Введение

Язык Visual C# .NET (C#) изучить относительно легко любому, кто знает другие объектно-ориентированные языки. Даже человек, знакомый с Visual Basic 6.0, которому нужен объектно-ориентированный язык, обнаружит, что понять C# несложно. Однако, несмотря на то, что C# вместе с .NET Framework предоставляют легкий путь создания простых приложений, все же 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, 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 приложений C# с исполняющей системой .NET, причем основное внимание сосредоточивается на новых средствах C# 4.0.

Шаблоны проектирования документируют передовой опыт в разработке приложений, который накоплен множеством программистов в течение длительного времени. Фактически платформа .NET Framework сама 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 появилась 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 технику функционального программирования можно было реализовать на C# (хотя и с некоторыми трудностями), новые средства языка C# 3.0 обеспечили среду, в которой функциональное программирование может органично переплетаться с обычным стилем императивного программирования C#. LINQ позволяет выразить операции запросов данных (которые обычно по природе своей являются функциональными), используя естественный для языка синтаксис. Однажды увидев, как работает LINQ, вы поймете, что можете сделать намного больше, чем простые запросы данных — его можно использовать для реализации сложных функциональных 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, 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, в книге будет показано, как применить все эти концепции в C#. Если вы — программист на Java или Visual Basic, потративший 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 была написана эта книга.

Как организована эта книга

Предполагается, что вы уже имеете практический опыт работы с некоторым объектно-ориентированным языком, таким как C++, Java или Visual Basic .NET. Поскольку язык C# унаследовал свой синтаксис от C++ и Java, в книге не дается полное 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 1–3.

- В главе 1, “Обзор C#”, дается первоначальное представление о том, как выглядит простое приложение C#, и предлагается описание некоторых базовых отличий среды программирования C# от среды “родного” C++.
- В главе 2, “C# и CLR”, развивается тема, поднятая в главе 1. Благодаря ней, вы кратко ознакомитесь с управляемой средой, внутри которой выполняется приложение C#. В главе рассказывается о сборках — базовых строительных блоках приложений, в которые компилируются файлы кода C#. Вдобавок будет показано, как метаданные делают сборки самодостаточными.
- В главе 3, “Обзор синтаксиса C#”, предлагается описание синтаксиса языка C#. Будут продемонстрированы две фундаментальных группы типов CLR: типы значений и ссылочные типы. Кроме того, будут описаны пространства имен и то, как их можно использовать для логического разделения типов и функциональности внутри приложений.
- Главы с 4 по 13 содержат углубленное описание применения полезных идиом, шаблонов проектирования и передовых приемов в программах и проектах C#. Хотя материал этих глав и выстроен в логическом порядке, но неизбежно в одних главах будут присутствовать ссылки на приемы или темы, описанные в других (последующих) главах. Избежать таких ситуаций почти невозможно, однако они сведены к минимуму.
- Глава 4, “Классы, структуры и объекты”, содержит подробности определения типов в C#. Вы узнаете больше о типах значений и ссылочных типах в CLR. Также кратко рассматривается встроенная поддержка интерфейсов внутри CLR и C#. Вы увидите, как работает наследование в C#, и узнаете, что каждый объект наследуется от типа `System.Object`. Эта глава также содержит исчерпывающую информацию об управляемой среде и о том, что нужно знать для определения удобных для нее типов. Начатое в этой главе обсуждение продолжается в последующих главах.
- Глава 5, “Интерфейсы и контракты”, посвящена интерфейсам и роли, которую они играют в языке C#. Интерфейсы представляют собой контракт функционально-

сти, которую могут реализовывать типы. Вы узнаете о многих способах реализации интерфейсов типами, а также о том, как исполняющая система выбирает, какие методы должны вызываться при обращении к методам интерфейса.

- В главе 6, "Перегрузка операций", детализируются способы создания специальной функциональности встроенных операций языка C#, когда они применяются к собственным типам. Вы увидите, как перегружать действие операций, хотя не все управляемые языки, которые компилируются в код для CLR, способны пользоваться перегруженными операциями.
- Глава 7, "Безопасность и обработка исключений", посвящена средствам обработки исключений языка C# и CLR. Хотя синтаксис подобен C++, создание безопасного и нейтрального в отношении исключений кода не так-то 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`. Будут также описаны некоторые новые возможности, добавленные в исполняющую систему .NET 2.0, которые позволят создавать более устойчивый к сбоям код.
- В главе 8, "Работа со строками", описаны 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 определяемые вами типы вести себя подобным образом за счет реализации интерфейса `IFormattable`. Дополнительно будут представлены средства глобализации каркаса и показано, как создавать собственные `CultureInfo` для культур и регионов, о которых .NET Framework не имеет понятия.
- В главе 9, "Массивы, типы коллекций и итераторы", рассматриваются разнообразные массивы и типы коллекций, доступные в C#. Можно создавать два типа многомерных массивов наряду с собственными типами коллекций, используя служебные классы коллекций. Будет показано, как определять прямые, обратные и двунаправленные итераторы, применяя новый синтаксис итераторов, введенный в C# 2.0, так что типы коллекций смогут успешно работать с операторами `foreach`.
- В главе 10, "Делегаты, анонимные функции и события", демонстрируются механизмы, используемые внутри C# для обеспечения обратных вызовов. В C# обратные вызовы инкапсулируются в вызываемые объекты, которые называются *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 будет показано, как на основе делегатов реализуется механизм уведомления публикации/подписки на события, позволяя отделять источник события от его потребителя.
- В главе 11, "Обобщения", рассматривается, пожалуй, наиболее впечатляющее средство, появившееся в C# 2.0 и CLR. Тем, кто имел дело с шаблонами C++, обобщения покажутся знакомыми, хотя между ними есть немало фундаментальных отличий. С помощью обобщений можно 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. Начиная с C# 4.0, применение обобщенных типов стало еще более интуитивно понятным за счет поддержки ковариантности и контравариантности. Теперь допускается присваивание одного обобщенного типа другому, что сокращает потребность в громоздких методах преобразований, которые требовались ранее.

- В главе 12, "Многопоточность в C#", показано то, что необходимо делать при создании многопоточных приложений в управляемой виртуальной исполняющей системе C#. Если вы знакомы с потоками в среде Win32, то заметите существенные отличия. Более того, управляемая среда предлагает более развитую инфраструктуру, облегчающую работу. Вы увидите, как делегаты с использованием шаблона IOU ("I Owe You") предоставляют удобный доступ к пулу потоков обработки. Вероятно, синхронизация — наиболее важная концепция, когда нужно заставить несколько потоков работать параллельно. В этой главе описаны различные средства синхронизации, доступные для приложений. В современном мире параллелизм выходит на передний план, поскольку вместо того, чтобы тратить астрономические суммы на создание все более быстрых процессоров, теперь больше внимания уделяется созданию процессоров многоядерных. В главе рассматриваются новые средства Parallel Extensions и Task Parallel Library (TPL), появившиеся в .NET 4.0.
- В главе 13, "В поисках канонических форм C#", дается обзор передовых приемов проектирования при определении новых типов, и показано, как сделать их такими, чтобы их применение было естественным, и чтобы их потребители не могли использовать их неправильно. Эта тема затрагивается и в других главах, но здесь она обсуждается во всех деталях. В главе дается подробный перечень всех моментов, которые следует принимать во внимание при определении новых типов на C#.
- В главе 14, "Расширяющие методы", рассматривается средство, новое для C# 3.0. Поскольку расширяющие методы можно вызывать 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 будет показано, как расширяющие методы могут открыть мир функционального программирования на C#.
- Глава 15, "Лямбда-выражения", посвящена еще одному новому средству C# 3.0. Лямбда-выражения дают возможность объявлять и создавать экземпляры делегатов, используя краткий и визуально выразительный синтаксис. Хотя той же цели могут служить и анонимные функции, они намного более многословны и синтаксически менее элегантны. В C# 3.0 лямбда-выражения могут быть преобразованы в деревья выражений. Это значит, что язык обладает встроенной способностью преобразовывать код в структуры данных. Само по себе это средство полезно, но не настолько, как в сочетании с языком интегрированных запросов (Language Integrated Query — LINQ). Вместе с расширяющими методами лямбда-выражения действительно формируют основу функционального программирования на C#.
- В главе 16, "LINQ: язык интегрированных запросов", описана кульминация всех новых средств C# 3.0. Используя выражения LINQ через новые LINQ-ориентированные ключевые слова C# 3.0, можно 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 применяться для манипуляций обычными объектами, а также информацией, полученной из баз данных SQL, наборов данных и XML — и это далеко не полный перечень.
- В главе 17, "Динамические типы", рассматривается новый тип `dynamic`, появившийся в C# 4.0. Этот тип привнес с собой простоту интеграции с динамическими языками .NET, включая объекты COM Automation. Ушли в прошлое времена, когда для интеграции с этими компонентами приходилось прибегать к неестественно выглядящему и трудно читаемому коду, поскольку реализация типа `dynamic` взяла на себя всю черновую работу. Реализация динамического типа опирается на

среду DLR (Dynamic Language Runtime — исполняющая система динамического языка) — тот же самый фундамент, на котором базируются, помимо прочих, такие языки, как IronRuby и IronPython. Используя тип `dynamic` в сочетании с такими типами DLR, как `ExpandoObject`, можно создавать и реализовывать действительно динамические типы на C#.

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