Предметно-ориентированные языки программирования

Domain-Specific Languages

Martin Fowler
With Rebecca Parsons

ADDISON-WESLEY

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Предметно-ориентированные языки программирования

Мартин Фаулер
При участии Ребекки Парсонс

ВИЛЬЯМС

Москва • Санкт-Петербург • Киев
2011

ББК 32.973.26-018.2.75
Ф28
УДК 681.3.07

Издательский дом "Вильямс"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция канд. техн. наук *И.В. Красикова*

Научный консультант докт. физ.-мат. наук *Д.А. Клюшин*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Фаулер, Мартин.
Ф28 Предметно-ориентированные языки программирования. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 576 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1738-6 (рус.)

ББК 32.973.26-018.2.75

Научно-популярное издание

Мартин Фаулер

Предметно-ориентированные языки программирования

Литературный редактор *Л.Н. Красножон*
Верстка *О.В. Романенко*
Художественный редактор *В.Г. Павлютин*
Корректор *Л.А. Гордиенко*

Подписано в печать 30.06.2011. Формат 70x100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 46,44. Уч.-изд. л. 33,4.
Тираж 1000 экз. Заказ № 0000.

Отпечатано с готовых диапозитивов в ГУП "Типография «Наука»"
199034, Санкт-Петербург, 9-я линия В. О., 12.

ООО "И. Д. Вильямс", 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-1738-6 (рус.)
ISBN 978-0-321-71294-3 (англ.)

Оглавление

Содержание

Посвящается Синди
— Мартин

Предисловие

Предметно-ориентированные языки (Domain-Specific Languages — DSL) были частью компьютерного мира еще до того, как я научился программировать. Спросите ветеранов Unix или Lisp, и они будут счастливы утомить вас до судорог рассказами о том, как предметно-ориентированные языки помогали им в их работе и какие трюки с их помощью они ухитрялись выполнять. Тем не менее этим языкам не суждено было стать заметной частью упомянутого компьютерного мира. Большинство людей знают о предметно-ориентированных языках только по чьим-то рассказам, и часто эти рассказы ограничиваются описанием только одной из сторон имеющихся методов.

Я написал эту книгу в надежде изменить ситуацию и предоставить вам как можно больше информации о методах предметно-ориентированных языков, чтобы вы могли принять осознанное решение об их использовании в своей работе и о том, какие именно методы предметно-ориентированных языков применять.

Предметно-ориентированные языки популярны по нескольким причинам, но я остановлюсь только на двух из них: повышение производительности труда разработчиков и улучшение связи с экспертами в предметной области. Правильно выбранный язык может сделать сложный блок кода существенно проще для понимания, что повышает производительность работающих с ним. Он также упрощает общение разработчика программного обеспечения со специалистами в предметной области, по сути предоставляя текст, который одновременно действует и как выполняемое программное обеспечение, и как описание проблемы, которое узкие специалисты в данной области знаний могут прочесть, чтобы понять, как их идеи представлены в программной системе. Трудно переоценить возможность говорить со специалистами на одном языке — эта возможность разрушает самый высокий из барьеров на пути общения между программистами и их клиентами и устраняет массу узких мест в разработке специализированного программного обеспечения.

Однако не хотелось бы и преувеличивать значение предметно-ориентированных языков. Я часто говорю, что всякий раз, рассматривая преимущества применения предметно-ориентированного языка для решения той или иной проблемы, вы должны выполнить еще одно рассмотрение, подставив на этот раз вместо слов “предметно-ориентированный язык” слово “библиотека”. Многое из того, что вы получаете с помощью предметно-ориентированного языка, можно получить, создав соответствующую программную структуру. Большинство предметно-ориентированных языков на самом деле представляют собой просто косметический фасад над библиотеками или программной структурой. В результате выгоды от предметно-ориентированных языков часто оказываются меньшими, чем думают неискушенные в этих вопросах люди; однако в любом

случае для принятия решения нужно четко понимать, какие именно положительные и отрицательные стороны имеет каждое из решений. Знание хорошо себя зарекомендовавших технологий существенно снижает стоимость построения предметно-ориентированного языка (и я надеюсь, что моя книга поможет вам получить эти знания). Фасад может быть декоративной деталью, но часто он так же полезен, как и само здание.

Почему сейчас ?

Предметно-ориентированные языки известны издавна, но в последние годы они породили значительный всплеск интереса. И именно в это время я решил потратить пару лет на эту книгу. Я не уверен, что смогу пояснить, чем вызван такой интерес к предметно-ориентированным языкам, но я поделюсь своей личной точкой зрения.

На рубеже нового тысячелетия возникло ощущение превосходящей разумные рамки стандартизации в языках программирования, по крайней мере в моем мире корпоративного программного обеспечения. Несколько лет Java рассматривался как Единственный Язык Будущего, и даже когда Microsoft выпустила свой C#, он был очень похож на Java. В новых разработках доминирующую позицию занимали компилируемые статические объектно-ориентированные языки программирования с C-образным синтаксисом (даже Visual Basic не избежал попыток приведения его к этому общему знаменателю).

Но вскоре стало ясно, что не все в порядке в королевстве Java/C#. В наличии имелась логика, не хотевшая укладываться в ложе этих языков программирования, что привело к возникновению конфигурационных файлов в формате XML. Программисты шутили, что в результате они пишут больше строк XML, чем на Java/C#. Отчасти это было связано с желанием изменить поведение программы во время выполнения, а отчасти — с желанием выразить аспекты поведения более дружественным и простым для пользователя способом. XML, несмотря на его "зашумленный" синтаксис, позволяет определять собственный словарь и предоставляет строгую иерархическую структуру.

Но "зашумленность" XML все же оказалась слишком большой. Люди жаловались на то, что от угловых скобок у них рябит в глазах, и возникло желание получить преимущества XML-файлов конфигурации без затрат, связанных с применением XML.

Наш рассказ постепенно достиг взрывоподобного явления Ruby On Rails. Независимо от области применения этот язык оказывал огромное влияние на работу программистов с библиотеками и структурными схемами. Бо́льшая часть методов работы сообщества Ruby представляет собой более гибкий подход, при котором взаимодействие с библиотекой должно было походить на программирование на специализированном языке. Этот способ мышления восходит к одному из старейших языков программирования — Lisp. При этом подходе и способе мышления можно увидеть цветение даже на каменистой почве Java/C#: в обоих языках все более популярными становятся интерфейсы, вероятно, из-за влияния таких продуктов, как JMock и Hamcrest.

Узнав обо всем этом, я почувствовал, что у меня имеется немалый пробел в знаниях. Я видел людей, использующих XML там, где более удобным и ничуть не более трудным было бы применение иного синтаксиса. Я видел людей, пытавшихся загнать Ruby в тесные рамки сложных выражений, в которых гораздо легче было бы применить пользовательский синтаксис. Я видел программистов, играющихся с синтаксическими анализаторами там, где можно было бы обойтись существенно меньшим количеством работы с гибкими интерфейсами в их обычных языках программирования.

Мне кажется, что причина всех этих перегибов — в недостатке знаний. Квалифицированные программисты недостаточно знакомы с технологиями предметно-ориентирован-

ных языков, чтобы принять обоснованное решение о том, какие из них использовать. Этот пробел в знаниях я бы и хотел восполнить.

Значение предметно-ориентированных языков

Более подробно на эту тему мы поговорим в главе 2, "Использование предметно-ориентированных языков", а пока что я вижу две основные причины, по которым вы должны быть заинтересованы в предметно-ориентированных языках (а значит, и в методах, описанных в этой книге).

Первая причина заключается в повышении производительности труда программиста. Рассмотрим фрагмент кода

```
input =~ /\d{3}-\d{3}-\d{4}/
```

Вы можете распознать в нем регулярное выражение, и, возможно, вы знаете, что оно означает. Регулярные выражения часто критикуют за излишнюю загадочность, но представьте, как бы выглядел обычный код сопоставления шаблону без регулярных выражений. Насколько легко было бы понять и модифицировать такой код в сравнении с регулярным выражением?

Предметно-ориентированные языки позволяют сделать некоторые частные задачи программирования более легкими для понимания, а значит, соответствующие программы можно будет быстрее писать, легче изменять, и они будут менее подвержены ошибкам.

Вторая причина заинтересованности в предметно-ориентированных языках выходит за рамки программирования. Поскольку такие языки меньше и их легче понимать, они позволяют специалистам в предметной области, не являющимся программистами, понимать код, управляющий важными аспектами их профессии. Анализ реального кода со специалистами в предметной области позволит значительно обогатить и сделать более полезным общение между программистами и их клиентами.

Обсуждая подобные темы, люди часто говорят, что предметно-ориентированные языки позволят полностью избавиться от программистов. Я очень скептически отношусь к этому аргументу; в конце концов, то же самое в свое время говорили и о COBOL. Хотя, конечно, есть языки (такие, как CSS), написанные людьми, которые не называют себя программистами. Однако для предметно-ориентированных языков чтение написанного кода имеет большее значение, чем написание. Если эксперты в предметной области читают и в основном понимают написанный программистами код, то им будет гораздо легче общаться с программистом, который этот код создавал.

Вторая причина использования предметно-ориентированных языков не так проста, но награда ст*о*ит затраченных усилий. Общение между программистами и их клиентами — самое узкое место в разработке программного обеспечения, так что все, что может облегчить такое общение, ст*о*ит затраченных усилий.

Не бойтесь объема этой книги

Толщина этой книги может немного испугать. Я сам всегда настороженно отношусь к толстым книгам, потому что у всех нас не так уж много времени, которое мы можем посвятить изучению чего-то нового (и которое гораздо важнее при выборе книги, чем ее цена). Поэтому я воспользовался форматом, который предпочитаю в таких случаях — двойной книги.

Двойная книга — это две книги под одной обложкой. Первая — обычная повествовательная книга, предназначенная для чтения от корки до корки. Моя цель в этой книге — предоставить краткий обзор темы, достаточный для понимания, но не для практической работы. Объем этой части — не более пары сотен страниц, что вполне можно позволить себе прочесть полностью.

Вторая (бо́льшая по объему) книга представляет собой справочный материал, который предназначен не для подробного чтения (хотя некоторые люди так и делают), а для того, чтобы при необходимости обратиться к ней, как к справочнику. Одни читатели полностью прорабатывают первую книгу, чтобы получить общее представление о теме, а затем обращаются ко второй части через предметный указатель — в поисках только того материала, который их интересует. Другие читают справочный раздел так же, как и описательный. При разделении книги на две я хотел помочь вам в выборе материала, который можно пропустить и в который следует погрузиться как следует. Решение за вами.

Я также попытался сделать справочную часть в достаточной степени самостоятельной, так что если вы захотите узнать, например, о построении деревьев, то можете прочесть только соответствующую часть книги и получить вполне приличное представление о том, что следует делать, даже если вы подзабыли материал первой части. Таким образом, как только вы прочтете первую часть книги, она станет справочником, что очень удобно при практической работе, когда вам нужно найти информацию о конкретных деталях.

Основная причина, по которой книга получилась такой большой, — я просто не придумал, как сделать ее покороче. Одна из главных моих целей в этой книге заключается в предоставлении читателю обзора большого количества различных методов предметно-ориентированных языков. Есть книги о генерации кода, о метапрограммировании в Ruby и об использовании генераторов анализаторов. Я же стремился охватить все эти методы, чтобы вы могли лучше понять их сходства и различия. Все они играют определенную роль в более широком ландшафте, и моя цель — так организовать путешествие по этому ландшафту, чтобы вы ознакомились не только с отдельными пейзажами, но и с каждой из его частей достаточно подробно, чтобы иметь возможность тут же начать работу с описанными методами.

О чем вы узнаете из этой книги

Я задумал эту книгу как руководство по различным видам предметно-ориентированных языков и подходам к их построению. Часто, начиная экспериментировать с предметно-ориентированными языками, люди используют какую-то одну технологию. Смысл этой книги — показать вам широкий спектр методов, чтобы вы могли оценить, какие из них лучше всего подходят в ваших обстоятельствах. Я представил подробную информацию и примеры реализации многих из этих методов. Естественно, я не могу показать вам все, что можно сделать, но для принятия первоначальных решений этого должно быть достаточно.

Из первых глав вы получите представление о том, что такое предметно-ориентированные языки, когда они могут пригодиться и какова их роль в сравнении с библиотеками. Вы узнаете, с чего начать при построении внешних и внутренних предметно-ориентированных языков. В главах о внешних предметно-ориентированных языках рассказывается о роли синтаксического анализатора, полезности генератора анализаторов и о способах применения анализаторов для синтаксического анализа внешнего предметно-ориентированного языка. В главах, посвященных внутренним предметно-ориентированным языкам, показано, как использовать языковые конструкции в стиле таких язы-