Книга для успешного старта

Ruby on Rails

Быстрая веб-разработка

O'REILLY® bhv®

Брюс А. Тейт, Курт Ниббс

Ruby on Rails

Up and Running

Bruce A. Tate and Curt Hibbs

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Брюс А. Тейт
Курт Ниббс

Ruby on Rails

Быстрая веб-разработка

Санкт-Петербург
«БХВ-Петербург»
2008

УДК 681.3.06
ББК 32.973.26-018.2
Т96

Тейт, Брюс А.

Т96 Ruby on Rails. Быстрая веб-разработка: Пер. с англ. / Брюс А. Тейт, Курт Ниббс. — СПб.: БХВ-Петербург, 2008. — 224 с.: ил.

ISBN 978-5-9775-0224-5

Книга представляет собой быстрое практическое пособие по созданию веб-приложений в среде разработки с открытым кодом Ruby on Rails. Приведены базовые понятия и основные приемы работы в данной среде. Рассмотрены работа с базами данных, построение отношений между таблицами и их отображение, создание контроллеров для представления основных действий приложения, организация просмотра страниц с помощью шаблонов, включение в приложения функциональных возможностей Ajax, написание тестов для компонентов, функциональных и комплексных тестов и другие вопросы.

Для веб-разработчиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Перевод с английского	*Юлии Макаровой*
Редактор	*Юрий Якубович*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 29.02.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 18,06.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0224-5 (рус.)
ISBN 978-0-596-10132-9 (англ.)

Оглавление

Об авторах

Брюс Тейт живет в городе Остин, штат Техас, у него двое детей, и он любит погонять на горном велосипеде и на байдарках. В 2001 году он основал независимую консалтинговую компанию J2Life, LLC, которая сейчас называется RapidRed. Она занимается обучением, реализацией и консультированием по вопросам, связанным с быстрой разработкой веб-приложений с использованием Ruby on Rails. Клиентами этой фирмы являются компании FedEx, Great West Life, AutoGas, TheServerSide и BEA. За свой двадцатилетний опыт работы он проработал 13 лет в компании IBM, а также занимал руководящие должности в различных начинающих фирмах. Брюс Тейт является известным оратором и уважаемым автором девяти книг, посвященных разработке программного обеспечения, включая провокационную книгу "Beyond Java" (издательство O'Reilly), удостоенную премии Джолта "Better, Faster, Lighter Java" (издательство O'Reilly), ориентированную на управление книгу "From Java to Ruby" (издательство Pragmatic) и абсолютный бестселлер "Горький вкус Java" (издательство "Питер").

Курт Хиббс был всегда немного помешан на новых технологиях и тенденциях в развитии технологий. Но он объясняет это тем, что попросту ленив, и поэтому ищет новые методы и технологии, чтобы облегчить свою работу и повысить производительность. Это привело к тому, что в 2001 году он открыл для себя Ruby (на то время не очень известный за пределами Японии) и создал несколько очень успешных проектов Ruby с открытым исходным текстом. На протяжении практически всей своей карьеры, которая началась в начале 1970-х годов, Курт был консультантом в известных компаниях, таких как Hewlet Packer, Intuit, Corel, WordStar, Charles Schwab, Vivendi Universal и многих других. Он также был руководителем в нескольких начинающих компаниях. Сейчас Курт работает главным специалистом по программному обеспечению в компании Боинг в Сент-Луисе.

О книге

Животное на обложке книги "Ruby on Rails в действии" — это дикий горный козел (Carpa pyrenaica). Он обитает в высокогорных районах Европы, Центральной Азии, Северной Африки. Большую часть времени горный козел проводит на высоте от 2 500 до 3 500 метров. Он известен своими очень длинными рогами, длина которых может достигать 90 см у особей мужского пола. Во время брачного периода особи мужского пола дерутся между собой рогами, чтобы отстоять свои права.

Хотя подобный подвиг вызывает опасения, согласно легенде рога горных козлов были такими сильными, что если горному козлу будет угрожать опасность, он может прыгнуть с обрыва и приземлиться на свои рога абсолютно невредимым.

Рисунок на обложке английского издания взят из музея естествознания в городе Риверсайд. Шрифт на обложке — Adobe ITC Garamond. Шрифт текста — Linotype Birka; шрифт заглавия — Adobe Myraid Condensed; шрифт кода — TheSans Mono Condensed фирмы LucasFont.

Предисловие

Феномен Ruby on Rails проносится по индустрии информационных технологий, с абсолютным пренебрежением к укоренившимся языкам программирования, общепринятым правилам или к коммерческой поддержке. Можно получить море информации о Ruby on Rails из статей в Интернете, из замечательных книг или даже из курсовых работ. Однако чего-то не хватает. Как опытному программисту, вооруженному лишь поверхностными знаниями языка программирования Ruby, удается выходить за пределы основ и программировать на каркасе Rails?

В книге "Ruby on Rails в действии" мы не будем повторять справочное руководство или пытаться заменить Google. Вместо этого мы постараемся дать полную картину того, как работают приложения в Rails, а также расскажем, где найти информацию, которая не охватывается в данной книге. Вы узнаете, как Rails динамически привязывает признаки ко всем моделям базы данных, которые называются объектами Active Record. Получив общее представление о Rails, вы сможете выносить больше пользы из лучших справочных руководств, тем самым расширяя свои знания.

Мы не будем загружать вас словами. Вместо этого, мы дадим вам теорию в контексте приложения, разрабатываемого вместе с вами в этой книге. Мы проведем вас через создание простого проекта, требующего не больше, чем дневник или простой веб-магазин, и имеющего настолько простую структуру, что даже новичок в Rails сможет быстро понять происходящее.

Мы не будем пытаться охватить каждый новый признак. Мы покажем основополагающие, по нашему мнению, признаки, которые формируют самые важные элементы. Мы также коснемся механизма миграции и каркаса Ajax, которые еще не так хорошо освещены.

Короче говоря, мы не будем пытаться создать полную библиотеку по Rails. Мы дадим вам основу, необходимую для начала работы.

Для кого предназначена эта книга?

Книга "Ruby on Rails в действии" предназначена для опытных разработчиков, которые ранее не работали с каркасом Rails и, возможно, на языке программирования Ruby. Для использования этой книги вам не нужно быть опытным программистом в Ruby. Хотя программистом вы все же должны быть. Вы должны обладать достаточными знаниями о выбранной вами платформе, чтобы писать программы, устанавливать программное обеспечение, запускать скрипты, используя системную консоль, редактировать файлы, использовать базу данных и понимать, как работают основные веб-приложения.

Условные обозначения

В книге используются следующие печатные обозначения:

- основной текст

  обозначает имена файлов, расширения файлов, имя пути, каталоги и программы Unix, а также "быстрые" клавиши (такие как Alt и Ctrl);

- **полужирный шрифт**

  выделяет пункты меню и элементы интерфейса, а также URL и электронные адреса;

- *курсив*

  обозначает новые термины;

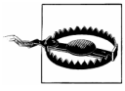
- `моноширинный шрифт`

  обозначает команды и вывод команд, а также содержимое файлов;

- **`моноширинный полужирный шрифт`**

  обозначает команды или другой текст, который должен напечатать пользователь.

Этот значок обозначает подсказку, предложение или примечание.



Этот значок обозначает предупреждение.

Использование примеров кода

Эта книга написана для того, чтобы помочь вам выполнить свою работу. В принципе, вы можете использовать код из этой книги в ваших программах и документации. Вам не нужно связываться с нами для разрешения, если вы не воспроизводите значительную часть кода. Например, написание программы, которая использует несколько небольших кусков кода из этой книги, не требует разрешения. Продажа или распространение компакт-дисков с примерами из книг издательства O'Reilly **требует** разрешения. Отвечая на вопросы, можете цитировать эту книгу и приводить примеры кода, для этого разрешение не нужно. Включение в документацию к вашему продукту значительной части кода из данной книги **требует** разрешения.

Вы можете найти пример кода на веб-странице книги "Ruby on Rails в действии": **http://www.oreilly.com/catalog/rubyrails/**. Там же вы найдете файл в ZIP-архиве, содержащий примерный код проекта, который получается после каждой главы. Каждый пример приложения пронумерован согласно главам. Если вы хотите пропустить главу, просто скачайте нужный файл.

Мы будем признательны за ссылки, хотя и не настаиваем на этом. Ссылки обычно включают название, автора, издательство и стандартный международный номер книги (ISBN). Например, ""Ruby on Rails в действии", Брюс Тейт и Курт Хибс, издательство O'Reilly Media, Inc., 2006. Все права защищены. 978-0-596-10132-9".

Если вам кажется, что вы не совсем честно используете примеры кода и вам требуется разрешение, свяжитесь с нами: **permissions@oreilly.com.**

Платформы

Ruby on Rails — это кроссплатформенный каркас. Но оболочки операционных систем Unix и Windows ведут себя по-разному. Чтобы быть последовательными, в этой книге мы будем использовать Windows. Вы можете легко запустить примеры из книги в операционных системах Unix или Mac OS X, но вы увидите несколько незначительных отличий.

- В Windows вы можете определять путь с помощью знаков прямая косая черта (/) или обратная косая черта (\). Мы постараемся быть последовательными и будем использовать прямую косую черту для определения пути.
- В Windows для запуска различных скриптов Ruby, которые составляют Rails, вам необходимо дополнительно печатать `ruby`. В среде Unix этого не требуется. Если вы используете Unix и вам надо напечатать команду `ruby script/server`, можете спокойно опустить `ruby`.

- В Windows для запуска процесса в отдельной оболочке вначале идет команда `start`. В Unix и Mac OS X добавьте знак `&` для запуска команды в фоновом режиме.

Safari® Enabled

Когда вы видите значок Safari® Enabled на обложке вашей любимой книги об информационных технологиях, это значит, что книга доступна онлайн через Книжную полку O'Reilly Network Safari.

Safari не просто предлагает электронные книги. Это виртуальная библиотека, которая дает возможность быстро искать среди тысяч лучших книг об информационных технологиях, вырезать и вставлять примеры кода, загружать главы и находить быстрые ответы, когда вам нужна самая точная и современная информация. Убедитесь в этом абсолютно бесплатно на сайте **http://www.safari.oreilly.com**.

Как с нами связаться

Мы проверяли информацию, изложенную в книге, и тестировали исходный код, но из-за большого объема текста и быстрой эволюции технологии вы можете обнаружить изменения в признаках или ошибки, допущенные нами. Сообщите нам, пожалуйста, о таких случаях по адресу:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

800-998-9938 (в США или Канаде)

707-829-0515 (международные и местные звонки)

707-829-0104 (факс)

Вы можете также послать электронное сообщение. Чтобы подписаться на рассылку или заказать каталог, пошлите электронное письмо на адрес:

info@oreilly.com

задать технический вопрос или оставить комментарий о книге, пошлите электронное письмо на адрес:

bookquestions@oreilly.com

Как уже упоминалось в предыдущем разделе, у нас есть веб-сайт этой книги, на котором вы можете найти код, список опечаток[1] (ранее замеченные ошибки и исправления доступные для просмотра) и другую информацию о книге. Сайт (на англ. языке) доступен по адресу:

http://www.oreilly.com/catalog/rubyrails/

Более подробную информацию об этой и других книгах можно найти на сайте издательства O'Reilly:

http://www.oreilly.com

Наши благодарности

Написание книги — очень трудоемкий процесс, требующий вдохновения, самоорганизации и настойчивости. Авторы с обложки получают всю славу (а возможно, и позор). Но свой вклад в книгу вносят многие люди. Мы хотели бы упомянуть тех, кто сделал написание этой книги таким увлекательным процессом.

Курт и Брюс хотели бы поблагодарить замечательную группу рецензентов, которые дали так много ценных комментариев, а именно Дэвида Мейбла, Мауро Чисио, Брук Хэндрик, Фэйзал Джодет, Шейн Клоусен, Лео де Блоу, Энн Бауман, Сес Хейверман, Дейва Гастингса и Ренди Хэнфорда. Мы также хотели бы поблагодарить Дэвида Гири за обогащение приложения Фотоальбом своими идеями.

Книга "Ruby on Rails в действии" была бы бессмысленна без неоценимого вклада команды разработчиков Ruby on Rails. Мы хотели бы поблагодарить Дэвида Хайнмаера Ханссона (создателя Rails), Флориана Вебера, Джеймис Бак, Джереми Кемпер, Леона Брида, Марселя Молина младшего, Майкла Козиарского, Николаса Секара, Сема Стефенсона, Скота Барона, Томаса Фукса и Тобиаса Лютке. Ruby — потрясающий язык, и мы хотели бы поблагодарить всех тех, кто сделал его таковым. Мы особенно признательны Юкихиро Мацумото (создателю Ruby), а также Дейву Томасу и Энди Ханту, без которых Ruby мог бы остаться фактически неизвестным за пределами Японии.

Брюс хотел бы особенно поблагодарить Курта, который присоединился к этому проекту, когда все уже казалось безнадежным. Также спасибо всем специалистам компании AutoGas, которые так эффективно испытывали эту технологию в контексте настоящего приложения разработки, особенно

[1] В русском переводе исправлены опечатки, замеченные и подтвержденные авторами к моменту перевода. — *Ред.*

команду разработчиков ядра, включая Мэтью Варгиза, Карла Хоншела, Чери Байерли, Криса Джиндорфа и Колби Блейсдела. Их коллективный опыт сформировал эту книгу в большей степени, чем можно подумать. Спасибо моему другу Лео из Голландии за его поддержку, хотя он в основном разработчик в Java. Он повлиял на меня больше, чем предполагает. Но больше всего я хочу поблагодарить свою семью. Кайла и Джулия, вы искорки в моей душе, которые поддерживают мой творческий огонь. Мэгги, ты мое вдохновение, я люблю тебя больше, чем могу выразить словами.

Курт хотел бы поблагодарить свою жену, Васану, за то, что позволила ему пропадать за компьютером до поздней ночи (а иногда и всю ночь) и не выражала своего недовольства. Я также хотел бы поблагодарить своих друзей в издательстве O'Reilly за то, что они предоставили мне возможность обсуждать необыкновенную продуктивность Ruby on Rails. Я хотел бы выразить особенную благодарность за опубликование моих статей на сайте ONLamp.com и сказать спасибо Майку Лоукидэсу за то, что он не сдался, когда я беспрерывно твердил ему, что не хочу писать книгу.

Глава 1

От нуля до шестидесяти. Введение в Rails

Rails можно назвать самым важным открытым проектом последних 10 лет. Он рекламируется как один из наиболее продуктивных каркасов для разработки веб-приложений всех времен и базируется на языке программирования Ruby, который приобретает все большую популярность. Что же уже произошло на сегодняшний день?

- В декабре 2006 года на прилавках книжных магазинов появилось больше книг о Rails, чем о любом из ведущих каркасов языка Java, включая JSF, Spring или Hibernate.
- Каркас Rails скачали как минимум 500 000 раз, и это только на втором году его существования, по данным на май 2006 года. Эта статистика выдерживает сравнение с самыми популярными открытыми каркасами в любом языке программирования.*
- Список подписчиков на почтовые рассылки о Rails пополняется сотнями адресатов в день, в то время как рассылки о каркасах для разработки приложений на других языках получают только десятки новых подписчиков в день.
- Каркас Rails привел к взрыву популярности языка программирования Ruby, который был малоизвестен до недавнего времени.
- Сбои в Rails вызывают горячие дебаты на форумах, посвященных другим языкам программирования. Сообщество программистов на Java, в частности, яростно обсуждало платформу Rails.

---

* Число 500 000 приводится официально. Статистика загрузок для популярного механизма доставки, который называется *gems*, позволяет легко отслеживать количество копий Rails, распространенных с его помощью, но есть и другие способы распространения, такие как Locomotive в Mac OS X. Реальная статистика загрузок может быть в два раза больше.

Не надо далеко ходить, чтобы найти огромные обзоры о Rails. Вы можете посмотреть несколько образовательных фильмов, которые демонстрируют Rails в действии. В них создатель Rails Дэвид Хэйнемеер показывает, как меньше, чем за 10 минут построить простое рабочее приложение, в котором выполняется и резервирование базы данных, и проверка данных. Но в отличие от многих сляпанных наспех окружений, которые вы видели раньше, Rails позволяет сделать все быстро и четко. Rails построит чистое приложение, основанное на философии *модель-представление-контроллер* (model-view-controller). Rails — это особый каркас.

Конечно, у Rails есть свои ограничения. В Ruby слабая поддержка *объектно-реляционной проекции* (Object-relational mapping, ORM) для унаследованных схем. Подход к этому в Ruby менее производителен, чем, например, в Java.* Кроме того, Ruby не имеет общепринятых интегрированных сред разработки. Каждый каркас имеет свои ограничения, и Rails не исключение. Но среди большого выбора приложений, достоинства Rails перекрывают все его недостатки.

Достоинства Rails

Из этой книги вы узнаете, как каркасу Rails удается процветать, не имея таких обширных библиотек, необходимых в других языках. Гибкость Ruby позволяет расширять ваши приложения такими способами, которые раньше были не доступны. Вы можете использовать свойство Rails, которое называется *скаффолдинг* (Scaffolding — строительные леса. — *Пер*.), для быстрого создания интерфейсов пользователя с базой данных. Затем вы дополняете приложение своим кодом, и скаффолдинг прячется за ним. Вы можете строить объекты модели, использующие базу данных, написав лишь несколько строчек кода, а Rails выполнит остальную утомительную работу.

Самая распространенная проблема для программистов в типичном проекте разработки заключается в построении интерфейсов пользователя для управления реляционной базой данных на основе интернет-технологий. При решении подобных задач Rails гораздо более продуктивен, чем любой другой каркас для разработки. И дело не в одном каком-нибудь революционном изобретении. Rails имеет черты, которые, в совокупности, делают вашу работу более продуктивной.

---

*Например, Hibernate поддерживает три вида наследования, а Rails поддерживает лишь наследование из одной таблицы. Hibernate поддерживает составные ключи, а подход Rails гораздо более ограничен.

Метапрограммирование

Техника метапрограммирования использует программы для написания программ. Другие каркасы применяют генерацию обширных кусков текста, что дает пользователям возможность продвинуться только на один шаг вперед, а скрипты настройки позволяют добавлять собственные команды только в небольшое количество тщательно отобранных точек. Метапрограммирование замещает эти две примитивные техники и устраняет их недостатки. Ruby — один из лучших языков для метапрограммирования, и Rails успешно использует это свойство.[**]

Active Record (Активная запись)

Rails вводит каркас Active Record, который сохраняет объекты в базу данных. Основываясь на одном из шаблонов проекта, которые каталогизированы Мартином Фаулером, версия Active Record в Rails находит нужные колонки в схеме базы данных и автоматически прикрепляет их к вашему объекту предметной области с помощью метапрограммирования. Этот метод обворачивания таблиц базы данных прост и элегантен, и к тому же предоставляет большие возможности.

Соглашение о конфигурации

Большинство каркасов для разработки веб-приложений для .NET или Java вынуждают вас писать страницы кодов конфигурации. Rails же, если вы придерживаетесь стандартных правил наименования, не потребует особых настроек. Фактически, следуя общим правилам, вы можете сократить код конфигурации в пять или более раз по сравнению с подобными каркасами Java.

Скаффолдинг (scaffolding)

Вы часто создаете временный код на ранних стадиях разработки, чтобы быстрее поднять приложение и увидеть, как работают вместе основные компоненты. Rails автоматически создает многие "строительные леса", которые вам понадобятся.

Встроенное тестирование

Rails создает простые автоматические тесты, которые вы можете в дальнейшем расширить. Rails также предоставляет код поддержки, который называется *harnesses* и *fixtures*, облегчающий написание и запуск контрольных примеров. Затем Ruby может выполнить все ваши автоматические тесты с помощью утилиты `rake`.

[**] Rails также использует генерацию команд, но гораздо больше полагается на метапрограммирование для выполнения сложных работ.

Три среды: разработка, тестирование и публикация

Rails дает вам три среды по умолчанию: разработка, тестирование и публикация. Каждая из них имеет свою специфику, что облегчает весь цикл разработки программного обеспечения. Например, Rails создает новую копию тестируемой базы данных для каждого запускаемого теста.

Есть еще много всего, включая Ajax для создания богатых пользовательских интерфейсов, частичные представления и помощники для повторного использования кода представления, встроенное кэширование, почтовый каркас и веб-сервисы. Мы не можем описать все свойства Rails в этой книге, однако мы подскажем, где найти дополнительную информацию. Но лучший способ оценить Rails — это увидеть его в действии, так что приступим.

Приводим Rails в действие

Вы можете вручную установить все компоненты для Rails, но в Ruby имеется инструмент под названием *gems*. Установщик gem заходит на веб-сайт **http://www.rubyforge.org/** и загружает прикладной компонент, называемый *gem*, вместе со всеми его составными частями. Чтобы установить Rails при помощи gems, запрашивая все необходимые составные части, нужно подать команду*

```
gem install rails --include-dependencies
```

Вот и все, Rails установлен. Одно предупреждение: вам также необходимо установить поддержку базы данных для имеющегося у вас сервера БД. Если у вас уже установлен MySQL, то больше ничего не требуется. Если нет, зайдите на сайт **http://rubyonrails.org/**, чтобы получить подробную информацию об установке Rails с другими базами данных.

Модель-представление-контроллер и Модель 2

В середине 1970-х годов стратегия модель-представление-контроллер (model-view-controller, MVC) появилась в обществе программистов в языке Smalltalk для предотвращения перемешивания бизнес-логики и логики представления. С помощью MCV вы помещаете бизнес-логику в отдельные модели предметной области (model) и изолируете ее от логики представления данных (view) моделей предметной области. Контроллер (controller) управляет навигацией между представлениями данных, обрабатывает введенные пользователем данные и распределяет подходящие объекты предметной области между моделью и представлением. С тех пор хорошие программисты используют MVC и реализуют MVC-приложения, используя каркасы, написанные на различных языках, включая Ruby.

Веб-разработчики используют немного другой вариант MVC, который называется Модель 2 (Model2). Модель 2 применяет те же принципы MVC, но приспо-

* Если вы хотите вводить код вместе с нами, убедитесь, что у вас установлены Ruby и gems. В *приложении 1* изложена подробная инструкция по установке.

сабливает их для веб-приложений, в которых нет текущего состояния. В приложениях Модели 2 браузер вызывает контроллер с помощью веб-стандартов. Контроллер взаимодействует с моделью для получения данных и подтверждает правильность данных, введенных пользователем, а затем подготавливает запрошенное представление модели предметной области. Далее, контроллер запускает нужный генератор представления, в зависимости от результатов проверки введенных пользователем данных или найденных данных. Уровень представления генерирует веб-страницу, используя данные, предоставленные контроллером. Затем каркас возвращает веб-страницу пользователю. Если в сообществе программистов Rails говорят о MVC, имеется в виду вариант Модель 2.

Модель 2 использовалась во многих успешных проектах, написанных на различных языках программирования. В Java самым распространенным каркасом Модели 2 является Struts. В Python ведущий каркас для разработки веб-приложений Zope использует Модель 2. О стратегии модель-представление-контроллер вы можете прочитать на сайте **http://en.wikipedia.org/wiki/Model-view-controller** (на английском языке; русская, сокращенная на нынешний момент версия этого текста находится по адресу **http://ru.wikipedia.org/wiki/Model-view-controller**).

Вот как создать проект в Rails:

```
> rails chapter-1
    create
    create  app/controllers
    create  app/helpers
    create  app/models
    create  app/views/layouts
    create  config/environments
    create  components
    create  db
    create  doc
    create  lib

    ...
    create  test/mocks/development
    create  test/mocks/test
    create  test/unit
    create  vendor
    ...
    create  app/controllers/application.rb
    create  app/helpers/application_helper.rb
    create  test/test_helper.rb
    create  config/database.yml
    ...
```

Мы сократили листинг, но картина ясна.

Структура

В каталогах, созданных во время установки, имеется место для вашего кода, скрипты для управления и создания приложения, а также много других полезных вещей. Позже мы подробно рассмотрим самые интересные из катологов. Сейчас же мы кратко расскажем о дереве каталогов в созданном нами проекте.

app

В этом каталоге организуются компоненты вашего приложения. У этого каталога есть подкаталоги, в которых содержится представление (views и helpers), контроллер (controllers) и внутренняя бизнес-логика (models).

components

Этот каталог содержит компоненты — маленькие замкнутые приложения, которые связывают модель, представление и контроллер.

config

В этом каталоге размещается код конфигурации, небольшое количество которого все же необходимо вашему приложению, включая конфигурацию вашей базы данных (в файле database.yml), структуру среды Rails (environment.rb) и маршрутизацию входящих веб-запросов (routing.rb). Вы можете также настроить поведение трех сред Rails — тестирование, разработка и применение, — используя файлы из каталога environments.

db

Обычно в приложении Rails есть объекты модели, которые обращаются к реляционной базе данных. Вы можете управлять реляционной базой данных, создавая скрипты и помещая их в этот каталог.

doc

В Ruby есть каркас под названием *RubyDoc*, который автоматически генерирует документацию для созданного вами кода. Вы можете помочь RubyDoc, создавая комментарии в коде. Этот каталог содержит все то, что генерирует RubyDoc, и документацию приложения.

lib

Сюда помещаются библиотеки, если они не должны находиться в другом месте (как, например, библиотеки поставщика).

log

Здесь находятся журналы регистрации ошибок. Rails создает скрипты для управления журналами регистрации различных ошибок. Здесь вы найдете отдельный журнал для сервера (server.log) и каждой среды Rails (development.log, test.log и production.log).

public

Подобно каталогу public для веб-сервера, этот каталог содержит веб-файлы, которые не изменяются, такие как файлы JavaScript (в каталоге public/javascripts), графические изображения (public/images), таблицы стилей (public/stylesheets), и файлы HTML (public).

scripts

Этот каталог содержит скрипты для запуска и управления различными инструментами, которые вы будете использовать в Rails. Например, там есть скрипты для генерации кода (`generate`) и запуска веб-сервера (`server`). Использованию этих скриптов будет еще уделяться много внимания в книге.

test

Здесь находятся тесты, написанные вами или созданные для вас Rails. Вы увидите подкаталог для mock-объектов (подкаталог mocks), тесты элементов (unit), фикстуры (fixtures) и функциональные тесты (functional). Мы подробно рассмотрим тестирование в *главе 7*.

tmp

Rails использует этот каталог для хранения временных файлов при промежуточной обработке.

vendor

В этом каталоге находятся библиотеки, написанные независимыми поставщиками (такие как библиотеки безопасности или утилиты для баз данных, не входящие в базовую поставку Rails).

Если исключить небольшие различия в разных версиях Rails, каждый проект Rails будет иметь одну и ту же структуру, с одинаковыми правилами наименования. Эта согласованность дает большое преимущество. Вы можете легко перемещаться между проектами Rails, не переучивая структуру проекта. Сам каркас Rails также использует это преимущество, т. к. разные каркасы Rails будут часто находить файлы, основываясь только на правилах наименования и структуре каталогов. Далее в этой главе вы увидите, как контроллер вызывает представления без ввода кода с вашей стороны.

Веб-сервер

Теперь, когда у нас есть проект, давайте запустим сервер. Напечатайте `cd chapter-1`, чтобы перейти к каталогу вашего проекта. Используйте скрипт `script/server` для запуска сервера WEBrick, сконфигурированного для разработки приложений. Если вы работаете в Windows, то перед каждым

вызовом скрипта должно стоять `ruby`, и вы можете использовать либо прямую, либо обратную косую черту. Если вы работаете в разновидности Unix, можете опустить ввод `ruby`:

```
> ruby script/server
=> Booting WEBrick...
=> Rails application started on http://0.0.0.0:3000
=> Ctrl-C to shutdown server; call with --help for options
[2006-05-11 07:32:08] INFO  WEBrick 1.3.1
[2006-05-11 07:32:08] INFO  ruby 1.8.4 (2005-12-24) [i386-mswin32]
[2006-05-11 07:32:08] INFO  WEBrick::HTTPServer#start: pid=94884
                                                        port=3000
```

Обратите внимание на несколько деталей.

- Сервер запустился на порте 3000. Вы можете изменить порт, отредактировав скрипт `script/server`. Дополнительные возможности конфигурирования описаны далее во врезке *"Конфигурирование сервера"*.
- Мы запустили WEBrick, простой веб-сервер, написанный на Ruby.
- Ruby позволяет использовать обратную косую в качестве разделителя в командной строке, но в Unix вы должны использовать прямую косую. Некоторые предпочитают обратную косую, т. к. это позволяет использовать функцию завершения кнопкой <Tab> в окне MS-DOS.

КОНФИГУРИРОВАНИЕ СЕРВЕРА

При необходимости вы можете конфигурировать порт, каталог для общедоступных файлов и другие возможности сервера, изменяя скрипт `script/server`. Ниже приведены настройки по умолчанию, выполняемые этим скриптом:

```
...
OPTIONS = {
  :port        => 3000,
  :ip          => "0.0.0.0",
  :environment => "development",
  :server_root => File.expand_path(File.dirname(__FILE__) +
"/../public/"),
  :server_type => WEBrick::SimpleServer
}
...
```

Вы часто встретите случаи написания конфигурации Ruby рядом с кодом, особенно в скриптах, подобных этому. Код между символами `{` и `}` — это *хеш-таблица* (hash map) в Ruby. Синтаксис `:key => "value"` преобразовывает `:key` в строку `"value"` в хеш-таблице (`OPTIONS` в предыдущем примере).

Обратите внимание на строку `:environment => "development"`; эта настройка запускает сервер в режиме разработки. Кроме прочего, режим разработки дает вам быстрый доступ к любому коду, которые вы хотите изменить, т. к. веб-сервер не помещает код в кэш-память.

Укажите в браузере адрес **http://127.0.0.1:3000/** или **http://localhost:3000/**. На экране вы увидите приветствие Rails, изображенное на рис. 1.1. Не волнуйтесь пока за детали запроса, главное, что Rails запущен и работает корректно.

Вы напечатали всего несколько слов и уже настроили среду разработки и веб-сервер, а также убедились, что сервер работает. В среде разработки вы можете держать сервер запущенным, перезагружаясь только для изменения конфигурации базы данных.

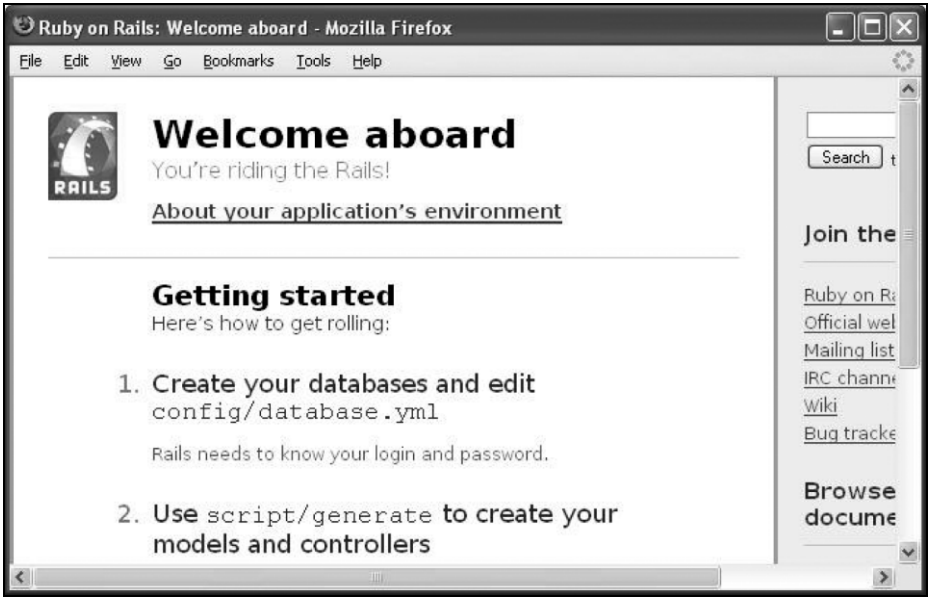


Рис. 1.1. Приветствие Rails

Выбор сервера

Rails может работать на разных веб-серверах. Большинство ваших разработок будут выполнены с использованием WEBrick, но вы, возможно, захотите запустить код разработки на другом сервере. Давайте рассмотрим имеющиеся серверы.

WEBrick

WEBrick, сервер по умолчанию для Rails, написан полностью на Ruby. Он поддерживает необходимые вам стандарты: HTTP для связей, HTML для веб-страниц и RHTML для встраивания кода Ruby в веб-страницы для динамического обновления содержимого. У WEBrick есть ряд важных преимуществ:

- он поставляется вместе с Ruby, поэтому он бесплатный и всегда доступен для использования или для создания пакетов к вашим проектам;
- он встроен в Rails, поэтому не надо прикладывать дополнительных усилий для интеграции;
- он может посылать прямые запросы приложению Rails, т. к. они оба написаны в Ruby;
- он прост в использовании.

Apache

Хотя WEBrick и один из самых удобных выборов, это не самый масштабируемый или гибкий выбор. Веб-сервер Apache — наиболее широко используемый веб-сервер в мире. Вы можете выбирать нужный из огромного множества плагинов для запуска десятков языков программирования или формирования других видов динамического содержания. Apache хорошо масштабируется, имеет замечательные кэширующие плагины и хорошую поддержку балансировщиков нагрузки и распылителей (sprayers) (механизмов, которые эффективно распределяют запросы между несколькими веб-серверами). Если вы ищете надежное решение, можете прекратить поиски, Apache — это то, что вам нужно.

lighttpd

Apache — хороший универсальный веб-сервер, но он не специализирован для некоторых специальных задач. lighttpd — это легковесный веб-сервер, который построен только ради одного — скорости. Он очень быстро обслуживает изображения и неизменяемое содержимое, такое как веб-страницы в формате HTML, а также поддерживает приложения через интерфейс приложения, который называется FastCGI. У lighttpd не так много гибких плагинов. как у веб-сервера Apache, и нет такой поддержки, но если вы ищете специализированный сервер для быстрого обслуживания неизменяемого содержимого и приложений Rails, lighttpd может оказаться хорошим решением. Он появился не очень давно, но любители Rails уважают его за скорость.

Mongrel

Хотя Apache и lighttpd очень быстрые и масштабируемые серверы, конфигурирование этих серверов для обслуживания приложений Rails может быть иногда не простой задачей, и уж точно это не так просто, как в WEBrick. Новый веб-сервер Mongrel может все изменить. Mongrel сочетает достоинства WEBrick (т. к. он написан в Ruby) и lighttpd (т. к. написан для скорости). Это сочетание могло бы сделать Mongrel прекрасным выбором как для разработки, так и для публикации. Он еще моложе lighttpd, но кажется таким перспективным, что одна крупная корпорация оказывает финансовую поддержку его разработке.

Другие веб-серверы

Теоретически любой веб-сервер, поддерживающий CGI (Common Gateway Interface, общий шлюзовой интерфейс), может обслуживать приложения Rails. К сожалению, CGI-интерфейс с Rails смертельно медленен, поэтому практически не подходит для публикации. Однако если вы работаете в специализированной среде, которая имеет свой собственный веб-сервер, вы, возможно, можете использовать его для обслуживания приложения Rails с помощью интерфейсов CGI и SCGI (Simple Common Gateway Interface, простой общий интерфейс шлюзов). Но сначала поищите информацию в Интернете, потому что, скорее всего, кто-нибудь это уже сделал и опубликовал инструкции. Например, если вы должны использовать ваше приложение Rails на Internet Information Server (IIS) от Microsoft, вы легко найдете инструкции, т. к. многие разработчики уже это делали. Возможно, вы обнаружите и другие веб-серверы, которые начинают использоваться для поддержки Rails.

Теперь сервер готов, пришло время написать какой-нибудь код. В этой главе мы остановимся на простых контроллерах и представлениях.

Создание контроллера

Вы уже видели, что Rails разбивает приложение на компоненты: модель, представление и контроллер. Мы начнем с контроллера. Для создания контроллера используйте скрипт `generate` (см. врезку *"script/generate"*). Сначала определяем тип объекта, который хотим создать, а затем имя нового контроллера.

```
> ruby script/generate controllers Greeting
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/greeting
      exists  test/functional/
```

```
create  app/controllers/greeting_controller.rb
create  test/functional/greeting_controller_test.rb
create  app/helpers/greeting_helpers.rb
```

Вы, наверное, не ожидали такой большой деятельности. Rails создал, как вы и хотели, контроллер — `greeting_controller.rb`. Но вы также получили и другие файлы:

application.rb

Пока еще нет контроллера для всего приложения, поэтому Rails создал его. Он пригодится позже, как место для размещения функций, важных для всего приложения, например, относящихся к безопасности.

views/greeting

Rails знает, что контроллеры и представления обычно идут парами, поэтому он создал каталог под названием views/greeting.

greeting_controller_test.rb

Rails также создал тест для вашего нового контроллера, т. к. большинство разработчиков Rails встраивают автоматические тесты, которые облегчают создание качественных приложений.

greeting_helper.rb

Помощники Rails (helpers) избавляют вас от повторения и написания лишнего кода.

Разработчики Ruby создавали Rails как инструмент для решения их собственных проблем, прежде чем выпустили его в качестве инструмента, который решит также и ваши проблемы. Вы видите пример прекрасной разработки, основанной на опыте. Первые пользователи Rails заметили, что сразу после создания контроллера им были необходимы дополнительные слои приложения, поэтому они модифицировали создание контроллера, чтобы избавить себя от нескольких нажатий клавиш. Разработчики Rails сами используют Rails.

SCRIPT/GENERATE

Генератор Rails — отличный помощник с поразительной продуктивностью. Он может помочь вам сгенерировать основные стандартные блоки для вашего приложения. Если вы забыли опции, напечатайте `ruby script/generate`, и вы получите следующие результаты (здесь они переведены на русский):

```
> ruby script/generate
Использование: script/generate  [options] generate [args]


Общие функции:
 -p, --pretend              Запустить, но не вводить изменений.
```

```
-f, --force                  Переписать уже существующие файлы.
-s, --skip                   Пропустить уже существующие файлы.
-q, --quiet                  Блокировать прямой выход.
-t, --backtrace              Отладка: Показать след в erRails.
-h, --help                   Показать эту подсказку.

Установленные генераторы
 Встроенные: controller, mailer, model, scaffold, web_service
```

Поскольку разные генераторы могут создать перекрывающие друг друга файлы, действия генератора могут оказаться разрушительными, если вы не будете внимательны. Не беспокойтесь, Rails вам поможет. Если вы не уверены в результатах работы генератора, лучше запустить его с опцией `--pretend`, чтобы точно знать, что он может сгенерировать.

Вы можете также составить список возможностей любого установленного генератора. Например, напечатав `ruby script/generate controller`, вы увидите опции, используемые при создании контроллера.

Вы можете устанавливать дополнительные генераторы. Например, вы можете использовать генератор логина, чтобы Rails мог создать модели, представления и контроллеры для базовой архитектуры аутентификации. Этот генератор также генерирует код для перехода из другой версии, scaffolding и даже вебсервис.

Чтобы найти генератор логина и другие имеющиеся генераторы, зайдите на **http://rubyonrails.org/show/Generators**. Для установки генератора воспользуйтесь gems. Например, чтобы установить генератор логина (`login_generator`) напечатайте:

```
gem install login_generator –s http://rubyonrails.org
```

Запуск контроллера

Давайте запустим приложение. Укажите в браузере адрес **http://127.0.0.1:3000/greeting**. Вы увидите сообщение об ошибке, говорящее вам о неизвестном действии `index`. Давайте выясним почему. Отредактируйте созданный в файле app/controller/greeting_controller.rb контроллер:

```
class GreetingController < ApplicationController
end
```

Вы пока еще не давали Ruby никаких команд, поэтому появление некоторых ошибок кажется логичным. Однако для устранения этой проблемы вам необходимо немного больше опыта. Рисунок 1.2 демонстрирует работу контроллеров.

Для управления контроллерами Rails использует каркас Action Pack. Веббраузеры связываются с серверами, посылая запросы через протокол HTTP. Для нашего приложения-приветствия запросом будет просто загрузить URL.

Первая часть URL идентифицирует устройство, а вторая — веб-ресурс. В Action Pack ресурс имеет как минимум три части: контроллер, действие, которое он должен выполнить, и идентификатор ресурса. Действия полностью соответствуют методам контроллера. Например, в этом URL

http://www.satulas.com/shopping_cart/total/45

часть `http://www.satulas.com/` идентифицирует веб-сервер, `shopping_cart` идентифицирует контроллер, `total` — действие, а `45` идентифицирует ресурс, вероятно корзину. Веб-сервер направляет входящий запрос в скрипт Ruby в каркасе Rails, который называется *диспетчер* (dispatcher). Rails имеет один диспетчер на веб-сервер. Диспетчер Rails анализирует URL и запускает нужное действие на нужном контроллере. Далее действие контроллера может вызвать модель и, в конечном счете, запускает представление.

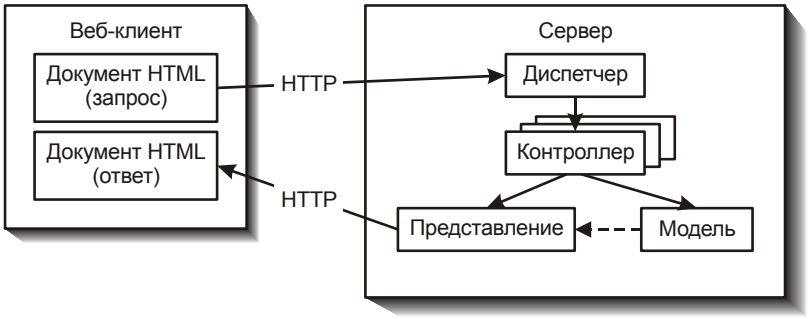


Рис. 1.2. Протекание процесса модель-представление-контроллер в Rails.

По умолчанию, если вы вызываете контроллер без указания действия, Rails вызывает действие `index`. Теперь причина ошибки понятна. Когда мы указали URL `app/controller/greeting`, мы не указали действие контроллера, поэтому Rails по умолчанию сослался на несуществующее действие `index`. Вы можете исправить ошибку, добавив метод под названием `index` к `GreetingController`. Давайте все упростим и заставим метод `index` сразу распечатать HTML, как это показано в примере 1.1.

Пример 1.1. Контроллер Rails отображает приветствие

```
Class GreetingController < ApplicationController
   def index
      render :text => "<h1>Welcome to your first Rails application<h1>"
   end
end
```

Сохраните код и обновите содержимое браузера — вы увидите веб-страницу, как на рис. 1.3. Даже если вы немного изменили код, вам не пришлось перезапускать сервер, повторно разворачивать приложение или делать еще что-нибудь, кроме обновления содержимого браузера. Быстрое оборотное время, которое называется *быстрая петля обратной связи* (rapid feedback loop), — отличительная особенность Ruby on Rails. Многие разработчики, начавшие работать в Rails, называют быструю петлю обратной связи именно тем свойством, которое больше всего повлияло на их продуктивность.

Рис. 1.3. Визуализация текста из контроллера

Построение представления

Сейчас у нас есть контроллер, который визуализирует текст, но примененный подход не может продвинуть вас существенно дальше. Если вы хотите следовать правилам модель-представление-контроллер, вы должны визуализировать текст в отдельном представлении, а не в контроллере. Сделанную нами сырую разработку легко исправить. Вместо написания чернового текста в контроллере, визуализируйте его в представлении. Как многие другие веб-каркасы, Rails может использовать шаблоны для представления. Для Rails шаблон — это просто HTML-страница с кодом на языке Ruby. Код Ruby выполняется на сервере, добавляя динамическое содержание HTML-странице.

Документация

В отличие от многих открытых проектов, Rails имеет отличную документацию, которая находится на сайте **http://api.rubyonrails.com**. Вы найдете обзоры, учебники и даже фильмы. Там же находится *интерфейс прикладного программирования* (Application Programming Interface, API) для последней версии Ruby on Rails, с полным набором документов для любого класса API в Rails. Этот же набор документов прилагается к установке Rails.

Отличная документация Rails появилась совсем не случайно. Как и к Java, к Ruby прилагается утилита под названием RubyDoc, которая генерирует доку-