете использовать их для логического разбиения типов и функциональности внутри ваших приложений.

Главы с 4 по 13 содержат углубленные описания применения полезных идиом, шаблонов проектирования и передовых приемов в ваших программах и проектах C#. Я попытался выстроить эти главы в логическом порядке, но неизбежно одни главы могут ссылаться на приемы или темы, описанные в других (последующих) главах.

Глава 4, “Классы, структуры и объекты”, содержит подробности определения типов в C#. Вы узнаете больше о типах значений и ссылочных типах в CLR. Я также коснусь “родной” поддержки интерфейсов внутри CLR и C#. Вы увидите, как работает наследование в C#, а также каким образом каждый объект наследуется

от типа `System.Object`. Эта глава также содержит богатую информацию об управляемой среде и о том, что вам нужно знать, чтобы определять типы, удобные для нее. Я представлю многие из таких тем в этой главе и продолжу дискуссию более подробно в последующих главах.

Глава 5, "Интерфейсы и контракты", посвящена интерфейсам и роли, которую они играют в языке C#. Интерфейсы представляют собой контракт функциональности, которую могут реализовывать типы. Вы узнаете о многих способах реализации интерфейсов типами, а также о том, как исполняющая система выбирает, какие методы нужно вызывать при вызове методов интерфейса.

Глава 6, "Перегрузка операций", детализирует способы создания специальной функциональности встроенных операций языка C#, когда они применяются к вашим собственным типам. Вы увидите, как перегрузить действие операций, хотя не все управляемые языки, которые компилируются в код для CLR, способны использовать перегруженные операции.

Глава 7, "Исключения: безопасность и обработка", посвящена средствам обработки исключений языка C# и CLR. Хотя синтаксис подобен C++, создание безопасного и нейтрального в отношении исключений кода не так просто — даже сложнее создания безопасного к исключениям кода на "родном" C++. Вы увидите, что написание устойчивого к сбоям, безопасного к исключениям кода вообще не требует применения конструкций `try`, `catch` или `finally`. Я также опишу некоторые новые возможности, добавленные в исполняющую систему .NET 2.0, которые позволят вам создавать более устойчивый к сбоям код, чем это было возможно в .NET 1.1.

Глава 8, "Работа со строками", описывает строки — первоклассный тип CLR — и методы их эффективного применения в C#. Значительная часть этой главы посвящена средствам строкового форматирования различных типов в .NET Framework и тому, как заставить определенные вами типы вести себя подобным образом — посредством реализации `Iformattable`. Дополнительно я представлю средства глобализации каркаса и расскажу, как создавать собственные `CultureInfo` для культур и регионов, о которых .NET Framework не имеет понятия.

Глава 9, "Массивы, типы коллекций и итераторы", расскажет о разнообразных массивах и типах коллекций, доступных в C#. Вы можете создавать два типа многомерных массивов наряду с собственными типами коллекций, используя служебные классы коллекций. Вы увидите, как определять прямые, обратные и двунаправленные итераторы, применяя новый синтаксис итераторов, представленный в C# 2.0, так что ваши типы коллекций смогут хорошо работать с операторами `foreach`.

Глава 10, "Делегаты, анонимные функции и события", продемонстрирует механизмы, используемые внутри C# для обеспечения обратных вызовов. C# пошел на один шаг дальше, и инкапсулирует обратные вызовы в вызываемые объекты, называемые *делегатами*. Вдобавок C# 2.0 позволяет создавать делегаты с сокращенным синтаксисом, называемые *анонимными функциями*. Анонимные функции подобны лямбда-функциям в функциональном программировании. К тому же вы увидите, как на основе делегатов каркас реализует механизм уведомления публикации/подписки на события, позволяя отделять источник события от его потребителя.

Глава 11, "Обобщения", представит, пожалуй, наиболее впечатляющее средство, появившееся в C# 2.0 и CLR. Тем, кто знаком с шаблонами C++, обобщения

покажутся знакомыми, хотя между ними есть многие фундаментальные отличия. Используя обобщения, вы можете создавать оболочку функциональности, внутри которой во время выполнения определяются более специфичные типы. Обобщения наиболее полезны для типов коллекций и обеспечивают значительную эффективность по сравнению с коллекциями из предыдущих версий .NET.

Глава 12, "Многопоточность в C#", описывает то, что необходимо сделать при создании многопоточных приложений в управляемой виртуальной исполняющей системе C#. Если вы знакомы с потоками в родной среде Win32, вы отметите существенные отличия. Более того, управляемая среда предоставляет более развитую инфраструктуру для облегчения работы. Вы увидите, как делегаты с использованием шаблона IOU ("I Owe You") предоставляют отличные ворота в пул потоков обработки. Вероятно, синхронизация — наиболее важная концепция, когда нужно заставить несколько потоков работать параллельно. В этой главе описаны различные средства синхронизации, доступные вашим приложениям.

Глава 13, "В поисках канонических форм C#", — это своего рода диссертация о передовых приемах проектирования при определении новых типов, и о том, как сделать их такими, чтобы их применение было естественным, и чтобы их потребители не могли использовать их неправильно. Я буду касаться этой темы и в других главах, но здесь она обсуждается в деталях. Эта глава снабжена подробным перечнем того, что следует принимать во внимание при определении новых типов на C#.

Глава 14, "Расширяющие методы", раскрывает средство, новое для C# 3.0. Поскольку вы можете вызывать расширяющие методы как обычные методы экземпляра с типом, который они расширяют, их можно воспринимать как развитие контракта типов. Но на самом деле они представляют собой нечто большее. В этой главе я покажу, как расширяющие методы могут открыть мир функционального программирования на C#.

Глава 15, "Лямбда-выражения", посвящена еще одному новому средству C# 3.0. Вы можете объявлять и создавать экземпляры делегатов, используя лямбда-выражения с применением краткого и визуально выразительного синтаксиса. Хотя той же цели могут служить и анонимные функции, они намного более многословны и менее синтаксически элегантны. Однако в C# 3.0 вы можете конвертировать лямбда-выражения в деревья выражений. То есть язык имеет встроенную способность преобразовывать код в структуры данных. Само по себе это средство полезно, но не настолько, как в сочетании в языком интегрированных запросов (Language Integrated Query — LINQ). В сочетании с расширяющими методами лямбда-выражения действительно завершают полный цикл функционального программирования на C#.

Глава 16, "LINQ: язык интегрированных запросов", — это кульминация всех новых средств C# 3.0. Используя выражения LINQ через новые LINQ-ориентированные ключевые слова C# 3.0, вы можете плавно интегрировать запросы к данным в код. LINQ формирует мост между обычным миром императивного программирования C# и миром функционального программирования запросов к данным. Выражения LINQ могут быть использованы для манипуляций нормальными объектами, равно как и данными, полученными из баз данных SQL, наборов данных и XML — и это далеко не полный перечень.

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: 115419, Москва, а/я 783

Украины: 03150, Киев, а/я 152