Введение

Спасибо вам за то, что выбрали эту книгу, и добро пожаловать в чудесный мир рефакторинга! Я надеюсь, что вы сочтете эту книгу полезной в повседневной программистской работе, что она поспособствует обсуждению проектных решений с коллегами, что она пригодится, когда вы решитесь приступить к улучшению какого-нибудь старого запутанного кода, и когда вы проснетесь ночью, приведет к кристаллизации в вашем сознании нескольких строк, которые никак не давались днем. Если это — ваше первое знакомство с рефакторингом, я рассчитываю, что эта книга в корне изменит ваш стиль программирования и мышления о коде. Этот предмет непрост для изучения, и вам судить, насколько хорошим учителем я смог для вас стать.

Я стал воспринимать рефакторинг в систематической манере после прочтения книги Мартина Фаулера (Martin Fowler) *Refactoring: Improving the Design of Existing Code* (Addison-Wesley, 1999 г.). Эта книга была одновременно и приземленной, и практичной, и помогла мне изучить некоторые непревзойденные приемы, которые я смог немедленно применить в реальных проектах. Книга не основывалась на какой-то сложной теории и не содержала никаких сложных математических формул.

Книга была написана на языке простом и понятном любому, кто занимается написанием кода. Вскоре после ее прочтения я заметил перемены в своем стиле программирования:

- я стал гораздо быстрее и увереннее обнаруживать проблематичный код и дефекты дизайна;
- я смог обдумывать решения этих проблем и эффективно решать их посредством рефакторинга;
- разговаривая с коллегами, я смог ясно и убедительно объяснять свои проектные решения.

Я перестал воспринимать код, как нечто незыблемое и неизменное во времени, и начал относиться к нему, как к пластичной, податливой форме, которую можно в любой момент трансформировать в соответствии со своим видением и текущими потребностями. Это стимулировало фундаментальные изменения в восприятии кода. Я понял, что существует способ модификации и расширения кода в эффективной, предсказуемой манере, с улучшением его дизайна в процессе.

Вскоре понимание рефакторинга стало распространяться в команде, в которой я работал, и я видел, что все больше и больше моих коллег изучили эту книгу. Некоторые даже сделали себе собственные копии. Я смог говорить с ними, используя терминологию рефакторинга, и сделать рефакторинг неотъемлемой частью процесса конструирования программного обеспечения. Даже менеджмент стал менять свои взгляды в этом направлении.

Отчасти из-за моего собственного интереса к изучению различных языков и технологий, отчасти по необходимости, которая часто возникала в процессе работы,

мне приходилось работать с разными командами и писать на разных языках программирования. Я попытался применить рефакторинг при работе с командами, программировавшими на C#, в той же манере, как делал это, работая с командами Java. Это прошло не совсем гладко. Вскоре я понял, что существует очень мало информации о рефакторинге, доступной программистам .NET. Хотя большинство приемов рефакторинга применимы сходным образом к любому объектно-ориентированному языку, все же существуют некоторые тонкие различия. Нет причин, которые мешали бы программистам изучать рефакторинг, глядя на примеры кода, написанного на их любимом языке.

Это вдохновило меня на написание книги о рефакторинге для Visual Basic .NET, а затем — и книги о рефакторинге в C# и ASP.NET. Я осознал, что есть реальная потребность в таком тексте, и что эта книга найдет практическое применение в руках многих, кто программирует на C#.

Для кого написана эта книга

Эта книга предназначена для опытных разработчиков C#, которые хотят познакомиться с миром рефакторинга. Чтобы получить максимальную отдачу от книги, вы должны обладать достаточным знанием языка программирования C# и иметь представление о принципах объектно-ориентированного программирования.

Если вы — начинающий программист, вы не сможете ограничиться этой книгой как первичным источником. Она не научит вас основам программирования на C#. Однако нет причин отказываться от изучения рефакторинга на ранних этапах вашей программистской карьеры. Как только вы научитесь программировать свои первые классы, вы тут же можете использовать эту книгу для того, чтобы научиться проектировать их правильно, а также тому, как исправлять ошибки, которые вы могли внести в первоначальный дизайн.

Вы можете найти эту книгу полезной, если пришли в C# из другого языка программирования, такого как C++, Java или VB.NET. Для такого программиста обычно не составляет труда понять синтаксис нового языка, но постичь сам "дух" языка удается не сразу. Эта книга поможет научиться программировать "в духе C#".

В этой книге я не выдвигаю никаких предположений относительно предметной области разрабатываемых приложений. Это может быть типичное веб-приложение, веб-служба, каркас, компонент, приложение онлайновой торговли, новый виджет Facebook либо игра; но что бы это ни было, до тех пор, пока оно написано на C#, вы найдете приемы, описанные в этой книге, весьма ценными.

Эта книга также имеет дело с некоторыми рефакторингами для ASP.NET. Это должно представлять особый интерес, если вы создаете веб-приложения на .NET и пишете код HTML и ASP.NET. Здесь я выхожу за рамки чистого кода C# и обращаюсь к коду и технологиям типа HTML, XHTML, CSS, HTTP и т.п. Хотя главы, относящиеся к ASP.NET, могут читать любые разработчики ASP.NET, приведенные здесь примеры написаны на C#.

Большая часть рефакторингов, с которыми я имею дело в этой книге — это стандартные приемы рефакторинга, применимые к любому полностью объектно-ориентированному языку. Это значит, что если вы программируете на некотором объектно-ориентированном языке, скажем, C++, до тех пор, пока вы знакомы с синтаксисом C# и можете читать примеры кода, вы сможете применить на практике информацию, представленную в этой книге.

Какие темы охватывает эта книга

Я постарался превратить эту книгу в исчерпывающее представление приемов рефакторинга на C# и некоторых концепций рефакторинга, специфичных для ASP.NET.

Эта книга охватывает следующие базовые концепции рефакторинга:

- запах кода;
- трансформации кода в ходе рефакторинга;
- некоторые базовые объектно-ориентированные принципы;
- использование инструментов рефакторинга для автоматизации процесса рефакторинга.

В книге используется единственный относительно большой учебный пример, который позволяет продемонстрировать практическое применение рефакторинга к реалистично крупной кодовой базе.

В дополнение к стандартным рефакторингам, применимым к любому объектно-ориентированному языку программирования, эта книга обучает некоторым специфичным для C# приемам рефакторинга. Она также демонстрирует некоторые специальные случаи использования рефакторинга. Например, вы узнаете, как преобразовать классический код C# в LINQ и как использовать расширяющие методы. Также будет затронута тема шаблонов проектирования.

Эта книга содержит огромное количество описаний важнейших запахов и рефакторингов. Однако она не претендует на звание полного каталога рефакторинга; в связи с ограниченностью во времени и объеме некоторые важные виды рефакторинга были исключены. Например, я не затронул такие приемы рефакторинга, как "Упрощение условной логики" (Simplify Conditional) или "Обращение условной логики" (Reverse Conditional) — приемы, доступные для автоматизации в инструменте Refactor! For ASP от Developer Express.

Также опущены многие рефакторинги "обращением", наподобие "Встраивание класса" (Inline Class). Эта книга предназначена для того, чтобы служить введением в рефакторинг. Мой опыт показал, что первая проблема, с которой сталкиваются программисты, вступая на путь рефакторинга — это плохо структурированный код, а такие приемы, как "Выделение метода" (Extract Method) или "Выделение класса" (Extract Class), как раз решают эту проблему. Их противоположности — "Встраивание метода" (Inline Method) и "Встраивание класса" (Inline Class) — помогут справиться с чрезмерно структурированным кодом. Они предназначены для того, чтобы исключать конструкции (например, методы и классы), необходимость в которых отпала. Это часто случается после применения интенсивного рефакторинга к кодовой базе.

Это не значит, что рефакторинги "обращением" менее важны. Потребность в них, скорее всего, возникнет на поздних стадиях процесса рефакторинга, после того, как вы овладеете основными приемами. Частью процесса обучения является достижение понимания, что процесс познания не кончается никогда. Например, с каждой новой версией подключаемого модуля Refactor! к общему арсеналу поддерживаемых приемов добавляются все новые рефакторинги.

Люди постоянно изобретают и открывают новые рефакторинги. По мере накопления опыта вы подготовите собственные приемы рефакторинга и, может быть, захотите поделиться ими с другими разработчиками. Я приглашаю вас идти дальше, за пределы настоящей книги — не ограничивайтесь тем, что почерпнете отсюда. Ищите и пытайтесь открыть новые приемы и методики рефакторинга. Таким образом, вы станете настоящим мастером в искусстве непрерывного совершенствования кода.

Как организована эта книга

Поскольку это первая книга, которая посвящена исключительно рефакторингу C# и ASP.NET, и, возможно, первая книга на эту тему для многих читателей, я надеюсь, что она принесет пользу на нескольких уровнях.

1. В качестве подробного введения в рефакторинг на C#.
2. В качестве подробного руководства по приемам рефакторинга и "дурным" запахам кода, с которым можно консультироваться в повседневной работе по программированию.
3. В качестве демонстрационного пособия по применению приемов рефакторинга в сценариях реальной жизни, на примере одного учебного примера — приложения "Прокат автомобилей", которое анализируется и модифицируется в конце каждой главы на протяжении всей книги.

Чтобы достичь этих целей, эта книга построена подобно любой технической книге. Новые концепции представляются и исследуются в логической последовательности — от базовых до наиболее сложных. Каждую концепцию сопровождает упрощенный иллюстративный пример. Таким образом, вы можете читать эту книгу в логической последовательности — от начала до конца. И, скорее всего, вы сделаете это еще раз или два после того, как откроете ее впервые.

В дополнение к основному повествованию вы увидите разбросанные по всему тексту врезки определений запахов, рефакторингов и принципов объектно-ориентированного проектирования. Назначение этих определений — предоставить сжатый обзор каждого субъекта. В случае запахов, например, определение содержит эвристические правила их обнаружения. В случае рефакторингов присутствует раздел под названием "Механизмы", содержащий описание последовательности шагов, которые потребуется предпринять, чтобы эффективно провести рефакторинг. Вы должны уметь пользоваться этими "рецептами" в своей повседневной работе, чтобы постоянно напоминать себе о том, как выполняется рефакторинг, как обнаруживается запах, что должен устранить рефакторинг, и т.д.

В конце большинства глав содержится дискуссия о том, как рефакторинги, запахи и принципы, раскрытые в главе, отражаются на учебном примере, включенном в книгу. Вы можете загрузить код, сопровождающий главу, и просматривать его, читая о трансформациях, которые претерпевает учебный пример в данной главе. Назначение учебного примера — продемонстрировать реалистическое применение рефакторинга. Часто в технических книгах можно встретить примеры кода, которые чрезмерно упрощены, чтобы проиллюстрировать материал. Хотя это и делает примеры яснее, равно как и облегчает автору задачу проиллюстрировать излагаемую теорию, читатель часто сталкивается в реальной жизни с гораздо более сложными ситуациями, когда масса неожиданных препятствий мешают ему применить изученную по книге технику к реальному рабочему коду. В этой книге я постарался предложить читателям намного более реалистичный сценарий в виде учебного примера приложения "Прокат автомобилей".

Книга делится на пять основных частей, которые ведут читателя от самых простых базовых концепций до более сложных — в логической, легкой для восприятия последовательности.

- В **главах 1–4** закладывается фундамент книги. Например, в главе 1 я говорю о рефакторинге в общих понятиях. В этой главе также развенчаны некоторые распространенные мифы о рефакторинге. В главе 2 вы почувствуете вкус ре-

факторинга на практике, а в главе 3 ознакомитесь с обзором инструментов, необходимых для автоматизации работы по рефакторингу. В главе 4 представлен учебный пример, который затем рассматривается на протяжении всей книги.

- В **главах 5–8** рассматриваются некоторые стандартные, базовые приемы рефакторинга. Вы узнаете из них о важности правильного выбора имен для конструкций кода и о разрушительных последствиях дублирования кода. Вы познакомитесь с такими стандартными приемами рефакторинга, как "Выделение метода" (Extract Method), и углубитесь в детали структурирования кода на уровне метода.
- В **главах 9–11** раскрываются некоторые более развитые приемы рефакторинга, которые позволяют получить максимум отдачи от объектно-ориентированных возможностей среды программирования. Хорошие объектно-ориентированные навыки важны для восприятия этой части книги. Вы узнаете здесь об обнаружении классов, создании иерархий наследования и реорганизации кода в крупном масштабе.
- В **главах 12 и 13** показано, как успешно применять рефакторинг для достижения более специфических целей. Например, вы увидите, что рефакторинг может быть совмещен с шаблонами проектирования для создания еще более изощренного дизайна. Вы увидите некоторые новые средства, которые пришли с версией языка C# из Visual Studio 2008, и узнаете, как можно использовать рефакторинг для получения максимума выгоды от этих новых средств. В главе 13 речь идет о применении LINQ, расширяющих методов и прочих средств C# 3.0. Будет даже произведен возврат к примеру из главы 2, чтобы предложить другое видение этого приложения — на этот раз, с точки зрения C# 3.0.
- В **главах 14 и 15** описан специфические рефакторинги для ASP.NET. В главе 14 закладывается фундамент для рефакторингов, рассмотренных в следующей главе. Она начинается с описания подключаемого модуля Refactor! for ASP.NET для Visual Studio, после чего объясняются исторические корни многих запахов, характерных для веб-приложений и HTML. В главе 15 вы узнаете, как сделать код совместимым с новейшими веб-стандартами, такими как XHTML и CSS, и как выиграть от применения механизмов повторного использования кода ASP.NET, подобных мастер-страницам и пользовательским элементам управления.

Что необходимо для использования этой книги

Чтобы успешно пользоваться этой книгой, понадобится следующее программное обеспечение.

- **Visual Studio .NET 2008.** Вам нужна, как минимум, версия Professional Edition, чтобы использовать подключаемый модуль Refactor! for ASP.NET. Вы сможете редактировать, выполнять код и проводить рефакторинг вручную в бесплатной редакции Visual C# Express Edition. В случае применения Visual Studio 2005 большая часть книги также будет полезна, за исключением главы 13, в которой рассматриваются исключительно средства, доступные в C# 3.0 и Visual Studio 2008.
- **Дополнительный модуль Refactor! for ASP.NET от Developer Express.** Загрузите последнюю версию этого бесплатного дополнения к Visual Studio с сайта Developer Express по адресу `http://www.devexpress.com/Products/Visual_Studio_Add-in/RefactorASP/`.

- **Microsoft SQL Server.** Этот продукт понадобится, чтобы запускать учебный пример, включенный в эту книгу. Это может быть версия MS SQL Server 2000, 2005 или 2008; версии Express также вполне достаточно для примера приложения, рассмотренного в книге.
- **Операционная система.** Вы можете использовать любую операционную систему, такую как Windows XP, Winodws Server 2003 или 2008, либо же Windows Vista или 7.

Соглашения

Чтобы помочь получить максимум пользы от текста и отслеживать, что происходит, на протяжении книги используется ряд соглашений.

Подсказки, советы и врезки, касающиеся текущего обсуждения, выделены и помечены курсивом.

Ниже перечислены используемые стили текста.

- Новые термины и важные слова *выделяются курсивом* при первом упоминании.
- Клавиатурные комбинации обозначаются как <Ctrl+A>.
- Имена файлов, URL-адреса и код внутри текста выглядят так:

  `persistence.properties`
- Код представляется двумя разными способами:
  - `моноширинный шрифт` — для большинства примеров кода;
  - **`полужирный моноширинный шрифт`** используется, чтобы подчеркнуть код, который особенно важен в текущем контексте.

Врезки “Запах”, “Рефакторинг” и “Принцип объектно-ориентированного проектирования”

В дополнение к упомянутым соглашениям, на протяжении книги вы встретите три типа врезок: “Запах”, “Рефакторинг” и “Принцип объектно-ориентированного проектирования”.

Врезка “Запах”

Эти врезки содержат сжатое описание кода с “душком”. Код с “душком” — важная концепция рефакторинга, и каждая такая врезка состоит из трех разделов.

- **Схематичная диаграмма.** Хотя и отдаленно напоминающие статические диаграммы классов UML, эти иллюстрации, конечно, не являются строгими в смысле соответствия канонам языка UML. Эти диаграммы призваны визуализировать трансформацию, которую производит с кодом данный рефакторинг. Не все рефакторинги сопровождаются схематичными диаграммами; некоторые маломасштабные рефакторинги уровня метода не подходят для такого рода иллюстраций. При создании диаграмм я вдохновлялся работой, выполненной Свеном Гортсом (Sven Gorts) в его работе *Refactoring Thumbnails* (Миниатюры рефакторинга). Дополнительную информацию об этих диаграммах вы найдете на его странице `www.refactoring.be/thumbnails.html`.
- **Обнаружение.** Здесь описаны некоторые эвристические методы обнаружения кода с “душком”.

- **Связанные с этим рефакторинги.** Здесь перечислены приемы рефакторинга, с помощью которых можно избавиться от этого запаха.
- **Обоснование.** Здесь объясняется негативный эффект от этого запаха в более теоретическом аспекте.

Врезка "Рефакторинг"

В таких врезках предоставляется собственно описание приема рефакторинга. Каждая врезка включает следующие разделы.

- **Мотивировка.** Здесь объясняется положительный эффект, оказываемые данным рефакторингом на код, а также то, как с его помощью улучшается дизайн приложения.
- **Связанные запахи.** Здесь перечисляются запахи кода, от которых помогает избавиться данный рефакторинг.
- **Механизмы.** Здесь предлагается пошаговый рецепт выполнения рефакторинга.

Врезка "Принцип объектно-ориентированного проектирования"

В этих врезках определены некоторые критичные принципы объектно-ориентированного проектирования, часто сопровождаемые короткими примерами в целях иллюстрации.

Указатель запахов

Для облегчения ссылок в следующей таблице перечислены запахи кода, описанные в настоящей книге, вместе с номерами страниц, где можно найти информацию о них.

Запах	Глава	Номер страницы
"Мертвый код" (Dead Code)	Глава 5	138
"Чрезмерная открытость" (Overexposure)	Глава 5	144
"Использование полностью уточненных имен вне раздела `using`" (Using Fully Qualified Names Outside ''Using'' Section)	Глава 5	151
"Неиспользуемые ссылки" (Unused References)	Глава 5	154
"Незначащие имена" (Unrevealing Names)	Глава 6	162
"Длинный метод" (Long Method)	Глава 7	189
"Комментарии" (Comments)	Глава 7	190
"Слепая обработка событий" (Event-Handling Blindness)	Глава 7	196
"Ленивый метод" (Lazy Method)	Глава 7	200
"Дублирование кода" (Duplicated Code)	Глава 7	202
"Магические литералы" (Magic Literals)	Глава 7	204
"Излишне обремененная временная переменная" (Overburdened Temporary Variable)	Глава 8	214
"Лишняя временная переменная" (Superfluous Temporary Variable)	Глава 8	218
"Класс данных" (Data Class)	Глава 9	254
"Проект, управляемый базой данных" (Database-Driven Design)	Глава 9	256

Запах	Глава	Номер страницы
“Большой класс” (Large Class)	Глава 9	266
“Процедурный проект” (Procedural Design)	Глава 9	270
“Отказ от наследства” (Refused Bequest)	Глава 10	305
“Неявные импорты” (Implicit Imports)	Глава 11	341
“Большое пространство имен” (Large Namespace)	Глава 11	346
“Несвязанное пространство имен” (Non-Coherent Namespace)	Глава 11	346
“Циклические зависимости” (Cyclic Dependencies)	Глава 11	353
“Документ отображается в разных браузерах по-разному” (Document Displays Differently in Different Browsers)	Глава 14	455
“Несовместимость документа с XHTML” (XHTML Document Non-Compliance)	Глава 14	458
“Презентационные элементы внутри документа” (Presentational Elements Inside the Document)	Глава 14	459

Указатель рефакторингов

В следующей таблице перечислены приемы рефакторинга, включенные в книгу, вместе с номерами страниц, где можно найти их описание.

Рефакторинг	Глава	Номер страницы
“Исключение мертвого кода” (Eliminate Dead Code)	Глава 5	140
“Сокращение уровня доступа” (Reduce Access Level)	Глава 5	147
“Перемещение элемента в более ограниченный регион” (Move Element to a More Enclosing Region)	Глава 5	149
“Замена полностью уточненных имен явным импортом” (Replace Fully Qualified Names with Explicit Imports)	Глава 5	152
“Удаление неиспользуемых ссылок” (Remove Unused References)	Глава 5	154
“Переименование” (Rename)	Глава 6	169
“Безопасное переименование” (Safe Rename)	Глава 6	175
“Выделение метода” (Extract Method)	Глава 7	185
“Встраивание метода” (Inline Method)	Глава 7	200
“Замена магического литерала константой” (Replace Magic Literal with Constant)	Глава 7	205
“Перемещение объявления к месту ссылки” (Move Declaration Near Reference)	Глава 8	211
“Перемещение инициализации к месту объявления” (Move Initialization to Declaration)	Глава 8	213
“Расщепление временной переменной” (Split Temporary Variable)	Глава 8	216
“Встраивание временной переменной” (Inline Temp)	Глава 8	219
“Замена временной переменной вызовом метода” (Replace Temp with Query)	Глава 8	221
“Введение поясняющей временной переменной” (Introduce Explaining Temporary Variable)	Глава 8	223

Указатель принципов объектно-ориентированного проектирования

В следующей таблице перечислены принципы объектно-ориентированного проектирования, включенные в эту книгу, вместе с номерами страниц, где они описаны.

Принцип объектно-ориентированного проектирования	Глава	Номер страницы
Открыто-закрыто	Глава 6	179
Одиночная ответственность	Глава 9	262
Программирование с абстракциями	Глава 10	298
Предпочтение композиции объектов наследованию классов	Глава 10	302
Эквивалентность повторного использования выпуску	Глава 11	347
Принцип нециклических зависимостей	Глава 11	356

Исходный код

Исходные коды примеров, рассмотренных в этой книге, доступны для загрузки на веб-сайте издательства по адресу `http://www.dialektika.com`.

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