5 Интернет-магазин

Основные темы:

- поэтапная разработка;
- примеры использования, последовательность страниц, структура данных;
- приоритеты.

Можно, конечно, потратить весь день на совместный разбор простых тестовых приложений, но это не поможет нам оплатить счета. Поэтому давайте нацелимся на что-нибудь более существенное и создадим на основе веб-технологий интернет-магазин под названием Depot.

Конечно, в мире и так достаточно всевозможных интернет-магазинов, но сотни разработчиков продолжают создавать все новые и новые подобные приложения. А чем мы хуже?

А если серьезно, то наша корзина покупателя поможет проиллюстрировать многие особенности Rails-разработки. Мы увидим, как создаются простые страницы обслуживания покупателей, как они привязываются к таблицам базы данных, как обрабатываются сессии и как создаются формы. На протяжении следующих 12 глав мы также будем обращаться к внешним темам, таким как блочное тестирование, безопасность и разметка страницы.

5.1. Поэтапная разработка

Разработка нашего приложения будет вестись поэтапно. Мы не станем пытаться определить буквально все до начала написания программного кода. Лучше мы выработаем объем спецификации, позволяющий приступить к программированию, и тут же создадим нечто работающее. Мы будем проводить практическую

апробацию идей, обобщать отзывы и продолжать работу, переходя к другому циклу мини-проектирования и разработки.

Придерживаться такого стиля программирования удается не всегда. Здесь требуется тесное сотрудничество с пользователями приложения, поскольку по мере продвижения разработки нам нужно собирать отзывы о работе продукта. Ошибаться можем и мы, и заказчик, обнаруживший, что, требуя от нас одно, он на самом деле хотел получить совсем другое. Не важно, кто стал виновником ошибки, но чем раньше мы ее обнаружим, тем меньше сил будет затрачено на ее исправление. В общем, применение этого стиля разработки характеризуется внесением многочисленных изменений в процессе работы.

Поэтому нам нужно воспользоваться таким набором инструментов, который не будет усложнять работу при изменении взглядов на создаваемый продукт. Если мы решили, что к таблице базы данных нужно добавить еще один столбец или изменить систему переходов между страницами, нам нужно иметь возможность войти в приложение и сделать все необходимое без объемных работ по программированию или долгой возни с настройками конфигурации. Вы сможете убедиться в том, что, когда дело касается внесения изменений, Ruby on Rails всегда на высоте — это идеальная среда для гибкого программирования.

Попутно мы будем создавать и поддерживать в рабочем состоянии набор тестов. Они помогут нам убедиться в том, что приложение всегда делает только то, для чего оно предназначено. Rails не только допускает создание подобных тестов, но и предоставляет вам исходный набор тестов при каждом определении нового контроллера.

Но вернемся к приложению.

5.2. Для чего предназначен Depot

Начнем с написания основной спецификации приложения Depot. Для этого рассмотрим самые общие примеры практики использования и составим краткое описание переходов по веб-страницам. Попытаемся также определить, какие данные понадобятся приложению (отдавая себе отчет в том, что наши исходные предположения могут оказаться неверными).

Примеры использования

Пример использования — это обыкновенное утверждение о том, как кто-нибудь использует систему. Консультанты изобретают подобные фразы, чтобы навесить ярлыки на вполне очевидные понятия, и то, что необычные слова всегда ценятся выше, чем обычные, которые на самом деле куда значимее, — это издержки бизнеса.

Примеры использования приложения Depot довольно просты (что для некоторых прозвучит трагически). Начнем с определения двух разных ролей или действующих субъектов: *покупателя* и *продавца*.

Покупатель использует Depot для просмотра продаваемого товара, выбора покупок и предоставления информации, необходимой для оформления заказа.

Продавец использует Depot для учета продаваемого товара, для определения заказов, ожидающих доставки, и для пометки доставленных заказов. (Продавцы также используют Depot, чтобы заработать кучу денег, выйти на пенсию и жить на тропических островах, но это уже тема какой-нибудь другой книги.)

И пока это все, что нам может понадобиться. Можно, конечно, мучительно уточнять, что такое «учет товара» и что подразумевается под «заказом, готовым к доставке», но зачем? Если есть какие-то неясные моменты, то вскоре, по мере демонстрации заказчику последовательных этапов нашей работы, все само собой прояснится.

Что касается отзывов, не забудем получить их прямо сейчас, чтобы убедиться в том, что наши начальные (весьма схематичные) примеры использования вполне отвечают представлениям нашего пользователя. Предположив, что наши примеры использования были приняты: давайте определим, как наше приложение будет работать с точки зрения различных пользователей.

Последовательность страниц

В любом случае нам нужно выработать замысел устройства основных страниц нашего приложения и составить примерное представление о том, как пользователи будут переходить со страницы на страницу. На ранней стадии разработки эта последовательность страниц, скорее всего, будет неполной, но она все же поможет нам сфокусироваться на том, что должно быть сделано, и определить последовательность действий.

Возможно, кому-то нравится создавать макеты страниц веб-приложения в Photoshop, Word или (как ни странно) с помощью HTML. А мне нравится работать с карандашом и бумагой. Так быстрее, и клиент также может включиться в работу, взяв карандаш и набросав варианты на бумаге.

Первый набросок последовательности страниц для покупателя показан на рис. 5.1. В нем нет ничего особенного. Покупатель видит страницу каталога, на которой он поштучно выбирает товар. Каждый выбранный товар попадает в корзину, которая демонстрируется после каждого выбора. Покупатель может продолжить покупки, используя страницы каталога, или подтвердить содержимое корзины и купить находящийся в ней товар. В процессе подтверждения мы получаем контактные и платежные сведения, а затем выставляем счет. Мы еще не знаем, как будем оформлять платеж, поэтому его детали в этой последовательности не имеют ясных очертаний.

Последовательность страниц продавца показана на рис. 5.2. Она также не отличается особой сложностью. После входа в программу продавец видит меню, позволяющее ему создать запись о товаре или просмотреть существующую запись, а также отправить существующие заказы. При просмотре товара продавец может выборочно отредактировать информацию о товаре или вовсе удалить касающуюся его запись.

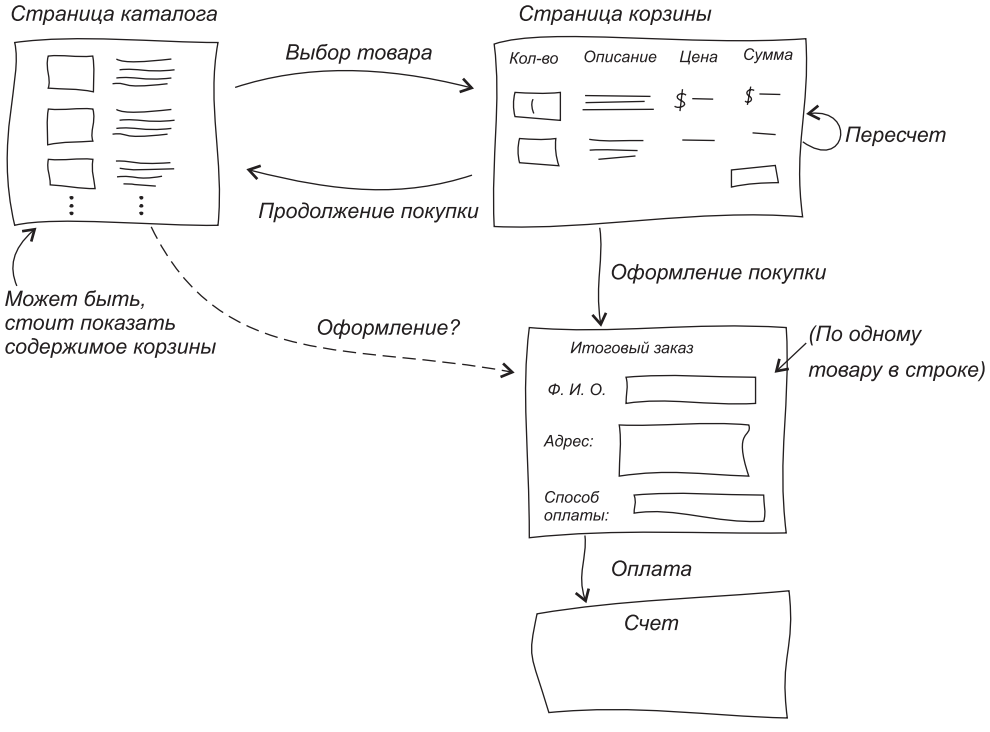

Рис. 5.1. Последовательность страниц, используемых покупателями

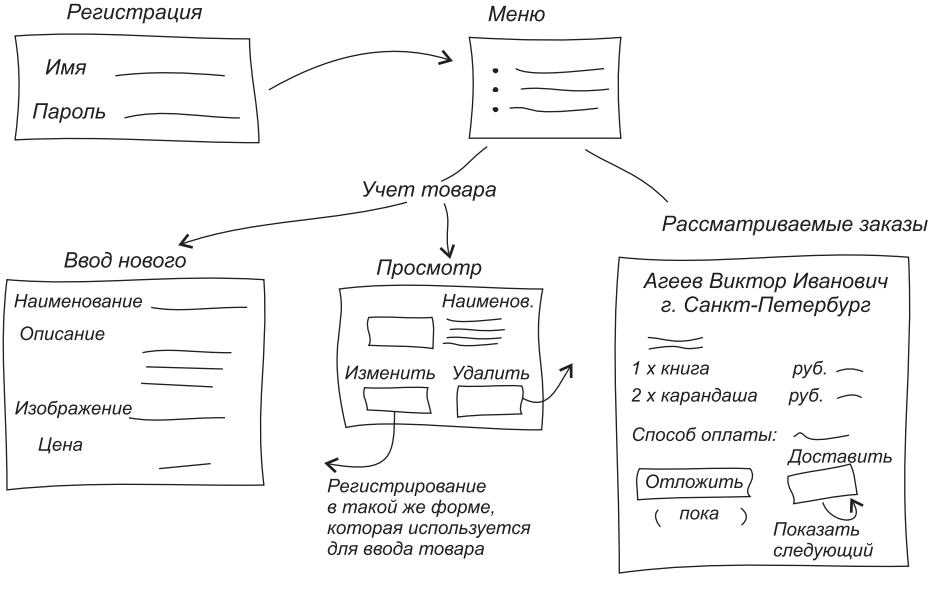

Рис. 5.2. Последовательность страниц, используемых продавцом

Вариант отправки сильно упрощен. Он предусматривает постраничный вывод каждого еще не отправленного заказа. Продавец может пропустить текущий заказ и перейти к следующему или может выбрать отправку заказа, используя, при необходимости, информацию со страницы.

Понятно, что функция отправки не выдержит никакого испытания практикой, но отправка относится к тем областям, реальное представление о которых составить довольно трудно. Проводить ее детальную проработку пока преждевременно, мы неизбежно наделаем массу ошибок.

Пока оставим все как есть, пребывая в уверенности, что сможем внести изменения, как только пользователи наберутся опыта работы с нашим приложением.

Данные

И, наконец, нужно подумать о данных, с которыми мы собираемся работать.

Заметьте, что здесь мы не употребляем такие слова, как «схемы» или «классы». Мы также не говорим о базах данных, таблицах, ключах и тому подобных вещах. Речь идет только о данных. На этой стадии разработки мы еще не знаем, будут ли использоваться базы данных.

Похоже, что на основании примеров использования и последовательности страниц мы будем иметь дело с теми данными, которые изображены на рис. 5.3. На мой взгляд, и здесь намного проще будет использовать карандаш и бумагу, а не какой-нибудь мудреный инструмент, но вы можете воспользоваться тем, что вам больше по душе.

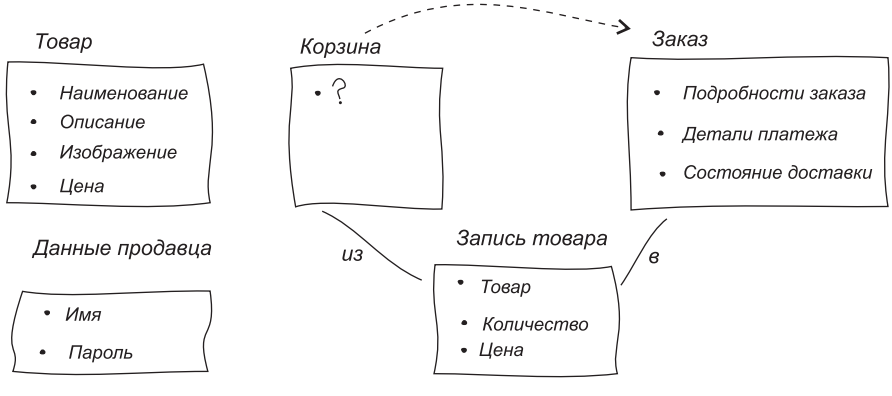

Рис. 5.3. Исходные предположения о составе данных, используемых в приложении

Работа с диаграммой, отображающей состав данных, наводит на некоторые размышления. Поскольку пользователь что-то покупает, нам нужно куда-то складывать покупки, поэтому я добавил корзину. Но что такое корзина, кроме как место для временного хранения списка товаров, представляется весьма смутно — я не в состоянии найти что-либо существенное, что можно было бы в ней хранить. Для обозначения этой неопределенности я поместил внутри прямоугольника, изображающего на диаграмме корзину, вопросительный знак.

Смею предположить, что эту неопределенность мы развеем в процессе создания приложения Depot.

В результате размышлений о данных общего характера возникает вопрос, какие сведения должны попадать в заказ. И тут я снова оставляю вопрос открытым — мы проясним его чуть позже, как только начнем демонстрировать пользователю наши первые шаги.

В конечном итоге вы могли заметить, что я продублировал цену товара в строке выбора. Здесь я несколько отступаю от правила «в начальной стадии все должно выглядеть как можно проще», но это нарушение продиктовано моим личным опытом. Если цена товара изменится, то это изменение не должно отражаться на том товаре, который попал в строку выбора текущих открытых заказов, поэтому в каждой строке выбора должна быть указана такая цена товара, которая существовала на момент оформления заказа.

Здесь я снова вместе с заказчиком проверил свои рассуждения и понял, что нахожусь на верном пути. (Вероятнее всего, мой заказчик сидел рядом со мной, пока я рисовал все эти три диаграммы.)

5.3. А теперь приступим к программированию

Итак, посидев рядом с заказчиком и проведя некоторый предварительный анализ, мы готовы сесть за компьютер и приступить к разработке с опорой на замысел, отображенный в трех исходных диаграммах. Но вскоре мы, наверное, их просто выбросим — они устареют, как только мы начнем получать отзывы о создаваемом приложении. Кстати, именно поэтому мы и не стали тратить на них слишком много времени — так будет проще с ними расстаться.

В следующих главах мы начнем создавать приложение, основываясь на нашем текущем представлении. Но пока мы не перевернули эту страницу, нужно ответить еще на один вопрос. С чего начать?

Я предпочитаю работать с заказчиком, поэтому мы можем выработать приоритеты совместными усилиями. В данном случае я обратил его внимание на то, что трудно разрабатывать что-нибудь другое, пока у нас не будет некоторого задела основных товаров, присутствующих в системе, поэтому я предложил потратить пару часов на то, чтобы получить начальную версию работоспособного средства по учету товаров. И он, конечно же, согласился.

ОБЩИЙ СОВЕТ ПО ВОЗВРАЩЕНИЮ НА ПРАВИЛЬНЫЙ КУРС

Весь материал данной книги был протестирован. При точном соблюдении описанного в ней сценария с использованием рекомендованных версий Rails и SQLite3 на Linux, Mac OS X или Windows все должно работать в соответствии с описаниями. Но могут случаться и отклонения от заданного маршрута. Никто из нас не застрахован от опечаток, и сопутствующие исследования не только возможны, но и категорически приветствуются. При этом следует знать, что они могут сбить вас с нужного курса. Но не стоит этого бояться: в конкретных разделах, где часто возникают подобные проблемы, будут появляться описания конкрет-

ных действий по возвращению на правильный курс. А общие советы на этот счет даются прямо сейчас.

Перезапускать сервер нужно будет только в нескольких случаях, когда это действие конкретно оговорено в данной книге. Но, когда вы попадете в полный тупик, можно будет попробовать перезапустить сервер.

«Волшебная» команда rake db:migrate:redo, о существовании которой вам нужно знать, подробно рассмотрена в части III. Она выполняет отмену и повторное использование последней миграции.

Если ваш сервер не примет какие-нибудь введенные в форму данные, обновите форму на своем браузере и повторите отправку данных.

6

Задача A: создание приложения

Основные темы:

- создание нового приложения;
- конфигурирование базы данных;
- создание моделей и контроллеров;
- добавление таблиц стилей;
- обновление разметки и представления.

Нашей первой задачей в разработке приложения станет создание веб-интерфейса, позволяющего вести учет товаров — вводить информацию о новых товарах, редактировать сведения об уже имеющихся, удалять ненужные товары и т. д. Мы разобьем разработку приложения на шаги, каждый из которых будет по времени выполнения занимать всего лишь несколько минут. Итак, приступим.

Обычно наши шаги будут состоять из нескольких составляющих этапов, например шаг B будет состоять из этапов B1, B2, B3 и т. д. В данном случае у шага будет два этапа.

6.1. Шаг A1: создание приложения по учету товаров

Основой приложения Depot является база данных. Если эта база уже установлена, сконфигурирована и протестирована, это может избавить вас от массы проблем. Если вы не уверены в том, что именно вам нужно, проще всего положиться на настройки по умолчанию. Если вы уже знаете, что вам нужно, Rails упростит вам описание выбранной конфигурации.

Создание Rails-приложения

В главе 2 мы уже видели, как создается новое Rails-приложение. Здесь мы сделаем то же самое. Перейдите в окно командной строки и наберите команду `rails new`, указав после нее имя нашего проекта. В данном случае наш проект называется `depot`, поэтому убедитесь в том, что вы не находитесь в каталоге уже существующего приложения, и наберите следующую команду:

```
work> rails new depot
```

Мы увидим, как по экрану побегут строки вывода. Когда этот процесс завершится, мы обнаружим новый каталог по имени depot. Именно с ним мы и будем работать.

```
work> cd depot
depot> dir /w
[.]            [..]           .gitignore     [app]          [config]
config.ru      [db]           [doc]          Gemfile        Gemfile.lock
[lib]          [log]          [public]       Rakefile       README
[script]       [test]         [tmp]          [vendor]
```

Создание базы данных

Для данного приложения будет использоваться база данных с открытым исходным кодом SQLite (которая вам понадобится, если вы намерены придерживаться представленного программного кода). В данной книге описывается использование SQLite версии 3.

SQLite 3 является базой данной, используемой при разработке Rails-приложений по умолчанию, и она была установлена вместе с Rails в главе 1 «Установка Rails». При работе с SQLite 3 не нужны никакие шаги по созданию базы данных, и поэтому нет никаких специальных учетных записей пользователей или паролей. Итак, сейчас вы начинаете убеждаться в преимуществах следования общему течению (или, в соответствии с избитой фразой пользователей Rails, следования соглашению о конфигурации).

Если вам важно использовать другой сервер базы данных, не SQLite 3, то команды, необходимые для создания базы данных, и порядок наделения полномочиями будут другими. Ряд полезных советов на этот счет можно найти на информационном ресурсе «Getting Started Rails Guide»[1].

Генерирование временной платформы

Ранее, на рис. 5.3, «Исходные предположения о составе данных, используемых в приложении», мы дали кратное описание основного содержимого таблицы товаров `products`. Давайте воплотим все это в реальность. Нам нужно создать таблицу базы данных и *модель* Rails, которая позволит нашему приложению использовать

[1] http://guides.rubyonrails.org/getting_started.html#configuring-a-database

эту таблицу, а также создать ряд *представлений* для формирования пользовательского интерфейса и *контроллер*, управляющий приложением.

Итак, давайте создадим для нашей таблицы `products` модель, представления, контроллер и миграцию. Работая с Rails, все это можно сделать с помощью одной команды, попросив Rails сгенерировать то, что называется *временной платформой* (scaffold) для заданной модели. Заметьте, что слово в командной строке[1], которая вскоре последует, используется в форме единственного числа — `Product`. В Rails модель автоматически отображается на таблицу базы данных, чье имя является формой множественного числа класса модели. В нашем случае мы запросили модель под названием `Product`, поэтому Rails связывает ее с таблицей по имени `products`. (А как же она найдет эту таблицу? Где ее искать, Rails подскажет запись `development` в файле config/database.yml. Для пользователей SQLite 3, это будет файл в каталоге db.)

```
depot> rails generate scaffold Product \
         title:string description:text image_url:string price:decimal
    invoke active_record
    create     db/migrate/20110711000001_create_products.rb
    create     app/models/product.rb
    invoke     test_unit
    create         test/unit/product_test.rbcreate
    create         test/fixtures/products.yml
     route resources :products
    invoke scaffold_controller
    create     app/controllers/products_controller.rb
    invoke     erb
    create         app/views/products
    create         app/views/products/index.html.erb
    create         app/views/products/edit.html.erb
    create         app/views/products/show.html.erb
    create         app/views/products/new.html.erb
    create         app/views/products/_form.html.erb
    invoke     test_unit
    create         test/functional/products_controller_test.rb
    invoke     helper
    create         app/helpers/products_helper.rb
    invoke         test_unit
    create             test/unit/helpers/products_helper_test.rb
    create app/assets/stylesheets/scaffolds.css.scss
    invoke assets
    invoke     coffee
    create         app/assets/javascripts/products.js.coffee
    invoke       scss
    create         app/assets/stylesheets/products.css.scss
    invoke     scss
    create         app/assets/stylesheets/scaffolds.css.scss
```

[1] Команда из-за своей длины не может целиком поместиться по ширине страницы. Чтобы ввести команду, располагающуюся на нескольких строках, нужно просто ставить обратный слэш в качестве последнего символа на всех строках, кроме последней, и тогда будет выдаваться приглашение на дополнительный ввод. Пользователям Windows вместо обратного слэша нужно будет поставить символ вставки (^).