深入浅出Rails（影印版）

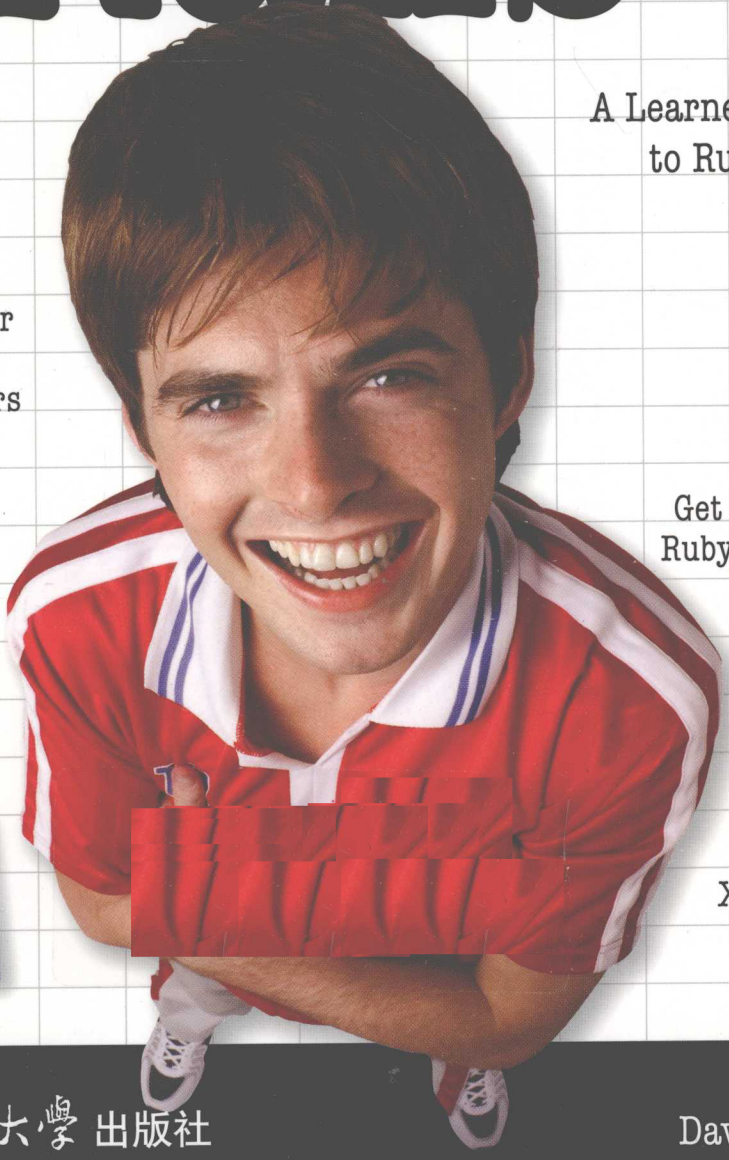
Head First Rails

A Learner's Companion to Ruby on Rails

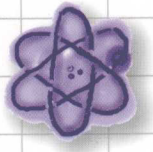
Master your data with Rails finders

Integrate your apps with Google Maps and never get lost again

Learn how Suzy validated all her dates and avoided another awful evening



Get inside Embedded Ruby, and take control of your apps



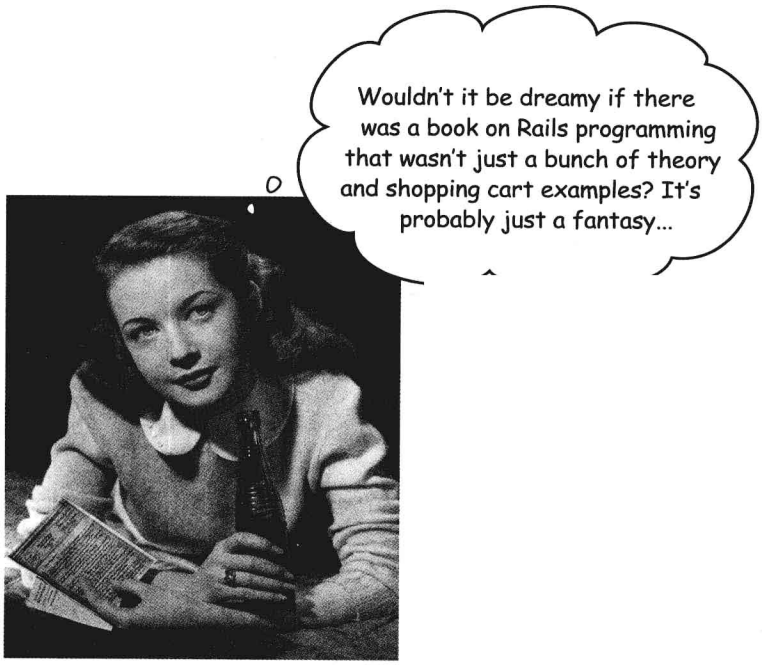
Add Ajax and XML to your Rails universe

O'REILLY® 東南大學出版社

David Griffiths 著

深入浅出Rails（影印版）

Head First Rails



David Griffiths

O'REILLY®

Beijing • Cambridge • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc.授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

深入浅出Rails：英文/(美)格里菲思（Griffiths, D.）著.—影印本.—南京：东南大学出版社，2010.6
书名原文：Head First Rails
ISBN 978-7-5641-2264-5

Ⅰ.①深… Ⅱ.①格… Ⅲ.①计算机网络－程序设计－英文 Ⅳ.① TP393.092

中国版本图书馆CIP数据核字（2010）第089088号

江苏省版权局著作权合同登记
图字：10-2010-151号

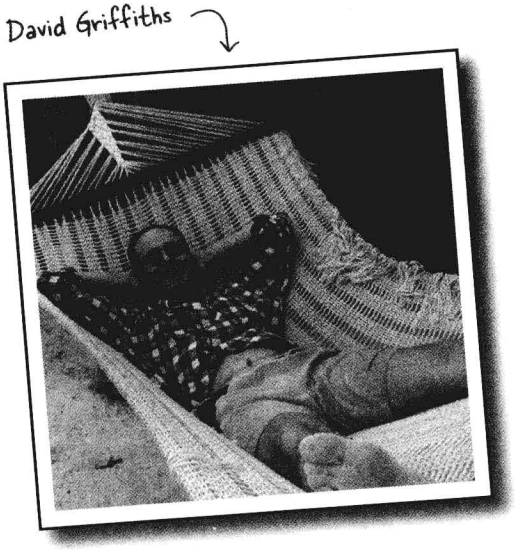
深入浅出Rails（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼2号　　邮编：210096
出 版 人：江　汉
网　　址：http://press.seu.edu.cn
电子邮件：press@seu.edu.cn
印　　刷：扬中市印刷有限公司
开　　本：787毫米×980毫米　12开本
印　　张：38.5印张
字　　数：644千字
版　　次：2010年6月第1版
印　　次：2010年6月第1次印刷
书　　号：ISBN 978-7-5641-2264-5
印　　数：1~1800册
定　　价：92.00元（册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

For Dawn, and in memory of my Mother, Joan Beryl Griffiths.

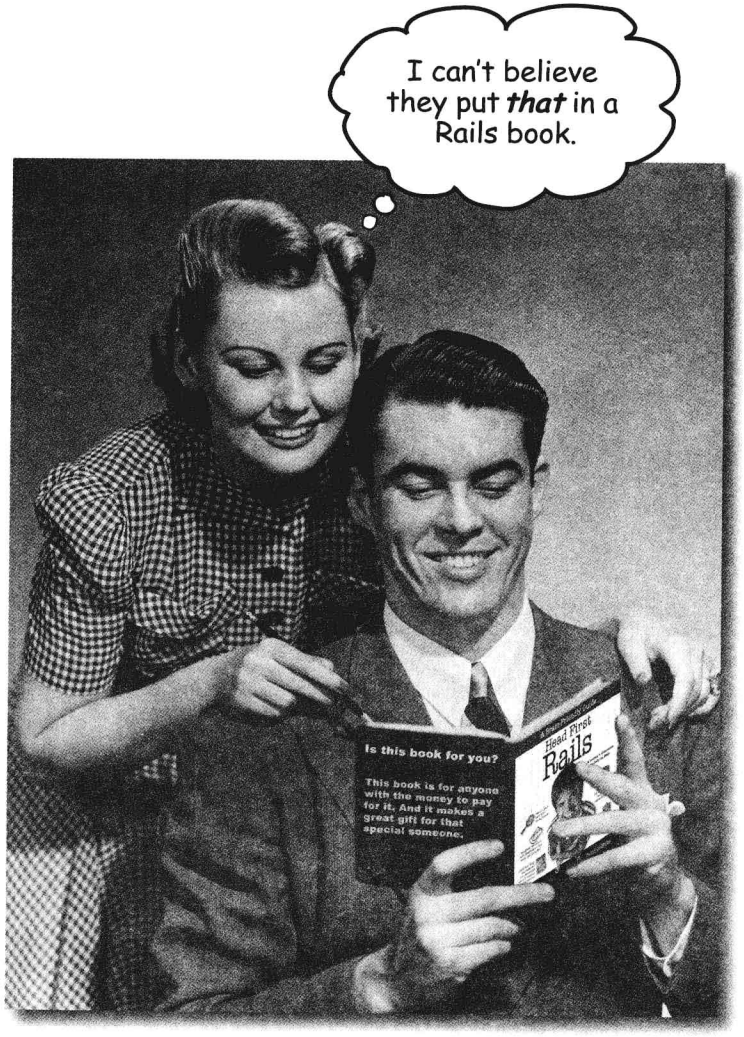
Author of Head First Rails



David Griffiths began programming at age 12, after watching a documentary on the work of Seymour Papert. At age 15 he wrote an implementation of Papert's computer language LOGO. After studying Pure Mathematics at University, he began writing code for computers and magazine articles for humans and he is currently working as an agile coach in the UK, helping people to create simpler, more valuable software. He spends his free time traveling with his lovely wife, Dawn.

how to use this book

Intro



In this section we answer the burning question:
"So why DID they put that in a Rails book?"

Who is this book for?

If you can answer "yes" to all of these:

1. Are you **comfortable with HTML**?
2. Do you have some experience of a computer language like **Java**, **C#** or **PHP**?
3. Do you want to build **cool stuff** for the web in a **fraction of the time** it used to take?

this book is for you.

Who should probably back away from this book?

If you can answer "yes" to any of these:

1. Are you someone who d**oesn't have any experience with HTML**?
2. Are you an **accomplished Rails developer looking for a reference book**?
3. Are you **afraid to try something different**? Would you rather have a root canal than mix stripes with plaid? Do you believe a technical book can't be serious if it anthropomorphizes clients and servers?

If this is the case, don't worry. Go pick up Head First HTML with CSS & XHTML by Elisabeth Freeman and Eric Freeman, and then come back to this book

this book is not for you.

[Note from marketing: this book is for anyone with a credit card.]

We know what you're thinking

"How can *this* be a serious Rails book?"

"What's with all the graphics?"

"Can I actually *learn* it this way?"

We know what your *brain* is thinking

Your brain craves novelty. It's always searching, scanning, *waiting* for something unusual. It was built that way, and it helps you stay alive.

So what does your brain do with all the routine, ordinary, normal things you encounter? Everything it *can* to stop them from interfering with the brain's *real* job—recording things that *matter*. It doesn't bother saving the boring things; they never make it past the "this is obviously not important" filter.

How does your brain *know* what's important? Suppose you're out for a day hike and a tiger jumps in front of you, what happens inside your head and body?

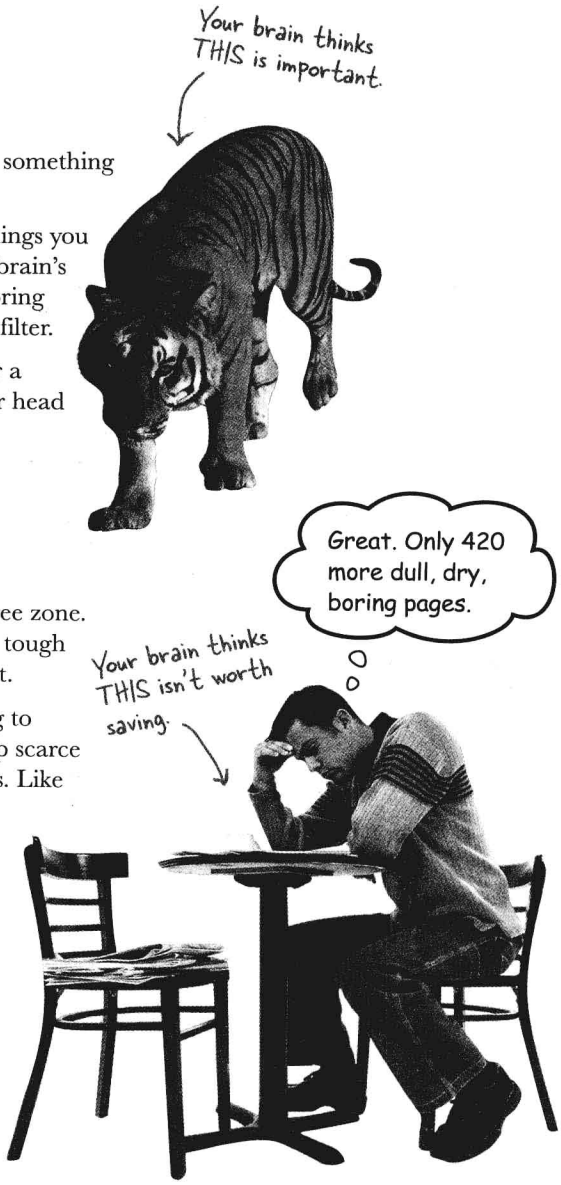
Neurons fire. Emotions crank up. *Chemicals surge.*

And that's how your brain knows...

This must be important! Don't forget it!

But imagine you're at home, or in a library. It's a safe, warm, tiger-free zone. You're studying. Getting ready for an exam. Or trying to learn some tough technical topic your boss thinks will take a week, ten days at the most.

Just one problem. Your brain's trying to do you a big favor. It's trying to make sure that this *obviously* non-important content doesn't clutter up scarce resources. Resources that are better spent storing the really *big* things. Like tigers. Like the danger of fire. Like the winners of the last three seasons of American Idol. And there's no simple way to tell your brain, "Hey brain, thank you very much, but no matter how dull this book is, and how little I'm registering on the emotional Richter scale right now, I really *do* want you to keep this stuff around."



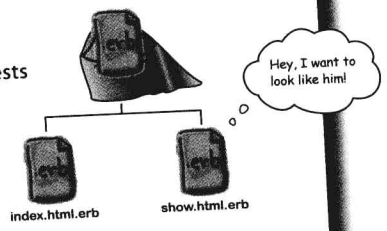
We think of a "Head First" reader as a learner.

So what does it take to *learn* something? First, you have to *get* it, then make sure you don't *forget* it. It's not about pushing facts into your head. Based on the latest research in cognitive science, neurobiology, and educational psychology, *learning* takes a lot more than text on a page. We know what turns your brain on.

Some of the Head First learning principles:

Make it visual. Images are far more memorable than words alone, and make learning much more effective (up to 89% improvement in recall and transfer studies). It also makes things more understandable. **Put the words within or near the graphics** they relate to, rather than on the bottom or on another page, and learners will be up to *twice* as likely to solve problems related to the content.

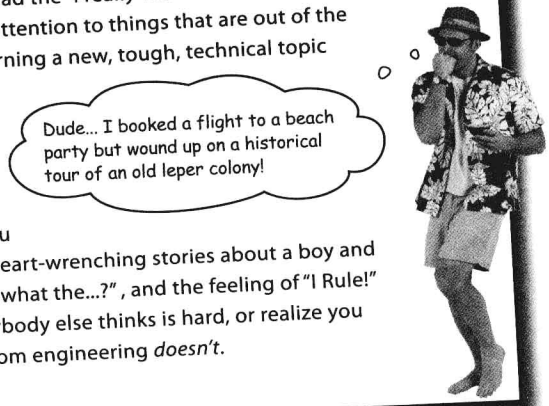
Use a conversational and personalized style. In recent studies, students performed up to 40% better on post-learning tests if the content spoke directly to the reader, using a first-person, conversational style rather than taking a formal tone. Tell stories instead of lecturing. Use casual language. Don't take yourself too seriously. Which would *you* pay more attention to: a stimulating dinner party companion, or a lecture?



Get the learner to think more deeply. In other words, unless you actively flex your neurons, nothing much happens in your head. A reader has to be motivated, engaged, curious, and inspired to solve problems, draw conclusions, and generate new knowledge. And for that, you need challenges, exercises, and thought-provoking questions, and activities that involve both sides of the brain and multiple senses.

Test Drive

Get—and keep—the reader's attention. We've all had the "I really want to learn this but I can't stay awake past page one" experience. Your brain pays attention to things that are out of the ordinary, interesting, strange, eye-catching, unexpected. Learning a new, tough, technical topic doesn't have to be boring. Your brain will learn much more quickly if it's not.



Touch their emotions. We now know that your ability to remember something is largely dependent on its emotional content. You remember what you care about. You remember when you *feel* something. No, we're not talking heart-wrenching stories about a boy and his dog. We're talking emotions like surprise, curiosity, fun, "what the...?", and the feeling of "I Rule!" that comes when you solve a puzzle, learn something everybody else thinks is hard, or realize you know something that "I'm more technical than thou" Bob from engineering *doesn't*.

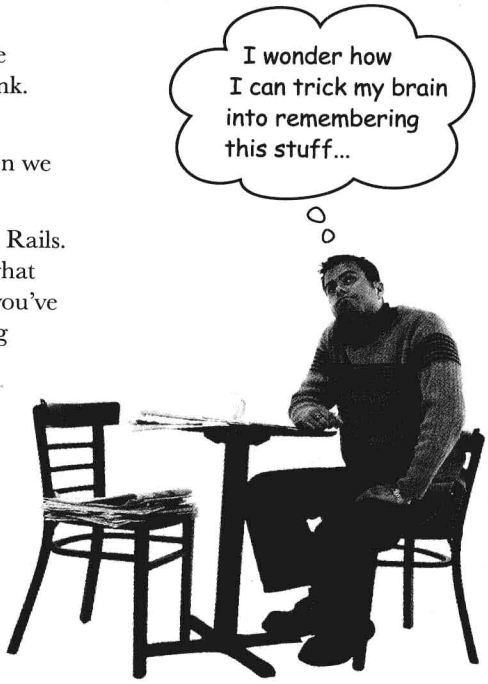
Metacognition: thinking about thinking

If you really want to learn, and you want to learn more quickly and more deeply, pay attention to how you pay attention. Think about how you think. Learn how you learn.

Most of us did not take courses on metacognition or learning theory when we were growing up. We were *expected* to learn, but rarely *taught* to learn.

But we assume that if you're holding this book, you really want to master Rails. And you probably don't want to spend a lot of time. If you want to use what you read in this book, you need to *remember* what you read. And for that, you've got to *understand* it. To get the most from this book, or *any* book or learning experience, take responsibility for your brain. Your brain on *this* content.

The trick is to get your brain to see the new material you're learning as Really Important. Crucial to your well-being. As important as a tiger. Otherwise, you're in for a constant battle, with your brain doing its best to keep the new content from sticking.



So just how *DO* you get your brain to treat Rails like it was a hungry tiger?

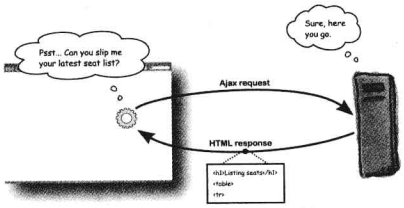
There's the slow, tedious way, or the faster, more effective way. The slow way is about sheer repetition. You obviously know that you *are* able to learn and remember even the dullest of topics if you keep pounding the same thing into your brain. With enough repetition, your brain says, "This doesn't *feel* important to him, but he keeps looking at the same thing *over* and *over* and *over*, so I suppose it must be."

The faster way is to do ***anything that increases brain activity,*** especially different *types* of brain activity. The things on the previous page are a big part of the solution, and they're all things that have been proven to help your brain work in your favor. For example, studies show that putting words *within* the pictures they describe (as opposed to somewhere else in the page, like a caption or in the body text) causes your brain to try to makes sense of how the words and picture relate, and this causes more neurons to fire. More neurons firing = more chances for your brain to *get* that this is something worth paying attention to, and possibly recording.

A conversational style helps because people tend to pay more attention when they perceive that they're in a conversation, since they're expected to follow along and hold up their end. The amazing thing is, your brain doesn't necessarily *care* that the "conversation" is between you and a book! On the other hand, if the writing style is formal and dry, your brain perceives it the same way you experience being lectured to while sitting in a roomful of passive attendees. No need to stay awake.

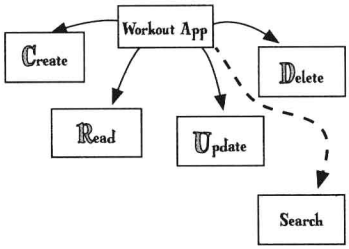
But pictures and conversational style are just the beginning...

Here's what WE did:



We used ***pictures***, because your brain is tuned for visuals, not text. As far as your brain's concerned, a picture really *is* worth a thousand words. And when text and pictures work together, we embedded the text *in* the pictures because your brain works more effectively when the text is *within* the thing the text refers to, as opposed to in a caption or buried in the text somewhere.

We used ***redundancy***, saying the same thing in *different* ways and with different media types, and *multiple senses*, to increase the chance that the content gets coded into more than one area of your brain.



We used concepts and pictures in ***unexpected*** ways because your brain is tuned for novelty, and we used pictures and ideas with at least *some* ***emotional*** *content*, because your brain is tuned to pay attention to the biochemistry of emotions. That which causes you to *feel* something is more likely to be remembered, even if that feeling is nothing more than a little ***humor***, ***surprise***, or ***interest.***

We used a personalized, ***conversational style***, because your brain is tuned to pay more attention when it believes you're in a conversation than if it thinks you're passively listening to a presentation. Your brain does this even when you're *reading*.

We included more than 80 ***activities***, because your brain is tuned to learn and remember more when you ***do*** things than when you *read* about things. And we made the exercises challenging-yet-do-able, because that's what most people prefer.

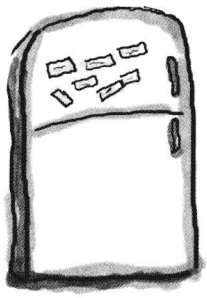
We used ***multiple learning styles***, because *you* might prefer step-by-step procedures, while someone else wants to understand the big picture first, and someone else just wants to see an example. But regardless of your own learning preference, *everyone* benefits from seeing the same content represented in multiple ways.

We include content for ***both sides of your brain***, because the more of your brain you engage, the more likely you are to learn and remember, and the longer you can stay focused. Since working one side of the brain often means giving the other side a chance to rest, you can be more productive at learning for a longer period of time.

And we included ***stories*** and exercises that present ***more than one point of view,*** because your brain is tuned to learn more deeply when it's forced to make evaluations and judgments.

We included ***challenges***, with exercises, and by asking ***questions*** that don't always have a straight answer, because your brain is tuned to learn and remember when it has to *work* at something. Think about it—you can't get your *body* in shape just by *watching* people at the gym. But we did our best to make sure that when you're working hard, it's on the *right* things. That ***you're not spending one extra dendrite*** processing a hard-to-understand example, or parsing difficult, jargon-laden, or overly terse text.

We used ***people***. In stories, examples, pictures, etc., because, well, because *you're* a person. And your brain pays more attention to *people* than it does to *things*.



Cut this out and stick it on your refrigerator.

Here's what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn't. Try new things.

1 Slow down. The more you understand, the less you have to memorize.

Don't just *read*. Stop and think. When the book asks you a question, don't just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2 Do the exercises. Write your own notes.

We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don't just *look* at the exercises. **Use a pencil.** There's plenty of evidence that physical activity *while* learning can increase the learning.

3 Read the "There are No Dumb Questions"

That means all of them. They're not optional sidebars, ***they're part of the core content!*** Don't skip them.

4 Make this the last thing you read before bed. Or at least the last challenging thing.

Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5 Talk about it. Out loud.

Speaking activates a different part of the brain. If you're trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You'll learn more quickly, and you might uncover ideas you hadn't known were there when you were reading about it.

6 Drink water. Lots of it.

Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

7 Listen to your brain.

Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it's time for a break. Once you go past a certain point, you won't learn faster by trying to shove more in, and you might even hurt the process.

8 Feel something.

Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9 Practice writing Rails applications!

There's only one way to truly master Rails programming: **program Rails applications**. And that's what you're going to do throughout this book. The best way to understand a subject is by **doing it**. Activity strengthens the neural pathways, so we're going to give you a **lot** of practice: every chapter has apps that we'll build. So don't just skip over them—a lot of learning happens when you build these apps yourself. And don't worry if you make mistakes. Your brain actually learns more quickly from mistakes than it does from successes. Finally, make sure you understand what's going on before moving on to the next part of the book. Each chapter builds on the chapters that come before it.

Read Me

This is a learning experience, not a reference book. We deliberately stripped out everything that might get in the way of learning whatever it is we're working on at that point in the book. And the first time through, you need to begin at the beginning because the book makes assumptions about what you've already seen and learned.

Before you begin this book you will need to get Ruby on Rails installed on your machine.

This is not a *how-to* book, so we don't have any chapters that give you instructions on how to install Ruby on Rails on your computer. It's better to get that kind of information from the web. You will need to install Ruby on Rails version 2.1 or above, as well as SQLite 3. You can find out more from
`http://www.rubyonrails.org/down`

This is not a reference book.

So don't expect to see lots and lots of pages explaining 15 different ways to do something. We want you to **understand** by **doing**, so right from the get-go, we'll give you just enough information to move your learning forward. By the end of the book, you will have a mental framework of how Rails works and what it can do. You will then be able to slot the reference material into your brain much more rapidly and meaningfully than you would have been able to before. Psychologists call this the ability to **chunk** information.

All of the code in this book is available on the Head First site.

We'll present all of the code you'll need as we go along. It's a **good idea** to program along with the book, and it's a **great idea** to play around with the code and make it do your own thing. But sometimes you may want a copy of the code used in each chapter, so we've made it available on the Head First Labs web site. Rails applications are quite self-contained, so there's no reason why you can't have the code that does what the *book says* it should do, alongside your own buffed and pimped out version. You can download the code from

`http://www.headfirstlabs.com/books/hfrails`

We don't fully explain every piece of code.

Rails can generate a lot of code for you, and we don't want you to get bogged down in line-by-line descriptions. We'll describe the important parts that you need to know, and then we'll move on. Don't worry—by the end of the book, all of the pieces should fall into place.

This is a Rails book, not a Ruby book.

Ruby is the language that the Rails framework is written in, and we'll teach you just enough Ruby as we go along. Don't worry—if you have some experience of another programming language like **C#** or **Java**, you'll do just fine. Rails is such a powerful system that you can get a very long way with just a little Ruby knowledge.

The activities are NOT optional.

The exercises and activities are not add-ons; they're part of the core content of the book. Some of them are to help with memory, some are for understanding, and some will help you apply what you've learned. Don't skip the exercises.

The redundancy is intentional and important.

One distinct difference in a Head First book is that we want you to really get it. And we want you to finish the book remembering what you've learned. Most reference books don't have retention and recall as a goal, but this book is about learning, so you'll see some of the same concepts come up more than once.

We don't show all the code all the time.

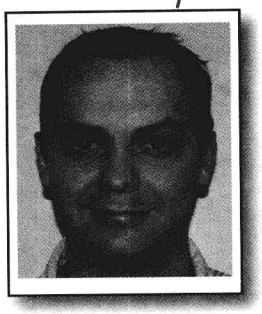
Our readers tell us that it's frustrating to wade through 10 slightly different versions of the same piece of code, so sometimes we will only show the parts of a script that have changed.

The chapters are skills-based not technology-based.

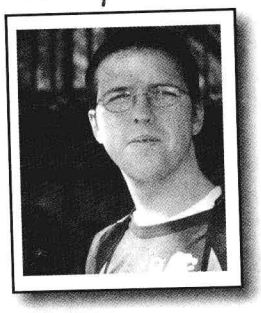
Each chapter will give you the skills to write more and more **advanced** and **valuable** applications. So we don't have chapters that just deal with talking to databases or designing a pretty interface. Instead, every chapter teaches you a little about the database, a little about the interface, and a little about several other parts of Rails. By the end of each one, you'll be able to say, "Cool—now I can build apps that can do **X**."

The technical review team

Andrew Bryan

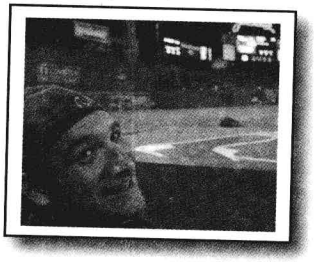


Jeremy Durham



Matt Harrington

Mike Isman



LuAnn Mazza

Eamon Walshe

Technical Reviewers:

Andrew Bryan is a software development and business consultant from Auckland, New Zealand. He is currently working for an online media and advertising company in Boston, where he lives with his lovely wife Angie.

Jeremy Durham has been building web applications using Ruby on Rails since early 2005, and has contributed to several Ruby libraries. He lives in Arlington, Massachusetts with his wife and two children.

Matt Harrington is a Northeastern University alumni and has been an avid programmer since age 9.

Mike Isman has been working with Ruby on Rails since he joined the eons.com team early in 2006, before Rails 1.0 was released. While working at Eons, Mike has also written smaller sites in Rails including the Life Expectancy Calculator at livingto100.com. He graduated in 2004 with a degree in Computer Science from the University of Rochester and has been busy doing web development ever since.

LuAnn Mazza is a Computer Analyst from Illinois.

Eamon Walshe is an Agile Coach with Exoftware and a former Distinguished Engineer with IONA Technologies. He is a fan of Rails because it allows developers to concentrate on what matters—delivering real business value, quickly.

Acknowledgments

My editors:

I owe a huge debt of gratitude to my editors, **Brett McLaughlin and Lou Barr**. They were always available for advice and support and whenever I came across a problem that seemed completely insoluble, they were not only able to identify exactly *what* was wrong, but *why* it was wrong and then come up with several ways of fixing it.

Brett McLaughlin

Lou Barr

I owe a very particular thank you to my wife, the author of *Head First Statistics*, **Dawn Griffiths**. This book would simply not have been completed on time had it not been for the immense amount of work she did on the final version.

This book is every bit as much hers as mine.

Dawn Griffiths

The O'Reilly team:

To **Caitrin McCullough** and **Karen Shaner**, who kept track of everything from contracts to web content.

To **Brittany Smith**, the book's Production Editor, for being a powerhouse of practical support.

To **Catherine Nolan**, for patiently guiding me through the first phase of the book.

To **Laurie Petrycki**, for her faith in the book and for allowing me to use her office in Cambridge.

And to **Kathy Sierra** and **Bert Bates,** the creators of the *Head First* Series, whose original vision has transformed the way technical books are written.

And not forgetting:

Brian Hanly, the CEO at *Exoftware*, and **Steve Harvey.** Their unstinting support and kindness made this book possible.

And finally the entire **technical review team** who had to perform an amazing amount of work in a very small amount of time.

I owe you all more than I can ever repay.

Safari® Books Online

When you see a Safari® icon on the cover of your favorite technology book that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at `http://safari.oreilly.com`.