Praise for Michael Hartl’s Books and Videos on Ruby on Rails™

“My former company (CD Baby) was one of the first to loudly switch to Ruby on Rails, and then even more loudly switch back to PHP (Google me to read about the drama). This book by Michael Hartl came so highly recommended that I had to try it, and the *Ruby on Rails™ Tutorial* is what I used to switch back to Rails again.”

—From the Foreword by Derek Sivers (sivers.org)
  Formerly: Founder, CD Baby
  Currently: Founder, Thoughts Ltd.

“Michael Hartl’s Rails Tutorial book is the #1 (and only, in my opinion) place to start when it comes to books about learning Rails. . . . It’s an amazing piece of work and, unusually, walks you through building a Rails app from start to finish with testing. If you want to read just one book and feel like a Rails master by the end of it, pick the *Ruby on Rails™ Tutorial.*”

—Peter Cooper
  Editor, Ruby Inside
“Grounded in the real world.”
—I Programmer (www.i-programmer.info), by Ian Elliot

“The book gives you the theory and practice, while the videos focus on showing you in person how it's done. Highly recommended combo.”
—Antonio Cangiano, Software Engineer, IBM

“The author is clearly an expert at the Ruby language and the Rails framework, but more than that, he is a working software engineer who introduces best practices throughout the text.”
—Greg Charles, Senior Software Developer, Fairway Technologies

“Overall, these video tutorials should be a great resource for anyone new to Rails.”
—Michael Morin, ruby.about.com

“Hands-down, I would recommend this book to anyone wanting to get into Ruby on Rails development.”
—Michael Crump, Microsoft MVP
RBUBY ON RAILS™ TUTORIAL

Second Edition
Visit informit.com/ruby for a complete list of available products.

The **Addison-Wesley Professional Ruby Series** provides readers with practical, people-oriented, and in-depth information about applying the Ruby platform to create dynamic technology solutions. The series is based on the premise that the need for expert reference books, written by experienced practitioners, will never be satisfied solely by blogs and the Internet.
Contents

Foreword to the First Edition by Derek Sivers xv
Foreword to the First Edition by Obie Fernandez xvii
Acknowledgments xix
About the Author xxi

Chapter 1  From Zero to Deploy  1

1.1  Introduction  3
  1.1.1  Comments for Various Readers  4
  1.1.2  “Scaling” Rails  7
  1.1.3  Conventions in This Book  7

1.2  Up and Running  9
  1.2.1  Development Environments  10
  1.2.2  Ruby, RubyGems, Rails, and Git  12
  1.2.3  The First Application  17
  1.2.4  Bundler  19
  1.2.5  rails server  23
  1.2.6  Model-view-controller (MVC)  25

1.3  Version Control with Git  27
  1.3.1  Installation and Setup  27
  1.3.2  Adding and Committing  30
  1.3.3  What Good Does Git Do You?  31
1.3.4 GitHub 32
1.3.5 Branch, Edit, Commit, Merge 34

1.4 Deploying 39
1.4.1 Heroku Setup 39
1.4.2 Heroku Deployment, Step One 40
1.4.3 Heroku Deployment, Step Two 40
1.4.4 Heroku Commands 41

1.5 Conclusion 43

Chapter 2 A Demo App 45

2.1 Planning the Application 45
2.1.1 Modeling Demo Users 47
2.1.2 Modeling Demo Microposts 48

2.2 The Users Resource 49
2.2.1 A User Tour 51
2.2.2 MVC in Action 56
2.2.3 Weaknesses of this Users Resource 62

2.3 The Microposts Resource 63
2.3.1 A Micropost Microtour 63
2.3.2 Putting the micro in Microposts 66
2.3.3 A User has_many Microposts 68
2.3.4 Inheritance Hierarchies 70
2.3.5 Deploying the Demo App 73

2.4 Conclusion 74

Chapter 3 Mostly Static Pages 77

3.1 Static Pages 82
3.1.1 Truly Static Pages 82
3.1.2 Static Pages with Rails 85

3.2 Our First Tests 93
3.2.1 Test-driven Development 93
3.2.2 Adding a Page 99

3.3 Slightly Dynamic Pages 103
3.3.1 Testing a Title Change 103
3.3.2 Passing Title Tests 106
3.3.3 Embedded Ruby 108
3.3.4 Eliminating Duplication with Layouts 111

3.4 Conclusion 114
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Exercises</td>
<td>114</td>
</tr>
<tr>
<td>3.6</td>
<td>Advanced Setup</td>
<td>117</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Eliminating <code>bundle exec</code></td>
<td>118</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Automated Tests with Guard</td>
<td>120</td>
</tr>
<tr>
<td>3.6.3</td>
<td>Speeding up Tests with Spork</td>
<td>123</td>
</tr>
<tr>
<td>3.6.4</td>
<td>Tests inside Sublime Text</td>
<td>127</td>
</tr>
<tr>
<td>4.1</td>
<td>Motivation</td>
<td>129</td>
</tr>
<tr>
<td>4.2</td>
<td>Strings and Methods</td>
<td>134</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Comments</td>
<td>134</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Strings</td>
<td>135</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Objects and Message Passing</td>
<td>138</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Method Definitions</td>
<td>141</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Back to the Title Helper</td>
<td>142</td>
</tr>
<tr>
<td>4.3</td>
<td>Other Data Structures</td>
<td>142</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Arrays and Ranges</td>
<td>142</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Blocks</td>
<td>146</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Hashes and Symbols</td>
<td>148</td>
</tr>
<tr>
<td>4.3.4</td>
<td>CSS revisited</td>
<td>152</td>
</tr>
<tr>
<td>4.4</td>
<td>Ruby Classes</td>
<td>153</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Constructors</td>
<td>153</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Class Inheritance</td>
<td>155</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Modifying Built-in Classes</td>
<td>158</td>
</tr>
<tr>
<td>4.4.4</td>
<td>A Controller Class</td>
<td>159</td>
</tr>
<tr>
<td>4.4.5</td>
<td>A User Class</td>
<td>161</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusion</td>
<td>164</td>
</tr>
<tr>
<td>4.6</td>
<td>Exercises</td>
<td>164</td>
</tr>
<tr>
<td>5.1</td>
<td>Adding Some Structure</td>
<td>167</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Site Navigation</td>
<td>169</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Bootstrap and Custom CSS</td>
<td>175</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Partials</td>
<td>181</td>
</tr>
<tr>
<td>5.2</td>
<td>Sass and the Asset Pipeline</td>
<td>187</td>
</tr>
<tr>
<td>5.2.1</td>
<td>The Asset Pipeline</td>
<td>187</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Syntactically Awesome Stylesheets</td>
<td>190</td>
</tr>
</tbody>
</table>
Chapter 6  Modeling Users  221

6.1  User Model  222
   6.1.1  Database Migrations  223
   6.1.2  The Model File  228
   6.1.3  Creating User Objects  230
   6.1.4  Finding User Objects  233
   6.1.5  Updating User Objects  235

6.2  User Validations  236
   6.2.1  Initial User Tests  236
   6.2.2  Validating Presence  239
   6.2.3  Length Validation  243
   6.2.4  Format Validation  245
   6.2.5  Uniqueness Validation  249

6.3  Adding a Secure Password  254
   6.3.1  An Encrypted Password  255
   6.3.2  Password and Confirmation  257
   6.3.3  User Authentication  260
   6.3.4  User Has Secure Password  263
   6.3.5  Creating a User  265

6.4  Conclusion  267

6.5  Exercises  268

Chapter 7  Sign Up  271

7.1  Showing Users  271
   7.1.1  Debug and Rails Environments  272
   7.1.2  A Users Resource  278
Chapter 7  User Authentication  279

7.1.3 Testing the User Show Page (with Factories)  282
7.1.4 A Gravatar Image and a Sidebar  286

7.2 Signup Form  292
7.2.1 Tests for User Signup  293
7.2.2 Using form_for  297
7.2.3 The Form HTML  301

7.3 Signup Failure  303
7.3.1 A Working Form  303
7.3.2 Signup Error Messages  308

7.4 Signup Success  312
7.4.1 The Finished Signup Form  313
7.4.2 The Flash  315
7.4.3 The First Signup  317
7.4.4 Deploying to Production with SSL  317

7.5 Conclusion  321
7.6 Exercises  321

Chapter 8  Sign In, Sign Out  325

8.1 Sessions and Signin Failure  325
8.1.1 Sessions Controller  326
8.1.2 Signin Tests  330
8.1.3 Signin Form  333
8.1.4 Reviewing Form Submission  336
8.1.5 Rendering with a Flash Message  339

8.2 Signin Success  343
8.2.1 Remember Me  343
8.2.2 A Working sign_in Method  349
8.2.3 Current User  351
8.2.4 Changing the Layout Links  355
8.2.5 Signin upon Signup  359
8.2.6 Signing Out  361

8.3 Introduction to Cucumber (Optional)  363
8.3.1 Installation and Setup  364
8.3.2 Features and Steps  365
8.3.3 Counterpoint: RSpec Custom Matchers  368

8.4 Conclusion  371
8.5 Exercises  372
Chapter 9  Updating, Showing, and Deleting Users  373

9.1  Updating Users  373
  9.1.1  Edit Form  374
  9.1.2  Unsuccessful Edits  380
  9.1.3  Successful Edits  382

9.2  Authorization  385
  9.2.1  Requiring Signed-in Users  386
  9.2.2  Requiring the Right User  390
  9.2.3  Friendly Forwarding  392

9.3  Showing All Users  396
  9.3.1  User Index  396
  9.3.2  Sample Users  403
  9.3.3  Pagination  404
  9.3.4  Partial Refactoring  410

9.4  Deleting Users  413
  9.4.1  Administrative Users  413
  9.4.2  The destroy Action  417

9.5  Conclusion  422

9.6  Exercises  424

Chapter 10  User Microposts  429

10.1  A Micropost Model  429
  10.1.1  The Basic Model  430
  10.1.2  Accessible Attributes and the First Validation  432
  10.1.3  User/Micropost Associations  433
  10.1.4  Micropost Refinements  439
  10.1.5  Content Validations  443

10.2  Showing Microposts  445
  10.2.1  Augmenting the User Show Page  446
  10.2.2  Sample Microposts  450

10.3  Manipulating Microposts  454
  10.3.1  Access Control  456
  10.3.2  Creating Microposts  459
  10.3.3  A Proto-feed  467
  10.3.4  Destroying Microposts  475

10.4  Conclusion  479

10.5  Exercises  480
# Chapter 11  Following Users  

11.1 The Relationship Model  
11.1.1 A Problem with the Data Model (and a Solution)  
11.1.2 User/Relationship Associations  
11.1.3 Validations  
11.1.4 Followed users  
11.1.5 Followers  

11.2 A Web Interface for Following Users  
11.2.1 Sample Following Data  
11.2.2 Stats and a Follow Form  
11.2.3 Following and Followers Pages  
11.2.4 A Working Follow Button the Standard Way  
11.2.5 A Working Follow Button with Ajax  

11.3 The Status Feed  
11.3.1 Motivation and Strategy  
11.3.2 A First Feed Implementation  
11.3.3 Subselects  
11.3.4 The New Status Feed  

11.4 Conclusion  
11.4.1 Extensions to the Sample Application  
11.4.2 Guide to Further Resources  

11.5 Exercises  

Index
This page intentionally left blank
My former company (CD Baby) was one of the first to loudly switch to Ruby on Rails, and then even more loudly switch back to PHP (Google me to read about the drama). This book by Michael Hartl came so highly recommended that I had to try it, and Ruby on Rails™ 3 Tutorial is what I used to switch back to Rails again.

Though I’ve worked my way through many Rails books, this is the one that finally made me get it. Everything is done very much “the Rails way”—a way that felt very unnatural to me before, but now after doing this book finally feels natural. This is also the only Rails book that does test-driven development the entire time, an approach highly recommended by the experts but which has never been so clearly demonstrated before. Finally, by including Git, GitHub, and Heroku in the demo examples, the author really gives you a feel for what it’s like to do a real-world project. The tutorial’s code examples are not in isolation.

The linear narrative is such a great format. Personally, I powered through the Rails Tutorial in three long days, doing all the examples and challenges at the end of each chapter. Do it from start to finish, without jumping around, and you’ll get the ultimate benefit.

Enjoy!

—Derek Sivers (sivers.org)
Formerly: Founder, CD Baby
Currently: Founder, Thoughts Ltd.
“If you want to learn web development with Ruby on Rails, how should I start?” For years Michael Hartl has provided the answer as author of the RailsSpace tutorial in our series and now the new Ruby on Rails™ 3 Tutorial that you hold in your hands (or PDF reader, I guess).

I’m so proud of having Michael on the series roster. He is living, breathing proof that us Rails folks are some of the luckiest in the wide world of technology. Before getting into Ruby, Michael taught theoretical and computational physics at Caltech for six years, where he received the Lifetime Achievement Award for Excellence in Teaching in 2000. He is a Harvard graduate, has a Ph.D. in Physics from Caltech, and is an alumnus of Paul Graham’s esteemed Y Combinator program for entrepreneurs. And what does Michael apply his impressive experience and teaching prowess to? Teaching new software developers all around the world how to use Ruby on Rails effectively! Lucky we are indeed!

The availability of this tutorial actually comes at a critical time for Rails adoption. We’re five years into the history of Rails and today’s version of the platform has unprecedented power and flexibility. Experienced Rails folks can leverage that power effectively, but we’re hearing growing cries of frustration from newcomers. The amount of information out there about Rails is fantastic if you know what you’re doing already. However, if you’re new, the scope and mass of information about Rails can be mind-boggling.

Luckily, Michael takes the same approach as his first book in the series, building a sample application from scratch, and writes in a style that’s meant to be read from start to finish. Along the way, he explains all the little details that are likely to trip up
beginners. Impressively, he goes beyond just a straightforward explanation of what Rails does and ventures into prescriptive advice about good software development practices, such as test-driven development. Neither does Michael constrain himself to a box delineated by the extents of the Rails framework—he goes ahead and teaches the reader to use tools essential to existence in the Rails community, such as Git and GitHub. In a friendly style, he even provides copious contextual footnotes of benefit to new programmers, such as the pronunciation of SQL and pointers to the origins of *lorem ipsum*. Tying all the content together in a way that remains concise and usable is truly a tour de force of dedication!

I tell you with all my heart that this book is one of the most significant titles in my Professional Ruby Series, because it facilitates the continued growth of the Rails ecosystem. By helping newcomers become productive members of the community quickly, he ensures that Ruby on Rails continues its powerful and disruptive charge into the mainstream. The *Rails Tutorial* is potent fuel for the fire that is powering growth and riches for so many of us, and for that we are forever grateful.

—Obie Fernandez, Series Editor
The *Ruby on Rails™ Tutorial* owes a lot to my previous Rails book, *RailsSpace*, and hence to my coauthor Aurelius Prochazka. I’d like to thank Aure both for the work he did on that book and for his support of this one. I’d also like to thank Debra Williams Cauley, my editor on both *RailsSpace* and the *Ruby on Rails™ Tutorial*; as long as she keeps taking me to baseball games, I’ll keep writing books for her.

I’d like to acknowledge a long list of Rubyists who have taught and inspired me over the years: David Heinemeier Hansson, Yehuda Katz, Carl Lerche, Jeremy Kemper, Xavier Noria, Ryan Bates, Geoffrey Grosenbach, Peter Cooper, Matt Aimonetti, Gregg Pollack, Wayne E. Seguin, Amy Hoy, Dave Chelimsky, Pat Maddox, Tom Preston-Werner, Chris Wanstrath, Chad Fowler, Josh Susser, Obie Fernandez, Ian McFarland, Steven Bristol, Pratik Naik, Sarah Mei, Sarah Allen, Wolfram Arnold, Alex Chaffee, Giles Bowkett, Evan Dorn, Long Nguyen, James Lindenbaum, Adam Wiggins, Tikhon Bernstam, Ron Evans, Wyatt Greene, Miles Forrest, the good people at Pivotal Labs, the Heroku gang, the thoughtbot guys, and the GitHub crew. Thanks to Jen Lindner, Patty Donovan (Laserwords), and Julie Nahil and Michael Thurston from Pearson for their help with the book. Finally, many, many readers—far too many to list—have contributed a huge number of bug reports and suggestions during the writing of this book, and I gratefully acknowledge their help in making it as good as it can be.
This page intentionally left blank
About the Author

**Michael Hartl** is the author of the *Ruby on Rails™ Tutorial*, the leading introduction to web development with Ruby on Rails. His prior experience includes writing and developing *RailsSpace*, an extremely obsolete Rails tutorial book, and developing Insoshi, a once-popular and now-obsole O social networking platform in Ruby on Rails. In 2011, Michael received a Ruby Hero Award for his contributions to the Ruby community. He is a graduate of Harvard College, has a Ph.D. in physics from Caltech, and is an alumnus of the Y Combinator entrepreneur program.
This page intentionally left blank
Welcome to *Ruby on Rails™ Tutorial*. The goal of this book is to be the best answer to the question, “If I want to learn web development with Ruby on Rails, where should I start?” By the time you finish the *Rails Tutorial*, you will have all the skills you need to develop and deploy your own custom web applications with Rails. You will also be ready to benefit from the many more advanced books, blogs, and screencasts that are part of the thriving Rails educational ecosystem. Finally, since the *Rails Tutorial* uses Rails 3, the knowledge you gain here represents the state of the art in web development. (The most up-to-date version of the *Rails Tutorial* can be found on the book’s website at http://railstutorial.org; if you are reading this book offline, be sure to check the online version of the Rails Tutorial book at http://railstutorial.org/book for the latest updates.)

Note that the goal of this book is *not* merely to teach Rails, but rather to teach *web development with Rails*, which means acquiring (or expanding) the skills needed to develop software for the World Wide Web. In addition to Ruby on Rails, this skillset includes HTML and CSS, databases, version control, testing, and deployment. To accomplish this goal, *Rails Tutorial* takes an integrated approach: You will learn Rails by example by building a substantial sample application from scratch. As Derek Sivers notes in the foreword, this book is structured as a linear narrative, designed to be read from start to finish. If you are used to skipping around in technical books, taking this linear approach might require some adjustment, but I suggest giving it a try. You can think of the *Rails Tutorial* as a video game where you are the main character and where you level up as a Rails developer in each chapter. (The exercises are the minibosses.)
In this first chapter, we’ll get started with Ruby on Rails by installing all the necessary software and by setting up our development environment (Section 1.2). We’ll then create our first Rails application, called (appropriately enough) first_app. The Rails Tutorial emphasizes good software development practices, so immediately after creating our fresh new Rails project we’ll put it under version control with Git (Section 1.3). And, believe it or not, in this chapter we’ll even put our first app on the wider web by deploying it to production (Section 1.4).

In Chapter 2, we’ll make a second project, whose purpose is to demonstrate the basic workings of a Rails application. To get up and running quickly, we’ll build this demo app (called demo_app) using scaffolding (Box 1.1) to generate code; since this code is both ugly and complex, Chapter 2 will focus on interacting with the demo app through its URIs (sometimes called URLs)¹ using a web browser.

The rest of the tutorial focuses on developing a single large sample application (called sample_app), writing all the code from scratch. We’ll develop the sample app using test-driven development (TDD), getting started in Chapter 3 by creating static pages and then adding a little dynamic content. We’ll take a quick detour in Chapter 4 to learn a little about the Ruby language underlying Rails. Then, in Chapter 5 through Chapter 9, we’ll complete the foundation for the sample application by making a site layout, a user data model, and a full registration and authentication system. Finally, in Chapter 10 and Chapter 11 we’ll add microblogging and social features to make a working example site.

The final sample application will bear more than a passing resemblance to a certain popular social microblogging site—a site that, coincidentally, was also originally written in Rails. Although of necessity our efforts will focus on this specific sample application, the emphasis throughout the Rails Tutorial will be on general principles, so that you will have a solid foundation no matter what kinds of web applications you want to build.

**Box 1.1 Scaffolding: Quicker, Easier, More Seductive**

From the beginning, Rails has benefited from a palpable sense of excitement, starting with the famous 15-minute weblog video by Rails creator David Heinemeier Hansson. That video and its successors are a great way to get a taste of Rails’ power,

---

¹. URI stands for Uniform Resource Identifier, while the slightly less general URL stands for Uniform Resource Locator. In practice, the URI is usually equivalent to “the thing you see in the address bar of your browser.”
and I recommend watching them. But be warned: They accomplish their amazing
15-minute feat using a feature called scaffolding, which relies heavily on generated
code, magically created by the Rails generate command.

When writing a Ruby on Rails tutorial, it is tempting to rely on the scaffolding
approach—it’s quicker, easier, more seductive. But the complexity and sheer amount
of code in the scaffolding can be utterly overwhelming to a beginning Rails developer;
you may be able to use it, but you probably won’t understand it. Following the
scaffolding approach risks turning you into a virtuoso script generator with little (and
brittle) actual knowledge of Rails.

In the Rails Tutorial, we’ll take the (nearly) polar opposite approach: Although
Chapter 2 will develop a small demo app using scaffolding, the core of the Rails
Tutorial is the sample app, which we’ll start writing in Chapter 3. At each stage of
developing the sample application, we will write small, bite-sized pieces of code—sim-
ple enough to understand, yet novel enough to be challenging. The cumulative effect
will be a deeper, more flexible knowledge of Rails, giving you a good background
for writing nearly any type of web application.

1.1 Introduction

Since its debut in 2004, Ruby on Rails has rapidly become one of the most powerful
and popular frameworks for building dynamic web applications. Everyone from scrappy
startups to huge companies have used Rails: 37signals, GitHub, Shopify, Scribd, Twitter,
LivingSocial, Groupon, Hulu, the Yellow Pages—the list of sites using Rails goes on
and on. There are also many web development shops that specialize in Rails, such
as ENTP, thoughtbot, Pivotal Labs, and Hashrocket, plus innumerable independent
consultants, trainers, and contractors.

What makes Rails so great? First of all, Ruby on Rails is 100 percent open-source,
available under the permissive MIT License, and as a result it also costs nothing to
download or use. Rails also owes much of its success to its elegant and compact design;
by exploiting the malleability of the underlying Ruby language, Rails effectively creates
a domain-specific language for writing web applications. As a result, many common
web programming tasks—such as generating HTML, making data models, and routing
URIs—are easy with Rails, and the resulting application code is concise and readable.

Rails also adapts rapidly to new developments in web technology and framework
design. For example, Rails was one of the first frameworks to fully digest and implement
the REST architectural style for structuring web applications (which we’ll be learning
about throughout this tutorial). And when other frameworks develop successful new techniques, Rails creator David Heinemeier Hansson and the Rails core team don’t hesitate to incorporate their ideas. Perhaps the most dramatic example is the merger of Rails and Merb, a rival Ruby web framework, so that Rails now benefits from Merb’s modular design, stable API, and improved performance.

Finally, Rails benefits from an unusually enthusiastic and diverse community. The results include hundreds of open-source contributors, well-attended conferences, a huge number of plugins and gems (self-contained solutions to specific problems such as pagination and image upload), a rich variety of informative blogs, and a cornucopia of discussion forums and IRC channels. The large number of Rails programmers also makes it easier to handle the inevitable application errors: The “Google the error message” algorithm nearly always produces a relevant blog post or discussion-forum thread.

1.1.1 Comments for Various Readers

The *Rails Tutorial* contains integrated tutorials not only for Rails, but also for the underlying Ruby language, the RSpec testing framework, HTML, CSS, a small amount of JavaScript, and even a little SQL. This means that, no matter where you currently are in your knowledge of web development, by the time you finish this tutorial you will be ready for more advanced Rails resources, as well as for the more systematic treatments of the other subjects mentioned. It also means that there’s a lot of material to cover; if you don’t already have experience programming computers, you might find it overwhelming. The comments below contain some suggestions for approaching the *Rails Tutorial* depending on your background.

**All readers:** One common question when learning Rails is whether to learn Ruby first. The answer depends on your personal learning style and how much programming experience you already have. If you prefer to learn everything systematically from the ground up, or if you have never programmed before, then learning Ruby first might work well for you, and in this case I recommend *Beginning Ruby* by Peter Cooper. On the other hand, many beginning Rails developers are excited about making web applications, and would rather not slog through a 500-page book on pure Ruby before ever writing a single web page. In this case, I recommend following the short interactive
tutorial at TryRuby,\textsuperscript{2} and then optimally do the free tutorial at Rails for Zombies\textsuperscript{3} to get a taste of what Rails can do.

Another common question is whether to use tests from the start. As noted in the introduction, the Rails Tutorial uses test-driven development (also called test-first development), which in my view is the best way to develop Rails applications, but it does introduce a substantial amount of overhead and complexity. If you find yourself getting bogged down by the tests, I suggest either skipping them on a first reading or (even better) using them as a tool to verify your code’s correctness without worrying about how they work. This latter strategy involves creating the necessary test files (called specs) and filling them with the test code exactly as it appears in the book. You can then run the test suite (as described in Chapter 5) to watch it fail, then write the application code as described in the tutorial, and finally re-run the test suite to watch it pass.

**Inexperienced programmers:** The Rails Tutorial is not aimed principally at beginning programmers, and web applications, even relatively simple ones, are by their nature fairly complex. If you are completely new to web programming and find the Rails Tutorial too difficult, I suggest learning the basics of HTML and CSS and then giving the Rails Tutorial another go. (Unfortunately, I don’t have a personal recommendation here, but Head First HTML looks promising, and one reader recommends CSS: The Missing Manual by David Sawyer McFarland.) You might also consider reading the first few chapters of Beginning Ruby by Peter Cooper, which starts with sample applications much smaller than a full-blown web app. That said, a surprising number of beginners have used this tutorial to learn web development, so I suggest giving it a try, and I especially recommend the Rails Tutorial screencast series\textsuperscript{4} to give you an “over-the-shoulder” look at Rails software development.

**Experienced programmers new to web development:** Your previous experience means you probably already understand ideas like classes, methods, data structures, and others, which is a big advantage. Be warned that if your background is in C/C++ or Java, you

\textsuperscript{2} http://tryruby.org
\textsuperscript{3} http://railsforzombies.org
\textsuperscript{4} http://railstutorial.org/screencasts
may find Ruby a bit of an odd duck, and it might take time to get used to it; just stick with it and eventually you’ll be fine. (Ruby even lets you put semicolons at the ends of lines if you miss them too much.) The Rails Tutorial covers all the web-specific ideas you’ll need, so don’t worry if you don’t currently know a **PUT** from a **POST**.

**Experienced web developers new to Rails:** You have a great head start, especially if you have used a dynamic language such as PHP or (even better) Python. The basics of what we cover will likely be familiar, but test-driven development may be new to you, as may be the structured REST style favored by Rails. Ruby has its own idiosyncrasies, so those will likely be new, too.

**Experienced Ruby programmers:** The set of Ruby programmers who don’t know Rails is a small one nowadays, but if you are a member of this elite group you can fly through this book and then move on to The Rails 3 Way by Obie Fernandez.

**Inexperienced Rails programmers:** You’ve perhaps read some other tutorials and made a few small Rails apps yourself. Based on reader feedback, I’m confident that you can still get a lot out of this book. Among other things, the techniques here may be more up-to-date than the ones you picked up when you originally learned Rails.

**Experienced Rails programmers:** This book is unnecessary for you, but many experienced Rails developers have expressed surprise at how much they learned from this book, and you might enjoy seeing Rails from a different perspective.

After finishing the Ruby on Rails Tutorial, I recommend that experienced programmers read The Well-Grounded Rubyist by David A. Black, which is an excellent in-depth discussion of Ruby from the ground up, or The Ruby Way by Hal Fulton, which is also fairly advanced but takes a more topical approach. Then move on to The Rails 3 Way to deepen your Rails expertise.

At the end of this process, no matter where you started, you should be ready for the many more intermediate-to-advanced Rails resources out there. Here are some I particularly recommend:

- RailsCasts by Ryan Bates: Excellent (mostly) free Rails screencasts
- PeepCode: Excellent commercial screencasts
• Code School: Interactive programming courses
• Rails Guides: Good topical and up-to-date Rails references
• RailsCasts by Ryan Bates: Did I already mention RailsCasts? Seriously: RailsCasts.

1.1.2 "Scaling" Rails

Before moving on with the rest of the introduction, I’d like to take a moment to address the one issue that dogged the Rails framework the most in its early days: the supposed inability of Rails to “scale”—i.e., to handle large amounts of traffic. Part of this issue relied on a misconception; you scale a site, not a framework, and Rails, as awesome as it is, is only a framework. So the real question should have been, “Can a site built with Rails scale?” In any case, the question has now been definitively answered in the affirmative: Some of the most heavily trafficked sites in the world use Rails. Actually doing the scaling is beyond the scope of just Rails, but rest assured that if your application ever needs to handle the load of Hulu or the Yellow Pages, Rails won’t stop you from taking over the world.

1.1.3 Conventions in This Book

The conventions in this book are mostly self-explanatory. In this section, I’ll mention some that may not be.

Both the HTML and PDF editions of this book are full of links, both to internal sections (such as Section 1.2) and to external sites (such as the main Ruby on Rails download page).

Many examples in this book use command-line commands. For simplicity, all command line examples use a Unix-style command line prompt (a dollar sign), as follows:

```
$ echo "hello, world"
hello, world
```

---

5. When reading the Rails Tutorial, you may find it convenient to follow an internal section link to look at the reference and then immediately go back to where you were before. This is easy when reading the book as a web page, since you can just use the Back button of your browser, but both Adobe Reader and OS X’s Preview allow you to do this with the PDF as well. In Reader, you can right-click on the document and select “Previous View” to go back. In Preview, use the Go menu: Go > Back.
Windows users should understand that their systems will use the analogous angle prompt `>:

C:\Sites> echo "hello, world"
hello, world

On Unix systems, some commands should be executed with `sudo`, which stands for “substitute user do.” By default, a command executed with `sudo` is run as an administrative user, which has access to files and directories that normal users can’t touch, such as in this example from Section 1.2.2:

```
$ sudo ruby setup.rb
```

Most Unix/Linux/OS X systems require `sudo` by default, unless you are using Ruby Version Manager as suggested in Section 1.2.2; in this case, you would type this instead:

```
$ ruby setup.rb
```

Rails comes with lots of commands that can be run at the command line. For example, in Section 1.2.5 we’ll run a local development web server as follows:

```
$ rails server
```

As with the command-line prompt, the Rails Tutorial uses the Unix convention for directory separators (i.e., a forward slash `/`). My Rails Tutorial sample application, for instance, lives in

```
/Users/mhartl/rails_projects/sample_app
```

On Windows, the analogous directory would be

```
C:\Sites\sample_app
```

The root directory for any given app is known as the Rails root, but this terminology is confusing and many people mistakenly believe that the “Rails root” is the root directory for Rails itself. For clarity, the Rails Tutorial will refer to the Rails root as
the *application root*, and henceforth all directories will be relative to this directory. For example, the `config` directory of my sample application is

```
/Users/mhartl/rails_projects/sample_app/config
```

The application root directory here is everything before `config`, that is,

```
/Users/mhartl/rails_projects/sample_app
```

For brevity, when referring to the file

```
/Users/mhartl/rails_projects/sample_app/config/routes.rb
```

I’ll omit the application root and simply write `config/routes.rb`.

The *Rails Tutorial* often shows output from various programs (shell commands, version control status, Ruby programs, etc.). Because of the innumerable small differences between different computer systems, the output you see may not always agree exactly with what is shown in the text, but this is not cause for concern.

Some commands may produce errors depending on your system; rather than attempt the Sisyphean task of documenting all such errors in this tutorial, I will delegate to the “Google the error message” algorithm, which among other things is good practice for real-life software development. If you run into any problems while following the tutorial, I suggest consulting the resources listed on the Rails Tutorial help page.6

### 1.2 Up and Running

I think of Chapter 1 as the “weeding out phase” in law school—if you can get your dev environment set up, the rest is easy to get through.

—Bob Cavezza, *Rails Tutorial* reader

It’s time now to get going with a Ruby on Rails development environment and our first application. There is quite a bit of overhead here, especially if you don’t have

---

extensive programming experience, so don’t get discouraged if it takes a while to get started. It’s not just you; every developer goes through it (often more than once), but rest assured that the effort will be richly rewarded.

1.2.1 Development Environments

Considering various idiosyncratic customizations, there are probably as many development environments as there are Rails programmers, but there are at least two broad types: text editor/command line environments, and integrated development environments (IDEs). Let’s consider the latter first.

IDEs

There is no shortage of Rails IDEs, including RadRails, RubyMine, and 3rd Rail. I’ve heard especially good things about RubyMine, and one reader (David Loeffler) has assembled notes on how to use RubyMine with this tutorial.7 If you’re comfortable using an IDE, I suggest taking a look at the options mentioned to see what fits with the way you work.

Text Editors and Command Lines

Instead of using an IDE, I prefer to use a text editor to edit text, and a command line to issue commands (Figure 1.1). Which combination you use depends on your tastes and your platform.

- **Text editor:** I recommend Sublime Text 2, an outstanding cross-platform text editor that is in beta as of this writing but has already proven to be exceptionally powerful. Sublime Text is heavily influenced by TextMate, and in fact is compatible with most TextMate customizations, such as snippets and color schemes. (TextMate, which is available only on OS X, is still a good choice if you use a Mac.) A second excellent choice is Vim,8 versions of which are available for all major platforms. Sublime Text is a commercial product, whereas Vim is free and open-source; both are industrial-strength editors, but Sublime Text is much more accessible to beginners.

7. https://github.com/perfectionist/sample_project/wiki
8. The vi editor is one of the most ancient yet powerful weapons in the Unix arsenal, and Vim is “vi improved.”
1.2 Up and Running

**Figure 1.1** A text editor/command line development environment (TextMate/iTerm).

- **Terminal:** On OS X, I recommend either use iTerm or the native Terminal app. On Linux, the default terminal is fine. On Windows, many users prefer to develop Rails applications in a virtual machine running Linux, in which case your command-line options reduce to the previous case. If developing within Windows itself, I recommend using the command prompt that comes with Rails Installer (Section 1.2.2).

If you decide to use Sublime Text, you might want to follow the setup instructions for Rails Tutorial Sublime Text.9 *Note:* Such configuration settings are fiddly and error-prone, so this step should only be attempted by advanced users.

**Browsers**

Although there are many web browsers to choose from, the vast majority of Rails programmers use Firefox, Safari, or Chrome when developing. The screenshots in Rails

---

Chapter 1: From Zero to Deploy

Tutorial will generally be of a Firefox browser. If you use Firefox, I suggest using the Firebug add-on, which lets you perform all sorts of magic, such as dynamically inspecting (and even editing) the HTML structure and CSS rules on any page. For those not using Firefox, both Safari and Chrome have a built-in “Inspect element” feature available by right-clicking on any part of the page.

A Note about Tools
In the process of getting your development environment up and running, you may find that you spend a lot of time getting everything just right. The learning process for editors and IDEs is particularly long; you can spend weeks on Sublime Text or Vim tutorials alone. If you’re new to this game, I want to assure you that spending time learning tools is normal. Everyone goes through it. Sometimes it is frustrating, and it’s easy to get impatient when you have an awesome web app in your head and you just want to learn Rails already, but have to spend a week learning some weird ancient Unix editor just to get started. But a craftsman has to know his tools, and in the end the reward is worth the effort.

1.2.2 Ruby, RubyGems, Rails, and Git

Practically all the software in the world is either broken or very difficult to use. So users dread software. They’ve been trained that whenever they try to install something, or even fill out a form online, it’s not going to work. I dread installing stuff, and I have a Ph.D. in computer science.

—Paul Graham, Founders at Work

Now it’s time to install Ruby and Rails. I’ve done my best to cover as many bases as possible, but systems vary, and many things can go wrong during these steps. Be sure to Google the error message or consult the Rails Tutorial help page if you run into trouble.

Unless otherwise noted, you should use the exact versions of all software used in the tutorial, including Rails itself, if you want the same results. Sometimes minor version differences will yield identical results, but you shouldn’t count on this, especially with respect to Rails versions. The main exception is Ruby itself: 1.9.2 and 1.9.3 are virtually identical for the purposes of this tutorial, so feel free to use either one.
Rails Installer (Windows)
Installing Rails on Windows used to be a real pain, but thanks to the efforts of the good people at Engine Yard—especially Dr. Nic Williams and Wayne E. Seguin—installing Rails and related software on Windows is now easy. If you are using Windows, go to Rails Installer and download the Rails Installer executable and view the excellent installation video. Double-click the executable and follow the instructions to install Git (so you can skip Section 1.2.2), Ruby (skip Section 1.2.2), RubyGems (skip Section 1.2.2), and Rails itself (skip Section 1.2.2). Once the installation has finished, you can skip right to the creation of the first application in Section 1.2.3.

Bear in mind that the Rails Installer might use a slightly different version of Rails from the one installed in Section 1.2.2, which might cause incompatibilities. To fix this, I am currently working with Nic and Wayne to create a list of Rails Installers ordered by Rails version number.

Install Git
Much of the Rails ecosystem depends in one way or another on a version control system called Git (covered in more detail in Section 1.3). Because its use is ubiquitous, you should install Git even at this early stage; I suggest following the installation instructions for your platform at the Installing Git section of Pro Git.

Install Ruby
The next step is to install Ruby. It’s possible that your system already has it; try running

```
$ ruby -v
ruby 1.9.3
```

to see the version number. Rails 3 requires Ruby 1.8.7 or later and works best with Ruby 1.9.x. This tutorial assumes that most readers are using Ruby 1.9.2 or 1.9.3, but Ruby 1.8.7 should work as well (although there is one syntax difference, covered in Chapter 4, and assorted minor differences in output).

As part of installing Ruby, if you are using OS X or Linux, I strongly recommend using Ruby Version Manager (RVM), which allows you to install and manage multiple versions of Ruby on the same machine. (The Pik project accomplishes a similar feat on Windows.) This is particularly important if you want to run different
versions of Ruby or Rails on the same machine. If you run into any problems with RVM, you can often find its creator, Wayne E. Seguin, on the RVM IRC channel (#rvm on freenode.net). If you are running Linux, I particularly recommend the installation tutorial for Linux Ubuntu and Linux Mint by Mircea Goia.

After installing RVM, you can install Ruby as follows:

```
rvm get head && rvm reload
rvm install 1.9.3
```

<wait a while>

Here the first command updates and reloads RVM itself, which is a good practice since RVM gets updated frequently. The second installs the 1.9.3 version of Ruby; depending on your system, it might take a while to download and compile, so don’t worry if it seems to be taking forever.

Some Linux users report having to include the path to a library called OpenSSL:

```
rvm install 1.9.3 --with-openssl-dir=$HOME/.rvm.usr
```

On some older OS X systems, you might have to include the path to the readline library:

```
rvm install 1.9.3 --with-readline-dir=/opt/local
```

(Like I said, lots of things can go wrong. The only solution is web searches and determination.)

After installing Ruby, you should configure your system for the other software needed to run Rails applications. This typically involves installing gems, which are self-contained packages of Ruby code. Since gems with different version numbers sometimes conflict, it is often convenient to create separate gemsets, which are self-contained bundles of gems. For the purposes of this tutorial, I suggest creating a gemset called `rails3tutorial2ndEd`:

```
rvm use 1.9.3@rails3tutorial2ndEd --create --default
```

10. If you haven’t used IRC before, I suggest you start by searching the web for “irc client <your platform>.” Two good native clients for OS X are Colloquy and LimeChat. And of course there’s always the web interface at http://webchat.freenode.net/?channels=rvm.

11. You might have to install the Subversion version control system to get this to work.
This command creates (\texttt{--create}) the gemset \texttt{rails3tutorial2ndEd} associated with Ruby 1.9.3 while arranging to start using it immediately (\texttt{use}) and setting it as the default (\texttt{--default}) gemset, so that any time we open a new terminal window the \texttt{1.9.3\@rails3tutorial2ndEd} Ruby/gemset combination is automatically selected. RVM supports a large variety of commands for manipulating gemsets; see the documentation at http://rvm.beginrescueend.com/gemsets. If you ever get stuck with RVM, running commands like these should help you get your bearings:

\begin{verbatim}
$ rvm --help
$ rvm gemset --help
\end{verbatim}

\textbf{Install RubyGems}

RubyGems is a package manager for Ruby projects, and there are many useful libraries (including Rails) available as Ruby packages, or \textit{gems}. Installing RubyGems should be easy once you install Ruby. In fact, if you have installed RVM, you already have RubyGems, since RVM includes it automatically:

\begin{verbatim}
$ which gem
/Users/mhartl/.rvm/rubies/ruby-1.9.3-p0/bin/gem
\end{verbatim}

If you don’t already have it, you should download RubyGems, extract it, and then go to the \texttt{rubygems} directory and run the setup program:

\begin{verbatim}
$ ruby setup.rb
\end{verbatim}

(If you get a permissions error here, recall from Section 1.1.3 that you may have to use \texttt{sudo}.)

If you already have RubyGems installed, you should make sure your system uses the version used in this tutorial:

\begin{verbatim}
$ gem update --system 1.8.24
\end{verbatim}

Freezing your system to this particular version will help prevent conflicts as RubyGems changes in the future.

When installing gems, by default RubyGems generates two different kinds of documentation (called \texttt{ri} and \texttt{rdoc}), but many Ruby and Rails developers find that the time to build them isn’t worth the benefit. (Many programmers rely on online documentation instead of the native \texttt{ri} and \texttt{rdoc} documents.) To prevent the automatic
generation of the documentation, I recommend making a gem configuration file called .gemrc in your home directory as in Listing 1.1 with the line in Listing 1.2. (The tilde “~” means “home directory,” while the dot . in .gemrc makes the file hidden, which is a common convention for configuration files.)

**Listing 1.1** Creating a gem configuration file.

```
$ subl ~/.gemrc
```

Here subl is the command-line command to launch Sublime Text on OS X, which you can set up using the Sublime Text 2 documentation for the OS X command line. If you’re on a different platform, or if you’re using a different editor, you should replace this command as necessary (i.e., by double-clicking the application icon or by using an alternate command such as mate, vim, gvim, or mvim). For brevity, throughout the rest of this tutorial I’ll use subl as a shorthand for “open with your favorite text editor.”

**Listing 1.2** Suppressing the ri and rdoc documentation in .gemrc.

```
install: --no-rdoc --no-ri
update: --no-rdoc --no-ri
```

**Install Rails**

Once you’ve installed RubyGems, installing Rails should be easy. This tutorial standardizes on Rails 3.2, which we can install as follows:

```
$ gem install rails -v 3.2.13
```

To check your Rails installation, run the following command to print out the version number:

```
$ rails -v
Rails 3.2.13
```

*Note:* If you installed Rails using the Rails Installer in Section 1.2.2, there might be slight version differences. As of this writing, those differences are not relevant, but in the future, as the current Rails version diverges from the one used in this tutorial, these differences may become significant. I am currently working with Engine Yard to create links to specific versions of the Rails Installer.
1.2 Up and Running

If you’re running Linux, you might have to install a couple of other packages at this point:

```
$ sudo apt-get install libxslt-dev libxml2-dev libsqlite3-dev # Linux only
```

### 1.2.3 The First Application

Virtually all Rails applications start the same way, with the `rails` command. This handy program creates a skeleton Rails application in a directory of your choice. To get started, make a directory for your Rails projects and then run the `rails` command to make the first application (Listing 1.3):

**Listing 1.3** Running `rails` to generate a new application.

```
$ mkdir rails_projects
$ cd rails_projects
$ rails new first_app

create README.rdoc
create Rakefile
create config.ru
create .gitignore
create Gemfile
create app
create app/assets/images/rails.png
create app/assets/javascripts/application.js
create app/assets/stylesheets/application.css
create app/controllers/application_controller.rb
create app/helpers/application_helper.rb
create app/mailers
create app/models
create app/views/layouts/application.html.erb
create app/mailers/.gitkeep
create app/models/.gitkeep
create config
create config/routes.rb
create config/application.rb
create config/environment.rb
.
.
create vendor/plugins
create vendor/plugins/.gitkeep
run bundle install
```
Fetching source index for https://rubygems.org/
.
.
Your bundle is complete! Use ‘bundle show [gemname]’ to see where a bundled gem is installed.

As seen at the end of Listing 1.3, running rails automatically runs the bundle install command after the file creation is done. If that step doesn’t work right now, don’t worry; follow the steps in Section 1.2.4 and you should be able to get it to work.

Notice how many files and directories the rails command creates. This standard directory and file structure (Figure 1.2) is one of the many advantages of Rails; it immediately gets you from zero to a functional (if minimal) application. Moreover, since the structure is common to all Rails apps, you can immediately get your bearings when looking at someone else’s code. A summary of the default Rails files appears in Table 1.1; we’ll learn about most of these files and directories throughout the rest of this book. In particular, starting in Section 5.2.1 we’ll discuss the app/assets directory,
Table 1.1  A summary of the default Rails directory structure.

<table>
<thead>
<tr>
<th>File/Directory</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>app/</td>
<td>Core application (app) code, including models, views, controllers, and helpers</td>
</tr>
<tr>
<td>app/assets</td>
<td>Applications assets such as cascading style sheets (CSS), JavaScript files, and images</td>
</tr>
<tr>
<td>config/</td>
<td>Application configuration</td>
</tr>
<tr>
<td>db/</td>
<td>Database files</td>
</tr>
<tr>
<td>doc/</td>
<td>Documentation for the application</td>
</tr>
<tr>
<td>lib/</td>
<td>Library modules</td>
</tr>
<tr>
<td>lib/assets</td>
<td>Library assets such as cascading style sheets (CSS), JavaScript files, and images</td>
</tr>
<tr>
<td>log/</td>
<td>Application log files</td>
</tr>
<tr>
<td>public/</td>
<td>Data accessible to the public (e.g., web browsers), such as error pages</td>
</tr>
<tr>
<td>script/rails</td>
<td>A script for generating code, opening console sessions, or starting a local server</td>
</tr>
<tr>
<td>test/</td>
<td>Application tests (made obsolete by the spec/ directory in Section 3.1.2)</td>
</tr>
<tr>
<td>tmp/</td>
<td>Temporary files</td>
</tr>
<tr>
<td>vendor/</td>
<td>Third-party code such as plugins and gems</td>
</tr>
<tr>
<td>vendor/assets</td>
<td>Third-party assets such as cascading style sheets (CSS), JavaScript files, and images</td>
</tr>
<tr>
<td>README.rdoc</td>
<td>A brief description of the application</td>
</tr>
<tr>
<td>Rakefile</td>
<td>Utility tasks available via the rake command</td>
</tr>
<tr>
<td>Gemfile</td>
<td>Gem requirements for this app</td>
</tr>
<tr>
<td>Gemfile.lock</td>
<td>A list of gems used to ensure that all copies of the app use the same gem versions</td>
</tr>
<tr>
<td>config.ru</td>
<td>A configuration file for Rack middleware</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Patterns for files that should be ignored by Git</td>
</tr>
</tbody>
</table>

part of the asset pipeline (new as of Rails 3.1) that makes it easier than ever to organize and deploy assets such as cascading style sheets and JavaScript files.

1.2.4 Bundler

After creating a new Rails application, the next step is to use Bundler to install and include the gems needed by the app. As noted briefly in Section 1.2.3, Bundler is run automatically (via bundle install) by the rails command, but in this section
we’ll make some changes to the default application gems and run Bundler again. This involves opening the **Gemfile** with your favorite text editor:

```
$ cd first_app/
$ subl Gemfile
```

The result should look something like Listing 1.4. The code in this file is Ruby, but don’t worry at this point about the syntax; Chapter 4 will cover Ruby in more depth.

**Listing 1.4** The default **Gemfile** in the **first_app** directory.

```ruby
source 'https://rubygems.org'

gem 'rails', '3.2.13'

# Bundle edge Rails instead:
# gem 'rails', :git => 'git://github.com/rails/rails.git'

gem 'sqlite3'

# Gems used only for assets and not required
# in production environments by default.
group :assets do
  gem 'sass-rails', '~> 3.2.3'
  gem 'coffee-rails', '~> 3.2.2'

  gem 'uglifier', '>= 1.2.3'
end


gem 'jquery-rails'

# To use ActiveModel has_secure_password
# gem 'bcrypt-ruby', '~> 3.0.0'

# To use Jbuilder templates for JSON
# gem 'jbuilder'

# Use unicorn as the web server
# gem 'unicorn'

# Deploy with Capistrano
# gem 'capistrano'

# To use debugger
# gem 'ruby-debug19', :require => 'ruby-debug'
```
Many of these lines are commented out with the hash symbol #; they are there to show you some commonly needed gems and to give examples of the Bundler syntax. For now, we won’t need any gems other than the defaults: Rails itself, some gems related to the asset pipeline (Section 5.2.1), the gem for the jQuery JavaScript library, and the gem for the Ruby interface to the SQLite database.

Unless you specify a version number to the `gem` command, Bundler will automatically install the latest version of the gem. Unfortunately, gem updates often cause minor but potentially confusing breakage, so in this tutorial we’ll include explicit version numbers known to work, as seen in Listing 1.5 (which also omits the commented-out lines from Listing 1.4).

Listing 1.5  A Gemfile with an explicit version of each Ruby gem.

```
source 'https://rubygems.org'

gem 'rails', '3.2.13'

group :development do
  gem 'sqlite3', '1.3.5'
end

# Gems used only for assets and not required
# in production environments by default.

group :assets do
  gem 'sass-rails', '3.2.4'
  gem 'coffee-rails', '3.2.2'
  gem 'uglifier', '1.2.3'
end

gem 'jquery-rails', '2.0.0'
```

Listing 1.5 changes the line for jQuery, the default JavaScript library used by Rails, from

```ruby
  gem 'jquery-rails'
```

to

```ruby
  gem 'jquery-rails', '2.0.0'
```
We’ve also changed

```ruby
gem 'sqlite3'
```
to

```ruby
group :development do
  gem 'sqlite3', '1.3.5'
end
```

which forces Bundler to install version **1.3.5** of the **sqlite3** gem. Note that we’ve also taken this opportunity to arrange for SQLite to be included only in a development environment (Section 7.1.1), which prevents potential conflicts with the database used by Heroku (Section 1.4).

Listing 1.5 also changes a few other lines, converting

```ruby
group :assets do
  gem 'sass-rails', '~> 3.2.3'
  gem 'coffee-rails', '~> 3.2.2'
  gem 'uglifier', '>= 1.2.3'
end
```
to

```ruby
group :assets do
  gem 'sass-rails', '3.2.4'
  gem 'coffee-rails', '3.2.2'
  gem 'uglifier', '1.2.3'
end
```

The syntax

```ruby
gem 'uglifier', '>= 1.2.3'
```

installs the latest version of the **uglifier** gem (which handles file compression for the asset pipeline) as long as it’s greater than version **1.2.3**—even if it’s, say, version **7.2**. Meanwhile, the code

```ruby
gem 'coffee-rails', '~> 3.2.2'
```

installs the gem **coffee-rails** (also needed by the asset pipeline) as long as it’s lower than version **3.3**. In other words, the `>=` notation always performs upgrades, whereas
the \texttt{\textgreater \textgreater 3.2.2} notation only performs upgrades to minor point releases (e.g., from \texttt{3.1.1} to \texttt{3.1.2}), but not to major point releases (e.g., from \texttt{3.1} to \texttt{3.2}). Unfortunately, experience shows that even minor point releases often break things, so for the \textit{Rails Tutorial} we’ll err on the side of caution by including exact version numbers for virtually all gems. (The only exception is gems that are in release candidate or beta stage as of this writing; for those gems, we’ll use \texttt{\textgreater \textgreater} so that the final versions will be loaded once they’re done.)

Once you’ve assembled the proper \texttt{Gemfile}, install the gems using \texttt{bundle install}:

\begin{verbatim}
$ bundle install
  Fetching source index for https://rubygems.org/
  .
  .
  .
\end{verbatim}

(If you’re running OS X and you get an error about missing Ruby header files (e.g., \texttt{ruby.h}) at this point, you may need to install Xcode. These are developer tools that came with your OS X installation disk, but to avoid the full installation I recommend the much smaller Command Line Tools for Xcode.)\textsuperscript{12} The \texttt{bundle install} command might take a few moments, but when it’s done our application will be ready to run. \textit{Note}: This setup is fine for the first app, but it isn’t ideal. Chapter 3 covers a more powerful (and slightly more advanced) method for installing Ruby gems with Bundler.

\subsection*{1.2.5 \texttt{rails server}}

Thanks to running \texttt{rails new} in Section 1.2.3 and \texttt{bundle install} in Section 1.2.4, we already have an application we can run—but how? Happily, Rails comes with a command-line program, or \texttt{script}, that runs a \textit{local} web server, visible only from your development machine.\textsuperscript{13}

\begin{verbatim}
$ rails server
  => Booting WEBrick
  => Rails application starting on http://0.0.0.0:3000
  => Call with -d to detach
  => Ctrl-C to shutdown server
\end{verbatim}

\footnotesize
\textsuperscript{12} https://developer.apple.com/downloads
\textsuperscript{13} Recall from Section 1.1.3 that Windows users might have to type \texttt{ruby rails server} instead.
Chapter 1: From Zero to Deploy

Figure 1.3 The default Rails page.

(If your system complains about the lack of a JavaScript runtime, visit the execjs page at GitHub for a list of possibilities. I particularly recommend installing Node.js.) This tells us that the application is running on port number 300014 at the address 0.0.0.0. This address tells the computer to listen on every available IP address configured on that specific machine; in particular, we can view the application using the special address 127.0.0.1, which is also known as localhost. We can see the result of visiting http://localhost:3000/ in Figure 1.3.

To see information about our first application, click on the link “About your application’s environment.” The result is shown in Figure 1.4. (Figure 1.4 represents the environment on my machine when I made the screenshot; your results may differ.)

---

14. Normally, websites run on port 80, but this usually requires special privileges, so Rails picks a less restricted higher-numbered port for the development server.
Of course, we don’t need the default Rails page in the long run, but it’s nice to see it working for now. We’ll remove the default page (and replace it with a custom home page) in Section 5.3.2.

1.2.6 Model-view-controller (MVC)

Even at this early stage, it’s helpful to get a high-level overview of how Rails applications work (Figure 1.5). You might have noticed that the standard Rails application structure (Figure 1.2) has an application directory called `app/` with three subdirectories: `models`, `views`, and `controllers`. This is a hint that Rails follows the model-view-controller (MVC) architectural pattern, which enforces a separation between “domain logic” (also called “business logic”) from the input and presentation logic associated with a graphical user interface (GUI). In the case of web applications, the “domain logic”
typically consists of data models for things like users, articles, and products, and the GUI is just a web page in a web browser.

When interacting with a Rails application, a browser sends a request, which is received by a web server and passed on to a Rails controller, which is in charge of what to do next. In some cases, the controller will immediately render a view, which is a template that gets converted to HTML and sent back to the browser. More commonly for dynamic sites, the controller interacts with a model, which is a Ruby object that represents an element of the site (such as a user) and is in charge of communicating with the database. After invoking the model, the controller then renders the view and returns the complete web page to the browser as HTML.

If this discussion seems a bit abstract right now, worry not; we’ll refer back to this section frequently. In addition, Section 2.2.2 has a more detailed discussion of MVC in
the context of the demo app. Finally, the sample app will use all aspects of MVC; we’ll cover controllers and views starting in Section 3.1.2, models starting in Section 6.1, and we’ll see all three working together in Section 7.1.2.

1.3 Version Control with Git

Now that we have a fresh and working Rails application, we’ll take a moment for a step that, while technically optional, would be viewed by many Rails developers as practically essential, namely, placing our application source code under version control. Version control systems allow us to track changes to our project’s code, collaborate more easily, and roll back any inadvertent errors (such as accidentally deleting files). Knowing how to use a version control system is a required skill for every software developer.

There are many options for version control, but the Rails community has largely standardized on Git, a distributed version control system originally developed by Linus Torvalds to host the Linux kernel. Git is a large subject, and we’ll only be scratching the surface in this book, but there are many good free resources online; I especially recommend Pro Git by Scott Chacon (Apress, 2009). Putting your source code under version control with Git is strongly recommended, not only because it’s nearly a universal practice in the Rails world, but also because it will allow you to share your code more easily (Section 1.3.4) and deploy your application right here in the first chapter (Section 1.4).

1.3.1 Installation and Setup

The first step is to install Git if you haven’t yet followed the steps in Section 1.2.2. (As noted in that section, this involves following the instructions in the Installing Git section of Pro Git.)

First-time System Setup

After installing Git, you should perform a set of one-time setup steps. These are system setups, meaning you only have to do them once per computer:

```bash
$ git config --global user.name "Your Name"
$ git config --global user.email your.email@example.com
```
I also like to use `co` in place of the more verbose `checkout` command, which we can arrange as follows:

```
$ git config --global alias.co checkout
```

This tutorial will usually use the full `checkout` command, which works for systems that don’t have `co` configured, but in real life I nearly always use `git co`.

As a final setup step, you can optionally set the editor Git will use for commit messages. If you use a graphical editor such as Sublime Text, TextMate, gVim, or MacVim, you need to use a flag to make sure that the editor stays attached to the shell instead of detaching immediately:15

```
$ git config --global core.editor "subl -w"
```

Replace `"subl -w"` with `"mate -w"` for TextMate, `"gvim -f"` for gVim, or `"mvim -f"` for MacVim.

**First-time Repository Setup**

Now we come to some steps that are necessary each time you create a new repository. First, navigate to the root directory of the first app and initialize a new repository:

```
$ git init
Initialized empty Git repository in /Users/mhartl/rails_projects/first_app/.git/
```

The next step is to add the project files to the repository. There’s a minor complication, though: By default Git tracks the changes of all the files, but there are some files we don’t want to track. For example, Rails creates log files to record the behavior of the application; these files change frequently, and we don’t want our version control system to have to update them constantly. Git has a simple mechanism to ignore such files: Simply include a file called `.gitignore` in the application root directory with some rules telling Git which files to ignore.16

---

15. Normally this is a feature, since it lets you continue to use the command line after launching your editor, but Git interprets the detachment as closing the file with an empty commit message, which prevents the commit from going through. I only mention this point because it can be seriously confusing if you try to set your editor to `subl` or `gvim` without the flag. If you find this note confusing, feel free to ignore it.

16. If you can’t see the `.gitignore` file in your directory, you may need to configure your directory viewer to show hidden files.
Looking again at Table 1.1, we see that the `rails` command creates a default `.gitignore` file in the application root directory, as shown in Listing 1.6.

**Listing 1.6** The default `.gitignore` created by the `rails` command.

```
# See http://help.github.com/ignore-files/ for more about ignoring files.
#
# If you find yourself ignoring temporary files generated by your text editor
# or operating system, you probably want to add a global ignore instead:
# git config --global core.excludesfile ˜/.gitignore_global

# Ignore bundler config
/.bundle

# Ignore the default SQLite database.
/db/*.sqlite3

# Ignore all logfiles and tempfiles.
/log/*.log
/tmp
```

Listing 1.6 causes Git to ignore files such as log files, Rails temporary (`tmp`) files, and SQLite databases. (For example, to ignore log files, which live in the `log/` directory, we use `log/*.log` to ignore all files that end in `.log`.) Most of these ignored files change frequently and automatically, so including them under version control is inconvenient; moreover, when collaborating with others they can cause frustrating and irrelevant conflicts.

The `.gitignore` file in Listing 1.6 is probably sufficient for this tutorial, but depending on your system you may find Listing 1.7 more convenient. This augmented `.gitignore` arranges to ignore Rails documentation files, Vim and Emacs swap files, and (for OS X users) the weird `.DS_Store` directories created by the Mac Finder application. If you want to use this broader set of ignored files, open up `.gitignore` in your favorite text editor and fill it with the contents of Listing 1.7.

**Listing 1.7** An augmented `.gitignore` file.

```
# Ignore bundler config
/.bundle

# Ignore the default SQLite database.
/db/*.sqlite3
```
# Ignore all logfiles and tempfiles.
/log/*.log
/tmp

# Ignore other unneeded files.
doc/
*.swp
*-
.project
.DS_Store

1.3.2 Adding and Committing

Finally, we’ll add the files in your new Rails project to Git and then commit the results. You can add all the files (apart from those that match the ignore patterns in `.gitignore`) as follows:

```sh
$ git add .
```

Here the dot “.” represents the current directory, and Git is smart enough to add the files *recursively*, so it automatically includes all the subdirectories. This command adds the project files to a *staging area*, which contains pending changes to your project; you can see which files are in the staging area using the `status` command:

```sh
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
#
new file: README.rdoc
new file: Rakefile
.
.
.
```

(The results are long, so I’ve used vertical dots to indicate omitted output.)

---

17. If in the future any unwanted files start showing up when you type `git status`, just add them to your `.gitignore` file from Listing 1.7.
1.3 Version Control with Git

To tell Git you want to keep the changes, use the `commit` command:

```
$ git commit -m "Initial commit"
[master (root-commit) df0a62f] Initial commit
42 files changed, 8461 insertions(+), 0 deletions(-)
create mode 100644 README.rdoc
create mode 100644 Rakefile
.
.
```

The `-m` flag lets you add a message for the commit; if you omit `-m`, Git will open the editor you set in Section 1.3.1 and have you enter the message there.

It is important to note that Git commits are local, recorded only on the machine on which the commits occur. This is in contrast to the popular open-source version control system called Subversion, in which a commit necessarily makes changes on a remote repository. Git divides a Subversion-style commit into its two logical pieces: A local recording of the changes (`git commit`) and a push of the changes up to a remote repository (`git push`). We’ll see an example of the push step in Section 1.3.5.

By the way, you can see a list of your commit messages using the `log` command:

```
$ git log
commit df0a62f3f091e53ffa799309b3e32c27b0b38eb4
Author: Michael Hartl <michael@michaelhartl.com>
Date: Thu Oct 15 11:36:21 2009 -0700

Initial commit
```

To exit `git log`, you may have to type `q` to quit.

1.3.3 What Good Does Git Do You?

It’s probably not entirely clear at this point why putting your source under version control does you any good, so let me give just one example. (We’ll see many others in the chapters ahead.) Suppose you’ve made some accidental changes, such as (D’oh!) deleting the critical `app/controllers/` directory:

```
$ ls app/controllers/
application_controller.rb
$ rm -rf app/controllers/
$ ls app/controllers/
ls: app/controllers/: No such file or directory
```
Here we’re using the Unix `ls` command to list the contents of the `app/controllers/` directory and the `rm` command to remove it. The `-rf` flag means “recursive force”, which recursively removes all files, directories, subdirectories, and so on, without asking for explicit confirmation of each deletion.

Let’s check the status to see what’s up:

```bash
$ git status
# On branch master
# Changed but not updated:
# (use "git add/rm <file>..." to update what will be committed)
# (use "git checkout -- <file>..." to discard changes in working directory)
# deleted:  app/controllers/application_controller.rb
# no changes added to commit (use "git add" and/or "git commit -a")
```

We see here that a file has been deleted, but the changes are only on the “working tree”; they haven’t been committed yet. This means we can still undo the changes easily by having Git check out the previous commit with the `checkout` command (and a `-f` flag to force overwriting the current changes):

```bash
$ git checkout -f
$ git status
# On branch master
nothing to commit (working directory clean)
$ ls app/controllers/
application_controller.rb
```

The missing directory and file are back. That’s a relief!

### 1.3.4 GitHub

Now that you’ve put your project under version control with Git, it’s time to push your code up to GitHub, a social code site optimized for hosting and sharing Git repositories. Putting a copy of your Git repository at GitHub serves two purposes: It’s a full backup of your code (including the full history of commits), and it makes any future collaboration much easier. This step is optional, but being a GitHub member will open the door to participating in a wide variety of open-source projects.
1.3 Version Control with Git

GitHub has a variety of paid plans, but for open-source code their services are free, so sign up for a free GitHub account if you don’t have one already. (You might have to follow the GitHub tutorial on creating SSH keys first.) After signing up, click on the link to create a repository and fill in the information as in Figure 1.6. (Take care not to initialize the repository with a README file, as rails new creates one of those automatically.) After submitting the form, push up your first application as follows:

```
$ git remote add origin git@github.com:<username>/first_app.git
$ git push -u origin master
```

These commands tell Git that you want to add GitHub as the origin for your main (master) branch and then push your repository up to GitHub. (Don’t worry about what the -u flag does; if you’re curious, do a web search for “git set upstream”.) Of course,
Chapter 1: From Zero to Deploy

Figure 1.7  A GitHub repository page.

...you should replace `<username>` with your actual username. For example, the command I ran for the `railstutorial` user was

```
$ git remote add origin git@github.com:railstutorial/first_app.git
```

The result is a page at GitHub for the first application repository, with file browsing, full commit history, and lots of other goodies (Figure 1.7).

1.3.5 Branch, Edit, Commit, Merge

If you’ve followed the steps in Section 1.3.4, you might notice that GitHub automatically shows the contents of the `README` file on the main repository page. In our case, since the project is a Rails application generated using the `rails` command, the `README` file is the one that comes with Rails (Figure 1.8). Because of the `.rdoc` extension on the file, GitHub ensures that it is formatted nicely, but the contents aren’t
Figure 1.8  The initial (rather useless) README file for our project at GitHub.

helpful at all, so in this section we’ll make our first edit by changing the README to
describe our project rather than the Rails framework itself. In the process, we’ll see a
first example of the branch, edit, commit, merge workflow that I recommend using
with Git.

Branch
Git is incredibly good at making branches, which are effectively copies of a repository
where we can make (possibly experimental) changes without modifying the parent files.
In most cases, the parent repository is the master branch, and we can create a new topic
branch by using checkout with the -b flag:

```bash
$ git checkout -b modify-README
Switched to a new branch 'modify-README'
$ git branch
master
  * modify-README
```
Chapter 1: From Zero to Deploy

Here the second command, `git branch`, just lists all the local branches, and the asterisk * identifies which branch we’re currently on. Note that `git checkout -b modify-README` both creates a new branch and switches to it, as indicated by the asterisk in front of the `modify-README` branch. (If you set up the `co` alias in Section 1.3, you can use `git co -b modify-README` instead.)

The full value of branching only becomes clear when working on a project with multiple developers, but branches are helpful even for a single-developer tutorial such as this one. In particular, the master branch is insulated from any changes we make to the topic branch, so even if we really screw things up, we can always abandon the changes by checking out the master branch and deleting the topic branch. We’ll see how to do this at the end of the section.

By the way, for a change as small as this one I wouldn’t normally bother with a new branch, but it’s never too early to start practicing good habits.

Edit
After creating the topic branch, we’ll edit it to make it a little more descriptive. I prefer the Markdown markup language to the default RDoc for this purpose, and if you use the file extension `.md` then GitHub will automatically format it nicely for you. So, first we’ll use Git’s version of the Unix `mv` (“move”) command to change the name, and then fill it in with the contents of Listing 1.8:

```
$ git mv README.rdoc README.md
$ subl README.md
```

Listing 1.8 The new `README` file, `README.md`.

```text
# Ruby on Rails Tutorial: first application

This is the first application for
[*Ruby on Rails Tutorial: Learn Rails by Example*](http://railstutorial.org/)
by [Michael Hartl](http://michaelhartl.com/).
```

Commit
With the changes made, we can take a look at the status of our branch:

18. See the chapter Git Branching in *Pro Git* for details.
At this point, we could use `git add` as in Section 1.3.2, but Git provides the `-a` flag as a shortcut for the (very common) case of committing all modifications to existing files (or files created using `git mv`, which don’t count as new files to Git):

```
$ git commit -a -m "Improve the README file"
2 files changed, 5 insertions(+), 243 deletions(-)
delete mode 100644 README.rdoc
create mode 100644 README.md
```

Be careful about using the `-a` flag improperly; if you have added any new files to the project since the last commit, you still have to tell Git about them using `git add` first.

Note that we write the commit message in the present tense. Git models commits as a series of patches, and in this context it makes sense to describe what each commit does, rather than what it did. Moreover, this usage matches up with the commit messages generated by Git commands themselves. See the GitHub post Shiny new commit styles for more information.

### Merge

Now that we’ve finished making our changes, we’re ready to `merge` the results back into our master branch:

```
$ git checkout master
Switched to branch 'master'
$ git merge modify-README
Updating 34f06b7..2c92bef
Fast forward
README.rdoc | 243 --------------------------------------------------
```
Note that the Git output frequently includes things like 34f06b7, which are related to Git’s internal representation of repositories. Your exact results will differ in these details, but otherwise should essentially match the output shown above.

After you’ve merged in the changes, you can tidy up your branches by deleting the topic branch using `git branch -d` if you’re done with it:

```bash
$ git branch -d modify-README
Deleted branch modify-README (was 2c92bef).
```

This step is optional, and in fact it’s quite common to leave the topic branch intact. This way you can switch back and forth between the topic and master branches, merging in changes every time you reach a natural stopping point.

As mentioned above, it’s also possible to abandon your topic branch changes, in this case with `git branch -D`:

```bash
# For illustration only; don’t do this unless you mess up a branch
$ git checkout -b topic-branch
$ <really screw up the branch>
$ git add .
$ git commit -a -m "Major screw up"
$ git checkout master
$ git branch -D topic-branch
```

Unlike the `-d` flag, the `-D` flag will delete the branch even though we haven’t merged in the changes.

**Push**

Now that we’ve updated the `README`, we can push the changes up to GitHub to see the result. Since we have already done one push (Section 1.3.4), on most systems we can omit `origin master` and simply run `git push`:

```bash
$ git push
```

As promised, GitHub nicely formats the new file using Markdown (Figure 1.9).
1.4 Deploying

Even at this early stage, we’re already going to deploy our (still-empty) Rails application to production. This step is optional, but deploying early and often allows us to catch any deployment problems early in our development cycle. The alternative—deploying only after laborious effort sealed away in a development environment—often leads to terrible integration headaches when launch time comes.19

Deploying Rails applications used to be a pain, but the Rails deployment ecosystem has matured rapidly in the past few years, and now there are several great options. These include shared hosts or virtual private servers running Phusion Passenger (a module for the Apache and Nginx20 web servers), full-service deployment companies such as Engine Yard and Rails Machine, and cloud deployment services such as Engine Yard Cloud and Heroku.

My favorite Rails deployment option is Heroku, which is a hosted platform built specifically for deploying Rails and other Ruby web applications.21 Heroku makes deploying Rails applications ridiculously easy—as long as your source code is under version control with Git. (This is yet another reason to follow the Git setup steps in Section 1.3 if you haven’t already.) The rest of this section is dedicated to deploying our first application to Heroku.

1.4.1 Heroku Setup

After signing up for a Heroku account, install the Heroku gem:

$ gem install heroku

---

19. Though it shouldn’t matter for the example applications in the Rails Tutorial, if you’re worried about accidentally making your app public too soon there are several options; see Section 1.4.4 for one.

20. Pronounced “Engine X.”

21. Heroku works with any Ruby web platform that uses Rack middleware, which provides a standard interface between web frameworks and web servers. Adoption of the Rack interface has been extraordinarily strong in the Ruby community, including frameworks as varied as Sinatra, Ramaze, Camping, and Rails, which means that Heroku basically supports any Ruby web app.
As with GitHub (Section 1.3.4), when using Heroku you will need to create SSH keys if you haven’t already, and then tell Heroku your public key so that you can use Git to push the sample application repository up to their servers:

$ heroku keys:add

Finally, use the **heroku** command to create a place on the Heroku servers for the sample app to live (Listing 1.9).

**Listing 1.9** Creating a new application at Heroku.

$ heroku create --stack cedar

Created http://stormy-cloud-5881.herokuapp.com/ | git@heroku.com:stormy-cloud-5881.herokuapp.com

Git remote heroku added

(The --stack cedar argument arranges to use the latest and greatest version of Heroku, called the Celadon Cedar Stack.) Yes, that’s it. The **heroku** command creates a new subdomain just for our application, available for immediate viewing. There’s nothing there yet, though, so let’s get busy deploying.

### 1.4.2 Heroku Deployment, Step One

To deploy to Heroku, the first step is to use Git to push the application to Heroku:

$ git push heroku master

### 1.4.3 Heroku Deployment, Step Two

There is no step two! We’re already done (Figure 1.10). To see your newly deployed application, you can visit the address that you saw when you ran **heroku create** (i.e., Listing 1.9, but with the address for your app, not the address for mine). You can also use an argument to the **heroku** command that automatically opens your browser with the right address:

$ heroku open

Because of the details of their setup, the “About your application’s environment” link doesn’t work on Heroku. Don’t worry; this is normal. The error will go away (in
Figure 1.10  The first Rails Tutorial application running on Heroku.

the context of the full sample application) when we remove the default Rails page in Section 5.3.2.

Once you’ve deployed successfully, Heroku provides a beautiful interface for administering and configuring your application (Figure 1.11).

1.4.4 Heroku Commands

There are many Heroku commands, and we’ll barely scratch the surface in this book. Let’s take a minute to show just one of them by renaming the application as follows:

```
$ heroku rename railstutorial
```

Don’t use this name yourself; it’s already taken by me! In fact, you probably shouldn’t bother with this step right now; using the default address supplied by Heroku is fine.
But if you do want to rename your application, you can arrange for it to be reasonably secure by using a random or obscure subdomain, such as the following:

hwpcbmze.herokuapp.com
seyjhflo.herokuapp.com
jhyicevg.herokuapp.com

With a random subdomain like this, someone could visit your site only if you gave him or her the address. (By the way, as a preview of Ruby’s compact awesomeness, here’s the code I used to generate the random subdomains:

```ruby
('a'..'z').to_a.shuffle[0..7].join
```

Pretty sweet.)

In addition to supporting subdomains, Heroku also supports custom domains. (In fact, the Ruby on Rails Tutorial site lives at Heroku; if you’re reading this book online, you’re looking at a Heroku-hosted site right now!) See the Heroku documentation for more information about custom domains and other Heroku topics.
1.5 Conclusion

We’ve come a long way in this chapter: installation, development environment setup, version control, and deployment. If you want to share your progress at this point, feel free to send a tweet or Facebook status update with something like this:

   I’m learning Ruby on Rails with @railstutorial! http://railstutorial.org

All that’s left is to actually start learning Rails! Let’s get to it.
This page intentionally left blank
Index

Note: Page numbers in *italics* indicate figures, those with *t* indicate tables, and those with *n* indicate footnotes.

**Symbols**

" (double quote character), 135
# (hash symbol), 21
/ (forward slash), 8
|| = construction, 354–355
! (not) operator, 139
&& (and) operator, 139
+ (plus) operator, 135
|| (or) operator, 139

code for (Listing 3.16), 102
footer partial with links for (Listing 5.25), 206
with HTML structure (Listing 3.21), 108
with HTML structure removed (Listing 3.28), 113
new, 102
refactoring, 103
StaticPages controller with added about action (Listing 3.15), 101
tests for static pages (Listing 5.27), 210
view for, with Embedded Ruby title
  (Listing 3.24), 110–111
writing a failing test for, 99–100
abstraction layers, 226n4
access control, 456–459
access control in manipulating Microposts,
  456–459
accessible attributes and first validation, 432–433
accessible attributes in model file, 230
actions, 85–86
Active Record, 222
callback, 253
count method, 295
creating user objects, 230–233
finding user objects, 233–235
updating user objects, 235–236
See also Validations
adding files, in Git, 30–31
administrative users, 413–417
attr_accessible, 416–417
attr_accessible attributes for User model
  without :admin attribute (Listing 9.42),
  417
deleting, 413–417
migration to add boolean admin attribute to
  users (Listing 9.40), 415

A

About page
  about route, adding (Listing 3.14), 100–101
  about view, adding, 101–102
  adding, 99–103
  adding code to test contents of (Listing 3.13),
    99
  code for (Listing 3.16), 102
  footer partial with links for (Listing 5.25), 206
  with HTML structure (Listing 3.21), 108
  with HTML structure removed (Listing 3.28),
    113
  new, 102
  refactoring, 103
  StaticPages controller with added about action
    (Listing 3.15), 101
administrative users (continued)
sample data populator code with admin user
(Listing 9.41), 416
tests for admin attribute (Listing 9.39), 414
user delete links (viewable by admins)
(Listing 9.45), 419
User model with admin boolean attribute, 415
administrative users, deleting, 413–417
Ajax
follow button with, 524–529
form for following a user using (Listing 11.35), 525
form for unfollowing a user using
(Listing 11.36), 525
JavaScript Embedded Ruby to create following
relationship (Listing 11.39), 529
problem solved by, 524
Ruby JavaScript to destroy following
relationship (Listing 11.40), 529
Ajax requests, responding to, 525–529
JS-ERb, 528–529
in Relationships controller (Listing 11.38), 527
tests for Relationships controller
(Listing 11.37), 526
ampersand (&), 534
anchor tag, 97
annotate, 229–230
annotated User model (Listing 6.5), 229–230
ApplicationController class with inheritance
(Listing 2.16), 72
Application Programmer Interface (API), 542
application root, 9, 28–29, 125, 161
Architectural Styles and the Design of Network-based
Software Architectures (Fielding), 60n4
arrays, in Ruby data structures, 142–145
asset directory in asset pipeline, 187–188
asset pipeline, Sass and, 187–190
asset directory, 187–188
efficiency in production, 189–190
manifest files, 188–189
preprocessor engines, 189
assignment, 352
See also Mass assignment
associations
Micropost resource, 68–70
user/micropost, 433–438
user/relationship, 491–494
associative arrays, 148
asynchronous JavaScript and XML. See Ajax
attr_accessible
administrative users, 416–417
attributes for User model without :admin
attribute (Listing 9.42), 417
making name and email attributes accessible
(Listing 6.6), 230
to prevent mass assignment, 230, 416–417
attribute accessor, 162
authenticate method
has_secure_password, 264, 338
moving the authenticate method into the
Sessions helper (Listing 10.27), 457–458
test for (Listing 6.29), 262–263
authentication, 260–263
adding authentication to Microposts controller
actions (Listing 10.28), 458
vs. authorization, 385
sessions and, 325–326
signin failure, 325–343
signin success, 343–363
See also Authenticate method
authenticity token, 301
authorization, 385–396
vs. authentication, 385
of following and followers pages, tests for
(Listing 11.28), 516–517
friendly forwarding, 392–396
for relationships controller, tests for
(Listing 11.33), 522–523
requiring right user, 390–392
requiring signed-in users, 386–389
automated testing, 77
Automattic, 286
avatar, 286n7
B
Bates, Ryan, 6, 7, 540
BCrypt cost factor in test environment, redefining
(Listing 7.11), 286
before filters, 373
adding a signed_in_user before filter
(Listing 9.12), 387
applied to every action in controller, 387
correct_user before filter in microposts,
477–478
correct_user before filter to protect edit/update pages (Listing 9.15), 391
 current_user boolean method, 391–392
 in requiring right user, 390–392
 restricting destroy action to admins (Listing 9.48), 422

Beginning Ruby (Cooper), 4, 5, 129, 543
Black, David A., 6, 543
blocks, in Ruby data structures, 146–148
Booleans, 133, 138–139, 142

Bootstrap
 adding bootstrap-sass gem to (Listing 5.3), 175
 adding to application.js (Listing 8.25), 358
 and custom CSS in layout structure, 175–186
 framework, 176, 317

browsers, 11–12
built-in Ruby classes, modifying, 158–159
bundle exec, eliminating, 118–119
 binstubs, 119
 RVM Bundler integration, 118–119
Bundler, 19–23
business logic, 25

C
callback, 253, 346–348
Capybara, 79
 in Cucumber step files, 367
 integration tests, 93
 signin tests, 330
 signup tests, 294
 syntax for CSS id, 471
 in test-driven development, 94–95
 test for destroying microposts, 477
 tests for user update action (Listing 9.9), 383

cascading style sheets (CSS), 152–153, 190–197
 asset directory, 187–188
 Bootstrap framework, 176, 317
 Capybara syntax for CSS id, 471
 custom CSS, 175–186
 efficiency in production, 189–190
 HTML source produced by CSS includes (Listing 4.7), 153
 layout links, 197–211
 manifest files, 188–189
 for microposts (Listing 10.24), 452–453
 mixins, 274–275
 nesting, 190–192

partials, 181–186
 preprocessor engines, 189
 in Ruby data structures, 152–153
 Sass, 187–197
 site navigation, 169–175
 structure, adding, 167–186
 for styling error messages (Listing 7.24), 311
 for user index (Listing 9.26), 400
 user signup, 211–215
 variables, 193–197

Celadon Cedar Stack, 40
chaining, 139, 421
checkout command, 28, 32
Chrome, 11–12, 103, 170
classes, 153–163
 built-in, modifying, 158–159
 code for example user (Listing 4.9), 161
 constructor, 153–154
 container class, 172
 controller, 159–161
 defining Word class in console (Listing 4.8), 156
 inheritance, 155–157
 user, 161–163
 class methods, 154–155
 class name converted to id, 493

Code School, 6, 543
code for example user (Listing 4.9), 161
columns, 10, 11
comments, 134–135
commit command, in Git, 31
config directory, 9, 88, 89
class methods, 154–155
constructor classes, 153–163
Contact page
 action for (Listing 5.18), 199
 adding, 197–199
 adding route for (Listing 5.17), 199
 footer partial with links for (Listing 5.25), 206
 for sample app, 114–117
 tests for (Listing 5.16), 198
 tests for static pages (Listing 5.27), 210
 view for (Listing 5.19), 199

container class, 172
content validations, Micropost model, 443–444
controller classes, 159–161
cookies, 349–351
 expiring 20 years in the future, 350
 remember token added to, 379
cookies (continued)
  remember token removed from, 363
  used as a hash, 349–351
Cooper, Peter, 4, 5, 543
correct_user before filter
  in microposts, 477–478
  to protect edit/update pages (Listing 9.15), 391
counting columns, 105n12
count method, 295
create action
  adding (empty) @feed_items instance variable
to (Listing 10.45), 474–475
completed, 313
completed Sessions controller create action (not yet working) (Listing 8.13), 343
handling signup failure (but not success)
  (Listing 7.21), 305
for microposts, 461
Microposts controller create action
  (Listing 10.30), 461
preliminary version of sessions create action
  (Listing 8.9), 337
for Sessions controller, 326, 336–338, 343, 395
Sessions create action with friendly forwarding
  (Listing 9.20), 395
in signup failure, 304–305
strategy for using, 304
tests for post-save behavior in (Listing 7.32), 323
user create action with save and redirect
  (Listing 7.25), 314
for Users controller, 425, 459
creating microposts, 459–467
adding micropost instance variable to home action (Listing 10.34), 463
adding microposts creation to Home page
  (Listing 10.31), 461
form partial for creating microposts
  (Listing 10.33), 463
Microposts controller create action
  (Listing 10.30), 461
partial for user info sidebar (Listing 10.32), 462
tests for (Listing 10.29), 460
updating error-messages partial from Listing 7.23 to work with other objects
  (Listing 10.35), 464
updating errors for editing users
  (Listing 10.37), 465
updating rendering of user signup errors
  (Listing 10.36), 465
cross-site request forgery (CSRF), 301
cross-site scripting attack, 481
CSS. See Cascading style sheets (CSS)
CSS: The Missing Manual (McFarland), 5
Cucumber, 363–371
  adding cucumber-rails gem to Gemfile
    (Listing 8.31), 364
  adding helper method and custom RSpec matcher (Listing 8.34), 371
  features and steps, 365–368
  features to test user signin (Listing 8.32), 366
  installation and setup, 364–365
  RSpec examples, equivalent, 368–371
  signin tests using, 363–371
  steps needed to get signin features to pass
    (Listing 8.33), 368
current user, 351–355
  defining assignment to (Listing 8.20), 352
  definition for (Listing 8.21), 353
  finding, using remember_token (Listing 8.22), 353
  non-nil, 356
  in signin success, 351–355
current_user? boolean method, 391–392
D
database indices, 254
database migration. See Migration
data model
  defined, 47
  micropost, 48–49
  user, 47–48
debug
  adding code for debug box, including Sass mixin (Listing 7.2), 274
  information, adding to site layout (Listing 7.1), 273–274
  information, restricting to development environment, 276
  information in sign up, 271–276
  output, 275
  in Rails environments, 276
default Rails directory structure, 19!
default Rails page, 24
  with the app environment, 25
default scope in Micropost model refinements, 440–441
demo app, 45–75
  conclusion, 74–75
  Microposts resource, 63–74
  planning the application, 45–49
  Users resource, 49–63
demo app, deploying, 73–74
dependent refinements in Micropost model, 441–443
deploying Rails, 39–42
destroy action
  adding factory for administrative users
    (Listing 9.43), 417–418
  adding working destroy action (Listing 9.46), 420–421
  in deleting users, 417–422
  before filter restricting destroy action to admins
    (Listing 9.48), 422
  test for protecting destroy action (Listing 9.47),
    421–422
tests for delete links (Listing 9.44), 418–419
user index /users with delete links, 420
destroying microposts
  ensuring that user’s microposts are destroyed
    along with user (Listing 10.16), 443
feed item partial with added delete link
  (Listing 10.47), 476
Microposts controller destroy action
  (Listing 10.49), 477–478
mockup of proto-feed with micropost delete links, 476
testing that microposts are destroyed when
users are (Listing 10.15), 442
tests for Microposts controller destroy action
  (Listing 10.48), 477
user home page after deleting
  second-most-recent micropost, 479
development environment, 9–27
  browsers, 11–12
  command lines, 10, 11
  IDEs, 10
  terminals, 11
text editors, 10, 11
time learning tools, 12
development log, 231–232, 450n4
directories
  standard directory and file structure, 18
  summary of default Rails directory structure, 19t
div tags, 171
doctype, 84
Document Object Model (DOM), 528
domain logic, 25
domain-specific language (DSL), 3, 94, 283
drb option, 125
duplication, eliminating, 103, 111–113
dynamic pages. See Slightly dynamic pages

E
each method, 146, 151, 245, 399, 533n11edit form, in updating users, 374–380
edits in updating users, successful, 382–384
edits in updating users, unsuccessful, 380–382
Emacs, 29
Embedded Ruby
  instance variables and, 162
  JavaScript, to create following relationship
    (Listing 11.39), 529
  slightly dynamic pages, 108–111
Embedded Ruby title
  view for About page with (Listing 3.24),
    110–111
  view for Help page with (Listing 3.23), 110
  view for Home page with (Listing 3.22), 109
empty? method, 138, 139, 310
encrypted passwords, 255–257
Engine Yard, 13, 16
Engine Yard Cloud, 39
environment loading, adding to Spork.prefork
  block (Listing 3.36), 124
equality comparison operator, 144
ERb. See Embedded Ruby
error messages, signup, 308–312
  code to display error messages on signup form
    (Listing 7.22), 309
CSS for styling error messages (Listing 7.24), 311
  failed signup with error messages, 312
  partial for displaying form submission error
    messages (Listing 7.23), 309
exceptions, 234n8
factories
  complete factory file, including new factory for microposts (Listing 10.12), 439
to simulate User model objects (Listing 7.8), 284
test for user show page (Listing 7.9), 285
testing user show page with, 282–286
Factory Girl, 283–286
  adding to Gemfile (Listing 7.7), 284
  in micropost refinements, 439–440
  sequence, defining (Listing 9.32), 407
  sequence, solving problems in, 406
  slow nature of running, 285–286
Faker gem, 403
  adding to Gemfile (Listing 9.29), 403
  lorem ipsum text, 450–451, 451n5
feed, 429
  proto-, 467–475
  RSS, 542
  status, 529–539
Fernandez, Obie, 6, 142n5, 543
Fielding, Roy, 60
Files
  standard directory and file structure, 18
  summary of default Rails directory structure, 19r
Firebug, 12, 301
Firefox, 11–12, 89, 170
first feed implementation, 532–535
flash, 315–317
  adding contents of flash variable to site layout
    (Listing 7.26), 315–316
  adding flash message to user signup
    (Listing 7.27), 317
ERb in site layout using content tag
  (Listing 7.33), 324
vs. flash.now, 316n11
message for failed signin, 339–343, 340
flash.now, 316n11, 342
follow and unfollow buttons, 519–529
  with Ajax, 524–529
  current user’s followers, 520
  profile of user to follow, with follow button, 486
profile with unfollow button and incremented followers count, 487
Relationships controller (Listing 11.34), 523–524
tests for (Listing 11.32), 521–522
tests for relationships controller authorization
  (Listing 11.33), 522–523
user profile with follow button, 514
users being followed by current user, 520
working follow button, 519–524
followed users in relationship model, 495–500
follower notifications, 541
followers, 500–503
  implementing user.followers using reverse relationships (Listing 11.16), 502
  model for user followers using reverse Relationship model, 500
  testing for reverse relationships (Listing 11.15), 501
followers relationship model, 500–503
follow form, 505–514
  adding followed_users and followers actions
to Users controller (Listing 11.18), 506
  adding follow form and follower stats to user profile page (Listing 11.27), 513
for following a user using (Listing 11.35), 525
form for following user (Listing 11.25), 512
form for unfollowing user (Listing 11.26), 512
partial for follow/unfollow form
  (Listing 11.23), 511
RESTful routes provided by custom rules in resource, 506r
routes added for user relationships
  (Listing 11.24), 512
for unfollowing a user using (Listing 11.36), 525
following
  adding following/follower relationships to sample data (Listing 11.17), 503–504
following? and follow! utility methods
  (Listing 11.12), 498
problem with the data model (and a solution), 485–490
relationship model, 484–503
sample following data, 503–505
user/relationship associations, 491–494
users, 503–544
utility methods, tests for (Listing 11.11), 497
following and followers pages, 515–519
    following and followers actions (Listing 11.30), 518
mockups of, 515–516
show_follow view used to render following and followers (Listing 11.31), 519
test for followed_users and followers pages (Listing 11.29), 517–518
tests for authorization of (Listing 11.28), 516–517
following data, sample, 503–505
following? method, 497–500
follow! method, 497–500
forgery, 112
format, validating, 245–248
form_for, 297–300
form_tag, 303, 334, 372
forward slash (/), 8
Fowler, Martin, 222n1
friendly forwarding, 392–396
    adding store_location to signed-in user before filter (Listing 9.19), 394–395
code to implement (Listing 9.18), 394
    Sessions create action with (Listing 9.20), 395
test for friendly forwarding (Listing 9.17), 393
full-table scan, 254
Fulton, Hal, 6, 543
functions, 91

g
gem configuration file
    creating (Listing 1.1), 16
    suppressing ri and rdoc documentation in (Listing 1.2), 16
Gemfile
    adding annotate gem to (Listing 6.4), 229
    adding bcrypt-ruby to (Listing 6.24), 255
    adding bootstrap-sass gem to (Listing 5.3), 175
    adding cucumber-rails gem to (Listing 8.31), 364
    adding Factory Girl to (Listing 7.7), 284
    adding Faker gem to (Listing 9.29), 403
default, in the first_app directory (Listing 1.4), 20
default Gemfile in the first app directory
    (Listing 1.4), 20
for demo app (Listing 2.1), 46
with explicit version of each Ruby gem
    (Listing 1.5), 21–22
including will paginate in (Listing 9.31), 405
needed to use PostgreSQL instead of SQLite
    (Listing 3.31), 117
for sample app (Listing 3.1), 78
for sample app (Listing 3.35), 123
for sample app, final (Listing 9.49), 423–424
for sample app including Guard (Listing 3.33), 120
gems, 14
gemsets, 14–15
generated code, scaffolding and, 3
generate script, 49, 86, 94
GET, 89–90
Git, 27–39
    adding files in, 30–31
    benefit of using, 31–32
    branches, 35–36
    commit command, 31
    committing, 36–37
    editing, 36
    first-time repository setup, 28–30
    first-time setup, 27–28
    installing, 13
    merging, 37–38
    pushing, 38–39
    README file, 34–35, 35
    README file, README.md (Listing 1.8), 36
    README file formatted with Markdown, 39
    status command, 30
GitHub, 32–34
    creating first app repository at, 33
    creating sample app repository at, 81
    initial README file for project at, 35
    repository page, 34
.gitignore
    augmented .gitignore file (Listing 1.7), 29–30
default .gitignore created by rails command
    (Listing 1.6), 29
Goia, Mircea, 14
Gravatar, 286–291
  adding sidebar to user show view (Listing 7.14), 290
  defining gravatar_for helper method (Listing 7.13), 288
  editing, 382–384
  SCSS for styling user show page, including sidebar (Listing 7.15), 290–291
  in sign up, 286–291
  user profile page /users/1 with default Gravatar, 289
  user show page /users/1 with sidebar and CSS, 291
  user show page with custom Gravatar, 289
  user show view with name and (Listing 7.12), 287
Guard
  automated tests with, 120–122
  Gemfile for sample app including (Listing 3.33), 120
  Spork with Guard, 126–127
gVim, 28

H
Hansson, David Heinemeier, 2, 4
hashes, 337
  nested (Listing 4.6), 151
  in Ruby data structures, 148–152
hash symbol, 21
has_secure_password
  authenticate method, 264, 338
  User, 263–265
have_selector method, 104
Head First HTML, 5
Help page
  code added to test (Listing 3.11), 98
  generated view for (Listing 3.8), 92
  with HTML structure (Listing 3.20), 107
  with HTML structure removed (Listing 3.27), 113
  tests for static pages (Listing 5.27), 210
  view for, with Embedded Ruby title (Listing 3.23), 110
Heroku
  commands, 41–42
  creating a new application at (Listing 1.9), 40
  deployment, 40–41
  setup, 39–40
hierarchies, inheritance, 70–73, 73
Home page
  adding follower stats to (Listing 11.21), 509
  adding microposts creation to (Listing 10.31), 461–462
  with follow stats, 511
  generated view for (Listing 3.7), 92
  with HTML structure (Listing 3.19), 107
  with HTML structure removed (Listing 3.26), 113
  with link to signup page (Listing 5.2), 173
  mockup with form for creating microposts, 459
  mockup with proto-feed, 467
  SCSS for Home page sidebar (Listing 11.22), 510
  with status feed, mockup of, 530
  with status feed and incremented following count, 488
  testing following/follower statistics on (Listing 11.19), 507
  view for, with Embedded Ruby title (Listing 3.22), 109
  view for, with HTML structure (Listing 3.19), 107
  with working status feed, 539
href, 97
HTML
About page with HTML structure removed (Listing 3.28), 113
About page with structure (Listing 3.21), 108
code for signin form (Listing 8.7), 334
for edit form defined in Listing 9.3 and shown in Figure 9.2. (Listing 9.4), 377
for form in Figure 7.12 (Listing 7.20), 301
initial user edit page with pre-filled name and email, 377
produced by CSS includes (Listing 4.7), 153
for signin form produced by Listing 8.7, 335
signup form, 301–303, 335
for signup form/signup for new users, 300
typical, with friendly greeting (Listing 3.3), 84
user edit action (Listing 9.2), 375
for user edit form (Listing 9.2), 377
HTTP, 89–90
HTTP verbs, 89, 90
hypertext reference (href), 97
hypertext transfer protocol. See HTTP
IDEs, 10
implicit return, 141
index, user. See User index
index action, simplified for demo app
(Listing 2.4), 61
index.html file, 82–84, 83
index page
with 100 sample users, 405
correspondence between pages and URIs for
Users resource, 52t
initial, for Users resource, 52
micropost, 68
with one user, 402
with second user, 55
tests for (Listing 9.23), 398–399
users with pagination, 410
inheritance
ApplicationController class with (Listing 2.16), 72
class, 155–157
classes, 155–157
hierarchies, 70–73, 73
Micropost class with (Listing 2.13), 71
MicropostsController class with (Listing 2.15), 72
User class with (Listing 2.12), 71
UsersController class with (Listing 2.14), 72
inheritance class, 155–157
inheritance hierarchies, 70–73, 73
initialization hash, 231
inspect method, 151
installing Rails, 16–17
instance variables, 61, 162
adding (empty) @feed items to create action
(Listing10.45), 474–475
adding to home action (Listing 10.41), 471
adding to the home action (Listing 10.34), 463
adding to user show action (Listing 10.22), 449
integrated development environments (IDEs), 10
integration tests, 93
See also Tests
interpolation, 136–137
string, 115, 133, 142, 162, 209
IRC client, 14n10
iTerm, 11

JavaScript
adding Bootstrap to application.js
(Listing 8.25), 358
to create following relationship (Listing 11.39), 529
to destroy following relationship
(Listing 11.40), 529
unobtrusive, 525
JavaScript Embedded Ruby (JS-ERb), 528–529
to create a following relationship
(Listing 11.39), 529
join method, 145, 534
JS-ERb. See JavaScript Embedded Ruby
(JS-ERb)
Katz, Yehuda, 364, 543

layout, filling in, 167–219
adding structure, 167–186
asset pipeline, Sass and, 187–190
conclusion, 215–216
exercises, 217–219
layout links, 197–211
stylesheets and, improving with Sass, 190–197
user signup, 211–215
layout files
duplication eliminated with, 103, 111–113
sample application site layout (Listing 3.25), 112
sample application site layout (Listing 4.1), 130
sample application site layout (Listing 4.3), 132
site layout with added structure (Listing 5.1), 169
layout links, 197–211
changing for signed-in users (Listing 8.24), 357–358
named routes, 205–207
Rails routes, 202–205
route tests, 200–202
RSpec, 207–211
test for links on layout (Listing 5.36), 218
layout links, changing, 355–359
adding Bootstrap JavaScript library to
application.js (Listing 8.25), 358
layout links, changing (continued)
  the signed in? helper method (Listing 8.23), 356
  for signed-in users (Listing 8.24), 357–358
  signin success and, 355–359
length validations, 243–244
  adding for name attribute (Listing 6.15), 244
  constraining micropost characters (Listing 2.9), 67
  test name for (Listing 6.14), 244
Linux, 13–14
Linux Mint, 14
Linux Ubuntu, 14
lists, unordered, 172
literal constructor, 153–154
literal strings, 135
Loeffler, David, 10
log, development, 231–232
log files, ignoring, 29
M
Macintosh OS X, 11
MacVim, 28
magic columns, 226, 232
manifest files in asset pipeline, 188–189
map method, 147–148, 533
mapping for site links, 198
mass assignment
  attr_accessible used to prevent, 230, 416
  invalid, ensuring Rails throws errors on (Listing 10.6), 436
memoization, 354n7
Merb, merger with Rails, 4
message passing in Ruby, objects and, 138–141
  messaging, 541
method chaining, 139, 421
method definitions, 141
micropost associations, 433–438
Micropost class with inheritance (Listing 2.13), 71
micropost data model, 48–49
micropost migration (Listing 10.1), 430
Micropost model, 429–444, 431
  accessible attributes and first validation, 432–433
  basic model, 430–432
  content validations, 443–444
  initial Micropost spec (Listing 10.2), 431
micropost migration (Listing 10.1), 430
refinements, 439–443
tests for (Listing 10.17), 443–444
tests for validity of new micropost
  (Listing 10.3), 432
user has many microposts (Listing 10.11), 438
user/micropost associations, 433–438
validation for user (Listing 10.4), 433
validations (Listing 10.18), 444
microposts
  adding to sample data (Listing 10.23), 451
  CSS for (Listing 10.24), 452–453
  destroying along with user (Listing 10.16), 443
  form partial for creating (Listing 10.33), 463
  ordering with default scope (Listing 10.14), 441
  partial for showing single micropost (Listing 10.21), 449
  sample microposts, 450–454
  summary of user/micropost association methods, 434t
testing that microposts are destroyed when users are (Listing 10.15), 442
testing the order of a user’s microposts (Listing 10.13), 440–441
Microposts, manipulating, 454–479
  access control, 456–459
  creating microposts, 459–467
  destroying microposts, 475–479
  micropost pagination links, 455
  proto-feed, 467–475
Microposts, showing, 445–454
  profile page with microposts, mockup of, 445
  sample microposts, 450–454
  user show page, augmenting, 446–450
Microposts controller
  create action (Listing 10.30), 461
  destroy action (Listing 10.49), 477–478
  in schematic form (Listing 2.8), 65–66
tests for destroy action (Listing 10.48), 477
MicropostsController class with inheritance (Listing 2.15), 72
Microposts resource, 63–75
  access control, 456–459
  associations, 68–70
  demo app, deploying, 73–74
  error messages for failed micropost creation, 69
  inheritance hierarchies, 70–73, 73
length validations, 243–244
micropost belonging to user (Listing 2.11), 69
between microposts and users, 70
microtour, 63–66
Rails routes with new rule (Listing 2.7), 65
RESTful routes provided by, 65t
routes for (Listing 10.25), 455
user has many microposts (Listing 2.10), 68
validations, 66–68
microtour, 63–66
migration
  to add boolean admin attribute to users
    (Listing 9.40), 415
micropost (Listing 10.1), 430
password, 256–257
Rake used in, 50
user model, 223–228
for User model (to create users table)
  (Listing 6.2), 225
mockups, 167–168
model annotation in model file, 228–230
model file, 228–230
  accessible attributes, 230
  model annotation, 228–230
modeling demo microposts, 48–49
modeling demo users, 47–48
modeling users, 221–269
  conclusion, 267
  exercises, 268–269
  passwords, 254–267
  user model, 222–236
  user validations, 236–254
model-view-controller (MVC), 25–27
  in action, 56–62
  in Rails, diagram of, 57
  schematic representation of, 26
motivation
  in Ruby, 129–133
  in status feed, 529–532
MVC. See Model-view-controller (MVC)

name attribute
  adding length validation for (Listing 6.15), 244
  failing test for validation of (Listing 6.11), 241
  validating presence of (Listing 6.9), 240
named routes
  footer partial with links (Listing 5.25), 206
  header partial with links (Listing 5.24),
    205–206
namespaces, 403
nested hashes (Listing 4.6), 151
nesting, 190–192
newline, 105n11
new status feed, 538–539
nil, 136
non-nil current user, 356

O
objects and message passing, in Ruby, 138–141
OS X. See Macintosh OS X

P
PagesController. See StaticPages controller
pagination, for showing all users, 404–410
  paginating users in index action (Listing 9.35),
    409
  tests for pagination (Listing 9.33), 407–408
pagination links, micropost, 455
palindrome? method, 155–156, 158
Paperclip gem, 287n8
partial refactoring, for showing all users, 410–412
  partials, 181–186
  adding CSS for site footer (Listing 5.13),
    185–186
  for displaying form submission error messages
    (Listing 7.23), 309
  for HTML shim (Listing 5.9), 183
  for the site footer (Listing 5.11), 184
  for the site header (Listing 5.10), 184
  site layout with footer partial (Listing 5.12),
    185
  site layout with partials for stylesheets and
    header (Listing 5.8), 182–183
  updating error-messages (Listing 10.35), 464
passwords, 254–267
  adding bcrypt-ruby to Gemfile (Listing 6.24),
    255
  and confirmation, 257–260
  creating a user, 265–267
  encrypted, 255–257
  ensuring that User object has password_digest
    column (Listing 6.25), 256
passwords (continued)
migration, 256–257
migration to add password_digest column to users table (Listing 6.26), 256
reminders, 540, 541
secure, adding, 254–260
test for password and password confirmation (Listing 6.28), 259–260
testing for password and password_confirmation attributes (Listing 6.27), 257
user authentication, 260–263
user has secure password, 263–265
User model with added password_digest attribute, 255
See also Authenticate method
Patterns of Enterprise Application Architecture (Fowler), 222n1
PeepCode, 6, 543
pending spec, 237
persistence, 223
Phusion Passenger, 39
Pik project, 13
pluralize text helper, 310
PostgreSQLn, 46–47, 115, 117, 223, 253n15
pound sign. See Hash symbol
preprocessor engines in asset pipeline, 189
presence, validating, 239–243
Preston-Werner, Tom, 286
private keyword, 348
production in asset pipeline, efficiency in, 189–190
profile images, 286, 382
profile links, 332
protected page
mockup of, 385
signin form after trying to access, 388
proto-feed, 467–475
adding feed instance variable to home action (Listing 10.41), 471
adding (empty) @feed items instance variable to create action (Listing 10.45), 474–475
adding status feed to Home page (Listing 10.44), 473
Home page after creating micropost, 474
Home page with, 473
mockup of Home page with, 467
preliminary implementation for micropost status feed (Listing 10.39), 469
single feed item partial (Listing 10.43), 472
status feed partial (Listing 10.42), 472
test for rendering feed on Home page (Listing 10.40), 470–471
tests for (Listing 10.38), 468
public/index.html file, 83
puts method, 136
R
Rails
approach to learning, 4–6
deploying, 39–42
development environment setup, 9–27
environments, 276–277
intermediate-to-advanced resources, 6–7
introduction, 3–9
Merb merger and, 4
Ruby and, importance of, 129–165 (See also Ruby)
running to generate new application (Listing 1.3), 17–18
scaling, 7
version control with Git, 27–39
Rails, installing
Git, installing, 13
Rails, installing (Windows), 13, 16–17
Ruby, installing, 13–15
RubyGems, installing, 15–16
The Rails 3 Way (Fernandez), 6, 82n5, 142n5, 543
RailsCasts, 6, 7, 540, 543
Rails command, 17–19
default .gitignore created by (Listing 1.6), 29 to generate new application (Listing 1.3), 17–18
Rails console, 134
Rails Guides, 6, 189, 202, 228, 506, 543
Rails Machine, 39
Rails root, 8–9
Rails routes, 202–205
adding mapping for the (Listing 5.23), 204
adding Users resource to (Listing 7.3), 279 commented-out hint for defining (Listing 5.22), 204 with new rule for Microposts resource (Listing 2.7), 65
with rule for Users resource (Listing 2.2), 58
for static pages (Listing 5.21), 202
Rails server, 23–25
*The Rails 3 Way* (Fernandez), 6, 142n5, 543
Rails Tutorial help page, 9n6
Rake, 50, 51
  task for populating database with sample users
  (Listing 9.30), 403–404
ranges, 145–146
README file
  Git, 34–35, 35
  improved, formatted with Markdown (Listing),
  39
  improved, for sample app (Listing 3.2), 80
  initial, for project at GitHub, 35
  new README file, README.md
    (Listing 1.8), 36
  updating, 80
Red, Green, Refactor, 94
  Green, 100–102
  Red, 99–100
  Refactor, 103
refactoring
  in adding static pages, 103
  compact, of Listing 11.38 (Listing 11.47),
  544
  first attempt at index view (Listing 9.36), 411
  partial, 410–412
  refactored following and followers actions
    (Listing 11.30), 518
refinements in Micropost model, 439–443
default scope, 440–441
dependent: destroy, 441–443
regular expression (regex), 246
relationship model, 484–503, 491
  adding belongs to associations to (Listing 11.6),
  494
  adding indices for relationships table
    (Listing 11.1), 490
  adding User model followed_users association
    (Listing 11.10), 496
  followed users, 495–500
  of followed users through user relationships,
  489
  followers, 500–503
  following? and follow! utility methods
    (Listing 11.12), 498
  implementing user.followers using reverse
  relationships (Listing 11.16), 502
  implementing user/relationships has_many
  association (Listing 11.4), 493
  problem with, 485–491
  for reverse relationships, 500–503
  test for unfollowing a user (Listing 11.12), 499
  test for user.followed_users attribute
    (Listing 11.9), 496
  testing for reverse relationships (Listing 11.15),
  501
  testing for user/relationships attribute
    (Listing 11.3), 492
  testing Relationship creation and attributes
    (Listing 11.2), 491–492
  testing Relationship model validations
    (Listing 11.7), 495
  testing user/relationships belongs_to association
    (Listing 11.5), 494
  tests for “following” utility methods
    (Listing 11.11), 497
  unfollowing user by destroying user
  relationship (Listing 11.14), 499–500
  for user followers using reverse relationship
  model, 500
  user/relationship associations, 491–494
  validations, 495
relationships attribute, 492
Relationships controller
  Ajax requests in, responding to (Listing 11.38),
  527
  follow and unfollow buttons (Listing 11.34),
  523–524
  responses to Ajax requests, tests for
    (Listing 11.37), 526
  reload method, 383
remember token, 344
  added to cookies, 379
  before_save callback to create (Listing 8.18),
  348–349
  cookie in local browser, 360
  current user found by using (Listing 8.22), 353
  first test for (Listing 8.15), 345
  migration to add to users table (Listing 8.16),
  346
  removed from cookies, 363
  test for valid (nonblank) (Listing 8.17), 347
remember token (continued)
User model with added remember_token attribute, 345
render, 183
replies, 541
repository setup, 28–30
request specs. See Tests
resources
advanced Rails, 4, 6
guide to further, 542–543
REST API, 542
REST architecture, 45, 59, 65, 86, 90
RESTful routes
provided by Microposts resource, 65
provided by Users resource, 65
reverse relationships, 500–503
followers using reverse relationship model, 500
implementing user.followers using reverse relationships (Listing 11.16), 502
testing for reverse relationships (Listing 11.15), 501
root, 8–9
routes in layout links
named, 205–207
Rails, 202–205
tests, 200–202
RSpec
adding helper method and custom RSpec matcher (Listing 8.34), 371
Cucumber equivalent, 368–371
custom matchers, 368–371
layout links, 207–211
request specs, 93, 368
RSS feed, 542
Rubular, 247, 248
Ruby, 129–165
comments, 134–135
conclusion, 164
exercises, 164–165
gems, 14
gemsets, 14–15
installing, 13–15
method definitions, 141
motivation, 129–133
objects and message passing, 138–141
strings, 135–138
title helper, 142
Ruby classes. See Classes
Ruby data structures, 142–153
arrays, 142–145
blocks, 146–148
cascading style sheets, 152–153
hashes and symbols, 148–152
ranges, 145–146
RubyGems, installing, 15–16
Ruby JavaScript (RJS)
to create following relationship (Listing 11.39), 529
to destroy following relationship (Listing 11.40), 529
RubyMine, 10
Ruby on Rails. See Rails
Ruby Version Manager (RVM), 8, 13, 118
The Ruby Way (Fulton), 6, 129, 543
S
Safari, 11–12, 89, 170
sample application, extensions to, 540–542
follower notifications, 541
messaging, 541
password reminders, 541
replies, 541
REST API, 542
RSS feed, 542
search, 542
signup confirmation, 541
sample users, showing all, 403–404
sandbox, 231, 252, 265
Sass, 187–197
asset pipeline and, 187–190
improving stylesheets with, 190–197
savel!, 497
scaffolding, 2–3
scaling Rails, 7
scope, 440–441
screencasts, 538, 542
SCSS
converting to CSS, 192
error messages styled with, 311
for Home page sidebar (Listing 11.22), 510
initial SCSS file converted to use nesting and variables (Listing 5.15), 195–197
rewriting, 193–194
Sass supported by, 190
for styling user show page, including sidebar
  (Listing 7.15), 290–291
search, 542
Secure Sockets Layer (SSL), 318
deploying production with, in signup success,
  317–321
Seguin, Wayne E., 13, 14
self, 157, 348
session hijacking attack, 318, 351
sessions
  authentication and, 325–326
  defined, 325–326
  destroying a session (user signout)
    (Listing 8.29), 362
  preliminary version of sessions create action
    (Listing 8.9), 337
  sessions create action with friendly forwarding
    (Listing 9.20), 395
  signin failure and, 325–326
  sign out method in Sessions helper module
    (Listing 8.30), 363
Sessions controller
  adding resource to get standard RESTful
    actions for sessions (Listing 8.2), 328
  completed Sessions controller create action (not
    yet working) (Listing 8.13), 343
  create action for, 326, 336–338, 343, 395
  signin failure and, 326–329
  tests for new session action and view
    (Listing 8.1), 327
short-circuit evaluation, 355
showing microposts. See Microposts, showing
  sidebar
    partial for the user info sidebar (Listing 10.32),
      462
  SCSS for Home page (Listing 11.22), 510
in SCSS for styling user show page
  (Listing 7.15), 290–291
in sign up, 288–291
signed in? helper method (Listing 8.23), 356
signed-in users
  authorization of, 386–389
  requiring, 386–389
signin, 325–372
  conclusion, 371–372
  Cucumber, signin tests using, 363–371
  exercises, 372
signin failure, 325–343
  flash message, rendering with, 339–343
  reviewing from submission, 336–338
  sessions, 325–326
  Sessions controller, 326–329
  signin form, 333–336, 335
  signin tests, 330–333
  signin form, 333–336, 335
  code for (Listing 8.7), 334
  HTML for signin form produced by Listing 8.7
    (Listing 8.8), 335
  initial failed signin, with create as in
    Listing 8.9., 336
  signin failure and, 333–336
  signing out, 361–363
    destroying a session (user signout)
      (Listing 8.29), 362
    sign out method in Sessions helper module
      (Listing 8.30), 363
  signin method, signin success and, 349–351
  signin success, 343–363
    current user, 351–355
    layout links, changing, 355–359
    remembering user signin status, 343–349
    signing out, 361–363
    sign_in method, 349–351
    signin upon signup, 359–361
  signin tests
    signin failure and, 330–333
    using Cucumber, 363–371
  signin upon signup, 359–361
sign up, 271–324
  conclusion, 321
  exercises, 321–324
  failure in (See Signup failure)
    Rails environments in, 276–277
    showing users, 271–291
    success in (See Signup success)
  signup confirmation, 541
signup failure, 303–312, 306
  apartial for displaying form submission
    error messages (Listing 7.23),
      309
  code to display error messages on signup form
    (Listing 7.23), 309
  create action that can handle (but not success)
    (Listing 7.21), 305
signup failure (continued)
  CSS for styling error messages (Listing 7.24), 311
d debug information, 307
 mockup of signup failure page, 304
 signup error messages, 308–312, 312
 working form, 303–308
 signup form, 292–303
  adding @user variable to the new action
 (Listing 7.18), 299
 CSS for (Listing 7.19), 300
 filled-in form with text and password fields, 302
 form for, using, 297–300
 form to sign up new users (Listing 7.17), 298
 HTML, 301–303
 HTML for form in figure 7.12 (Listing 7.20), 301
 for new users, 300
 tests for signing up users (Listing 7.16), 296–297
tests for user signup, 293–297
 using form_for, 297–300
 signup page
  initial (stub) (Listing 5.33), 214
 linking the button to (Listing 5.34), 215
 route for (Listing 5.32), 214
 signing in user upon signup (Listing 8.27), 361
 signin upon signup, 359–361
 testing that newly signed-up users are also signed in (Listing 8.26), 360–361
 Users controller, 212
 signup success, 312–321
 deploying production with SSL, 317–321
 finished signup form, 313–315
 first signup, 317
 flash, 315–319
 mockup of, 314
 signup URI, in user signup, 213–215
 site navigation in filling in layout, 169–175
 Home page with link to signup page
 (Listing 5.2), 173
 site layout with added structure (Listing 5.1), 169
 skeleton for a shuffle method attached to the
 String class (Listing 4.11), 165
 skeleton for a string shuffle function
 (Listing 4.10), 164
 slightly dynamic pages, 103–113
duplication, eliminating with layouts, 103, 111–113
 Embedded Ruby, 108–111
 instance variables and Embedded Ruby, 162
 passing title tests, 106–108
 testing a title change, 103–107
 testing title page, 103–106
 spike, 93
 split method, 143
 Spork, 123–127
  adding environment loading to Spork.prefork
 block (Listing 3.36), 124
 configuring RSpec to automatically use
 (Listing 3.37), 125
 Gemfile for sample app (Listing 3.35), 123
 Guardfile updated for Spork (Listing 3.38), 126
 Guard with Spork, 126–127
 speeding up tests with, 123–127
 SQL injection, 470
 SQLite Database Browser, 226, 227, 266
 Stack Overflow, 301, 492n4
 staging area, 30
 static pages, 77–128
 conclusion, 114
 exercises, 114–117
 test-driven development, 93–99
 testing, 93–103
 See also Slightly dynamic pages
 static pages, adding, 99–103
 green, 100–102
 red, 99–100
 refactor, 103
 static pages, advanced setup, 117–128
 bundle exec, eliminating, 118–119
 Guard, automated tests with, 120–122
 Spork, speeding up tests with, 123–127
 Sublime Text, tests inside, 127–128
 static pages, making, 82–92
 with Rails, 85–92
 truly static pages, 82–85
 undoing things, 87–88
 StaticPages controller
 with about action (Listing 3.15), 101
 generating (Listing 3.4), 86
 inheritance hierarchy for, 160
made by Listing 3.4 (Listing 3.6), 91
routes for home and help actions in
(Listing 3.5), 88
spec with base title (Listing 3.29), 115–116
spec with title tests (Listing 3.18), 105
stats, 505–514
adding follower stats to Home page
(Listing 11.21), 509
adding follow form and follower stats to user
profile page (Listing 11.27), 513
Home page with follow stats, 511
mockup of stats partial, 505
a partial for displaying follower stats
(Listing 11.20), 508
SCSS for Home page sidebar (Listing 11.22),
510
testing following/follower statistics on the
Home page (Listing 11.19), 507
stats form, 505–514
status command, in Git, 30
status feed, 529–539
adding completed feed to User model
(Listing 11.42), 532
final implementation of from_users_followed_by
(Listing 11.45), 537–538
final tests for (Listing 11.41), 531–532
first cut at from_users_followed_by
(Listing 11.43), 535
first feed implementation, 532–535
home action with paginated feed
(Listing 11.46), 538
Home page with working status feed, 539
improving from_users_followed_by
(Listing 11.44), 536
mockup of a user’s Home page with, 530
motivation and strategy, 529–532
new, 538–539
partial for a single feed item (Listing 10.43),
472
preliminary implementation for micropost
(Listing 10.39), 469
subselects, 535–538
for user following users, 531
strategy in status feed, 529–532
string interpolation, 115, 133, 142, 162,
209
string literals, 135
strings
double-quoted, 137–138
printing, 136–137
in Ruby, 135–138
single-quoted, 137–138
structure in filling in layout, 167–186
bootstrap and custom CSS, 175–186
partials, 181–186
site navigation, 169–175
stylesheets. See Cascading style sheets (CSS)
Sublime Text, tests inside, 127–128
Sublime Text 2, 10, 16, 127
subselects in status feed, 535–538
sudo, 8
superclass method, 155
symbols, 148–152
system setups, 27
T
TDD. See Test-driven development (TDD)
terminals, 11
ternary operator, 481, 482
test-driven development (TDD), 5
Green, 100–102
Red, 99–100
Red, Green, Refactor, 94
Refactor, 103
Spork, 123–127
in testing static pages, 93–99
testing tools, 93
tests
for admin attribute (Listing 9.39), 414
for authorization of following and followers
pages (Listing 11.28), 516–517
automated tests with Guard, 120–122
for Contact page (Listing 5.16), 198
for creating microposts (Listing 10.29), 460
for delete links (Listing 9.44), 418–419
for destroy action in Microposts controller
(Listing 10.48), 477
for email format validation (Listing 6.16),
245–246
for follow and unfollow buttons
(Listing 11.32), 521–522
for “following” utility methods (Listing 11.11),
497
for friendly forwarding (Listing 9.17), 393
tests (continued)
  for full_title helper (Listing 5.37), 219
  Guard, automated tests with, 120–122
  for index page (Listing 9.23), 398–399
  integration tests, 93
  for Micropost model (Listing 10.17), 443–444
  for Microposts controller destroy action
    (Listing 10.48), 477
  for micropost’s user association (Listing 10.8),
    437
  for new session action and view (Listing 8.1),
    327
  for pagination (Listing 9.33), 407–408
  passing title, 106–108
  for post-save behavior in (Listing 7.32), 323
  for proto-feed (Listing 10.38), 468
  for Relationships controller (Listing 11.37),
    526
  for relationships controller authorization
    (Listing 11.33), 522–523
  for Relationships controller authorization
    (Listing 11.33), 522–523
  for responses to Ajax requests (Listing 11.37),
    526
  for reverse relationships (Listing 11.15),
    501
  for routes in layout links, 200–202
  for showing microposts on user show page
    (Listing 10.19), 446
  for signin, using Cucumber, 363–371
  for signin failure, 330–333
  for signing up users (Listing 7.16), 296–297
  for signin tests using Capybara, 294, 330
  for signin tests using Cucumber, 363–371
  spec with title tests (Listing 3.18), 105
  speeding up with Spork, 123–127
  static pages (Listing 5.27), 210
  for static pages, 93–99
  for static pages (Listing 5.27), 210
  for status feed, final (Listing 11.41), 531–532
  Sublime Text, tests inside, 127–128
  for title change, 103–106
  title test (Listing 3.17), 104
  user, initial, 236–239
  for user index page (Listing 9.23), 398–399
  for user show page (Listing 7.9), 285
  for user signup, 293–297
  for user’s microposts attribute (Listing 10.9),
    437–438
  for user update action (Listing 9.9), 383
  for user validations, initial, 236–239
  for utility methods, (Listing 11.11), 497
  for validity of new micropost (Listing 10.3),
    432
  text editors, 10, II
  TextMate, 10, 28, 105n12
  time helpers, 350
  timestamps, 225
  title change
    passing title tests, 106–107
    testing, 103–106
  title helper, 142
    tests for full_title helper (Listing 5.37), 219
  title test (Listing 3.17), 104
  toggle method, 414
  tools, learning, 12
  Torvalds, Linus, 27

U
  underscore method, 493n5
  unfollow and follow buttons. See Follow and
    unfollow buttons
  unfollow form, using Ajax (Listing 11.36),
    525
  unfollowing a user
    by destroying a user relationship
      (Listing 11.14), 499–500
    test for (Listing 11.13), 499
  uniqueness, validating, 249–254
  Unix-style command line, 7
  unobtrusive JavaScript, 525
  unordered list tag, 172
  update action. See User update action
  updating users, 373–384
    edit form, 374–380
    successful edits, 382–384
    unsuccessful edits, 380–382
  URIs
    adding to users link (Listing 9.28), 401–402
    correspondence between pages and URIs for
      Users resource, 527
    defined, 2n1
    signup, in user signup, 213–215
    test for “Users” link (Listing 9.27), 401
URLs
  correspondence between pages and Users resource, 52t
  defined, 2n1

user
  administrative, 413–417
  creating, 265–267
  current user? method (Listing 9.16), 392
  destroying, 499–500
  has secure password, 263–265
  new user view with partial (Listing 9.51), 425
  paginating, 404–410
  requiring signed-in users, 386–389
  requiring the right user, 390–392
  sample users, 403–404
  showing, 271–291
  signin status, remembering, 343–349
  stub view for showing user information (Listing 7.4), 280
  summary of user/micropost association methods/updating, 434t
  tests, initial, 236–239

user authentication. See Authentication

user authorization. See Authorization

user class, 161–163

User class with inheritance (Listing 2.12), 71

user data model, 47–48

user edit form
  adding test for Settings link (Listing 9.5), 378
  HTML for (Listing 9.2), 377
  mockup of, 374
  partial for new and edit form fields (Listing 9.50), 425
  tests for user update action (Listing 9.9), 383
  updating error-messages partial from Listing 7.23 to work with other objects (Listing 10.35), 464

user update action (Listing 9.10), 384

user.followers method, 500

user has_many microposts (Listing 10.11), 438

user index, 396–403
  adding URI to users link (Listing 9.28), 401–402
  CSS for (Listing 9.26), 400
  first refactoring attempt at index view (Listing 9.36), 411
  including will paginate in Gemfile (Listing 9.31), 405
  mockup of, 397
  paginating users in index action (Listing 9.35), 409
  pagination, 404–410
  with pagination (Listing 9.34), 408
  partial refactoring, 410–412
  partial to render single user (Listing 9.37), 412
  refactored (Listing 9.38), 412
  requiring signed-in user for index action (Listing 9.22), 398
  for showing all users, 396–403
  test for “Users” link URI (Listing 9.27), 401
  testing that index action is protected (Listing 9.21), 396–397
  tests for pagination (Listing 9.33), 407–408
  user index action (Listing 9.24), 399
  user index view (Listing 9.25), 400

user info sidebar, partial for (Listing 10.32), 462

user index page
  page 2 of, 411
  tests for (Listing 9.23), 398–399

users with 100 sample users, 405
users with only one user, 402
users with pagination, 410

User microposts, 429–482
  conclusion, 479–480
  exercises, 480–482
  manipulating, 454–479
  model, 429–444, 431
  resources, 63–74
  showing, 445–454

User model, 222–236
  accessible attributes, 230
  with added password_digest attribute, 255
  adding annotate gem to Gemfile (Listing 6.4), 229
User model (continued)
annotated User model (Listing 6.5), 229–230
brand new (Listing 6.3), 228
for demo application (Listing 2.5), 61
generating (Listing 6.1), 224
making name and email attributes accessible
(Listing 6.6), 230
migration for (to create a users table)
(Listing 6.2), 225
migrations, 223–228
model file, 228–230
user objects, 230–236
user objects
creating, 230–233
finding, 233–235
updating, 235–236
user profile page, mockup of, 445
user/relationship associations, 491–494
implementing has_many association
(Listing 11.4), 493
See also Relationship model
users, deleting, 413–422
administrative users, 413–417
destroy action, 417–422
users, following, 483–544
conclusion, 539–543
current user’s profile, 484
exercises, 543–544
finding a user to follow, 485
Home page with status feed and incremented following count, 488
implementation of user following, 488
model of followed users through user relationships, 489
profile of user to follow, with follow button, 486
profile with unfollow button and incremented followers count, 487
resources, guide to further, 542–543
sample application, extensions to, 540–542
status feed, 529–539
test for unfollowing (Listing 11.13), 499
web interface for, 503–529
See also Relationship model
users, showing all, 396–412
pagination, 404–410
partial refactoring, 410–412
sample users, 403–404
user index, 396–403
users, showing in sign up, 271–291
debug information, 272–276
Gravatar, 286–291
Rails environments, 276–277
sidebar, 288–291
user show page, testing, 282–286
Users resource, 278–281
users, updating, 373–385
edit form, 374–380
successful edits, 382–384
unsuccessful edits, 380–382
Users controller, 212
adding followed_users and followers
actions to Users controller (Listing 11.18), 506
class with inheritance (Listing 2.14), 72
create action for, 425, 459
initial, with new action (Listing 5.29), 212
in schematic form (Listing 2.3), 58
with show action (Listing 7.5), 281
testing the user show page with factories,
282–286
in user signup, 212
user show page, 53, 282–286
adding sidebar to user show view (Listing 7.14), 290
adding title and heading for user profile page
(Listing 7.10), 285
defining gravatar_for helper method
(Listing 7.13), 288
factories to simulate User model objects
(Listing 7.8), 284
Factory Girl added to Gemfile (Listing 7.7), 284
in Microposts, augmenting, 446–450
recap of initial User pages spec (Listing 7.6), 282–283
reddefining BCrypt cost factor in test
environment (Listing 7.11), 286
SCSS for styling, including sidebar
(Listing 7.15), 290–291
tests for (Listing 7.9), 285
user profile page /users/1 with default
Gravatar, 289
at /users/1 after adding Users resource, 282
Users controller with show action (Listing 7.5), 281
user show page /users/1 with sidebar and CSS, 291
user show page with custom Gravatar, 289
user show view with name and (Listing 7.12), 287
user signup, 211–215
  adding flash message to (Listing 7.27), 317
errors, updating rendering of (Listing 10.36), 465
signup URI, 213–215
tests for, 293–297
users controller, 212
Users resource, 49–63
  adding to the routes file (Listing 7.3), 279
correspondence between pages and URLs, 52
MVC in action, 56–62
Rails routes with rule for (Listing 2.2), 58
RESTful routes provided by, 65
in sign up, 278–281
weaknesses of, 62–63
Users resource tour, 51–56
user update action (Listing 9.10), 384
  initial (Listing 9.8), 381
tests for (Listing 9.9), 383
user validations, 236–254
  format, 245–248
  length, 243–244
  presence, 239–243
  uniqueness, 249–254
user tests, initial, 236–239

v
validations
  commenting out a validation to ensure a failing test (Listing 6.10), 241
e-mail format with regular expression (Listing 6.17), 246
  format, 245–248
initial user pages spec (Listing 7.6), 282
length, 243–244
length, adding for name attribute (Listing 6.15), 244
Microposts resource, 66–68
migration for enforcing email uniqueness (Listing 6.22), 252
of name attribute, failing test for (Listing 6.11), 241
for password attribute (Listing 6.27), 257
practically blank default User spec (Listing 6.7), 237
of presence, 239–243
of presence of name and email attributes (Listing 6.13), 243
of presence of name attribute (Listing 6.9), 240
Relationship data model, 495
Relationship model, adding (Listing 11.8), 495
in relationship model, 495
test for name length (Listing 6.14), 244
test for presence of email attribute (Listing 6.12), 243
test for rejection of duplicate email addresses (Listing 6.18), 249
test for rejection of duplicate email addresses, insensitive to case (Listing 6.20), 250
testing Relationship model validations (Listing 11.7), 495
tests for email format validation (Listing 6.16), 245–246
of uniqueness, 249–254
of uniqueness of email addresses (Listing 6.19), 250
of uniqueness of email addresses, ignoring case (Listing 6.21), 251
user, 236–254
validations, Micropost model, 432–444
  accessible attributes and first, 432–433
  content validations, 443–444
  first validation, accessible attributes and, 432–433
tests for validity of new micropost (Listing 10.3), 432
  for user (Listing 10.4), 433
variables in improving stylesheets, 193–197
version control. See Git
Vim, 10, 12, 29, 82
virtual attributes, 257

W
web interface for following users, 503–529
  adding following/follower relationships to sample data (Listing 11.17), 503–504
follow button with Ajax, working, 524–529
web interface for following users (continued)
follow form, 505–514
following and followers pages, 515–519
follow/unfollow buttons, working, 519–524
sample following data, 503–505
stats, 505–514
Webrat, 79n1
*The Well-Grounded Rubyist* (Black), 6, 129, 543
will paginate method, 408
Windows, 11

wireframes, 167
wrapping long words, helper for (Listing 10.50), 481

Y
YAML, 276n3

Z
zero-offset, 143