Об этой книге

Эта книга в основном и в первую очередь посвящена внедрению (или инъекциям) зависимостей (ВЗ, Dependency Injection). Кроме того, это издание о технологии .NET, но это не так существенно. Коды примеров в нем написаны на языке C#, но большинство обсуждаемого в книге материала может быть легко применено к другим языкам и платформам. Я, например, изучал большинство рассматриваемых принципов и паттернов, читая книги, в которых коды примеров были написаны на языках Java или C++.

Внедрение зависимостей — это набор связанных принципов и паттернов. Оно является в большей степени процессом обдумывания и проектирования кода, а не конкретной технологией. Основная цель использования ВЗ — создание удобного в сопровождении программного обеспечения, построенного на объектно-ориентированной парадигме.

Все применяемые в данном издании концепции относятся к объектно-ориентированному программированию (ООП). Проблема, с которой справляется ВЗ (создание удобного в сопровождении программного кода), универсальна, но решение, предлагаемое ВЗ, находится в области объектно-ориентированного программирования на языках со статической типизацией: C#, Java, Visual Basic .NET, C++ и им подобных. Вы не сможете применить ВЗ в процедурном программировании. Этот подход — внедрение зависимостей — также, по-видимому, плохо применим в функциональных или динамических языках.

Само по себе ВЗ невелико, но оно тесно связано с большим комплексом принципов и паттернов, используемых в проектировании и разработке объектно-ориентированного программного обеспечения. Несмотря на то что основным предметом этой книги является последовательное и всестороннее рассмотрение ВЗ, в ней также обсуждается множество других тем с точки зрения той специфической перспективы, которую дает ВЗ. Целью этого издания является не просто дать вам представление о тонкостях ВЗ, но и повысить вашу квалификацию в области разработки объектно-ориентированного программного обеспечения.

Для кого предназначена эта книга

Мне бы хотелось сказать, что эта книга предназначена для всех разработчиков платформы .NET. Но сообщество .NET-программистов сегодня очень велико и объединяет разработчиков веб-приложений, настольных приложений, приложений для смартфонов, RIA (Rich Internet Application — насыщенные интернет-приложения), интеграции, офисной автоматизации, систем управления контентом (CMS) и даже игр. И хотя .NET является объектно-ориентированным инструментом, не все из этих разработчиков создают объектно-ориентированный код.

Это издание об объектно-ориентированном программировании, поэтому читатели, как минимум, должны быть заинтересованы в применении ООП и понимать суть интерфейсов[1]. Наличие нескольких лет опыта в программировании, а также знание паттернов или SOLID (http://ru.wikipedia.org/wiki/SOLID_(объектно-ориентированное_программирование — *Примеч. ред.*), несомненно будет дополнительным преимуществом. Не уверен, что начинающие программисты получат много пользы от чтения этой книги, так как она в большей степени адресована опытным разработчикам и архитекторам программного обеспечения.

Все примеры в издании написаны на языке C#, поэтому читатели, программирующие на других языках платформы .NET, должны понимать C#. Читатели, работающие с объектно-ориентированными языками типа Java или C++, не поддерживаемыми платформой .NET, также могут счесть эту книгу полезной, поскольку информация, специфичная для платформы .NET, излагается здесь относительно поверхностно. Мне довелось прочесть множество книг по паттернам проектирования, примеры в которых были даны на языке Java, и я легко разобрался в большинстве из них, поэтому надеюсь, что и читатель не растеряется в подобной ситуации.

Наш маршрут

Содержимое книги разбито на четыре части. В идеале мне бы хотелось, чтобы вы прочли ее полностью, а затем использовали нужные части в качестве справочника, но я понимаю, что у вас могут быть и другие приоритеты. Из этих соображений большая часть глав написана таким образом, что вы можете выбрать нужную главу и начать чтение прямо с нее.

Однако первая часть книги все же обязательна для прочтения. Она содержит общую вводную информацию о внедрении зависимостей, и ее лучше читать по порядку. Вторая часть представляет собой каталог паттернов и других технологических компонентов. Третья часть содержит анализ ВЗ, выполненный с трех различных точек зрения. Четвертая часть книги содержит описание двух библиотек *контейнеров внедрения зависимостей*.

- Часть 1 — это общее введение в ВЗ. Если вы не в курсе, что именно представляет собой ВЗ, то эта глава является хорошей отправной точкой. Но даже если вы знаете что-либо об ВЗ, вы, возможно, захотите ознакомиться с содержимым части 1, поскольку она определяет терминологию, используемую в остальной части книги. В главе 1 дается общее представление о ВЗ, а также обсуждаются назначение и достоинства этой технологии. Глава 2 описывает большой и всесторонний практический пример. В главе 3 объясняется, как библиотеки контейнеров внедрения зависимостей используются вместе с другими компонентами. По сравнению с другими частями книги, часть 1 имеет гораздо более линейную структуру. Поэтому, чтобы максимально полно усвоить материал, каждую из глав этой части нужно прочесть от начала и до конца.
- Часть 2 является каталогом паттернов, антипаттернов и рефакторинга. В ней вы найдете инструкции по применению ВЗ, а также описание сложностей, с ко-

[1] В контексте ООП. — *Примеч. пер.*

торыми вы можете столкнуться на этом пути. Глава 4 содержит каталог паттернов, а глава 5, соответственно, — антипаттернов. В главе 6 описываются обобщенные решения наиболее распространенных проблем. Поскольку главы из этой части предназначены для использования в качестве каталога, то все они состоят из практически не связанных между собой разделов, которые можно читать независимо друг от друга и в произвольной последовательности.

- Часть 3 содержит анализ ВЗ по трем различным аспектам: композиция объектов (object composition), управление их жизненным циклом (lifecycle management) и перехват (interception).

  В главе 7 обсуждаются вопросы реализации ВЗ на основе уже существующих фреймворков приложений, таких как WCF, ASP.NET MVC, WPF и некоторых других. Во многих случаях материал главы 7 может быть использован в качестве каталога способов реализации ВЗ для определенного набора фреймворков. В главе 8 описывается, как осуществлять управление жизненным циклом зависимостей, избегая при этом утечки ресурсов. Поскольку структура этой главы менее строга, чем структура предыдущих глав, значительная часть ее может использоваться в качестве каталога широко известных жизненных стилей. Глава 9 описывает порядок сборки приложений с использованием сквозных аспектов приложения (cross-cutting concerns). В этой главе мы получим реальные результаты от проделанной ранее работы, так что я считаю ее кульминацией всей части.

- Часть 4 содержит описание библиотек контейнеров внедрения зависимостей. В двух главах рассмотрены контейнеры Castle Windsor и StructureMap.

Чтобы разговор о принципах и паттернах ВЗ не был привязан к API определенных контейнеров, большинство материала в этой книге, за исключением части 4, написано без ссылок на какой-нибудь конкретный контейнер. По этой причине контейнеры интенсивно обсуждаются только в части 4. Я твердо уверен, что благодаря общему обсуждению это издание будет полезным в течение более длительного периода времени.

Условные обозначения и возможности для скачивания

В этой книге содержится много примеров кода. Большинство из них написано на C#, но в разных местах встречается и код XML. Исходные коды в листингах и тексте книги приводятся моноширинным шрифтом, чтобы отделить его от обычного текста.

Весь исходный код в издании написан на языке C# в среде Visual Studio 2010. Приложения, имеющие архитектуру ASP.NET MVC, созданы в соответствии со стандартами ASP.NET MVC 3.

Лишь немногие технологии, рассмотренные в этой книге, базируются исключительно на возможностях современных языков программирования. Я начинал со слабо связанного кода на платформе .NET 1.1, и большинство примеров могло бы быть реализовано без каких-либо дополнительных ухищрений. Как бы то ни было, я хотел достичь разумного баланса между консервативным и современным

стилями программирования. Когда я пишу профессиональный код, я использую самые передовые возможности языка для достижения наилучшего результата. Но в этой книге нет ничего сложнее дженериков (generics) и LINQ. Меньше всего мне хотелось бы произвести неверное впечатление: дескать, внедрение зависимостей применимо только в ультрасовременных языках.

Написание кода в качестве примеров для любой книги связано с характерными проблемами. В отличие от экрана современного компьютера, книга позволяет размещать в строке только очень короткие фрагменты кода. Было очень заманчиво писать код в сжатом стиле, используя краткие, но малопонятные имена методов и переменных. Такой код труден для понимания, даже когда написан в реальных программах, и работа с ним ведется в интегрированной среде разработки и отладчике. Но в книге работать с ним крайне сложно. Я убежден, что следует использовать настолько удобочитаемые имена, насколько это возможно. Чтобы соответствовать всем этим требованиям, мне зачастую приходилось прибегать к нестандартным переносам. Весь код при этом остается компилируемым, но время от времени такое форматирование выглядит немного смешно.

В коде интенсивно используется ключевое слово `var`. В профессиональном коде я практически не применяю это слово, но нахожу его полезным в коде примеров, когда оно используется совместно с явными объявлениями, так как интегрированная среда разработки в данной ситуации не может помочь. В целях экономии места я также использую слово `var` всюду, где, по моему мнению, явные объявления не являются необходимыми.

Ключевое слово `class` часто применяется как синоним слову `type`. В .NET классы, структуры, интерфейсы, перечисления и т. д. являются типами, но поскольку слово `type` в обычной речи, не связанной с программированием, имеет множество разнообразных смыслов, то его использование сделало бы текст менее понятным.

Большинство кода в этом издании относится к всеобъемлющему примеру, используемому на протяжении всей книги: онлайновому магазину, дополненному вспомогательными внутренними приложениями. Возможно, вы надеялись увидеть более захватывающий пример, но я выбрал его по нескольким причинам.

- Это хорошо известная проблемная область для большинства читателей. И хотя она может показаться скучной, я думаю, что она обладает определенными достоинствами, поскольку до сих пор мало исследовалось применение внедрения зависимостей в данной области.
- Я не представляю какую-либо другую предметную область, настолько широкую, чтобы в ней можно было реализовать все разнообразные сценарии, которые я держу в голове.

Я написал массу кода для использования в качестве примеров, и большая часть этого кода даже не вошла в книгу. Фактически я написал почти весь мой код, следуя принципам разработки через тестирование (Test-Driven Development, TDD), но поскольку эта книга не о TDD, я практически не включал в нее модульные тесты.

Исходные коды всех примеров из этой книги доступны на сайте издательства Manning: http://manning.com/DependencyInjectionin.NET. Файл `ReadMe.txt` в корневом каталоге скачанных материалов содержит инструкции по компиляции и запуску кода.