Addison-Wesley Professional Ruby Series

RUBY ON RAILS™ TUTORIAL

THIRD EDITION

LEARN WEB DEVELOPMENT WITH RAILS

MICHAEL HARTL

FOREWORDS BY DEREK SIVERS AND OBIE FERNANDEZ

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

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 of CD Baby
Currently: founder of 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

"For the self-motivated reader who responds well to the 'learn by doing' method and is prepared to put in the effort, then this comes highly recommended."

—Ian Elliot, reviewer, I Programmer

"*Ruby on Rails*™ *Tutorial* is a lot of work but if you're careful and patient, you'll learn a lot."

—Jason Shen, tech entrepreneur, blogger at The Art of Ass-Kicking

"Michael Hartl's *Ruby on Rails*™ *Tutorial* seamlessly taught me about not only Ruby on Rails, but also the underlying Ruby language, HTML, CSS, a bit of JavaScript, and even some SQL—but most importantly it showed me how to build a web application (Twitter) in a short amount of time."

—Mattan Griffel, co-founder & CEO of One Month

"Although I'm a Python/Django developer by trade, I can't stress enough how much this book has helped me. As an undergraduate, completely detached from industry, this book showed me how to use version control, how to write tests, and, most importantly—despite the steep learning curve for setting up and getting stuff running—how the end-result of perseverance is extremely gratifying. It made me fall in love with technology all over again. This is the book I direct all my friends to who want to start learning programming/building stuff. Thank you Michael!"

—Prakhar Srivastav, software engineer, Xcite.com, Kuwait

"It doesn't matter what you think you will be developing with in the future or what the framework *du jour* is; if you want to learn how to build something, there is no better place to start than with this tutorial. And for all the 'non-technical' people out there who want to see their ideas come to life, who are considering hiring contractors, paying for a class, or 'founder dating' in the search for a technical co-founder: stop. Take a step back. Forget about your idea for a short while and immerse yourself in this tutorial to learn what it takes to put something together. You and your software-related projects will be better for it."

—Vincent C., entrepreneur and developer

"It has to be the best-written book of its type I've ever seen, and I can't recommend it enough."

—Daniel Hollands, administrator of Birmingham.IO

"For those wanting to learn Ruby on Rails, Hartl's *Ruby on Rails*™ *Tutorial* is (in my opinion) the best way to do it."

—David Young, software developer and author at deepinthecode.com

"This is a great tutorial for a lot of reasons, because aside from just teaching Rails, Hartl is also teaching good development practices."

—Michael Denomy, full-stack web developer

"Without a doubt, the best way I learned Ruby on Rails was by building an actual working app. I used Michael Hartl's *Ruby on Rails*™ *Tutorial*, which showed me how to get a very basic Twitter-like app up and running from scratch. I cannot recommend this tutorial enough; getting something up and going fast was key; it beats memorization by a mile."

—James Fend, serial entrepreneur, JamesFend.com

"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, [Hartl's] 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

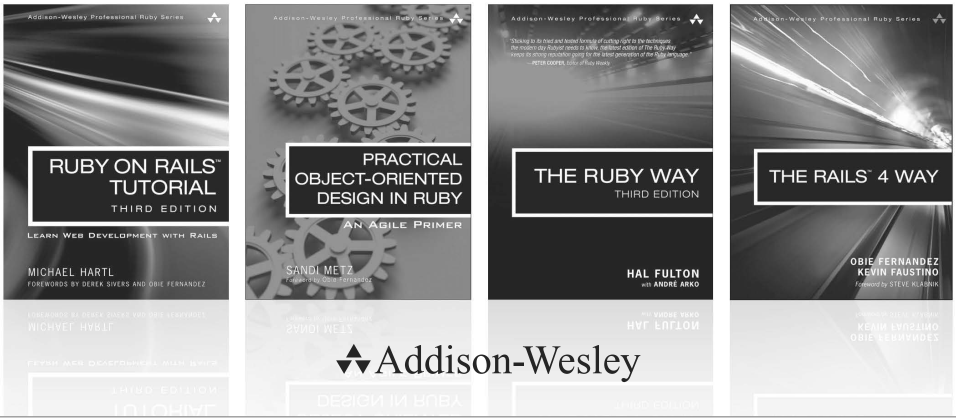
Ruby on Rails™ Tutorial

Third Edition

Addison-Wesley

Professional Ruby Series

Obie Fernandez, Series Editor



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.

Make sure to connect with us!
informit.com/socialconnect

informIT.com the trusted technology learning source | Addison-Wesley | Safari Books Online

ALWAYS LEARNING PEARSON

Ruby on Rails™ Tutorial

Learn Web Development with Rails

Third Edition

Michael Hartl

Addison-Wesley

New York • Boston • Indianapolis • San Francisco
Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382–3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact international@pearsoned.com.

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Hartl, Michael, author.
[Ruby on rails 3 tutorial]
Ruby on rails tutorial : learn web development with rails / Michael Hartl.—Third edition.
pages cm
Includes index.
ISBN 978-0-13-407770-3 (pbk. : alk. paper)—ISBN 0-13-407770-9 (pbk. : alk. paper)
1. Ruby on rails (Electronic resource) 2. Web site development.
3. Ruby (Computer program language) I. Title.
TK5105.8885.R83H37 2015
006.7—dc23

2014049130

Copyright © 2015 Michael Hartl

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, 200 Old Tappan Road, Old Tappan, New Jersey 07675, or you may fax your request to (201) 236-3290.

The source code in *Ruby on Rails™ Tutorial, Third Edition*, is released under the MIT License.

ISBN 13: 978-0-13-407770-3
ISBN 10: 0-13-407770-9
Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.
First printing, April 2015

Editor-in-Chief
Mark L. Taub

Acquisitions Editor
Debra Williams Cauley

Managing Editor
John Fuller

Full-Service Production Manager
Julie B. Nahil

Copy Editor
Jill E. Hobbs

Indexer
Richard Evans

Proofreader
Andrea Fox

Reviewer
Will Sommers

Assistant Editor
Kim Boedigheimer

Cover Designer
Chuti Prasertsith

Compositor
LaurelTech

Contents

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

Chapter 1 From Zero to Deploy 1

- 1.1 Introduction 4
  - 1.1.1 Prerequisites 5
  - 1.1.2 Conventions in This Book 6
- 1.2 Up and Running 8
  - 1.2.1 Development Environment 9
  - 1.2.2 Installing Rails 11
- 1.3 The First Application 12
  - 1.3.1 Bundler 15
  - 1.3.2 `rails server` 21
  - 1.3.3 Model–View–Controller 23
  - 1.3.4 Hello, World! 25
- 1.4 Version Control with Git 29
  - 1.4.1 Installation and Setup 30
  - 1.4.2 What Are the Benefits of Using Git? 32
  - 1.4.3 Bitbucket 33
  - 1.4.4 Branch, Edit, Commit, Merge 36

1.5 Deploying 41
1.5.1 Heroku Setup 43
1.5.2 Heroku Deployment, Step 1 45
1.5.3 Heroku Deployment, Step 2 45
1.5.4 Heroku Commands 45
1.6 Conclusion 47
1.6.1 What We Learned in This Chapter 47
1.7 Exercises 48

Chapter 2 A Toy App 51

2.1 Planning the Application 52
2.1.1 A Toy Model for Users 54
2.1.2 A Toy Model for Microposts 55
2.2 The Users Resource 55
2.2.1 A User Tour 58
2.2.2 MVC in Action 60
2.2.3 Weaknesses of This Users Resource 70
2.3 The Microposts Resource 71
2.3.1 A Micropost Microtour 71
2.3.2 Putting the *Micro* in Microposts 74
2.3.3 A User `has_many` Microposts 76
2.3.4 Inheritance Hierarchies 78
2.3.5 Deploying the Toy App 81
2.4 Conclusion 82
2.4.1 What We Learned in This Chapter 83
2.5 Exercises 84

Chapter 3 Mostly Static Pages 87

3.1 Sample App Setup 87
3.2 Static Pages 91
3.2.1 Generated Static Pages 92
3.2.2 Custom Static Pages 100

3.3 Getting Started with Testing 101
3.3.1 Our First Test 104
3.3.2 Red 106
3.3.3 Green 107
3.3.4 Refactor 110
3.4 Slightly Dynamic Pages 111
3.4.1 Testing Titles (Red) 112
3.4.2 Adding Page Titles (Green) 113
3.4.3 Layouts and Embedded Ruby (Refactor) 116
3.4.4 Setting the Root Route 122
3.5 Conclusion 124
3.5.1 What We Learned in This Chapter 125
3.6 Exercises 125
3.7 Advanced Testing Setup 127
3.7.1 MiniTest Reporters 128
3.7.2 Backtrace Silencer 129
3.7.3 Automated Tests with Guard 129

Chapter 4 Rails-Flavored Ruby 137

4.1 Motivation 137
4.2 Strings and Methods 142
4.2.1 Comments 143
4.2.2 Strings 144
4.2.3 Objects and Message Passing 146
4.2.4 Method Definitions 149
4.2.5 Back to the Title Helper 151
4.3 Other Data Structures 152
4.3.1 Arrays and Ranges 152
4.3.2 Blocks 155
4.3.3 Hashes and Symbols 158
4.3.4 CSS Revisited 162
4.4 Ruby Classes 164
4.4.1 Constructors 165
4.4.2 Class Inheritance 166

4.4.3 Modifying Built-in Classes 169
4.4.4 A Controller Class 170
4.4.5 A User Class 172
4.5 Conclusion 175
4.5.1 What We Learned in This Chapter 175
4.6 Exercises 176

Chapter 5 Filling in the Layout 179

5.1 Adding Some Structure 179
5.1.1 Site Navigation 180
5.1.2 Bootstrap and Custom CSS 187
5.1.3 Partials 194
5.2 Sass and the Asset Pipeline 200
5.2.1 The Asset Pipeline 200
5.2.2 Syntactically Awesome Style Sheets 203
5.3 Layout Links 210
5.3.1 Contact Page 211
5.3.2 Rails Routes 213
5.3.3 Using Named Routes 215
5.3.4 Layout Link Tests 216
5.4 User Sign-up: A First Step 219
5.4.1 Users Controller 220
5.4.2 Sign-up URL 221
5.5 Conclusion 223
5.5.1 What We Learned in This Chapter 224
5.6 Exercises 225

Chapter 6 Modeling Users 227

6.1 User Model 228
6.1.1 Database Migrations 229
6.1.2 The Model File 235
6.1.3 Creating User Objects 235

6.1.4 Finding User Objects 238
6.1.5 Updating User Objects 240
6.2 User Validations 241
6.2.1 A Validity Test 242
6.2.2 Validating Presence 243
6.2.3 Length Validation 247
6.2.4 Format Validation 249
6.2.5 Uniqueness Validation 254
6.3 Adding a Secure Password 261
6.3.1 A Hashed Password 261
6.3.2 User Has Secure Password 264
6.3.3 Minimum Password Length 265
6.3.4 Creating and Authenticating a User 267
6.4 Conclusion 269
6.4.1 What We Learned in This Chapter 270
6.5 Exercises 271

Chapter 7 Sign Up 275

7.1 Showing Users 275
7.1.1 Debug and Rails Environments 276
7.1.2 A Users Resource 282
7.1.3 Debugger 287
7.1.4 A Gravatar Image and a Sidebar 288
7.2 Sign-up Form 293
7.2.1 Using `form_for` 294
7.2.2 Sign-up Form HTML 298
7.3 Unsuccessful Sign-ups 302
7.3.1 A Working Form 302
7.3.2 Strong Parameters 306
7.3.3 Sign-up Error Messages 308
7.3.4 A Test for Invalid Submission 313
7.4 Successful Sign-ups 316
7.4.1 The Finished Sign-up Form 316
7.4.2 The Flash 319

7.4.3 The First Sign-up 322
7.4.4 A Test for Valid Submission 323
7.5 Professional-Grade Deployment 326
7.5.1 SSL in Production 326
7.5.2 Production Web Server 327
7.5.3 Ruby Version Number 329
7.6 Conclusion 330
7.6.1 What We Learned in This Chapter 330
7.7 Exercises 331

Chapter 8 Log In, Log Out 335

8.1 Sessions 336
8.1.1 Sessions Controller 336
8.1.2 Login Form 339
8.1.3 Finding and Authenticating a User 343
8.1.4 Rendering with a Flash Message 346
8.1.5 A Flash Test 348
8.2 Logging In 351
8.2.1 The `log_in` Method 351
8.2.2 Current User 353
8.2.3 Changing the Layout Links 357
8.2.4 Testing Layout Changes 361
8.2.5 Login upon Sign-up 366
8.3 Logging Out 368
8.4 Remember Me 371
8.4.1 Remember Token and Digest 372
8.4.2 Login with Remembering 377
8.4.3 Forgetting Users 385
8.4.4 Two Subtle Bugs 387
8.4.5 “Remember Me” Check Box 392
8.4.6 Remember Tests 397
8.5 Conclusion 404
8.5.1 What We Learned in This Chapter 405
8.6 Exercises 406

Chapter 9 Updating, Showing, and Deleting Users 411

9.1 Updating Users 411
- 9.1.1 Edit Form 412
- 9.1.2 Unsuccessful Edits 417
- 9.1.3 Testing Unsuccessful Edits 419
- 9.1.4 Successful Edits (with TDD) 420

9.2 Authorization 424
- 9.2.1 Requiring Logged-in Users 425
- 9.2.2 Requiring the Right User 431
- 9.2.3 Friendly Forwarding 436

9.3 Showing All Users 440
- 9.3.1 Users Index 441
- 9.3.2 Sample Users 445
- 9.3.3 Pagination 448
- 9.3.4 Users Index Test 452
- 9.3.5 Partial Refactoring 454

9.4 Deleting Users 456
- 9.4.1 Administrative Users 456
- 9.4.2 The `destroy` Action 460
- 9.4.3 User Destroy Tests 462

9.5 Conclusion 466
- 9.5.1 What We Learned in This Chapter 467

9.6 Exercises 468

Chapter 10 Account Activation and Password Reset 471

10.1 Account Activation 471
- 10.1.1 Account Activations Resource 473
- 10.1.2 Account Activation Mailer Method 480
- 10.1.3 Activating the Account 493
- 10.1.4 Activation Test and Refactoring 502

10.2 Password Reset 506
- 10.2.1 Password Resets Resource 508
- 10.2.2 Password Resets Controller and Form 512
- 10.2.3 Password Reset Mailer Method 516

10.2.4 Resetting the Password 523
10.2.5 Password Reset Test 530
10.3 Email in Production 532
10.4 Conclusion 534
10.4.1 What We Learned in This Chapter 536
10.5 Exercises 536
10.6 Proof of Expiration Comparison 539

Chapter 11 User Microposts 541

11.1 A Micropost Model 541
11.1.1 The Basic Model 542
11.1.2 Micropost Validations 544
11.1.3 User/Micropost Associations 547
11.1.4 Micropost Refinements 550
11.2 Showing Microposts 555
11.2.1 Rendering Microposts 555
11.2.2 Sample Microposts 559
11.2.3 Profile Micropost Tests 563
11.3 Manipulating Microposts 568
11.3.1 Micropost Access Control 569
11.3.2 Creating Microposts 572
11.3.3 A Proto-feed 580
11.3.4 Destroying Microposts 586
11.3.5 Micropost Tests 589
11.4 Micropost Images 593
11.4.1 Basic Image Upload 594
11.4.2 Image Validation 597
11.4.3 Image Resizing 601
11.4.4 Image Upload in Production 603
11.5 Conclusion 606
11.5.1 What We Learned in This Chapter 608
11.6 Exercises 609

Chapter 12 Following Users 613

12.1 The Relationship Model 614

12.1.1 A Problem with the Data Model (and a Solution) 615

12.1.2 User/Relationship Associations 621

12.1.3 Relationship Validations 623

12.1.4 Followed Users 625

12.1.5 Followers 628

12.2 A Web Interface for Following Users 630

12.2.1 Sample Following Data 631

12.2.2 Stats and a Follow Form 632

12.2.3 Following and Followers Pages 642

12.2.4 A Working Follow Button the Standard Way 651

12.2.5 A Working Follow Button Created with Ajax 653

12.2.6 Following Tests 658

12.3 The Status Feed 661

12.3.1 Motivation and Strategy 662

12.3.2 A First Feed Implementation 663

12.3.3 Subselects 667

12.4 Conclusion 671

12.4.1 Guide to Further Resources 672

12.4.2 What We Learned in This Chapter 673

12.5 Exercises 673

Index 675

This page intentionally left blank

Foreword to the First Edition

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.

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)
Founder, CD Baby

* [Author note:] This is not typical! Getting through the entire book usually takes *much* longer than three days.

This page intentionally left blank

Foreword to the Third Edition

Rails is now ten years old and adoption shows no sign of slowing down. Along with the perpetual growth, we've seen a paradoxical tragedy unfolding. According to many people, Rails is now one of the hardest tech stacks for beginners to adopt. The complexity of choices to make when starting out is very high and there are ten years of blog posts and books out there, most of which are obsolete and broken to some degree or another. The supreme irony is that getting started with Rails today involves a lot of configuration, perhaps not of Rails itself, but of the myriad libraries that are recommended for use with it. In case you've forgotten, or weren't paying attention in 2005 when Rails debuted, the main goal was to achieve convention over configuration.

To some extent, we've replicated the Java web beast that we once fought hard to slay. Argh.

Don't get too depressed though, because that would be missing the point. The good news is that once you're past the daunting learning curve, Rails remains one of the most powerful and efficient stacks available for building API backends and content-driven websites.

Now maybe you're considering using this book to kick off your journey up the Rails learning curve. Trust me, it's the right choice. I've known Michael Hartl for almost ten years now and he is a highly intelligent man. Just look for his credentials elsewhere in the book and you'll see what I mean. But never mind the prestigious degrees, the approach he has adopted for this latest edition of our best-selling *Ruby* *on Rails*™ *Tutorial* proves just how smart he is. Instead of doubling down on the opinionated approach (like another series author I know, ahem), he's gone in the opposite direction! By getting less opinionated, he has lowered the barrier for Rails newcomers in significant ways.

First of all, he dispenses with any sort of local installation or configuration. He also eschews complex configuration options (like Spork and RubyTest), which are likely to trip up novices. All code examples run in a standardized cloud-based environment accessible via a simple web browser.

Second, he throws out tons of content from the previous edition and embraces the Rails "default stack," including its built-in MiniTest testing framework. The resulting elimination of many external dependencies (RSpec, Cucumber, Capybara, Factory Girl) makes the Rails learning curve quite a bit easier to climb, at the expense of having to rewrite big swaths of the book.

Over the years, in no small part due to his work on this book franchise, Michael has become a master of writing training materials grounded in practical, useful knowledge. And as in the past, this edition includes basics of vital tools such as Git and GitHub. Testing is front and center, which most would agree is the proper emphasis for beginners. Michael's well-polished examples always utilize small, bite-sized pieces of code—simple enough to understand and novel enough to be challenging. By the time you finish the book and are playing around with your very own little Twitter clone, you're sure to possess a deeper, more flexible knowledge of Rails. Most importantly, you'll have a foundation flexible enough to get you coding up nearly any type of web application.

Godspeed!

—Obie Fernandez,
Series Editor

Acknowledgments

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, Mark Bates, 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. 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*, one of the leading introductions to web development, and is a cofounder of the Softcover self-publishing platform. His prior experience includes writing and developing *RailsSpace*, an extremely obsolete Rails tutorial book, and developing Insoshi, a once-popular and now-obsolete 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

This page intentionally left blank

CHAPTER 2
A Toy App

In this chapter, we'll develop a toy demo application to show off some of the power of Rails. The purpose is to get a high-level overview of Ruby on Rails programming (and web development in general) by rapidly generating an application using *scaffold generators*, which create a large amount of functionality automatically. As discussed in Box 1.2, the rest of the book will take the opposite approach, developing a full sample application incrementally and explaining each new concept as it arises. For a quick overview (and some instant gratification), though, there is no substitute for scaffolding. The resulting toy app will allow us to interact with it through its URLs, giving us insight into the structure of a Rails application, including a first example of the *REST architecture* favored by Rails.

As with the forthcoming sample application, the toy app will consist of *users* and their associated *microposts* (thus constituting a minimalist Twitter-style app). The functionality will be utterly under-developed, and many of the steps will seem like magic, but worry not: The full example will develop a similar application from the ground up starting in Chapter 3, and I will provide plentiful forward-references to later material. In the meantime, have patience and a little faith—the whole point of this tutorial is to take you *beyond* this superficial, scaffold-driven approach to achieve a deeper understanding of Rails.

2.1 Planning the Application

In this section, we'll outline our plans for the toy application. As in Section 1.3, we'll start by generating the application skeleton using the `rails new` command with a specific Rails version number:

```
$ cd ~/workspace
$ rails _4.2.0_ new toy_app
$ cd toy_app/
```

If the command above returns an error like "Could not find 'railties'," it means you don't have the right version of Rails installed, and you should double-check that you followed the command in Listing 1.1 exactly as written. (If you're using the cloud IDE as recommended in Section 1.2.1, note that this second app can be created in the same workspace as the first. It is not necessary to create a new workspace. To get the files to appear, you may need to click the gear icon in the file navigator area and select "Refresh File Tree.")

Next, we'll use a text editor to update the `Gemfile` needed by Bundler with the contents of Listing 2.1.

Listing 2.1 A `Gemfile` for the toy app.

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

gem 'rails',                '4.2.0'
gem 'sass-rails',           '5.0.1'
gem 'uglifier',             '2.5.3'
gem 'coffee-rails',         '4.1.0'
gem 'jquery-rails',         '4.0.3'
gem 'turbolinks',           '2.3.0'
gem 'jbuilder',             '2.2.3'
gem 'sdoc',                 '0.4.0', group: :doc

group :development, :test do
  gem 'sqlite3',     '1.3.9'
  gem 'byebug',      '3.4.0'
  gem 'web-console', '2.0.0.beta3'
  gem 'spring',      '1.1.3'
end

group :production do
  gem 'pg',             '0.17.1'
  gem 'rails_12factor', '0.0.2'
end
```

Note that Listing 2.1 is identical to Listing 1.14.

As in Section 1.5.1, we'll install the local gems while suppressing the installation of production gems using the `--without production` option:

```
$ bundle install --without production
```

Finally, we'll put the toy app under version control with Git:

```
$ git init
$ git add -A
$ git commit -m "Initialize repository"
```

You should also create a new repository by clicking the "Create" button at Bitbucket (Figure 2.1), and then push up to the remote repository:

```
$ git remote add origin git@bitbucket.org:<username>/toy_app.git
$ git push -u origin --all # pushes up the repo and its refs for the first time
```

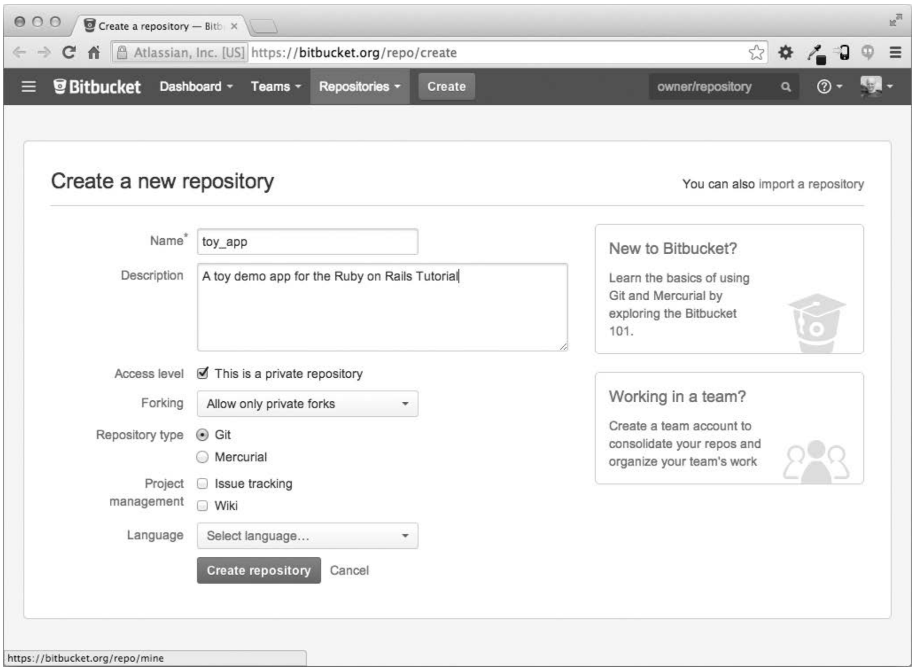
Finally, it's never too early to deploy, which I suggest doing by following the same "hello, world!" steps provided in Listing 1.8 and Listing 1.9.[1] Then commit the changes and push up to Heroku:

```
$ git commit -am "Add hello"
$ heroku create
$ git push heroku master
```

(As in Section 1.5, you may see some warning messages, which you should ignore for now. We'll eliminate them in Section 7.5.) Apart from the address of the Heroku app, the result should be the same as in Figure 1.18.

Now we're ready to start making the app itself. The typical first step when making a web application is to create a *data model*, which is a representation of the structures needed by our application. In our case, the toy app will be a microblog, with only users and short (micro)posts. Thus, we'll begin with a model for *users* of the app (Section 2.1.1), and then we'll add a model for *microposts* (Section 2.1.2).

1. The main reason for this is that the default Rails page typically breaks at Heroku, which makes it hard to tell if the deployment was successful.

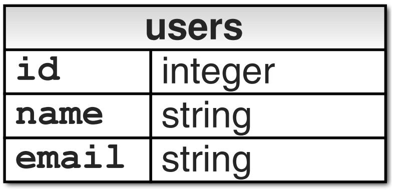

Create a repository — Bitb
Atlassian, Inc. [US] https://bitbucket.org/repo/create
Bitbucket
Dashboard
Teams
Repositories
Create
owner/repository
New to Bitbucket?
Learn the basics of using Git and Mercurial by exploring the Bitbucket 101.
Working in a team?
Create a team account to consolidate your repos and organize your team's work
Create repository
Cancel
https://bitbucket.org/repo/mine

Create a new repository

You can also import a repository

Name*	toy_app
Description	A toy demo app for the Ruby on Rails Tutorial
Access level	[✓ This is a private repository]
Forking	Allow only private forks
Repository type	[✓ Git] Mercurial
Project management	Issue tracking, Wiki
Language	Select language...

Figure 2.1 Creating the toy app repository at Bitbucket.

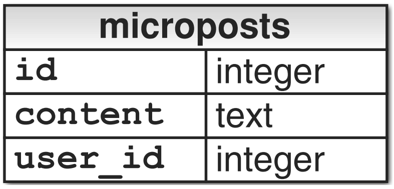


users	
id	integer
name	string
email	string

Figure 2.2 The data model for users.

2.1.1 A Toy Model for Users

There are as many choices for a user data model as there are different registration forms on the web; we'll go with a distinctly minimalist approach. Users of our toy app will have a unique `integer` identifier called `id`, a publicly viewable `name` (of type `string`), and an `email` address (also a `string`) that will double as a username. A summary of the data model for users appears in Figure 2.2.



microposts	
`id`	integer
`content`	text
`user_id`	integer

Figure 2.3 The data model for microposts.

As we'll see starting in Section 6.1.1, the label `users` in Figure 2.2 corresponds to a *table* in a database, and the `id`, `name`, and `email` attributes are *columns* in that table.

2.1.2 A Toy Model for Microposts

The core of the micropost data model is even simpler than the one for users: A micropost has only an `id` and a `content` field for the micropost's text (of type `text`).[2] There's an additional complication, though: We want to *associate* each micropost with a particular user. We'll accomplish this by recording the `user_id` of the owner of the post. The results are shown in Figure 2.3.

We'll see in Section 2.3.3 (and more fully in Chapter 11) how this `user_id` attribute allows us to succinctly express the notion that a user potentially has many associated microposts.

2.2 The Users Resource

In this section, we'll implement the users data model in Section 2.1.1, along with a web interface to that model. This combination will constitute a *Users resource*, which will allow us to think of users as objects that can be created, read, updated, and deleted through the web via the HTTP protocol. As promised in the introduction, our Users resource will be created by a scaffold generator program, which comes standard with each Rails project. I urge you not to look too closely at the generated code; at this stage, doing so will simply confuse you.

Rails scaffolding is generated by passing the `scaffold` command to the `rails generate` script. The argument of the `scaffold` command is the singular version of

2. Because microposts are short by design, the `string` type is actually big enough to contain them, but using `text` better expresses our intent, while also giving us greater flexibility should we ever wish to relax the length constraint.

the resource name (in this case, `User`), together with optional parameters for the data model's attributes:[3]

```
$ rails generate scaffold User name:string email:string
      invoke  active_record
      create    db/migrate/20140821011110_create_users.rb
      create    app/models/user.rb
      invoke    test_unit
      create      test/models/user_test.rb
      create      test/fixtures/users.yml
      invoke  resource_route
       route    resources :users
      invoke  scaffold_controller
      create    app/controllers/users_controller.rb
      invoke    erb
      create      app/views/users
      create      app/views/users/index.html.erb
      create      app/views/users/edit.html.erb
      create      app/views/users/show.html.erb
      create      app/views/users/new.html.erb
      create      app/views/users/_form.html.erb
      invoke    test_unit
      create      test/controllers/users_controller_test.rb
      invoke    helper
      create      app/helpers/users_helper.rb
      invoke      test_unit
      create        test/helpers/users_helper_test.rb
      invoke    jbuilder
      create      app/views/users/index.json.jbuilder
      create      app/views/users/show.json.jbuilder
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/users.js.coffee
      invoke    scss
      create      app/assets/stylesheets/users.css.scss
      invoke  scss
      create    app/assets/stylesheets/scaffolds.css.scss
```

By including `name:string` and `email:string`, we have arranged for the User model to have the form shown in Figure 2.2. (There is no need to include a parameter for `id`; it is created automatically by Rails for use as the *primary key* in the database.)

3. The name of the scaffold follows the convention of *models*, which are singular, rather than resources and controllers, which are plural. Thus, we have `User` instead of `Users`.

To proceed with the toy application, we first *migrate* the database using *Rake* (Box 2.1):

```
$ bundle exec rake db:migrate
==  CreateUsers: migrating ==================================================
-- create_table(:users)
   -> 0.0017s
==  CreateUsers: migrated (0.0018s) =========================================
```

This simply updates the database with our new `users` data model. (We'll learn more about database migrations starting in Section 6.1.1.) To ensure that the command uses the version of Rake corresponding to our `Gemfile`, we need to run `rake` using `bundle exec`. On many systems, including the cloud IDE, you can omit `bundle exec`, but it is necessary on some systems, so we'll include it here for completeness.

With that, we can run the local web server in a separate tab (Figure 1.7) as follows:[4]

```
$ rails server -b $IP -p $PORT    # Use only `rails server` if running locally
```

Now the toy application should be available on the local server as described in Section 1.3.2. (If you're using the cloud IDE, be sure to open the resulting development server in a new *browser* tab, not inside the IDE itself.)

Box 2.1 Rake

In the Unix tradition, the *make* utility has played an important role in building executable programs from source code. Indeed, many a computer hacker has committed to muscle memory the line

```
onfig re    ma e    s do ma e install
```

commonly used to compile code on Unix systems (including Linux and Mac OS X).

Rake is *Ruby make*, a make-like language written in Ruby. Rails uses Rake extensively, especially for the innumerable little administrative tasks that must be carried out when developing database-backed web applications. The `rake db:migrate`

4. The `rails` script is designed so that you don't need to use `bundle exec`.

command is probably the most widely used, but there are many others; you can see a list of database tasks using `-T db`:

```
ndle e e  ra e    d
```

To see all the Rake tasks available, run

```
ndle e e  ra e
```

The list generated by this command is likely to be overwhelming, but don't worry: You don't have to know all (or even most) of these commands. By the end of the *Rails Tutorial*, you'll know all the most important ones.

2.2.1 A User Tour

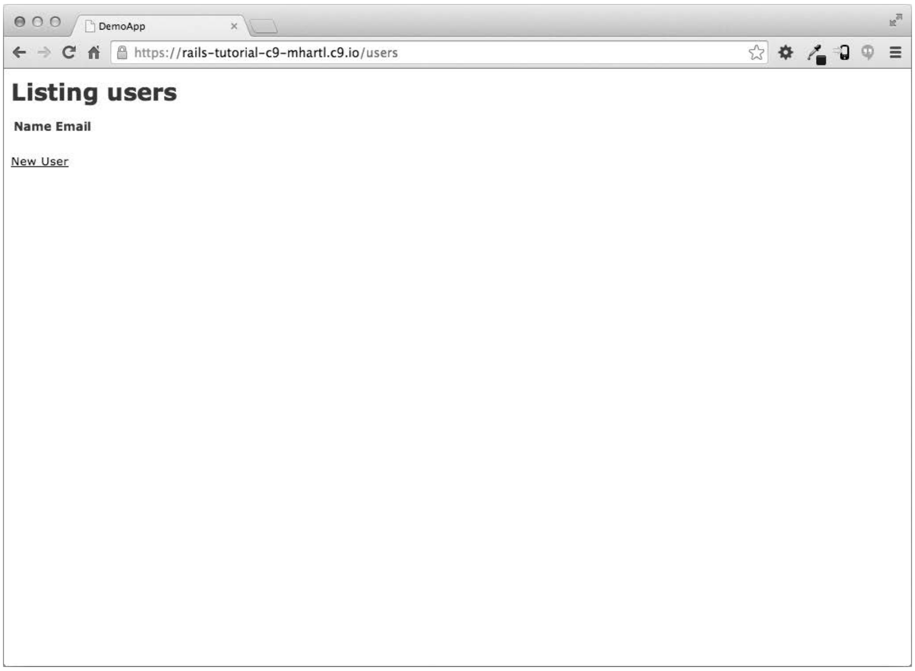
If we visit the root URL at / (read "slash," as noted in Section 1.3.4), we get the same default Rails page shown in Figure 1.9. In generating the Users resource scaffolding, however, we have also created a large number of pages for manipulating users. For example, the page for listing all users is at /users, and the page for making a new user is at /users/new. The rest of this section is dedicated to taking a whirlwind tour through these user pages. As we proceed, it may help to refer to Table 2.1, which shows the correspondence between pages and URLs.

We start with the page to show all the users in our application, called `inde` ; as you might expect, initially there are no users (Figure 2.4).

To make a new user, we visit the `ne` page, as shown in Figure 2.5. (Since the http://0.0.0.0:3000 or cloud IDE part of the address is implicit whenever we are developing locally, we'll omit it from now on.) In Chapter 7, this will become the user sign-up page.

Table 2.1 The correspondence between pages and URLs for the Users resource.

URL	Action	Purpose
/users	`index`	page to list all users
/users/1	`show`	page to show user with id `1`
/users/new	`new`	page to make a new user
/users/1/edit	`edit`	page to edit user with id `1`



Listing users

Name Email

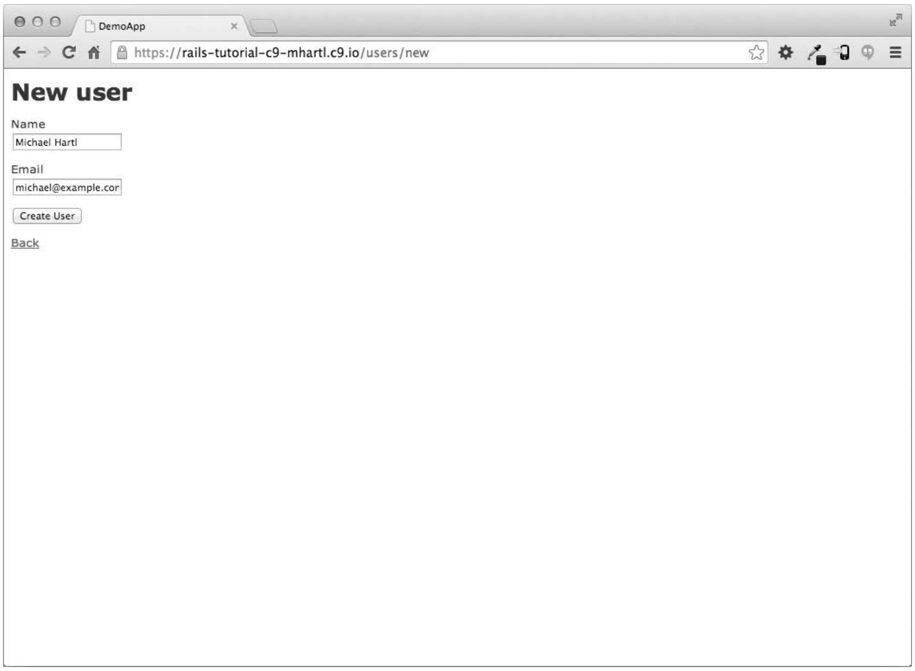
New User

Figure 2.4 The initial index page for the Users resource (/users).

We can create a user by entering name and email values in the text fields and then clicking the Create User button. The result is the user `sho` page, seen in Figure 2.6. (The green welcome message is accomplished using the *flash*, which we'll learn about in Section 7.4.2.) Note that the URL is /users/1; as you might suspect, the number `1` is simply the user's `id` attribute from Figure 2.2. In Section 7.1, this page will become the user's profile.

To change a user's information, we visit the `edit` page (Figure 2.11 on page 67). By modifying the user information and clicking the Update User button, we change the information for the user in the toy application (Figure 2.8). (As we'll see in detail starting in Chapter 6, this user data is stored in a database back-end.) We'll add user edit/update functionality to the sample application in Section 9.1.

Now we'll create a second user by revisiting the `ne` page and submitting a second set of user information; the resulting user `inde` is shown in Figure 2.9. Section 7.1 will develop the user index into a more polished page for showing all users.



New user

Name
Michael Hartl

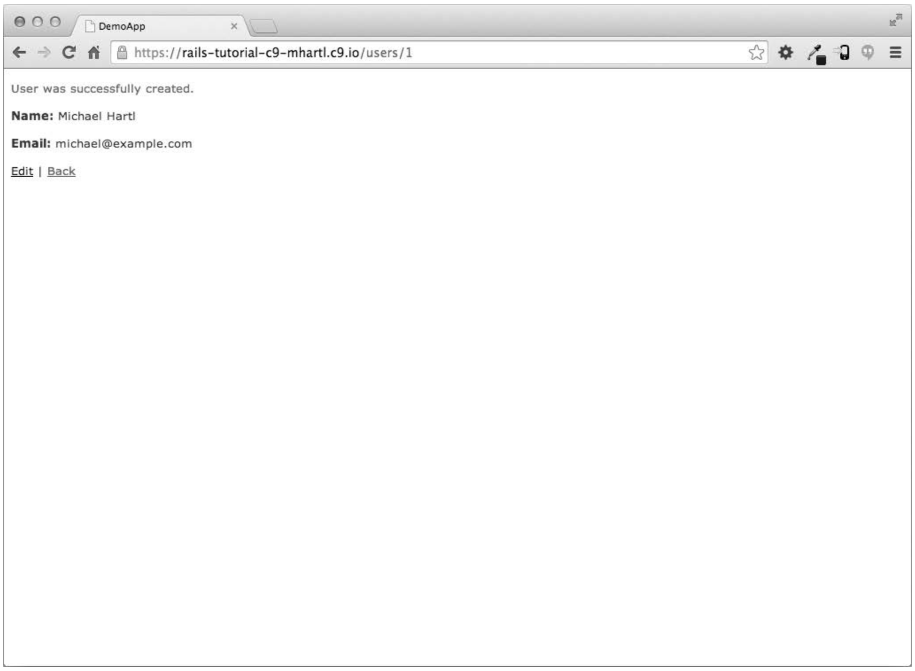
Email
michael@example.com

Figure 2.5 The new user page (/users/new).

Now that we know how to create, show, and edit users, we come finally to the process of destroying them (Figure 2.10). You should verify that clicking on the link in Figure 2.10 destroys the second user, yielding an index page with only one user. (If it doesn't work, make sure that JavaScript is enabled in your browser; Rails uses JavaScript to issue the request needed to destroy a user.) Section 9.4 adds user deletion to the sample app, taking care to restrict its use to a special class of administrative users.

2.2.2 MVC in Action

Now that we've completed a quick overview of the Users resource, let's examine one particular part of it in the context of the Model–View–Controller (MVC) pattern introduced in Section 1.3.3. Our strategy will be to describe the results of a typical browser hit—a visit to the user index page at /users—in terms of the MVC pattern (Figure 2.7).



User was successfully created.

Name: Michael Hartl

Email: michael@example.com

Edit | Back

Figure 2.6 The page to show a user (/users/1).

Here is a summary of the steps shown in Figure 2.7:

1. The browser issues a request for the /users URL.
2. Rails routes /users to the `index` action in the Users controller.
3. The `index` action asks the User model to retrieve all users (`User.all`).
4. The User model pulls all the users from the database.
5. The User model returns the list of users to the controller.
6. The controller captures the users in the `@users` variable, which is passed to the `index` view.
7. The view uses embedded Ruby to render the page as HTML.
8. The controller passes the HTML back to the browser.[5]

5. Some references indicate that the view returns the HTML directly to the browser (via a web server such as Apache or Nginx). Regardless of the implementation details, I prefer to think of the controller as a central hub through which all the application's information flows.

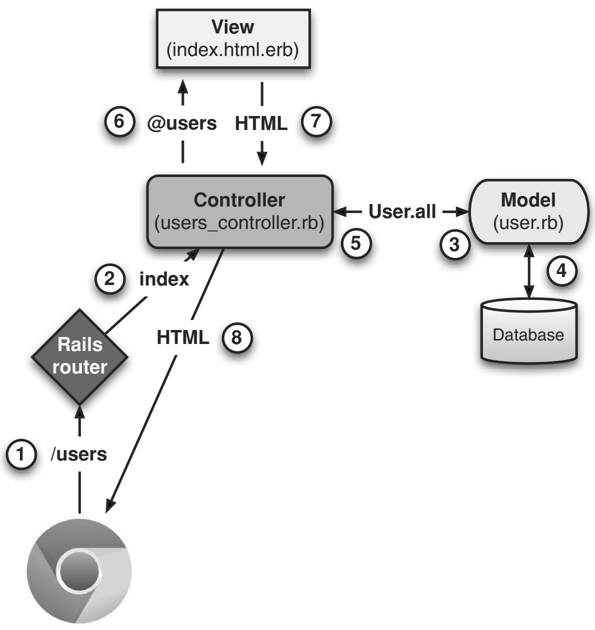
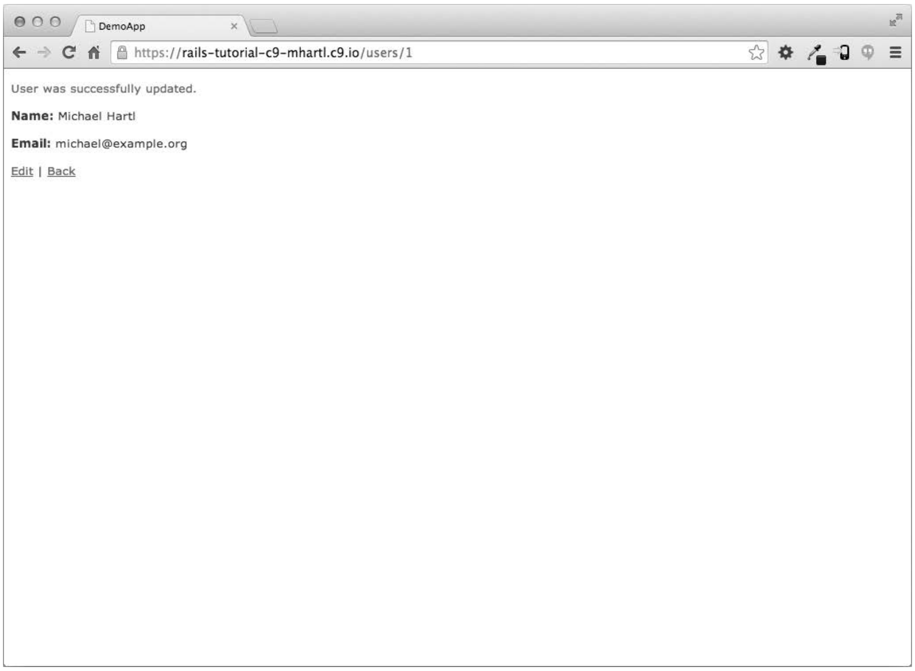


Figure 2.7 A detailed diagram of MVC in Rails.

Now let's take a look at these in more detail. We start with a request issued from the browser—that is, the result of typing a URL in the address bar or clicking on a link (Step 1 in Figure 2.7). This request is sent to the *Rails router* (Step 2), which it dispatches to the proper *controller action* based on the URL (and, as we'll see in Box 3.2, the type of request). The code to create the mapping of user URLs to controller actions for the Users resource appears in Listing 2.2; this code effectively sets up the table of URL/action pairs seen in Table 2.1. (The strange notation `:users` is a *symbol*, which we'll learn about in Section 4.3.3.)

Listing 2.2 The Rails routes, with a rule for the Users resource.
config/routes.rb

```
Rails.application.routes.draw do
  resources :users
  .
  .
  .
end
```



User was successfully updated.

Name: Michael Hartl

Email: michael@example.org

Edit | Back

Figure 2.8 A user with updated information.

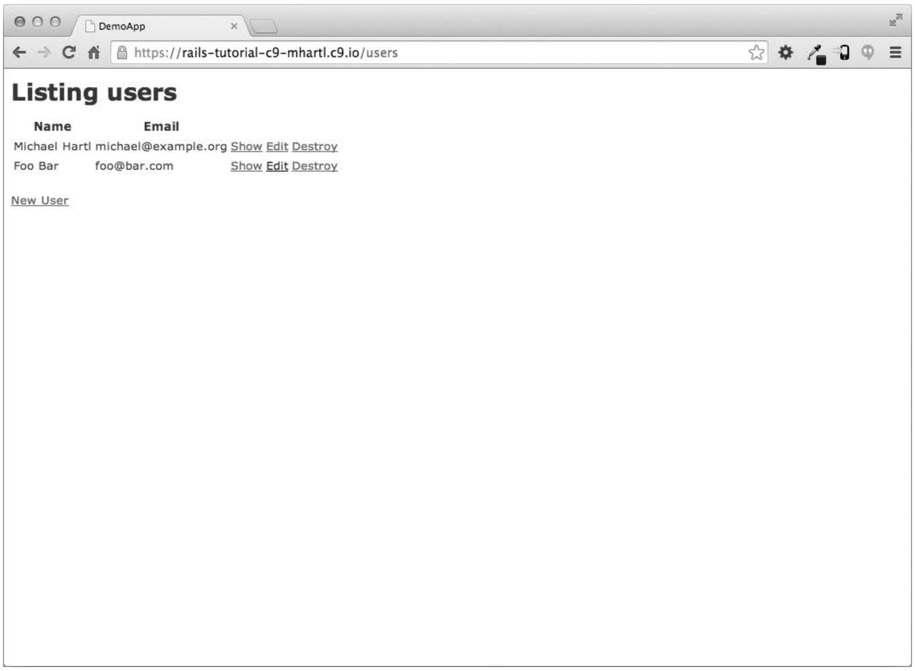
While we're looking at the routes file, let's take a moment to associate the root route with the users index, so that "slash" goes to /users. Recall from Listing 1.10 that we changed

```
# root 'welcome#index'
```

to read

```
root 'application#hello'
```

so that the root route went to the `hello` action in the Application controller. In the present case, we want to use the `index` action in the Users controller, which we can arrange using the code shown in Listing 2.3. (At this point, I also recommend removing the `hello` action from the Application controller if you added it at the beginning of this section.)



Listing users

Name	Email			
Michael Hartl	michael@example.org	Show	Edit	Destroy
Foo Bar	foo@bar.com	Show	Edit	Destroy

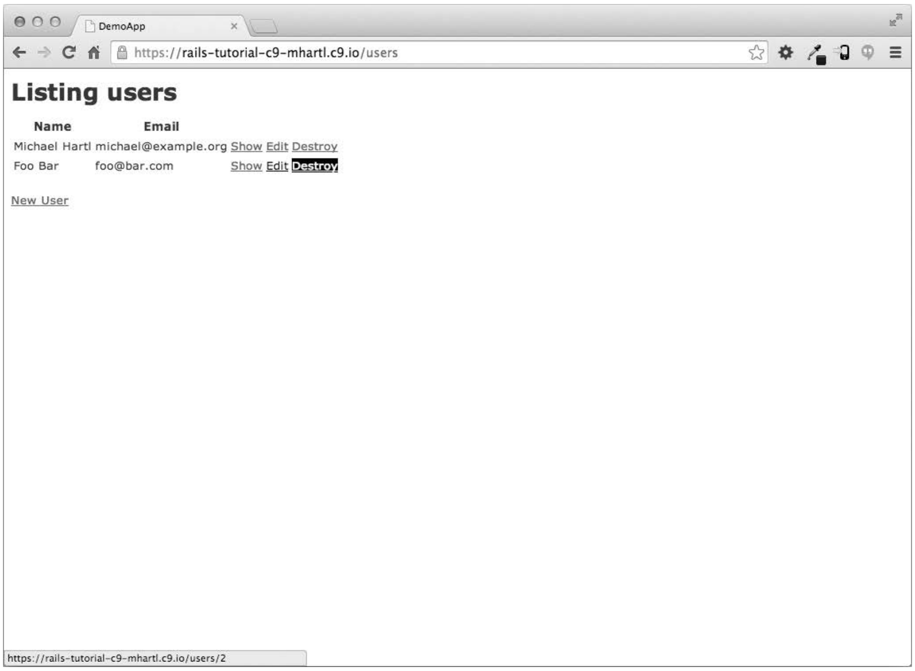
New User

Figure 2.9 The user index page (/users) with a second user.

Listing 2.3 Adding a root route for users.
config/routes.rb

```
Rails.application.routes.draw do
  resources :users
  root 'users#index'
  .
  .
  .
end
```

The pages from the tour in Section 2.2.1 correspond to *actions* in the Users *controller*, which is a collection of related actions. The controller generated by the scaffolding is shown schematically in Listing 2.4. In this listing, the notation `class UsersController < ApplicationController`, is an example of a Ruby *class* with *inheritance*.



Listing users

Name	Email			
Michael Hartl	michael@example.org	Show	Edit	Destroy
Foo Bar	foo@bar.com	Show	Edit	Destroy

New User

https://rails-tutorial-c9-mhartl.c9.io/users/2

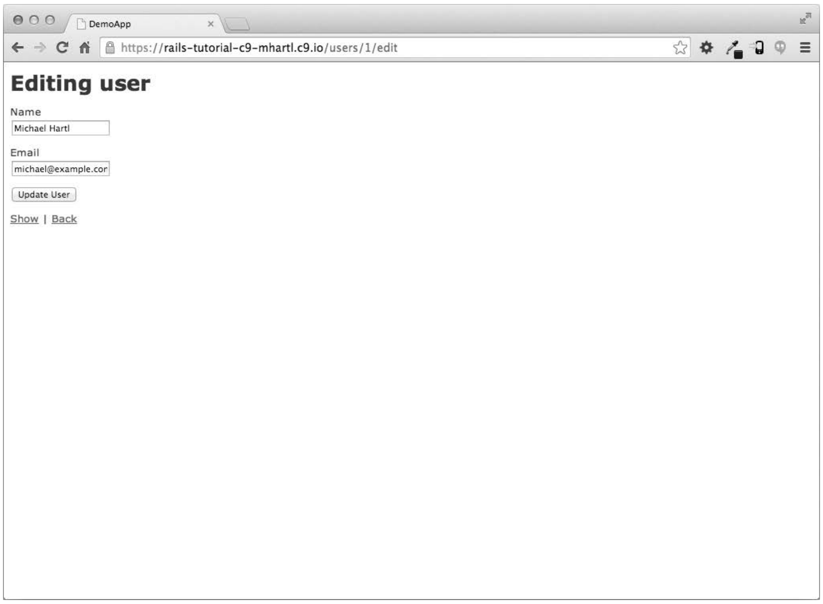
Figure 2.10 Destroying a user.

(We'll discuss inheritance briefly in Section 2.3.4 and cover both subjects in more detail in Section 4.4.)

Listing 2.4 The Users controller in schematic form.
app/controllers/users_controller.rb

```
class UsersController < ApplicationController
  .
  .
  .
  def index
    .
    .
    .
  end
  def show
    .
    .
    .
  end

  def new
    .
    .
    .
  end

  def edit
    .
    .
    .
  end

  def create
    .
    .
    .
  end

  def update
    .
    .
    .
  end

  def destroy
    .
    .
    .
  end
end
```

You may notice that there are more actions than there are pages; in this listing, `index`, `show`, `new`, and `edit` actions all correspond to pages from Section 2.2.1, but there are additional `create`, `update`, and `destroy` actions as well. These actions don't typically render pages (although they can); instead, their main purpose is to modify information about users in the database. This full suite of controller actions, summarized in Table 2.2, represents the implementation of the REST architecture in Rails (Box 2.2), which is based on the idea of *representational state transfer* identified and named by



Editing user

Name
Michael Hartl

Email
michael@example.cor

Update User

Show | Back

Figure 2.11 The user edit page (/users/1/edit).

Table 2.2 RESTful routes provided by the Users resource in Listing 2.2.

HTTP request	**URL**	**Action**	**Purpose**
	/users	`index`	page to list all users
	/users/1	`show`	page to show user with id `1`
	/users/new	`new`	page to make a new user
	/users	`create`	create a new user
	/users/1/edit	`edit`	page to edit user with id `1`
	/users/1	`update`	update user with id `1`
	/users/1	`destroy`	delete user with id `1`

computer scientist Roy Fielding.[6] As seen in Table 2.2, there is some overlap in the URLs; for example, both the user `show` action and the `update` action correspond to the URL /users/1. The difference between them is the HTTP request method to which they respond. We'll learn more about HTTP request methods starting in Section 3.3.

6. Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

Box 2.2 REpresentational State Transfer (REST)

If you read much about Ruby on Rails web development, you'll see a lot of references to "REST," which is an acronym for REpresentational State Transfer. REST is an architectural style for developing distributed, networked systems and software applications such as the World Wide Web and web applications. Although this theory is rather abstract, in the context of Rails applications REST means that most application components (such as users and microposts) are modeled as *resources* that can be created, read, updated, and deleted—operations that correspond both to the CRUD operations of relational databases and to the four fundamental HTTP request methods: , , , and .[7] (We'll learn more about HTTP requests in Section 3.3 and especially Box 3.2.)

As a Rails application developer, the RESTful style of development helps you make choices about which controllers and actions to write: You simply structure the application using resources that get created, read, updated, and deleted. In the case of users and microposts, this process is straightforward, since they are naturally resources in their own right. In Chapter 12, we'll see an example where REST principles allow us to model a subtler problem, "following users," in a natural and convenient way.

To examine the relationship between the Users controller and the User model, let's focus on a simplified version of the `index` action, shown in Listing 2.5. (The scaffold code is ugly and confusing, so it's suppressed here.)

Listing 2.5 The simplified user `index` action for the toy application.
app/controllers/users_controller.rb

```
class UsersController < ApplicationController
  .
  .
  .
  def index
```

7. Earlier versions of Rails used for data updates, but is the more appropriate method according to the HTTP standard.

```
    @users = User.all
  end
  .
  .
  .
end
```

This `index` action includes the line `@users = User.all` (Step 3 in Figure 2.7), which asks the User model to retrieve a list of all users from the database (Step 4), and then places them in the variable `@users` (pronounced "at-users") (Step 5). The User model itself appears in Listing 2.6; although it is rather plain, it comes equipped with a large amount of functionality because of inheritance (Section 2.3.4 and Section 4.4). In particular, by using the Rails library called *Active Record*, the code in Listing 2.6 arranges for `User.all` to return all the users in the database.

Listing 2.6 The User model for the toy application.
app/models/user.rb

```
class User < ActiveRecord::Base
end
```

Once the `@users` variable is defined, the controller calls the *view* (Step 6), shown in Listing 2.7. Variables that start with the `@` sign, called *instance variables*, are automatically available in the views; in this case, the `index.html.erb` view in Listing 2.7 iterates through the `@users` list and outputs a line of HTML for each one. (Remember, you aren't supposed to understand this code right now. It is shown only for purposes of illustration.)

Listing 2.7 The view for the user index.
app/views/users/index.html.erb

```
<h1>Listing users</h1>

<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th colspan="3"></th>
    </tr>
  </thead>

<% @users.each do |user| %>
  <tr>
    <td><%= user.name %></td>
    <td><%= user.email %></td>
    <td><%= link_to 'Show', user %></td>
    <td><%= link_to 'Edit', edit_user_path(user) %></td>
    <td><%= link_to 'Destroy', user, method: :delete,
                                     data: { confirm: 'Are you sure?' } %></td>
  </tr>
<% end %>
</table>

<br>

<%= link_to 'New User', new_user_path %>
```

The view converts its contents to HTML (Step 7), which is then returned by the controller to the browser for display (Step 8).

2.2.3 Weaknesses of This Users Resource

Though good for getting a general overview of Rails, the scaffold Users resource suffers from a number of severe weaknesses.

- **data validations** Our User model accepts data such as blank names and invalid email addresses without complaint.
- **authentication** We have no notion of logging in or out, and no way to prevent any user from performing any operation.
- **tests** This isn't technically true—the scaffolding includes rudimentary tests—but the generated tests don't test for data validation, authentication, or any other custom requirements.
- **style or layout** There is no consistent site styling or navigation.
- **real understanding** If you understand the scaffold code, you probably shouldn't be reading this book.

2.3 The Microposts Resource

Having generated and explored the Users resource, we turn now to the associated Microposts resource. Throughout this section, we recommend comparing the elements of the Microposts resource with the analogous user elements from Section 2.2; the two resources parallel each other in many ways. The RESTful structure of Rails applications is best absorbed by this sort of repetition of form—indeed, seeing the parallel structure of Users and Microposts even at this early stage is one of the prime motivations for this chapter.

2.3.1 A Micropost Microtour

As with the Users resource, we'll generate scaffold code for the Microposts resource using `rails generate scaffold`, in this case implementing the data model from Figure 2.3:[8]

```
$ rails generate scaffold Micropost content:text user_id:integer
      invoke  active_record
      create    db/migrate/20140821012832_create_microposts.rb
      create    app/models/micropost.rb
      invoke    test_unit
      create      test/models/micropost_test.rb
      create      test/fixtures/microposts.yml
      invoke  resource_route
       route    resources :microposts
      invoke  scaffold_controller
      create    app/controllers/microposts_controller.rb
      invoke    erb
      create      app/views/microposts
      create      app/views/microposts/index.html.erb
      create      app/views/microposts/edit.html.erb
      create      app/views/microposts/show.html.erb
      create      app/views/microposts/new.html.erb
      create      app/views/microposts/_form.html.erb
      invoke    test_unit
      create      test/controllers/microposts_controller_test.rb
      invoke    helper
      create      app/helpers/microposts_helper.rb
```

8. As with the User scaffold, the scaffold generator for microposts follows the singular convention of Rails models; thus, we have `generate Micropost`.

```
      invoke      test_unit
      create        test/helpers/microposts_helper_test.rb
      invoke    jbuilder
      create      app/views/microposts/index.json.jbuilder
      create      app/views/microposts/show.json.jbuilder
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/microposts.js.coffee
      invoke    scss
      create      app/assets/stylesheets/microposts.css.scss
      invoke  scss
   identical    app/assets/stylesheets/scaffolds.css.scss
```

(If you get an error related to Spring, just run the command again.) To update our database with the new data model, we need to run a migration as in Section 2.2:

```
$ bundle exec rake db:migrate
==  CreateMicroposts: migrating ===========================================
-- create_table(:microposts)
   -> 0.0023s
==  CreateMicroposts: migrated (0.0026s) ====================================
```

Now we are in a position to create microposts in the same way we created users in Section 2.2.1. As you might guess, the scaffold generator has updated the Rails routes file with a rule for Microposts resource, as seen in Listing 2.8.[9] As with users, the `resources :microposts` routing rule maps micropost URLs to actions in the Microposts controller, as seen in Table 2.3.

Listing 2.8 The Rails routes, with a new rule for Microposts resources.
config/routes.rb

```
Rails.application.routes.draw do
  resources :microposts
  resources :users
  .
  .
  .
end
```

9. The scaffold code may have extra newlines compared to Listing 2.8. This is not a cause for concern, as Ruby ignores extra newlines.

Table 2.3 RESTful routes provided by the Microposts resource in Listing 2.8.

HTTP request	URL	Action	Purpose
	/microposts	`index`	page to list all microposts
	/microposts/1	`show`	page to show micropost with id `1`
	/microposts/new	`new`	page to make a new micropost
	/microposts	`create`	create a new micropost
	/microposts/1/edit	`edit`	page to edit micropost with id `1`
	/microposts/1	`update`	update micropost with id `1`
	/microposts/1	`destroy`	delete micropost with id `1`

The Microposts controller itself appears in schematic form in Listing 2.9. Note that, apart from `MicropostsController` in replacing of `UsersController`, Listing 2.9 is *identical* to the code in Listing 2.4. This is a reflection of the REST architecture common to both resources.

Listing 2.9 The Microposts controller in schematic form.
app/controllers/microposts_controller.rb

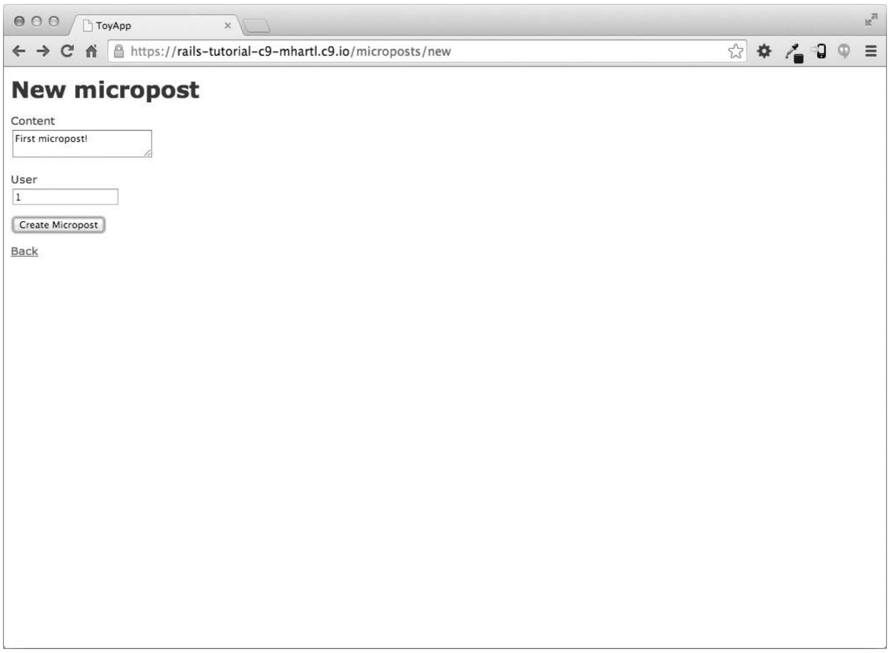
```
class MicropostsController < ApplicationController
  .
  .
  .
  def index
    .
    .
    .
  end

  def show
    .
    .
    .
  end

  def new
    .
    .
    .
  end
  def edit
    .
    .
    .
  end

  def create
    .
    .
    .
  end

  def update
    .
    .
    .
  end

  def destroy
    .
    .
    .
  end
end
```

To make some actual microposts, we enter information at the new microposts page, /microposts/new, as seen in Figure 2.12.

At this point, go ahead and create a micropost or two, taking care to make sure that at least one has a `user_id` of `1` to match the ID of the first user created in Section 2.2.1. The result should look something like Figure 2.13.

2.3.2 Putting the *Micro* in Microposts

Any *micro*post worthy of the name should have some means of enforcing the limits on the length of the post. Implementing this constraint in Rails is easy with *validations*; to accept microposts with at most 140 characters (la Twitter), we use a *length* validation. At this point, you should open the file `app/models/micropost.rb` in your text editor or IDE and fill it with the contents of Listing 2.10.



New micropost

Content

First micropost!

User

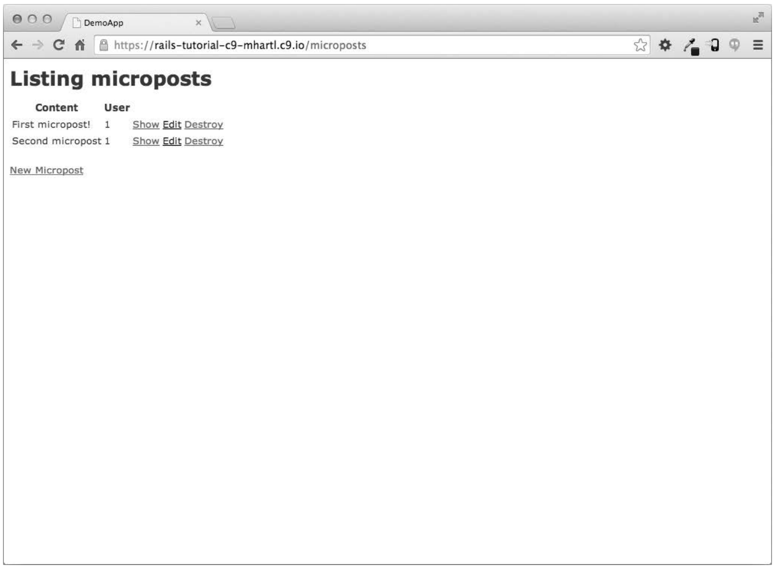
1

Figure 2.12 The new micropost page (/microposts/new).

Listing 2.10 Constraining microposts to be at most 140 characters.
app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  validates :content, length: { maximum: 140 }
end
```

The code in Listing 2.10 may look rather mysterious—we'll cover validations more thoroughly starting in Section 6.2—but its effects are readily apparent if we go to the new micropost page and enter more than 140 characters for the content of the post. As seen in Figure 2.14, Rails renders *error messages* indicating that the micropost's content is too long. (We'll learn more about error messages in Section 7.3.3.)



Listing microposts

Content	User			
First micropost!	1	Show	Edit	Destroy
Second micropost	1	Show	Edit	Destroy

New Micropost

Figure 2.13 The micropost index page (/microposts).

2.3.3 A User `has_many` Microposts

One of the most powerful features of Rails is the ability to form *associations* between different data models. In the case of our User model, each user potentially has many microposts. We can express this relationship in code by updating the User and Micropost models as in Listing 2.11 and Listing 2.12, respectively.

Listing 2.11 A user has many microposts.
app/models/user.rb

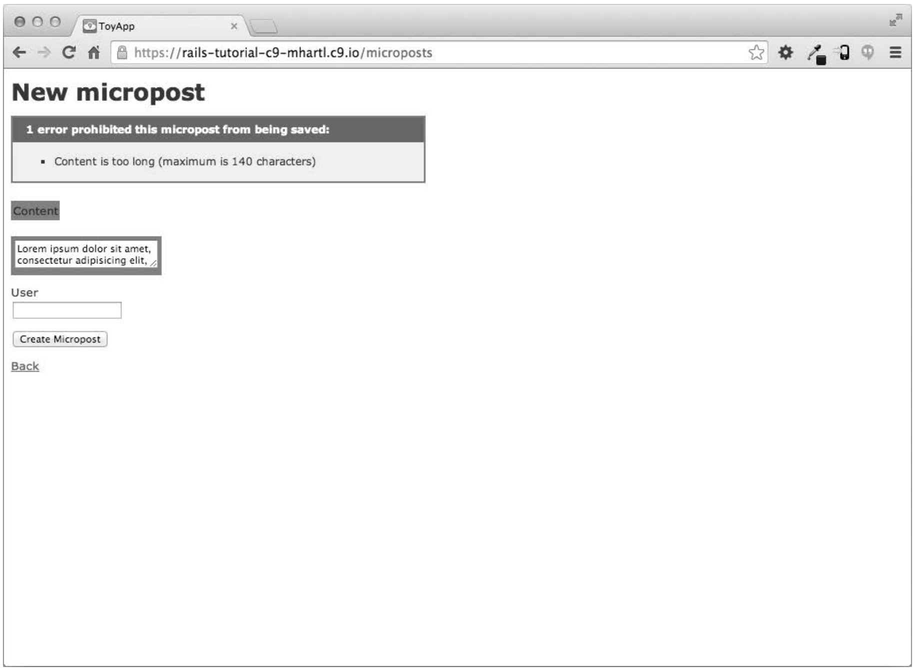
```
class User < ActiveRecord::Base
  has_many :microposts
end
```

Listing 2.12 A micropost belongs to a user.
app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  belongs_to :user
  validates :content, length: { maximum: 140 }
end
```

We can visualize the result of this association in Figure 2.15. Because of the `user_id` column in the `microposts` table, Rails (using Active Record) can infer the microposts associated with each user.

In Chapter 11 and Chapter 12, we will use the association of users and microposts both to display all of a user's microposts and to construct a Twitter-like micropost feed.



New micropost

1 error prohibited this micropost from being saved:

- Content is too long (maximum is 140 characters)

Content

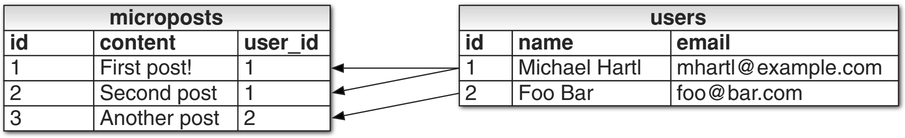
Lorem ipsum dolor sit amet, consectetur adipisicing elit,

User

Create Micropost

Back

Figure 2.14 Error messages for a failed micropost creation.



microposts		
id	content	user_id
1	First post!	1
2	Second post	1
3	Another post	2

users		
id	name	email
1	Michael Hartl	mhartl@example.com
2	Foo Bar	foo@bar.com

Figure 2.15 The association between microposts and users.

For now, we can examine the implications of the user—micropost association by using the *console*, which is a useful tool for interacting with Rails applications. We first invoke the console with `rails console` at the command line, and then retrieve the first user from the database using `User.first` (putting the results in the variable `first_user`):[10]

```
$ rails console
>> first_user = User.first
=> #<User id: 1, name: "Michael Hartl", email: "michael@example.org",
created_at: "2014-07-21 02:01:31", updated_at: "2014-07-21 02:01:31">
>> first_user.microposts
=> [#<Micropost id: 1, content: "First micropost!", user_id: 1, created_at:
"2014-07-21 02:37:37", updated_at: "2014-07-21 02:37:37">, #<Micropost id: 2,
content: "Second micropost", user_id: 1, created_at: "2014-07-21 02:38:54",
updated_at: "2014-07-21 02:38:54">]
>> micropost = first_user.microposts.first  # Micropost.first would also work.
=> #<Micropost id: 1, content: "First micropost!", user_id: 1, created_at:
"2014-07-21 02:37:37", updated_at: "2014-07-21 02:37:37">
>> micropost.user
=> #<User id: 1, name: "Michael Hartl", email: "michael@example.org",
created_at: "2014-07-21 02:01:31", updated_at: "2014-07-21 02:01:31">
>> exit
```

(I include `exit` in the last line just to demonstrate how to exit the console. On most systems, you can also use Ctrl-D for the same purpose.[11]) Here we have accessed the user's microposts using the code `first_user.microposts`. With this code, Active Record automatically returns all the microposts with `user_id` equal to the ID of `first_user` (in this case, `1`). We'll learn much more about the association facilities in Active Record in Chapter 11 and Chapter 12.

2.3.4 Inheritance Hierarchies

We end our discussion of the toy application with a brief description of the controller and model class hierarchies in Rails. This discussion will make sense only if you have some experience with object-oriented programming (OOP); if you haven't studied OOP, feel free to skip this section. In particular, if you are unfamiliar with *classes* (discussed in Section 4.4), you might want to loop back to this section at a later time.

10. Your console prompt might be something like , but the examples use since Ruby versions will vary.

11. As with "Ctrl-C," the convention is to write "Ctrl-D" even though it's really "Ctrl-d."

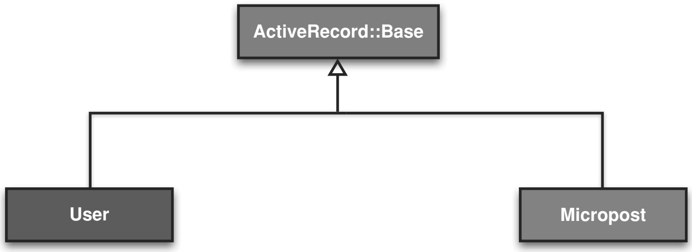


Figure 2.16 The inheritance hierarchy for the User and Micropost models.

We start with the inheritance structure for models. Comparing Listing 2.13 and Listing 2.14, we see that both the User model and the Micropost model inherit (via the left angle bracket <) from `ActiveRecord::Base`, which is the base class for models provided by Active Record; a diagram summarizing this relationship appears in Figure 2.16. It is by inheriting from `ActiveRecord::Base` that our model objects gain the ability to communicate with the database, treat the database columns as Ruby attributes, and so on.

Listing 2.13 The `User` class, highlighting inheritance.
app/models/user.rb

```
class User < ActiveRecord::Base
  .
  .
  .
end
```

Listing 2.14 The `Micropost` class, highlighting inheritance.
app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  .
  .
  .
end
```

The inheritance structure for controllers is only slightly more complicated. Comparing Listing 2.15 and Listing 2.16, we see that both the Users controller and the

Microposts controller inherit from the Application controller. Examining Listing 2.17, we see that `ApplicationController` itself inherits from `ActionController::Base`; this is the base class for controllers provided by the Rails library Action Pack. The relationships between these classes are illustrated in Figure 2.17.

Listing 2.15 The `UsersController` class, highlighting inheritance.
app/controllers/users_controller.rb

```
class UsersController < ApplicationController
  .
  .
  .
end
```

Listing 2.16 The `MicropostsController` class, highlighting inheritance.
app/controllers/microposts_controller.rb

```
class MicropostsController < ApplicationController
  .
  .
  .
end
```

Listing 2.17 The `ApplicationController` class, highlighting inheritance.
app/controllers/application_controller.rb

```
class ApplicationController < ActionController::Base
  .
  .
  .
end
```

As with model inheritance, both the Users and Microposts controllers gain a large amount of functionality by inheriting from a base class (in this case, `ActionController::Base`), including the ability to manipulate model objects, filter inbound HTTP requests, and render views as HTML. Given that all Rails controllers inherit from `ApplicationController`, rules defined in the Application controller

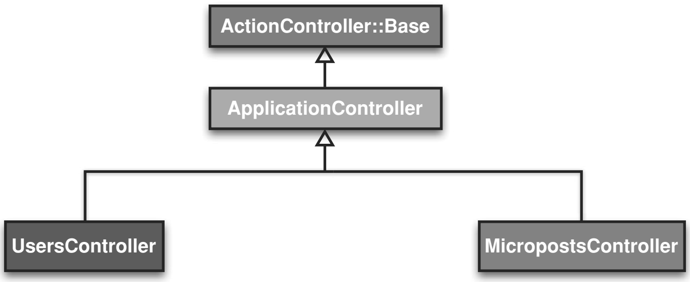


Figure 2.17 The inheritance hierarchy for the Users and Microposts controllers.

automatically apply to every action in the application. For example, in Section 8.4 we'll see how to include helpers for logging in and logging out of all of the sample application's controllers.

2.3.5 Deploying the Toy App

With the completion of the Microposts resource, now is a good time to push the repository up to Bitbucket:

```
$ git status
$ git add -A
$ git commit -m "Finish toy app"
$ git push
```

Ordinarily, you should make smaller, more frequent commits, but for the purposes of this chapter a single big commit at the end is fine.

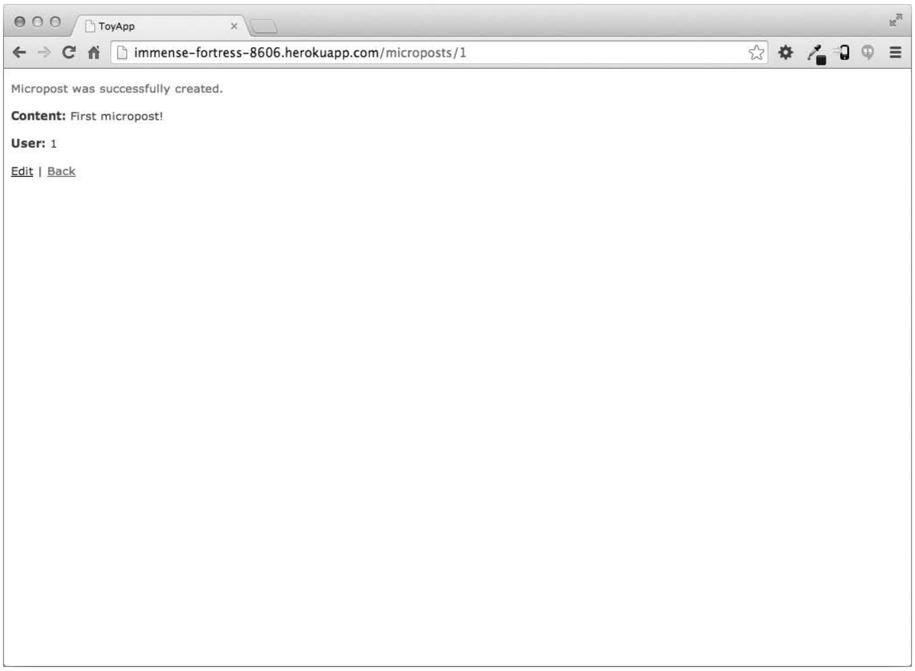
At this point, you can also deploy the toy app to Heroku as explained in Section 1.5:

```
$ git push heroku
```

(This assumes you created the Heroku app in Section 2.1. Otherwise, you should run `heroku create` and then `git push heroku master`.)

To get the application's database to work, you'll also have to migrate the production database:

```
$ heroku run rake db:migrate
```



Micropost was successfully created.

Content: First micropost!

User: 1

Edit | Back

Figure 2.18 Running the toy app in production.

This updates the database at Heroku with the necessary user and micropost data models. After running the migration, you should be able to use the toy app in production, with a real PostgreSQL database back-end (Figure 2.18).

2.4 Conclusion

We've now come to the end of the high-level overview of a Rails application. The toy app developed in this chapter has several strengths and a host of weaknesses.

Strengths

- High-level overview of Rails
- Introduction to MVC
- First taste of the REST architecture

Beginning data modeling

A live, database-backed web application in production

ea nesses

No custom layout or styling

No static pages (such as “Home” or “About”)

No user passwords

No user images

No logging in

No security

No automatic user/micropost association

No notion of “following” or “followed”

No micropost feed

No meaningful tests

o real understanding

The rest of this tutorial is dedicated to building on these strengths and eliminating the weaknesses.

2.4.1 What We Learned in This Chapter

Scaffolding automatically creates code to model data and interact with it through the web.

Scaffolding is good for getting started quickly but is bad for gaining understanding.

Rails uses the Model–View–Controller (MVC) pattern for structuring web applications.

As interpreted by Rails, the REST architecture includes a standard set of URLs and controller actions for interacting with data models.

Rails supports data validations to place constraints on the values of data model attributes.

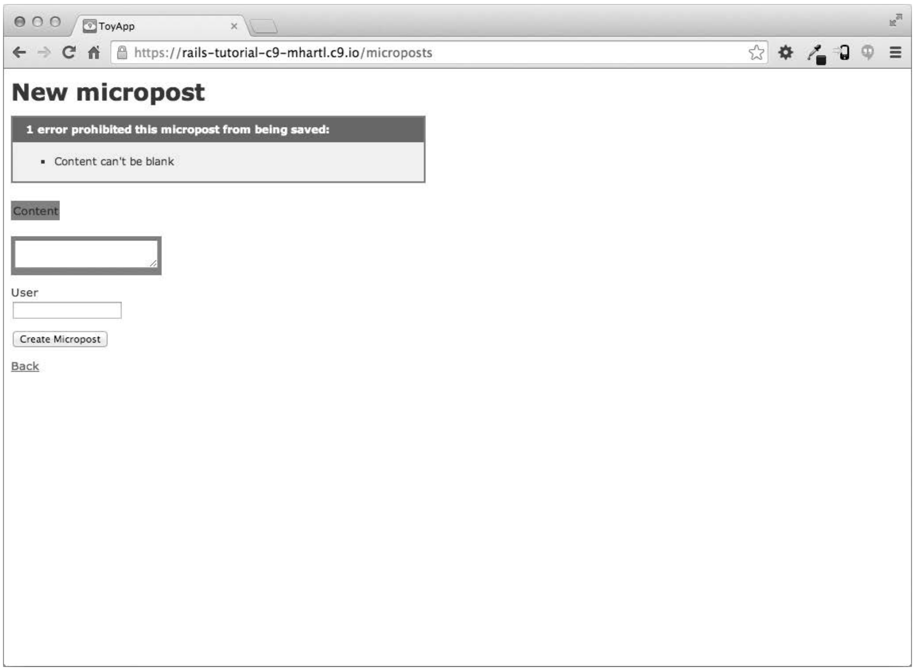
Rails comes with built-in functions for defining associations between different data models.

We can interact with Rails applications at the command line using the Rails console.

2.5 Exercises

Note: To receive a copy of the *Solutions Manual for Exercises*, with solutions to every exercise in the *Ruby on Rails*™ *Tutorial*, visit www.railstutorial.org/ word , where " word " is the last word of the Figure 3.9 caption.

1. The code in Listing 2.18 shows how to add a validation for the presence of micropost content to ensure that microposts can't be blank. Verify that you get the screen shown in Figure 2.19.
2. Update Listing 2.19 by replacing `FILL_IN` with the appropriate code to validate the presence of name and email attributes in the User model (Figure 2.20).



New micropost

1 error prohibited this micropost from being saved:

- Content can't be blank

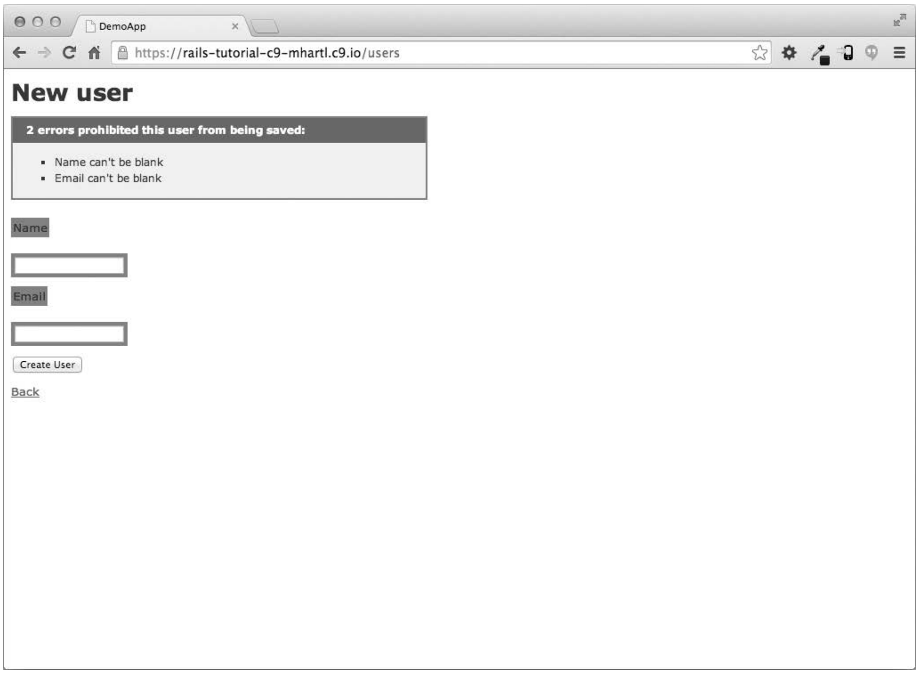
Content

User

Create Micropost

Back

Figure 2.19 The effect of a micropost presence validation.



New user

2 errors prohibited this user from being saved:

- Name can't be blank
- Email can't be blank

Name

Email

Create User

Back

Figure 2.20 The effect of presence validations on the User model.

Listing 2.18 Code to validate the presence of micropost content.
app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  belongs_to :user
  validates :content, length: { maximum: 140 },
                      presence: true
end
```

Listing 2.19 Adding presence validations to the User model.
app/models/user.rb

```
class User < ActiveRecord::Base
  has_many :microposts
   validates FILL_IN, presence: true
   validates FILL_IN, presence: true
end
```

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

/ (slash)
  root URL, 58
  Unix directory separator, 7
( ) (parentheses)
  in statement syntax, 383n
  and the ternary operator, 395
! (exclamation point)
  bang methods, 154
  not operator, 148
? (question mark)
  boolean indicator, 147
  in query parameters, 483
 (double quotes)
  mixing with single quotes, 128
  in string literals, 144
 (curly braces)
  block indicator, 156
  hash indicator, 158, 163
\ (backslash)
  as escape character, 146
  as literal character, 146
# (hash symbol)
  comment indicator, 18, 143
  styling CSS IDs (Listing 5.7), 193–194, 657
  URL stub, 184
 (vertical bars)
  block variable indicator, 156
  or operator, 148
 (dollar sign)
  command-line prompt, 6
  dollar-sign syntax, 657
  Sass variables, 207
: (colon), as a hashkey, 159–161
!! (exclamation points), coercing boolean values, 149, 269n22
!= (exclamation point, equal sign), not equal operator, 153
 (single quotes), mixing with double quotes, 128
[ ] (square brackets), accessing elements, 153
@ (at sign), instance variables, 69, 207
/* ... */ (slash, asterisk...), CSS comments, 189
&& (ampersands), and operator, 148, 346
+ (plus operator), concatenating strings, 144
 (less-than signs), adding to arrays, 154
== (equal signs), equality operator, 153
 = (vertical bars, equal sign), or equals assignment operator, 355–356
=> (equal, greater than), hashrocket, 159

A
-a flag, 39–40
About page
  creating, 109–110
  mockup, *217*
  sample page, 110
  testing, 104–111
Abstraction layers, 232n
Acceptance tests, 421
Account activation
  edit action (Listing 10.28), 499
  generalized authenticated? method (Listings 10.23 to 10.26), 496–498
  metaprogramming, 494
  overview, 471–473, 493–501
  preventing unactivated users from logging in (Listing 10.29), 499–501
  versus remembering, login, and password reset, 473*t*
  sign-up page with warning message, *501*
  successfully activated profile page, *500*
Account activation mailer method
  account activation HTML view (Listing 10.12), 484
  account activation text view (Listing 10.11), 483
  Action Mailer library, 480
  adding account activation to user sign-up (Listing 10.20), 490
  assert_match method (Listing 10.17), 488–489
  commenting out failing tests (Listing 10.21), 491
  creating an activation email (Listing 10.10), 481–482
  email, sent in production, *534*
  email previews (Listings 10.13 to 10.15), 484–486
  escaping out special characters, 483
  generating a mailer (Listings 10.6 to 10.9), 480–481
  Home page with activation message, *492*
  mailing the account activation link (Listing 10.10), 482
  query parameters, 483
  sample activation email (Listing 10.22), 491–493
  setting the test domain host (Listing 10.18), 489
  User mailer test (Listings 10.16, 10.17), 486–489
  User mailer test, screen mockups, *487*, *488*
Account activation resource
  activating fixture users (Listing 10.5), 479
  activating seed users (Listing 10.4), 478
  activation token, 474
  adding a resource (Listing 10.1), 474
  method reference technique, 476
  migration for account activation (Listing 10.2), 475
  overview, 473–479
  User model with account activation attributes (Listing 10.3), *475*
Account activation test
  activation via user model object (Listing 10.34), 505
  integration test (Listing 10.30), 502–503
  overview, 502–506
  sending email via user model object (Listing 10.33), 504–505
  user activation methods in the User model (Listing 10.32), 504
  user sign-up test (Listing 10.30), 502–503
Action Mailer library, 480
Actions, corresponding views (Listings 3.7, 3.8), 99–100
Activation tokens, 474
Active Record. *See also* SQL (Structured Query Language); Validating user data
  definition, 228
  finding User objects, 238–240
  initializing objects, 236
  making and saving a model, 235–238
  uniqueness, at the database level, 257–258
  uniqueness, validating (Listings 6.23 to 6.26, 6.28), 254–261
  validating users, 241–242. *See also* Validating user data
Active Record::RecordNotFound exception, 239
Active relationships, 618–619, *619*

add_column method (Listing 6.32), 262
add_index method, 259
Adler, Yoel, 438n
admin attribute, 456–459
admin? boolean method, 457–459
Administrative users
  admin attribute, 456–459
  admin? boolean method, 457–459
  deleting (Listings 9.50, 9.51), 456–459, *458*
  identifying, 456–459
Ajax (Listings 12.33 to 12.38), 653–658
Ampersands (&&), and operator, 148, 346
Anonymous functions, 553
any? method, 311
app/assets directory, 200
ApplicationController class (Listing 2.17), 80
application.html.erb file, 119–120
Applications
  getting started. *See* Hello world program
  interacting with. *See* Console
  samples. *See* Hello world program; Static pages; Toy application
Architectural Styles and the Design of Network-based Software Architecture (Fielding), 67n
Arrays. *See also* Ranges
  ! (exclamation point), bang methods, 154
  != (exclamation point, equal sign), not equal operator, 153
  [ ] (square brackets), accessing elements, 153
  == (equal signs), equality operator, 153
  adding to, 154
  bang methods, 154
  classes, 165
  converting ranges to, 154–155
  converting strings to, 152
  converting to strings, 154
  versus hashes, 159
  indexing, 152
  mutating, 154
  pulling out elements, 155
  selecting every element, 155
  of strings, 249
  zero-offset, 152
assert method (Listings 6.4 to 6.5), 243
assert_match method (Listing 10.17), 488–489
assert_select method
  examples of (Listing 5.22), 219*t*
  sign-up failure, 314
  testing for HTML tags (Listing 3.22), 112–113
Asset directories, 200–201
Asset pipeline
  asset directories, 200–201
  definition, 200
  manifest files (Listing 5.14), 201–202
  preprocessor engines, 202
  production efficiency, 202–203
Associations between Micropost models (Listings 11.9 to 11.12), 548*t*
Associations between Micropost models (Listings 11.9, 11.12), 545–550
Associative arrays. *See* Hashes
At sign (@), instance variables, 69, 207
attr_accessible method, 307–308
attr.accessor method (Listing 4.13), 230
Attribute accessors, creating, 173
authenticate method
  authenticating passwords, 268
  authenticating users (Listing 8.5), 345–346
  examples (Listing 8.5), 346*t*
authenticated? method (Listings 10.23 to 10.26), 496–498
Authenticating users (Listing 8.5), 345–346, 346*t*
Authentication
  versus authorization, 424
  passwords, 267–269
Authentication systems, rolling your own, 227–228
Authenticity token, sign-up forms, 302
Authorization, friendly forwarding
  create action (Listing 9.29), 439–440
  implementation code (Listing 9.27), 437–438
  logged_in_user before filter (Listing 9.28), 437–438
  overview, 436–440
  redirect_back_or method (Listing 9.29), 439–440
  testing for (Listing 9.26), 436–437

Authorization, requiring logged-in users
adding a logged_in_user method (Listings 9.12), 426
before_action command (Listings 9.12, 9.16, 9.18), 425–431
before filters (Listings 9.12, 9.14, 9.16, 9.18), 425–431
logged_in_user before filter (Listings 9.12, 9.14, 9.16, 9.18), 426–431
logging in a test user (Listing 9.14), 428
login form mockup, a protected page, *425*
login form mockup, access failure, *427*
overview, 425–431
testing edits (Listing 9.17), 429–430
Authorization, requiring the right user
adding a user to the fixture file (Listing 9.20), 431
correct_user before filter (Listings 9.22, 9.25), 433–436
current_user? method (Listing 9.24), 434–435
overview, 431–436
testing a correct user (Listing 9.22), 433–434
testing the wrong user (Listing 9.21), 432
Authorization *versus* authentication, 424
Automated testing, 101–104, 129–135
aux options, 134–135
Avatar, definition, 288n

Backing up code, 33–36
Backslash (\)
as escape character, 146
as literal character, 146
Backtrace silencer (Listing 3.41), 129
Bang bang (!!), coercing boolean values, 149, 269n22
Bang methods, 154
bar, word origin, 144
Base titles, 139
Bash command-line interface, 13
Bates, Mark, 13
bcrypt hash function (Listing 6.33), 263–264, 269n21
Before filters (Listings 9.12, 9.14, 9.16, 9.18), 425–431
before_action command (Listings 9.12, 9.16, 9.18), 425–431
before_filter command, 426n
Beginning Ruby (Cooper), 5
belongs_to associations (Listings 11.9, 12.3), 547–550, 623
Bitbucket (Listing 1.12)
overview, 33–36, *36*
pushing changes up to, 41
Block variables, definition, 155–156
Blocks
as closures, 156n
definition, 155–156
map method, 157
multiline, 156
overview, 155–158
Books and publications. *See also* Online resources
Architectural Styles and the Design of Network-based Software Architecture (Fielding), 67n
Beginning Ruby (Cooper), 5
“Class and Instance Variables in Ruby,” 169
Conquering the Command Line (Bates), 187
Learn Ruby on Rails (Kehoe), 5–6
Learn to Program (Pine), 5
Patterns of Enterprise Application Architecture (Fowler), 228n1
Pro Git, 38n
“The Rails 3 Asset Pipeline in (about) 5 Minutes” (Erasmus), 200n
The Rails 4 Way (Fernandez), 151n5
“Rails Routing from the Outside In,” 633n
Ruby on Rails™ Tutorial website, 1n
“Shiny new commit styles,” 40
Boolean values
combining, 148
in control flow, 147
identifying, 147
Bootstrap, drop-down menus (Listing 8.17), 361
Bootstrap and custom CSS. *See also* Site layout
adding CSS (Listings 5.4, 5.5), 189–190
adding the bootstrap.sass gem (Listing 5.3), 188
color (Listing 5.7), 194
Home page mockup, *190–191*

  installing Bootstrap, 188
  overview, 187–188
  site logo (Listing 5.7), 194, *195*
  typography (Listing 5.6), 192–194, *193*
bootstrap.sass gem (Listing 5.3), 188
bootstrap-will_paginate method (Listing 9.40), 448–449
box_sizing, 280–281
branch command, 37–38
Branches. *See also* Git, branching
  current branch, identifying, 37–38
  deleting, 41
  editing, 39–40
  listing, 37–38
  merging, 40–41
  pushing changes up to Bitbucket, 41
  status, displaying, 39–40
Built-in classes, modifying, 169–170
Built-in functions, 138
bundle command, 20n9, 20–21, 93*t*
bundle exec, 57
Bundler, 15–21
Buttons, web page
  follow buttons, *616*, *640*, 658–660
  linking to (Listing 5.34), 221–223
  unfollow buttons, *617*, *641*, 658–660
byebug gem (Listing 3.2), 287

C
Callbacks, databases, 260
CamelCase, 94
CarrierWave uploader
  adding to the Gemfile (Listing 11.56), 594
  configuring for production (Listings 11.65, 11.66), 604–606
  description, 594
  MiniMagick interface (Listing 11.64), 601–603
  uploading images (Listings 11.57 to 11.60), 594–597
  uploading images in production (Listings 11.65, 11.66), 603–606
  validating images, 597–601
Cascading style sheets (CSS). *See* CSS (cascading style sheets)
Case sensitive indices, 259–261
Case sensitivity
  downcasing email addresses (Listing 6.31), 260
  email addresses (Listings 6.25 to 6.26), 255–256
cat command (Listing 1.11), 34
cd (change directory) command, 13
C lio, Paulo J nior, 387n
Change directory (cd) command, 13
change method (Listing 6.2), 232
Changes, overwriting, 33
checkout (co) command, 30, 33, 37
Clark, Mike, 6
Clark, Nicole, 6
"Class and Instance Variables in Ruby," 169
Class methods, 166
Class names, converting to IDs, 622
Classes. *See also* CSS classes
  arrays, 165
  attribute accessors, creating, 173
  built-in, modifying, 169–170
  checking for palindromes, 166–169
  class methods, 166
  constructors, 165–166
  defining (Listing 4.12), 167–169
  definition, 99
  description, 164
  getter methods, creating, 173
  hashes, 165
  hierarchy, determining, 166–169
  inheritance, 166–169, *168*
  instance method, 166
  instance variables, creating, 173
  instances of, 166
  literal constructors, 165–166
  named constructors, 165–166
  new method, 165–166
  self keyword, 168–169
  setter methods, creating, 173
  Static Pages controller, 170–172
  superclass method, 166–169
  User class (Listing 4.13), 172–174
Client computer, definition, 97–98
Closures, 156n

Cloud integrated development environment. *See* Cloud9
Cloud storage
  for micropost images, 604–606
  S3 (Simple Storage Service), 604–606
Cloud9. *See also* Installing Rails on Cloud9; Local development environment
  creating a new workspace, *11*
  Find in Files global search, 9
  getting started, 9
  local servers, sharing, *24*
  Rails Tutorial Cloud9 workspace, 9
  setting indentation, 10, *12*
  signing up, 9n7
  user interface, *10*
co (checkout) command, 30, 33, 37
Code generation, undoing, 94–95
Code School, 6
.coffee file extension, 202
Colon (:), as a hashkey, 159–161
Color (Listing 5.7), 194
Columns, in databases, 230
Commands. *See also specific commands*
  Heroku, 45–47
  Unix, 14*t*
Commenting out failing tests (Listing 10.21), 491
Comments
  /* ... */ (slash, asterisk...), CSS comments, 189
  # (hash symbol), comment indicator, 18, 143
  in the console, 143
commit command, 31, 39–40
Commit messages, displaying, 32
Committing changes, 31, 39–40
config directory, listing contents of, *96*
Confirmation of passwords, 264–265, 267–269
Conquering the Command Line (Bates), 13, 187
Console
  comments, 143
  definition, 142
  environments, 143, 278–280
  exiting, 143
  irb configuration (Listing 4.8), 142
  message chaining, 148
  message passing, 146–149
  method definitions, 149–151
  in Micropost resource, 78
  objects, 146–149
  prompts, 78n
  starting, 142–143
  strings, 144–146
  toy application, 78
console command, 78
Constructors, 165–166
Contact page
  adding actions (Listing 5.19), 212
  adding routes (Listing 5.18), 212
  integration test for layout links (Listing 5.25), 216–219
  named routes (Listings 5.23 to 5.24), 215–216
  root routes (Listing 5.22), 213–215
  testing (Listing 5.16), 211–212
  view for (Listing 5.20), 213
content_for function, 117n14
Controller class. *See* Static Pages controller
Controller tests. *See* Testing, Static Pages controller
Controllers, generating, 92–100. *See also* Sessions controller; Static Pages controller
Conventions in this book, 6–8
Converting
  arrays to strings, 154
  ranges to arrays, 154–155
  strings to arrays, 152
Cookies
  decrypting, 379–380
  encrypting, 378–379
  expiration, 377–378
  permanent, 377–385
  remembering user login, 372
  session hijacking, 372
  signed, 378–379
  stealing, 372
cookies method, 336
cookies.permanent method, 377–385
Cooper, Peter, 5
Coordinated Universal Time (UTC), 237n
correct_user before filter (Listings 9.22, 9.25), 433–436

count method
  showing microposts, 558
  sign-up failure, 315
Counter cache, showing microposts, 558
cp (copy files) command, 13
create action
  creating microposts (Listing 11.35), 572–580
  creating sessions (Listing 8.4), 343
  friendly forwarding (Listing 9.29), 439–440
  logging in, 337
  password reset (Listing 10.40), 515
  sign-up success (Listing 7.23), 317–318
create command, 44–45
Create directory (mkdir) command, 13
create method, creating users, 237–238, 267–269
create_table method (Listing 6.2), 232
Creating
  About page, 109–110
  account activation email (Listing 10.10), 481–482
  attribute accessors, 173
  database indices, 257.258
  fixtures (Listing 8.19), 364
  following relationships (Listings 12.37, 12.38), 657–658
  getter methods, 173
  Help pages, 92–93
  instance variables, 173
  microposts. *See* Microposts, creating
  persistent sessions, 372
  remember me checkbox (Listings 8.47 to 8.49), 391–397
  sample users (Listings 9.38, 9.39), 446–448, *448*
  sessions (Listing 8.4), 343
  Sessions controller, 336–337
  setter methods, 173
  Static Pages controller class, 170–172
  temporary sessions (Listings 8.12, 8.13), 351–353
  User objects, 235–238
  user profile pages, 282–286
  users, 58–60, 237–238, 267–269
Creating, microposts
  adding to the Home page (Listing 11.36), 573–574
  create action (Listing 11.35), 572–580
  creating (Listings 11.35 to 11.38), 572–575
  form for creating microposts (Listing 11.36, 11.38), *572*, 572–575
  pluralize method, 574
  redefining error messages (Listings 11.40 to 11.44), 576–577
  user info sidebar (Listing 11.37), 574
Cropping images, *421*
CSRF (cross-site request forgery), 301
CSS (cascading style sheets). *See also* Site layout
  (curly braces), hash indicator, 163
  including style sheets (Listing 4.11), 162–164
  rules, bundling and reusing. *See* Sass mixins (Listing 7.2)
  Sass mixins (Listing 7.2), 280–281
  stylesheet_link_tag function, 138, 164
  styling error messages with (Listing 7.20), 312–313
  tool for. *See* Sass utility
CSS classes, 184
.css file extension, 189
Ctrl-C keystroke
  case sensitivity, 22
  restarting the console, 143
  shutting down a server, 22
Ctrl-D keystroke
  case sensitivity, 78
  exiting Guard, 135
  exiting the console, 78
curl utility, 186–187
Curly braces ( )
  block indicator, 156
  hash indicator, 158, 163
Current user, finding (Listing 8.14), 353–357
current_user, updating (Listing 8.36), 384
current_user method (Listing 8.14), 353–357
current_user? method (Listing 9.24), 434–435
Custom environments, 278n

D
-d flag, 41
-D flag, 41

Data models
  associations (Listings 2.11, 2.12), 76–78, *77*
  toy application (Listing 2.6), 54–55, *54–55*, 69
Data structures. *See also* CSS (cascading style sheets)
  arrays, 152–155
  blocks, 155–158
  hashes, 158–162
  ranges, 152–155
  symbols, 158–162
Data validation, toy application, 74–76. *See also* Validating user data
Database indices, creating, 257, 258
Database migration. *See also* Rake db:migrate command
  definition, 231
  dropping tables, 234
  magic columns, 232, 237
  migrating down, 234
  migrating the User model (Listing 6.2), 231–234
  migrating up, 232
  migration file name, 232
  reversing. *See* Undoing, database migration
  rolling back. *See* Undoing, database migration
  in a sandbox, 235
  timestamp, 232
  undoing, 95, 234, 235, 268n
Databases. *See also* User model
  callbacks, 260
  case-sensitive indices, 259–261
  columns, 230
  full-table scan, 258
  interacting with. *See* Active Record; SQL (Structured Query Language)
  method reference convention, 260
  resetting, 268n. *See also* Undoing, database migration
  reversing changes. *See* Undoing, database migration
  rolling back changes. *See* Undoing, database migration
  rows, 230
  tables, 230
  tasks, listing, 58
Date and time, User objects, 237
db/development.sqlite3 file, 233
db:rollback command, 234
db/seeds.rb file (Listing 9.39), 447
de Ron, Niels, 387n
Debug, and Rails environments (Listing 7.1), 276–282
Debug box, sample code (Listing 7.21), 280
Debug information, sign-up failure (Listing 7.16), 303–306
debug method (Listing 7.1), 277
debugger
  byebug gem (Listing 3.2), 287
  use for, 287–288
  with Users controller (Listing 7.6), 287
Decrypting cookies, 379–380
Default scope for Micropost models (Listings 11.13 to 11.17), 551–553
default_scope method (Listing 11.16), 552–553
defined? method (Listing 8.50), 398–399
DELETE request, 98
DELETE requests from web browsers, 359n
Deleting. *See also* destroy action; Destroying
  branches, 41
  files and directories, 32
Deleting users
  administrative users (Listings 9.50, 9.51), 456–459, *458*
  destroy action (Listings 9.52 to 9.54), 460–462
  overview, 456
  strong parameters, 459
  testing destroyed users (Listings 9.55 to 9.57), 462–465
  users index with delete links, mockup, *457*, *461*
dependent: :destroy method (Listings 11.19), 554–555
Deployment
  README file, *42*
  tools for, 42. *See also* Heroku
  toy application, 81–82, *82*
Deployment, professional grade
  overview, 326
  SSL certificates, 327

SSL (Secure Sockets Layer) in production (Listing 7.27), 326–327
destroy action
  deleting sessions (Listings 8.26 to 8.28), 369–371
  deleting users (Listings 9.52 to 9.54), 460–462
  destroying microposts, 587–589
  logging out, 337
destroy command, 94–95
Destroying. *See also* DELETE request; Deleting
  following relationships (Listing 12.38), 658
  Micropost model users (Listings 11.18, 11.19), 554–555
  microposts, 586–589
  sessions (Listings 8.26 to 8.28), 369–371
  User objects, 238
  users, toy application, 60, *65*
Development environment
  cloud. *See* Cloud9
  local, 9n5
  setting up. *See* Cloud9
Development environments, 278n
development.sqlite3 file, 233
Dictionary attacks, protection from, 269n21
Directories
  contents, listing, 32
  deleting, 32
Displaying. *See also* Listing; Showing
  branch status, 39–40
  commit messages, 32
  email addresses, 284
  .gitignore file, *134*
  IP binding address, 21
  port number, 21
  sign-up error messages (Listings 7.17, 7.18), 308–313
  user names, 284
Displaying user data
  custom environments, 278n
  debug, and Rails environments (Listing 7.1), 276–282
  development environments, 278n
  lorem ipsum text, *277*, 560n–561n
  production environments, 278n
  Rails environments, 278–280
  show page, mockups, *285–286*
  test environments, 278n
  user email addresses, 284
  user names, 284
  user profile page, mockup, *276*
  Users controller with a show action (Listing 7.5), 285
div tag, 183–184
Doctype declarations (Listing 3.21), 112
Document divisions, 184
Document type (Listing 3.21), 112
Dollar sign ( )
  Command-line prompt, 6
  dollar-sign syntax, 657
  Sass variables, 207
Double quotes (“ ”)
  mixing with single quotes, 128
  in string literals, 144
Downcasing email addresses (Listing 6.31), 260
Drop-down menus, mockup, *362*
drop_table command, 234
DRY (Don’t Repeat Yourself) principle, 116
Duplication, eliminating. *See* Refactoring code
Dynamic content, page titles
  base titles, 139
  on every page. *See* Base titles
  full_title helper function (Listings 4.2, 4.3, 4.9), 139, 151
  HTML structure of a typical web page (Listing 3.21), 112
  layout files (Listing 3.32), 119–120
  list of sample pages, 111*t*
  missing page titles, 139
  overview, 111
  RED testing, 112–113
  templates for, 117
  yield function, 117

E

each method, 161
edit action
  account activation (Listing 10.28), 499
  password reset (Listing 10.50), 524–525
  updating users (Listings 9.1 to 9.4), 412–417
Edit form, updating users (Listing 9.3), 415

Edit form mockups
  initial edit page, *415*
  unsuccessful edits, *419*
  updating users, *415*
Edit view (Listing 9.2), 414
Editing
  branches, 39–40
  Git branches, 39
  Rails router, 27–28
  user information, toy application, 59, *62*, *63*
elseif statement, 147
Email. *See also* Account activation mailer method; Password reset mailer method
  account activation, sample (Listing 10.22), 491–493
  email via user model object (Listing 10.33), 504–505
  fixtures (Listing 6.29), 259
  generating an account activation mailer (Listings 10.6 to 10.9), 480–481
  mailing the account activation link (Listing 10.10), 482
  User mailer test (Listings 10.16, 10.17), 486–489
  User mailer test, screen mockups, *487*, *488*
Email, sending in production
  configuring SendGrid (Listing 10.55), 532–534
  overview, 532–534
Email addresses
  displaying, 284
  user names as, 230n
  validating user data (Listings 6.18 to 6.19, 6.21, 6.23), 253–254
Email addresses, validating
  case sensitivity (Listings 6.25 to 6.26), 255–256
  downcasing (Listing 6.31), 260
  format (Listings 6.18 to 6.19, 6.21), 249–254
  with regular expressions (Listing 6.21), 252–254, 252*t*
  uniqueness (Listings 6.23 to 6.26, 6.28), 254–261
  uniqueness at the database level, 257–258
Email previews, (Listings 10.13 to 10.15), 484–486
Embedded Ruby
  definition, 117
  dynamic page titles (Listings 3.28 to 3.31), 117–119
Embedded Ruby (Listings 3.28 to 3.35), 116–121
empty method, and the nil value, 148
empty? method, 311
Enabling SSL, 326–327
Encrypting cookies, 378–379
Encryption *versus* hashing, 262n19
Environments, in the console, 143. *See also* Development environment; Production environment; Test environment
Equal, greater than (=), hashrocket, 159
Equal signs (==), equality operator, 153
Erasmus, Michael, 200n
ERb. *See* Embedded Ruby
.erb file extension, 202
Error messages
  failed micropost creation, 75, *77*
  Googling, 5
  invalid multibyte character, 48n
  Rails Tutorial help, 7
  README.doc file missing, 36
  redefining, microposts (Listings 11.40 to 11.44), 576–577
  Spring process ID (pid) file will be overwritten, 124n
  toy application, *77*
  You have not declared a Ruby version..., 329
error_explanation (Listing 7.20), 312–313
_error_messages.html.erb partial file (Listing 7.19), 310–311
Escaping out special characters, 483
Exceptions
  definition, 239
  raising (Listing 8.53), 401–402
  removing (Listing 8.57), 404
Exclamation point (!)
  bang methods, 154
  not operator, 148

Exclamation point, equal sign (!=), not equal operator, 153
Exclamation points (!!), coercing boolean values, 149, 269n22
Expressions, short-circuit evaluation, 356. *See also* Regular expressions

F

f flag, 33
Failed password reset, *529*
Failure-page mockups, *303*, *305*
Faker gem
  adding to the Gemfile (Listing 9.38), 445–446
  creating sample data, 446–447
  lorem ipsum text, *277*, 560n–561n
false command, 232
Feed. *See* Microposts, proto-feed
@feed_items instance variable (Listings 11.46 to 11.49), 582–586
Fernandez, Obie, 151
Fielding, Roy, 67
field_with_errors class (Listing 7.20), 312–313
File compression, uglifier gem, 19
File navigator, advanced testing, *132–133*
Files
  deleting, 32
  Git staging area, 31
Find in Files global search, 9
find method (Listing 7.5), 354–357
find_by method, finding users (Listing 8.5), 345–346, 354–357
Finding
  User objects, 238–240
  users (Listing 8.5), 343–346, 354–357
  users (Listing 8.14), 353–357
first_user variable, 78
Fixtures
  adding a digest method (Listings 8.18, 8.19), 363–364
  creating (Listing 8.19), 364
  definition, 362
Flash, in sessions
  code for failed login (Listing 8.9), 350
  flash test (Listing 8.7), 348–350
  flash.now method (Listing 8.9), 350
  rendering with a flash message (Listing 8.6), 346–348, 347*t*
Flash messages (Listings 7.24, 7.25), 319–321
flash.now method (Listing 8.9), 350
Follow buttons
  mockups, *616*, *640*
  testing (Listing 12.39), 658–660
Followed users (Listings 12.8 to 12.10), 625–628
Followers
  definition, 615
  Relationship model (Listings 12.12, 12.13), 628–630
  testing for (Listing 12.13), 630
Followers page, following user mockups, *644*, *647*, *648*
followers.include? method, *629*
Following an unfollowed user, *654*
Following associations (Listings 12.8 to 12.10), 625–628
Following page
  following user mockups, *643*
  users following mockups, *646*
Following relationships
  creating (Listings 12.37, 12.38), 657
  destroying (Listing 12.38), 658
  testing (Listing 12.39), 658–660
Following users, definition, 616
Following users, Relationship model
  active relationships, 618–619, *619*
  adding indices (Listing 12.1), 620
  association methods, summary of, 623*t*
  belongs_to associations (Listing 12.3), 623
  data model, *620*
  data model problem, 615–621
  enforcing uniqueness, 620
  followed users (Listings 12.8 to 12.10), 625–628
  followers (Listings 12.12, 12.13), 628–630
  followers.include? method, *629*
  following association (Listings 12.8 to 12.10), 625–628
  foreign keys, 622
  has_many associations, 621–622
  has_many :through association (Listings 12.8 to 12.10), 625–628

Following users, Relationship model (*continued*)
  multiple-key indices, 620
  passive relationships (Listing 12.12), 618–619, 628–630
  passive relationships, model for, *629*
  Twitter-style relationships, 618
  user.followers method (Listings 12.12, 12.13), 628–630
  user.following method (Listings 12.8 to 12.10), 625–628
  validations (Listings 12.4 to 12.6), 623–625
Following users, status feed
  from_users_followed_by method (Listings 12.45, 12.46), 667–671
  home action (Listing 12.48), 669
  initial status feed (Listing 12.43), 663–666
  join method, 664
  map method, 663–666
  motivation and strategy (Listing 12.41), 662–663
  sample, *661*
  subselects (Listings 12.45 to 12.48), 667–671
Following users, web interface
  access control for relationships (Listings 12.30, 12.31), 651–653
  displaying statistics, *632*, *638*
  displaying statistics (Listings 12.15 to 12.23), 632–642
  follow buttons, creating (Listings 12.30 to 12.38), 651–653
  followers action (Listing 12.25), 644–645
  following action (Listing 12.25), 644–645
  following/followers pages (Listings 12.24 to 12.28), 642–650
  follow/unfollow forms (Listings 12.15 to 12.23), 632–642
  Relationships controller (Listing 12.32), 652
  resources block, 632
  routes for user relationships (Listing 12.20), 638–639
  sample following data (Listing 12.14), 631–632
  show_follow view (Listing 12.26), 645–647
  testing authorizations (Listing 12.24), 642–643
Following-user mockups
  following an unfollowed user, *654*
  status feed on the live web, *671*
  unfollowed user, *653*
  user followers page, *644*, *647*, *648*
  users following page, *643*, *646*
Following-user mockups, Home page
  with follow statistics, *638*
  incremented followers count, *618*
  status feed, *618*, *661*, *670*
Following-user mockups, user profiles
  basic page, *614*
  finding a user to follow, *615*
  follow button, *616*, *640*
  incremented followers count, *617*
  unfollow button, *617*, *641*
foo, word origin, 144
foobar, word origin, 144
.footer classes (Listing 5.11), 197n
footer tag (Listing 5.11), 197n
Footers
  Home page mockup, *199*
  partials for (Listings 5.11 to 5.13), 197–199
  site layout (Listings 5.11 to 5.13), 197–199
Foreign key reference, 543
Foreign keys, 622
forget method (Listing 8.38), 385–386
forgot password form. *See also* Password reset
  basic, *507*
  invalid email address, *516*
  modified, *514*
  valid email address, *521*
forgot password link, *507*
form tag, sign-up forms, 300–301
form_for helper, 339–340
form_for method, 294–298
form_tag helper, 340n
Fowler, Martin, 228n1
from_users_followed_by method (Listings 12.45, 12.46), 667–671
FUBAR, word origin, 144
Full-table database scan, 258
full_title helper function (Listings 4.2, 4.3, 4.9), 139, 151

Functions. *See also* Methods
  anonymous, 553
  built-in, 138
  definition, 99
  helpers. *See also specific helpers*
  implicit return, 150
  versus methods, 147n
  nameless. *See* Anonymous functions
Functions, helpers
  definition, 138
  specific to a particular controller, 139n

gem command
  installing Rails, 11–12
  specifying a version range (Listings 1.1, 1.3, 1.4, 1.5), 11, 13–21
Gemfile
  static pages (Listing 3.2), 88–89
  updating (Listing 2.1), 52–53
generate command (Listing 3.4), 92–93, 94–95
generate model command, 231
Generating code. *See* Scaffolding
GET request, 98
Getter methods, creating, 173
Getting started
  with Cloud9, 9
  first application. *See* Hello world program
  lowering the barrier, 2
  overview, 2–4
Git
  adding the public key, 34, *35*
  backing up code, 33–36
  benefits of, 32–33
  Bitbucket (Listing 1.12), 33–36, *36*
  cat command (Listing 1.11), 34
  checkout command, 33, 37
  deleting files and directories, 32
  f flag, 33
  GitHub, 33
  hosting sites, 33
  listing directory contents, 32
  ls command, 32
  overwriting current changes, 33
  preventing conflicts Spring server, 131
  q command, 32
  quitting, 32
  README.doc file missing, 36, *38*
  README.md file (Listing 1.13), 39
  -rf flag, 32
  rmfile command, 14*t*, 32
Git, branching
  branch command, 37–38
  branch status, displaying, 39–40
  current branch, identifying, 37–38
  deleting branches, 41
  editing branches, 39–40
  listing branches, 37–38
  merging results, 40–41
  pushing changes up to Bitbucket, 41
Git, installation and setup
  adding messages to the commit, 31
  commit command, 31, 39–40
  commit messages, displaying, 32
  committing changes, 31, 39–40
  first-time repository, 30–32
  first-time system, 30
  .gitignore file, 31
  log command, 32
  -m flag, 31
  overview, 30–32
  staging area for files, 31
  status command, 31
Git, repositories
  backing up, 33
  pushing changes up to, 33–36, 41
GitHub, 33
.gitignore file
  adding Spring server (Listing 3.43), 132–133
  displaying, *134*
  generating, 31
Googling error messages, 5
Goto Anything command, 87n
.gravatar_edit class (Listing 7.10), 292n
gravatar_for helper function (Listing 7.8), 289–290
Gravatars (globally recognized avatars) (Listings 7.7, 7.8), 288–293
GREEN flag, 8

GREEN (passing) tests (Listings 3.15, 3.20), 107–110
Guard program, 129–135

H

Haml template system, 117n13
Hansson, David Heinemeier, 4, 102n
Hash symbol (#)
  comment indicator, 18, 143
  styling CSS IDs (Listing 5.7), 193–194, 657
  URL stub, 184
Hashed passwords (Listings 6.32, 6.33), 261–264
Hashes
  : (colon), as a hashkey, 159–161
  (curly braces), hash indicator, 158, 163
  => (equal, greater than), hashrocket, 159
  versus arrays, 159
  classes, 165
  computational cost, 363
  definition, 262
  each method, 161
  indices, 158
  inspect method, 162
  keys, 158
  as last argument, 163
  nested (Listing 4.10), 161
  p function, 162
  printing, 162
  symbols as hash keys, 159–161
Hashing
  versus encryption, 262n
  MD5 hashing algorithm, 289
has_many associations
  Relationship model, 621–622
  toy application, 76–78, *77*
has_many relationships, *548*
has_many :through association (Listings 12.8 to 12.10), 625–628
has_secure_password method (Listing 6.34), 261–265, 269
header tag, 183–184
Headers
  partials for (Listing 5.10), 196–197
  site layout (Listing 5.10), 183, 196–197
Hello world program
  controller action (Listing 1.8), 25–29
  creating a skeleton, 12
  default page, *26*
  making a workspace directory (Listing 1.2), 12–13
  MVC (Model-View-Controller) architecture, 23–25, *27*
  Rails router, editing, 27–28
  root route (Listings 1.9, 1.10), 27–28
  root URL, 27–28
  viewing in a browser, *29*
Help, Rails Tutorial, 7
help action (Listings 3.4, 3.5, 3.8), 92–93, 96, 100
Help pages
  cleaned-up views (Listings 3.33 to 3.36), 121–122
  creating, 92–93
  custom, sample, *102*
  custom HTML (Listing 3.10), 100–101
  help action (Listings 3.4, 3.5, 3.8), 92–93, 96
  with HTML structure removed (Listings 3.33 to 3.36), 121–122
  refactored views (Listings 3.33 to 3.36), 121–122
Heroku
  commands, 45–47
  creating and configuring an account (Listing 1.15), 44–45
  deployment, 45, 329
  heroku command, 44–45
  heroku create command, 44–45
  heroku open command, 45
  heroku rename command, 45–47
  PostgreSQL database, 43
  renaming applications, 45–47
  setup (Listing 1.14, 1.15), 43–45
  SSH key, adding, 44
heroku command, 44–45
heroku create command, 44–45
heroku open command, 45
heroku rename command, 45–47
Heroku Toolbelt, 44
hexdigest method, 289

Hidden fields, password reset (Listing 10.49), 523
Hierarchy, Static Pages controller, examining, 170–172, *171*
Hierarchy of classes, determining, 166–169
Highlighted lines in code samples, 8
home action
  creating Home pages (Listings 3.4, 3.5), 92–93, 96
  status feed, following-users page (Listing 12.48), 669
Home page. *See also* Site layout
  with account activation message, *492*
  with activation message, *492*
  adding microposts (Listing 11.36), 573–574
  custom HTML (Listing 3.9), 100
  embedded Ruby title (Listings 3.28 to 3.31), 117–119
  full HTML structure (Listings 3.24 to 3.26), 114–116
  generated view (Listing 3.7), 99
  link to the sign-up page (Listing 5.2), 186
  raw view, *97*
  root route, setting (Listing 3.7), 122–123, *123*
  with a title, *115*
Home page, follower statistics
  adding (Listing 12.17), 635–636
  styling (Listing 12.18), 636–637
Home page mockups
  Bootstrap and custom CSS, *190–191*
  CSS styling, *191*
  custom page, *101*
  with debug information, *281*
  footer, *199*
  form for creating microposts, *572*, *579*, *581*
  with proto-feed, *581*
  site logo (Listing 5.7), 194, *195*
  site navigation, *181*
  spacing, *191*
  typography, *193*
  without custom CSS, *182*
Home page mockups, following users
  with follow statistics, *638*
  incremented followers count, *618*
  status feed, *618*, *661*, *670*
HTML
  doctype declarations (Listing 3.21), 112
  document divisions, 184
  document type (Listing 3.21), 112
  header tag, 183–184
  li: (list item) tag, 185
  lists, 185
  nav tag, 185
  placing markup in a separate file. *See* Partials
  section tag, 184
  selectors, 112
  self-closing tags, 187n
  structure of a typical web page (Listing 3.21), 112
  testing for specific tags, 112
  ul: (unordered list) tag, 185
  whitespace sensitivity, 112, 185n
HTML shims (Listing 5.9), 183, 196
html.erb file extension, 117
HTTP (Hypertext Transfer Protocol)
  remembering a user's identity. *See* Sessions
  stateless protocol, 336
HTTP request methods, 97–98. *See also specific methods*

I

IDE (integrated development environment). *See* Cloud9
IDs, CSS, 184
if-else statement, 358
if-then statements, writing. *See* Ternary operator
Image cropping, *421*
Images. *See also* Microposts, images
  representing users. *See* Gravatars
  setting alternative display of, 187
image_tag helper function
  rendering images (Listing 11.60), 597
  setting alternate images (Listing 5.2), 186–187
Implicit return, functions, 150
Indentation, setting in Cloud9, 10, *12*
index action
  showing all users (Listing 9.33), 443
  toy application (Listing 2.5), 68–70
index action redirect (Listing 9.31), 442

Index page, showing all users
  adding the URL to the users link (Listing 9.36), 444–445
  bootstrap-will_paginate method (Listing 9.40), 448–449
  CSS for users index (Listing 9.35), 444
  index action (Listing 9.33), 443
  index action redirect (Listing 9.31), 442
  index view (Listing 9.34), 443–444
  overview, 441–445
  paginate method (Listing 9.42), 450–452
  pagination (Listings 9.40 to 9.42), 449–450, *451*
  partial refactoring (Listings 9.46 to 9.48), 454–456
  requiring a logged_in_user before filter (Listing 9.32), 442
  testing index action redirect (Listing 9.31), 442
  testing pagination (Listings 9.43, 9.44), *451–452*, 452–454
  will_paginate method (Listings 9.40, 9.41), 448–450
Index page, toy application (Listing 2.7), 58–59, *59*, 69–70, *76*
Index page mockups, showing all users
  basic page, *441*
  with one hundred users, *448*
  with one user, *446*
  with pagination, *451–452*
Indexing arrays, 152
Indices, 158
Inheritance, 166–169, *168*, 235
Inheritance hierarchies, toy application
  ApplicationController class (Listing 2.16), 80
  for controllers (Listings 2.15 to 2.17), 79–80
  MicropostsController class (Listing 2.16), 80
  for models (Listings 2.13 to 2.15), 79
  overview, 77–78, 80–81, *81*
  UsersController class (Listing 2.15), 80
Initialization hash, 236
Initializing objects, 236
inspect method, 162, 250
Installing
  Bootstrap, 188
  Gemfile (Listing 1.4), 15–21
  Git. *See* Git, installation and setup
  Rails on a local system (Listing 1.6), 21
Installing Rails on Cloud9
  bundle command, 20n9, 20–21
  Bundler, 15–21
  command-line program, 21
  default directory structure, 19*t*
  gem command, 11–12
  installing Gemfile (Listing 1.4), 15–21
  opening a new terminal tab, *22–23*
  overview, 11–12
  rails new command (Listing 1.3), 12–15
  rails server (Listings 1.6, 1.7), 21–23, *21–23*
  sample directory structure, *16*
  script, 21
  shortcuts, 93*t*
  with a specific version number (Listings 1.1, 1.3, 1.5), 11, 13–21
InstallRails, 9n5
Instance methods, 166
Instance variables, creating, 173
Instances of classes, 166
Integrated development environment (IDE). *See* Cloud9
Integration testing
  account activation (Listing 10.30), 502–503
  assertions (Listing 5.25), 216–219, 219*t*
  Contact page (Listing 5.16), 211–212
  for invalid sign-up submissions (Listing 7.21), 313–316
  for layout links (Listing 5.25), 216–219
  multiple assertions per test, 217n
  for valid sign-up submissions (Listing 7.26), 323–325
"Invalid multibyte character" error message, 48n
IP binding address, specifying and displaying, 21
 IP environment variable, specifying, 21
irb configuration (Listing 4.8), 142
Iteration over innumerable data structures. *See* Blocks

join method, 154, 664

- Kehoe, Daniel, 5
- Keys, 158
- kill command, 135

L

- Labels. *See* Symbols
- Lambda, 553
- Layout. *See* Site layout
- Layout files (Listing 3.32), 119–120
- *Learn Ruby on Rails* (Kehoe), 5–6
- *Learn to Program* (Pine), 5
- Length validation (Listing 2.10), 74–76
- Less-than signs ( ), adding to arrays, 154
- li: (list item) tag, 185
- lib/assets directory, 200
- Links. *See also* Site layout
  - named routes, 210–211*t*
  - stub URLs, 184
- link_to helper function, 184, 186
- Listing. *See also* Displaying; Showing
  - branches, 37–38
  - directory contents, 32
  - users, toy application, 58–60, *61*, *64*
- Lists, HTML, 185
- Literal constructors, 165–166
- Literal strings, 144
- Local development environment, 9
- Local servers
  - running. *See* rails server command
  - sharing, *24*
- log command, 32
- Logged-in status, determining (Listing 8.15), 358–359
- logged_in? method (Listing 8.15), 358–359
- logged_in_user method
  - friendly forwarding (Listing 9.28), 437–438
  - Microposts controller (Listing 11.32), 570–571
  - requiring logged-in users (Listings 9.12, 9.14, 9.16, 9.18), 425–431
- Logging in. *See also* Remember me; Sessions
  - changing layout links (Listings 8.16, 8.17), 357–361
  - current user, finding (Listing 8.14), 353–357
  - current_user method (Listing 8.14), 353–357
  - find method (Listing 7.5), 354–357
  - find_by method, 354–357
  - logged in status, determining (Listing 8.15), 358–359
  - logged_in? method (Listing 8.15), 358–359
  - log_in method (Listings 8.12, 8.13), 351–353
  - overview (Listing 8.11), 351
  - and redirecting to the profile page (Listing 8.13), 352–353
  - session method (Listings 8.12, 8.13), 351–353
  - successful login mockup, *358*
  - upon sign-up (Listings 8.22 to 8.24), 366–368
  - user profile mockup, *358*
  - validating user login (Listing 8.20), 365–366
- Logging in, testing layout changes
  - mockup, drop-down menu, *362*
  - mockup, logged-in user, *362*
  - overview, 361–366
- Logging out
  - only if logged in (Listing 8.42), 389–390
  - overview (Listings 8.26 to 8.28), 368–371
  - from a persistent session (Listing 8.39), 386–387
  - from persistent sessions (Listing 8.39), 386–387
  - a test for logout (Listings 8.28, 8.40), 387–392
- Login form, code for (Listing 10.38), 512–513
- Login form mockups
  - access failure, *427*
  - drop-down menus (Listing 8.17), *362*
  - a protected page, *425*
  - "remember me" checkbox, *392*, *394*
  - successful login, 358
- log_in method (Listings 8.12, 8.13), 351–353
- Login page with forgot password link, *510*
- Login *versus* remembering, account activation, and password reset, 473*t*
- log_in_as method (Listing 8.50), 398–399
- Logo. *See* Site logo
- log_out method (Listings 8.26 to 8.28), 369–371

lorem ipsum text, *277*, 560n–561n
Lorem.sentence method (Listing 11.24), 560
Lovitt, Michael, 252n13
ls command, 32

M

-m flag, 31
Magic columns, 232, 237
make utility. *See* Rake language
Manifest files (Listing 5.14), 201–202
map method
  blocks, 157
  status feed, following-users page, 663–666
Mapping user URLs to
  controller actions (Listing 2.2), toy application, 62
  correspondence between pages, 58*t*
Markup, placing in a separate file. *See* Partials
Mass assignment, 306–308
MD5 hashing algorithm, 289
Memoization, 354n
Merging branches, 40–41
Message chaining, 148
Message passing, 146–149
Messages, adding to a commit, 31. *See also* Error messages; Sign-up error messages
Metaprogramming, 494
Method definitions in the console, 149–151
Method reference convention, 260
Method reference technique, account activation, 476
Methods. *See also* Functions
  calling themselves. *See* Class methods
  versus functions, 147n
  remembering variable assignments. *See* Memoization
Micropost controller (Listing 2.9), 73–74
Micropost model. *See also* Showing microposts
  associations between models (Listings 11.9, 11.12), 545–550
  associations between models, summary of, 548*t*
  basic model, 542–543
  belongs_to relationships (Listing 11.9), 547–550
  data model, *542*
  default scope (Listings 11.13 to 11.17), 551–553
  default_scope method (Listing 11.16), 552–553
  dependent: :destroy (Listings 11.18, 11.19), 554–555
  destroying users (Listings 11.18, 11.19), 554–555
  foreign key reference, 543
  generating, 542–543
  has_many relationships, *548*
  migration with added index (Listing 11.1), 542–543
  multiple key index, 543
  ordering posts (Listings 11.13 to 11.17), 551–553
  overview, 541–542
  validation (Listings 11.2 to 11.8), 544–547
Micropost model, mockups
  Home page with form for creating microposts, *572*, *579*, *581*
  Home page with proto-feed, *581*
  user profile page with code for microposts, *560*
  user profile page with microposts, *556*, *564*, 565–568
  user profile page with pagination links, *566*
  user profile page with unstyled microposts, *562*
Micropost model, testing user profiles
  associating fixture microposts with users, *564*
  associating fixture microposts with users (Listing 11.26), 563–567
  micropost pagination, 565–568
  profile structure (Listing 11.27), 567–568
Micropost page, toy application, *75*
Micropost resource. *See* Toy application, Micropost resource
Microposts
  access control (Listings 11.31), 569–571
  authorization tests for Micropost controller (Listings 11.31), 569–570
  definition, 541
  feed. *See* Microposts, proto-feed

logged_in_user method (Listing 11.32), 570–571
routes for the Micropost resource (Listing 11.30), 568–569*t*
samples. *See* Micropost model, mockups
Microposts, creating
adding to the Home page (Listing 11.36), 573–574
create action (Listing 11.35), 572–580
creating (Listings 11.35 to 11.43), 572–575
form for creating microposts (Listing 11.36, 11.38), *572*, 572–575
pluralize method, 574
redefining error messages (Listings 11.40 to 11.44), 574
user info sidebar (Listing 11.37), 574
Microposts, destroying
adding a delete link (Listing 11.50), 586–587
destroy action, 587–589
Microposts, images. *See also* CarrierWave uploader
adding to Micropost model (Listing 11.57), 595
basic uploading (Listings 11.57 to 11.60), 594–597
cloud storage, 604–606
display size, constraining, 601n
format validation (Listing 11.61), 597–600
image_tag helper (Listing 11.60), 597
picture attribute, 594–597, *595*
resizing (Listing 11.64), 601–603, *602*, *603*
size validation (Listings 11.62, 11.63), 599–601
Microposts, mockups
Home page after deleting a micropost, *589*
Home page with a new micropost, *586*
Home page with a proto-feed, *581*, *585*
proto-feed with delete links, *587*
Microposts, proto-feed
failed micropost submission (Listing 11.49), 584–585
@feed_items instance variable (Listings 11.46 to 11.49), 582–586
Home page with a proto-feed, *581*
overview, 580–586
SQL injection, security hole, 582
status feed (Listings 11.46 to 11.49), 582–586
where method (Listing 11.45), 581–582
Microposts, testing
adding a micropost with a different owner (Listing 11.52), 590
deletion with user mismatch (Listing 11.53), 590–591
integration test for the micropost interface (Listing 11.54), 591–592
@microposts instance variable (Listing 11.22), 558
MicropostsController class (Listing 2.16), 80
Migrating down, 234
Migrating up, 232
Migration file name, 232
Migrations. *See* Database migration; Rake db:migrate command
mini-backtrace gem (Listing 3.41), 129
MiniMagick, CarrierWave (Listing 11.64), 601–603
MiniTest assertions, list of, 112n
MiniTest reports (Listing 3.40), 128
minitest-reporters gem, 128
Missing page titles, 139
mkdir (create directory) command, 13
Mockingbird, 180n, 275n
Mockups, tool for, 180n
Modeling users. *See* User model
Moses, Riley, 255n
Multiline blocks, 156
Multiple key index, 543
Multiple-key indices, 620
Mutating arrays, 154
mv (move files) command, 13
MVC (Model-View-Controller) architecture
controllers, 23–25
default Rails page, *24*
flow diagram, 61, *67*
models, 23–25
requests, 24
schematic view, *27*
sharing a local server, *24*
toy application, 60–70
views, 23–25

Named constructors, 165–166
Named routes
  in links, 210–211*t*
  in site layout (Listings 5.23 to 5.24), 215–216
Naming conventions
  CamelCase, 94
  snake case, 94
  User model, 231
nav tag, 185
navbar class, 184
navbar-fixed-top class, 184
navbar-inverse class, 184
navbar-nav class, 185
Navigating
  the filesystem, 87n
  site layout, 180–187
Nested hashes (Listing 4.10), 161
Nesting Sass style sheets, 203–205 (Listing 5.15)
new action, creating a Sessions controller, 336–337
new command (Listing 1.3), 12–15
new method, as named constructor, 165–166
newline
  in CSS (cascading style sheets), 163–164
  definition, 163
Newlines, in scaffolding (Listing 2.8), 72n
Nginx, 42
nil value
  in a boolean context, 149
  definition, 145
  and the empty method, 148
  message chaining, 148
  responding to methods, 148
  testing for, 148
"Nothing at all" value. *See* nil value

O
Objects
  in the console, 146–149
  converting to strings, 148
  definition, 146
  message passing, 146–149
Octothorpe. *See* Hash symbol (#), comment indicator
ol (ordered list) tag, 557
One Month Rails, 5
Online resources. *See also* Books and publications
  Code School, 6
  help for *Rules Tutorial*, 7n
  One Month Rails, 5
  Pragmatic Studio: Online Ruby and Rails courses, 6
  Rails Guides, 6
  RailsApps, 6
  RailsCasts, 6
  Tealeaf Academy, 6
  Thinkful, 6
  Try Ruby tutorial, 5
  Turing School of Software and Design, 6
  version control, 29
open command, 45
Opening new terminal tabs, *22–23*
Ordered list (ol) tag, 557
Ordering micropost posts (Listings 11.13 to 11.17), 551–553
Overwriting current changes, 33

P
p function, 162
paginate method (Listing 9.42), 450–452
Pagination
  microposts, 558–559, 565–568, *566*
  showing all users (Listings 9.40 to 9.42), 449–450, *451*
  will_paginate method, 558–559
Palindromes, checking for, 166–169
params hash, sign-up failure (Listing 7.16), 304–306
params variable (Listing 7.1), 277
Parentheses (( ))
  in statement syntax, 383n
  and the ternary operator, 395
Partial refactoring (Listings 9.46 to 9.48), 454–456
Partials
  for headers (Listing 5.10), 196–197
  for HTML shims (Listing 5.9), 196
  overview (Listing 5.8), 194
  render helper function, 196

sharing across views, 196n
for site footer (Listings 5.11 to 5.13), 197–199
Passive relationships
model for, *629*
Relationship model (Listing 12.12), 618–619, 628–630
Password reset. *See also* Forgot password form
adding a link to (Listing 10.37), 509–510
adding password reset methods (Listing 10.41), 515
adding reset methods to the User model (Listing 10.52), 528
create action (Listing 10.40), 515
edit action (Listing 10.50), 524–525
form for (Listings 10.49 to 10.52), 523–528, *526*
generating a controller for (Listings 10.36 to 10.41), 508–516
hidden fields (Listing 10.49), 523
login form, code for (Listing 10.38), 512–513
overview, 506–508
password reset view (Listing 10.39), 513
reset attributes, *512*
resetting the password (Listings 10.49 to 10.52), 523–528
update action (Listing 10.51), 525–528
User model, *512*
Password reset, mockups
failed password reset, *529*
forgot password form, *507*, *514*, *516*, *521*
forgot password link, *507*
login page with forgot password link, *510*
password reset form, *526*
reset email, HTML version, *519*
reset email, text version, *520*
reset password form, *508*
successful password reset, *530*
Password reset mailer method
HTML email template (Listing 10.44), 517
mailing the password reset link, 516–518
plain-text email template (Listing 10.43), 517
sample reset email from the server log (Listing 10.48), 521–522
testing (Listing 10.46), 519–520
working preview method (Listing 10.45), 518
Password reset test
failed password reset, mockup, *529*
integration test (Listing 10.53), 530–532
overview, 530–532
successful password reset, mockup, *530*
Password reset view (Listing 10.39), 513
Password security
authentication, 267–269
bcrypt hash function (Listing 6.33), 263–264, 269n21
confirmation, 264–267
hash, definition, 262
hashed passwords (Listings 6.32, 6.33), 261–264
hashing *versus* encryption, 262n
minimum length, 265–267
overview, 262
salted hashed, 269n21
User model, *263*
virtual, definition, 262n18
virtual attributes, 262
Password test *versus* account activation, remembering, and login, 473*t*
password_confirmation attribute, 262
password_digest attribute (Listing 6.32), 262–263. *See also* Hashed passwords
Passwords
allowing blank (Listing 9.10), 423
duplicates across users, 374
PATCH request, 98
Patterns of Enterprise Application Architecture (Fowler), 228n1
Permanent cookies, 377–385
Persistence, 229
Persistent sessions
creating, 372
logging out (Listing 8.39), 386–387
testing (Listing 8.55), 403
updating current_user (Listing 8.36), 384
Philips, Andy, 262n
Phusion Passenger, 42
picture attribute, 594–597, *595*
Pictures. *See* Images
pid file will be overwritten, error message, 124n
Pine, Chris, 5

pluralize method, 574
pluralize text helper, 311
Plus operator (+), concatenating strings, 144
PORT environment variable, 21
Port number, specifying and displaying, 21
POST method, 98
PostgreSQL database, 43
Pound sign. *See* Hash symbol (#), comment indicator
Pragmatic Studio: Online Ruby and Rails courses, 6
Preprocessor engines, 202
Printing
  hashes, 162
  strings, 145
private keyword, 307
Pro Git, 38n
Proc file, creating from a block, 553
Production environments, 278n
Profile page, successful account activation, *500*
Programs that write other programs. *See* Metaprogramming
Proto-feed
  failed micropost submission (Listing 11.49), 584–585
  @feed_items instance variable (Listings 11.46 to 11.49), 582–586
  Home page with a proto-feed, *581*
  overview, 580–586
  SQL injection, security hole, 582
  status feed (Listings 11.46 to 11.49), 582–586
  where method (Listing 11.45), 581–582
provide function, 117n14
ps command, 134–135
pull-right class, 185
push method, adding to arrays, 154
Pushing changes up to Git repositories, 33–36, 41
PUT method, 68
puts function, 145

q command, 32
Query parameters, account activation, 483
Question mark (?)
  boolean indicator, 147
  in query parameters, 483
Quitting Git, 32

R
Rails
  console. *See* Console
  creator of, 4
  definition, 4
  environments, 278–280
  learning Ruby first, 5
  open source, 5
  prerequisites, 5–6
  shortcuts, 93*t*
"The Rails 3 Asset Pipeline in (about) 5 Minutes" (Erasmus), 200n
The Rails 4 Way (Fernandez), 151n5
Rails Guides, 6
Rails router, editing, 27–28
Rails routes (Listing 5.22), 213–215. *See also* Named routes; Root routes
"Rails Routing from the Outside In," 633n
rails server command (Listings 1.6, 1.7). *See also* Installing Rails on Cloud9
  default RAILS page, *24*
  IP binding address, specifying and displaying, 21
  IP environment variable, 21
  overview, 21–23
  PORT environment variable, 21
  port number, specifying and displaying, 21
Rails Tutorial Cloud9 workspace, 9
Rails Tutorial help, 7
RailsApps, 6
RailsCasts, 6
rake command, 232
rake db:migrate command, 57–58
Rake language definition, 57–58
Ranges, 152–155. *See also* Arrays
Reading data from the web. *See* GET request
README file
  mockup, *42*
  sample code (Listing 3.3), 90
README.doc file missing, 36, *38*

README.md file (Listing 1.13), 39
RecordNotFound exception, 239
RED flag, 8
RED (failing) tests (Listings 3.13 to 3.18), 106–108
Red—Green—Refactor cycle, 106–107
redirect_back_or method (Listing 9.29), 439–440
Refactoring code
  definition, 103
  removing duplication (Listings 3.28 to 3.35), 116–121
  smelly code, 110–111
Regex. *See* Regular expressions
Regressions, 103
Regular expressions
  Rubular tool, 252–254, *253*
  validating email addresses (Listing 6.21), 252–254, 252*t*
  validating user data (Listing 6.21), 251–254, 252*t*
Relationship model. *See* Following users, Relationship model
Remember branch, testing (Listings 8.53 to 8.57), 401–404
Remember me. *See also* Logging in
  cookies, 372
  duplicate user passwords, 374
  forget method (Listing 8.38), 385–386
  overview, 371
  persistent sessions, 372
  "remember me" checkbox, *392*, *394*
  remember_digest attribute (Listings 8.30 to 8.32), 373–377, 373*t*
  remember_token method (Listing 8.32), 375–377
  tokens (Listings 8.30 to 8.32), 373–377, *385*
Remember me, at login
  authenticated? method (Listings 8.33, 8.43 to 8.45), 380–381, 390–391
  overview, 377–385
  persistent sessions (Listing 8.36), 384
  remembering a logged-in user (Listings 8.34, 8.35), 381–385
Remember me checkbox
  creating (Listings 8.47 to 8.49), 391–397
  CSS for (Listing 8.48), 393–394
  login form mockups, *392*, *394*
  testing (Listings 8.50, 8.51), 400–404
remember me function, testing
  defined? method (Listing 8.50), 398–399
  log_in_as method (Listing 8.50), 398–399
  persistent sessions (Listing 8.55), 403
  raising exceptions (Listing 8.53), 401–402
  remember branch (Listings 8.53 to 8.57), 403
  remember me checkbox (Listings 8.50, 8.51), 400–404
  remembering cookies (Listing 8.51), 400–401
  removing raised exceptions (Listing 8.57), 404
remember_digest attribute (Listings 8.30 to 8.32), 373–377, 373*t*
Remembered options, 89n
Remembering cookies (Listing 8.51), 400–401
Remembering *versus* login, account activation and password reset, 473*t*
remember_token method (Listing 8.32), 375–377
rename command, 45–47
Renaming applications, 45–47
render helper function, 196
Rendering microposts (Listings 11.21 to 11.23), 555–559
Repositories, creating. *See* Git, installation and setup
Reset attributes, password reset, *512*
Reset password form, *508*
Resetting
  databases. *See* Undoing, database migrations
  passwords (Listings 10.49 to 10.52), 523–528
Resizing micropost images (Listing 11.64), 601–603, *602*, *603*
REST (REpresentational State Transfer), 66–68
RESTful routes
  for the Micropost resource (Listing 11.30), 568–569, 569*t*
  toy application, 68*t*, 73*t*
RESTful URLs, 283–284, 284*t*
-rf flag, 32
rmfile command, 14*t*, 32

Root routes
  adding (Listing 2.3), 63–64
  defining (Listing 5.22), 213–215
  definition (Listings 1.9, 1.10), 27–28
  setting (Listing 3.7), 122–123, *123*
Root URL, 27–28
Routing. *See* Named routes; Rails routes; Root routes
Rows, databases, 230
Rubular tool, 252–253, *253*
Ruby language
  classes. *See* Classes
  data structures. *See* Data structures
  make utility. *See* Rake language
  version numbers, displaying, 329–330
Ruby on Rails. *See* Rails
Ruby on Rails™ Tutorial website, 1n
Rules (CSS), bundling and reusing. *See* Sass mixins (Listing 7.2)

S

S3 (Simple Storage Service), 604–606
Salted hashed passwords, 269n21
Sample applications. *See* Hello world program; Static pages; Toy application
Sample data, creating
  lorem ipsum text, *277*, 560n–561n
  Lorem.sentence method (Listing 11.24), 560
  microposts (Listings 11.24, 11.25), 560–561
  users (Listings 9.38, 9.39), 446–448, *448*
Sandbox, 235
.sass file extension, 203n
Sass mixins (Listing 7.2), 280–281
Sass utility, asset pipeline
  asset directories, 200–201
  definition, 200
  manifest files (Listing 5.14), 201–202
  preprocessor engines, 202
  production efficiency, 202–203
Sass utility, style sheets. *See also* CSS (cascading style sheets)
  nesting, 203–205 (Listing 5.15)
  overview, 203
  variable names, 207
  variables (Listing 5.15), 206–210
save method, 236
Saving User objects to the database, 236
scaffold command
  naming conventions, 56n
  toy application (Listing 2.8), 55–56, 71–74
Scaffolding
  description, 3–4
  extra newlines, 72n
  generate scaffold command, 3
  Micropost resource (Listing 2.8), 71–74
  toy application, 52–54
.scss file extension, 189, 202
section tag, 184
Secure Sockets Layer (SSL). *See* SSL (Secure Sockets Layer)
Security, preventing unactivated users from logging in (Listing 10.29), 499–501. *See also* Authentication; Authorization; Password security
Selecting array elements, 155
Selectors, 112
self keyword, 168–169
Self-closing tags, 187n
SendGrid (Listing 10.55), 532–534
Server computer, definition, 97–98
Server log errors, *283*
Servers, sharing local, *24*
Session hijacking, 326, 372
session method (Listings 8.12, 8.13), 351–353
Sessions. *See also* Logging in
  cookies, 336
  definition, 336
  destroying (Listings 8.26 to 8.28), 369–371
  reviewing form submission, create action (Listing 8.4), 343
Sessions, creating
  permanent. *See* Persistent sessions
  temporary (Listings 8.12, 8.13), 351–353
Sessions, finding and authenticating users
  authenticate method (Listing 8.5), 345–346, 346*t*
  authenticating users (Listing 8.5), 345–346, 346*t*
  failed login, mockup, *344*
  find_by method (Listing 8.5), 345–346

finding users (Listing 8.5), 345–346
overview, 343
Sessions, flash
code for failed login (Listing 8.9), 348–350
flash test (Listing 8.7), 348–350
flash.now method (Listing 8.9), 350
rendering with a flash message (Listing 8.6), 346–348, 347*t*
Sessions, login form creation
code sample (Listing 8.2), 341
form_for helper, 339–340
form_tag helper, 340n
HTML sample (Listing 8.3), 341–342
mockup of completed form, *342*
overview, 339–342
Sessions, login form mockups
added links, *342*
basic form, *338*
completed form, *342*
with flash message, *347*
flash persistence, *348*
login failure, *340*, *344*, *347*
Sessions, persistent
creating, 372
logging out (Listing 8.39), 386–387
testing (Listing 8.55), 403
updating current_user (Listing 8.36), 384
Sessions controller
adding resources (Listing 8.1), 337
overview, 336–339
routes, custom, 338*t*
routes, listing, 338–339
Setter methods, creating, 173
Shims (Listing 5.9), 183, 196
"Shiny new commit styles," 40
Shivs. *See* HTML shims
Short-circuit evaluation, 356
Shortcuts, Rails commands, 93*t*. *See also* Ctrl-C keystroke; Ctrl-D keystroke
Show page, mockups, *285–286*
Showing. *See also* Displaying; Listing
Showing all users
db/seeds.rb file (Listing 9.39), 447
Faker gem, adding to the Gemfile (Listing 9.38), 445–446
overview, 440
sample users, creating (Listings 9.38, 9.39), 446–448, *448*
Showing all users, index page
adding the URL to the users link (Listing 9.36), 444–445
bootstrap-will_paginate method (Listing 9.40), 448–449
CSS for users index (Listing 9.35), 444
index action (Listing 9.33), 443
index action redirect (Listing 9.31), 442
index view (Listing 9.34), 443–444
overview, 441–445
paginate method (Listing 9.42), 450–452
pagination (Listings 9.40 to 9.42), 449–450, *451*
partial refactoring (Listings 9.46 to 9.48), 454–456
requiring a logged_in_user before filter (Listing 9.32), 442
testing index action redirect (Listing 9.31), 442
testing pagination (Listings 9.43, 9.44), *451–452*, 452–454
will_paginate method (Listings 9.40, 9.41), 448–450
Showing all users, index page mockups
basic page, *441*
with one hundred users, *448*
with one user, *446*
with pagination, *451–452*
Showing microposts. *See also* Micropost model
count method, 558
counter cache, 558
@microposts instance variable (Listing 11.22), 558
number of posts per user, 558
paginating posts, 558–559
profile page with microposts, mockup, *556*
rendering microposts (Listings 11.21 to 11.23), 555–559
time_ago_in_words method (Listing 11.21), 557
will_paginate method (Listing 11.23), 558–559

Side effects of functions, 145
Sidebar (Listings 7.9, 7.10), 291–293
Signed cookies, 378–379
Signing up on Cloud9, 9n7
Sign-up, logging in (Listings 8.22 to 8.24), 366–368
Sign-up error messages
  any? method, 311
  displaying (Listings 7.17, 7.18), 308–313
  empty? method, 311
  error_explanation (Listing 7.20), 312–313
  _error_messages.html.erb partial file (Listing 7.19), 310–311
  field_with_errors class (Listing 7.20), 312–313
  mockups, *309*, *314*, *318*
  overview, 308–313
  pluralize text helper, 311–312
  styling with CSS (Listing 7.20), 312–313
Sign-up failure
  assert_select method, 314
  attr_accessible method, 307–308
  count method, 314
  debug information (Listing 7.16), 303–306
  failure-page mockups, *305*
  mass assignment, 306–308
  params hash (Listing 7.16), 304–306
  private keyword, 307
  strong parameters (Listing 7.16), 306–308
  testing for invalid submissions (Listing 7.21), 313–316
  testing for valid submissions (Listing 7.26), 323–325
  a working form (Listings 7.15, 7.16), 301–306
Sign-up forms
  authenticity token, 301
  CSRF (cross-site request forgery), 301
  form tag, 300–301
  form_for method, 294–298
  mockups, *292*, *295–297*, *314*
  overview, 293
Sign-up forms, HTML
  overview, 298–299
  sample HTML (Listing 7.15), 298–302
Sign-up page
  with account activation warning message, *501*
  adding an @user variable (Listing 7.12), 294–295
  assigning a route (Listing 5.33), 221–223
  custom stub view, 222
  first user sign-up, 322–323
  flash messages (Listings 7.24, 7.25), 319–321
  linking a button (Listing 5.34), 221–223
  linking to the Home page (Listing 5.2), 186
  Users controller, creating (Listing 5.28), 220
Sign-up page, mockups, *223*, *229*, *295–297*
  filled-in form, *301*
  first user sign-up, *322–323*
  flash messages, *323*
  sign-up failure (Listings 7.15, 7.16), 302–306, *305*
Sign-up success
  create action (Listing 7.23), 317–318
  finished sign-up form (Listing 7.22), 316–319, *317*
  overview, 316
  welcome message, *317*
Sign-up test (Listing 10.30), 502–503
Simple Storage Service (S3), 604–606
Single quotes ('), mixing with double quotes, 128
Site layout. *See also* Bootstrap; Links
  alternate images, 186–187
  document divisions, 184
  footers (Listings 5.11 to 5.13), 197–199
  headers (Listing 5.10), 183, 196–197
  navigation, 180–187
  site logo (Listing 5.7), 194
  web page (Listings 3.28 to 3.35), 116–121
Site logo (Listing 5.7), 194
Site navigation
  application.html.erb file (Listing 5.1), 180–187
  Home page mockup, *181*
  updating the site layout file (Listing 5.1), 180–187
Siwek, Katarzyna, 162n
Size validation, micropost images (Listing 11.62, 11.63), 599–601
Skeletons (code). *See* Scaffolding

Slash (/)
  root URL, 58
  Unix directory separator, 7
Slash, asterisk... (/* ... */), CSS comments, 189
Smelly code, 110–111
Snake case, 94
Spacing, Home page mockup, *191*
split method, 152
spring command, 135
Spring process ID (pid) file will be overwritten, error message, 124n
Spring server
  adding to .gitignore file (Listing 3.43), 132–133
  preventing conflicts with Git, 131
SQL (Structured Query Language)
  alternatives to. *See* Active Record
  pronouncing "SQL," 228n2
SQL injection, security hole, 582
SQLite Database Browser
  pronouncing "SQLite," 233n
  screen shots, *234*, *268*
Square brackets ([ ]), accessing elements, 152
SSH key, adding to Heroku, 44
SSL (Secure Sockets Layer)
  enabling, 326–327
  in production (Listing 7.27), 326–327
  in production deployment (Listing 7.26), 326–327
  session hijacking, 326
SSL certificates, 327
Stabby lambda, 553
Stack trace. *See* Backtrace
State, maintaining. *See* Cookies; Remember me
Stateless protocol, 336
Static pages. *See also* About page; Contact page; Help page; Home page; Sign-up page
  custom pages (Listings 3.9, 3.10), 100–101
  embedded Ruby (Listings 3.28 to 3.35), 116–121
  Gemfile (Listing 3.2), 88–89
  generating (Listing 3.4), 92–100
  generating a controller, 92–100. *See also* Static Pages controller
  generating a sample app (Listing 3.1), 88
  help action (Listings 3.4, 3.5, 3.8), 92–93, 96, 100
  home action (Listings 3.4, 3.5), 92–93, 96
  layouts (Listings 3.28 to 3.35), 116–121
  README file (Listing 3.3), 90
  refactoring (Listings 3.28 to 3.35), 116–121
  root route, setting (Listing 3.7), 122–123, *123*
  setup, 87–91
Static pages, dynamic titles
  base titles, 139
  on every page. *See* Base titles
  full_title helper function (Listings 4.2, 4.3, 4.9), 139, 151
  HTML structure of a typical web page (Listing 3.21), 112
  layout files (Listing 3.32), 119–120
  list of sample pages, 111*t*
  missing page titles, 139
  overview, 111
  Red testing, 112–113
  templates for, 117
  yield function, 117
Static Pages controller
  adding page titles, 113–116
  classes, creating, 170–172
  code sample (Listing 3.6), 98
  contents of config directory, *96*
  generating (Listing 3.4), 92–93
  help action (Listings 3.4, 3.5, 3.8), 92–93, 96, 100
  hierarchy, examining, 170–172, *171*
  home action (Listings 3.4, 3.5), 92–93, 96
  Home page, generated view (Listing 3.7), 99
  Home page, raw view, *97*
  Rails shortcuts, 93*t*
Static Pages controller, testing
  default (Listing 3.11), 104–105
  generating a test file (Listing 3.1), 104–105
  GREEN (passing) tests (Listings 3.15, 3.20), 107–110
  page titles (Listings 3.24 to 3.26), 113–116
  RED (failing) tests (Listings 3.13 to 3.18), 106–108
  Red—Green—Refactor cycle, 106–107

Static Pages controller, testing (*continued*)
refactoring code, 110–111
validating the test suite (Listing 3.12), 105
status command, 31
Status feed
following-user mockups, *671*
micropost (Listings 11.46 to 11.49), 582–586
Status feed, following-users page
from_users_followed_by method (Listings 12.45, 12.46), 667–671
home action (Listing 12.48), 669
initial status feed (Listing 12.43), 663–666
join method, 664
map method, 664
motivation and strategy (Listing 12.41), 662–663
sample, *661*
subselects (Listings 12.45 to 12.48), 667–671
Stealing cookies, 372
String literals, 144
Strings
arrays of, 249
assigning values to, 144
concatenating, 144
in the console, 144–146
converting objects to, 148
converting to arrays, 152
definition, 144
double-quoted, 144–145
interpolating, 144
printing, 145
single-quoted, 145–146
Strong parameters
deleting users, 460–462
sign-up failure (Listing 7.16), 303–306
Structured Query Language (SQL)
alternatives to. *See* Active Record
pronouncing “SQL,” 228n2
Stub URLs, linking to, 184
Stub view for user information (Listing 7.4), 284
Style sheets, CSS (Listing 4.11), 162–164
Style sheets, Sass utility
nesting (Listing 5.15), 203–205
overview, 203
variable names, 207
variables (Listing 5.15), 206–210
stylesheet_link_tag function, 138, 164
Styling with CSS (Listing 7.20), 312–313
Subselects (Listings 12.45 to 12.48), 667–671
Successful password reset, *530*
superclass method, 166–169
Symbols
as hash keys, 159–161
overview, 158–162

T
Table attacks, protection from, 269n21
Tables
databases, 230
dropping from databases, 234
TDD (test-driven development), 103–104. *See also* Testing
Tealeaf Academy, 6
Templates for pages, 117
Ternary operator, 395–396
Test environments, 278n
Test suites, 102
Testing. *See also* Integration testing; TDD (test-driven development)
About page, 104–111
authorization tests for Micropost controller (Listings 11.31), 569–570
authorizations, following users (Listing 12.24), 642–643
automated, 101–104
commenting out failed tests (Listing 10.21), 491
the correct user (Listing 9.22), 433–434
destroyed users (Listings 9.55 to 9.57), 462–465
edits (Listing 9.17), 429–430
failing tests, writing. *See* TDD (test-driven development)
for followers (Listing 12.13), 630
following relationships (Listing 12.39), 658–660
for friendly forwarding (Listing 9.26), 436–437

index action redirect (Listing 9.31), 442
logging in a test user (Listing 9.14), 428
logout (Listings 8.4, 8.28), 370–371
logout (Listings 8.28, 8.40), 387–392
MiniTest assertions, list of, 112n
model tests. *See* Validating user data
multiple assertions per test, 217n
for nil values, 148
pagination (Listings 9.43, 9.44), *451–452*, 452–454
password reset mailer method (Listing 10.46), 519–520
persistent sessions (Listing 8.55), 403
refactoring code, 103
regressions, 103
for specific HTML tags, 112
timing of, 103–104
unfollow buttons (Listing 12.39), 658–660
unsuccessful user updates (Listing 9.6), 419–420
the wrong user (Listing 9.21), 432
Testing, account activation
activation via user model object (Listing 10.34), 505
integration test (Listing 10.30), 502–503
overview, 502–506
sending email via user model object (Listing 10.33), 504–505
setting the test domain host (Listing 10.18), 489
user activation methods in the User model (Listing 10.32), 504
user sign-up test (Listing 10.30), 502–503
Testing, advanced
adding Spring to .gitignore file (Listing 3.43), 132–133
automated, 129–135
backtrace silencer (Listing 3.41), 129
custom Guardfile (Listing 3.42), 130–131
file navigator, *132–133*
Guard program, 129–135
MiniTest reports (Listing 3.40), 128
Testing, Micropost user profiles
associating fixture microposts with users (Listing 11.26), 563–567, *564*
micropost pagination, 565–568
profile structure (Listing 11.27), 567–568
Testing, Static Pages controller
default (Listing 3.11), 104–105
generating a test file (Listing 3.1), 104–105
GREEN (passing) tests (Listings 3.15, 3.20), 107–110
page titles (Listings 3.24 to 3.26), 113–116
RED (failing) tests (Listings 3.13 to 3.18), 106–108
Red—Green—Refactor cycle, 106–107
refactoring code, 110–111
validating the test suite (Listing 3.12), 105
Text editors, aligning columns, 164n
Thinkful, 6
Thomas, Dave, 6
Time_ago_in_words method (Listing 11.21), 557
Timestamps
database migrations, 232
User objects, 237
UTC (Coordinated Universal Time), 237n
Titles on static pages
adding, 113–116
base titles, 139
on every page. *See* Base titles
full_title helper function (Listings 4.2, 4.3, 4.9), 139, 151
HTML structure of a typical web page (Listing 3.21), 112
layout files (Listing 3.32), 119–120
list of sample pages, 111*t*
missing page titles, 139
overview, 111
templates for, 117
yield function, 117
Titles on static pages, testing
Home page title (Listing 4.4), 140–141
page titles (Listings 3.22, 3.24 to 3.26), 112–116
TLS (Transport Layer Security), 326n. *See also* SSL (Secure Sockets Layer)
to_a method, 154–155
Tokens (Listings 8.30 to 8.32), 373–377, *385*
Torvalds, Linus, 29

to_s method, 148
touch command, 109
Toy application
  deploying, 81–82, *82*
  micropost data model, 55, *55*
  planning, 52–55
  scaffolding, 52–54
  updating the Gemfile (Listing 2.1), 52–53
  user data model (Listing 2.6), 54–55, *54–55*, 69. *See also* Toy application, Users resource
Toy application, inheritance hierarchies
  ApplicationController class (Listing 2.16), 80
  for controllers (Listings 2.15 to 2.17), 79–80
  MicropostsController class (Listing 2.16), 80
  for models (Listings 2.13 to 2.15), 79
  overview, 77–78, 80–81, *81*
  UsersController class (Listing 2.15), 80
Toy application, Micropost resource
  data model associations (Listings 2.11, 2.12), 76–78, *77*
  data validation, 74–76
  error messages, *77*
  first_user variable, 78
  has_many associations, 76–78, *77*
  index page, *76*
  length validation (Listing 2.10), 74–76
  Micropost controller (Listing 2.9), 73–74
  micropost page, *75*
  Rails console, 78
  RESTful routes, 73*t*
  scaffolding (Listing 2.8), 71–74
Toy application, Users resource
  bundle exec, 57
  controller actions, 68*t*
  correspondence between pages and URLs, 58*t*
  creating users, 58–60
  destroying users, 60, *65*
  editing user information, 59, *62*, *63*
  index action (Listing 2.5), 68–70
  index page, 58–59, *59*
  index page (Listing 2.7), 69–70
  listing users, 58–60, *61*, *64*
  mapping user URLs to controller actions (Listing 2.2), 62
  migrating the database, 57–58
  MVC (Model-View-Controller), 60–70
  overview, 55
  rake db:migrate command, 57–58
  Rake language, 57–58
  REST (REpresentational State Transfer), 66–68
  RESTful routes, 68*t*
  root route, adding (Listing 2.3), 63–64
  scaffold command (Listing 2.8), 55–56, 56n
  user page, 58, *60*
  Users controller (Listing 2.4), 64–66
  weaknesses, 70
Transport Layer Security (TLS), 326n. *See also* SSL (Secure Sockets Layer)
Try Ruby tutorial, 5
Tuite, Colm, 179n
Turing School of Software and Design, 6
Twitter-style relationships, 618
Typography (Listing 5.6), 192–194, *193*

U

uglifier gem, 19
ul: (unordered list) tag, 185
underscore method, 622
Undoing
  code generation, 94–95
  database migrations, 95, 234, 235, 268n
Unfollow buttons
  mockups, *617*, *641*
  testing (Listing 12.39), 658–660
Unfollowed user, mockup, *653*
Uniform Resource Identifier (URI), 3n
Uniform Resource Locator (URL)
  correspondence between pages, 58*t*
  definition, 3n
Uniqueness, email addresses
  at the database level
  validating (Listings 6.23 to 6.26, 6.28), 254–261
Unix command line
  / (forward slash), directory separator, 7
  ? (dollar sign), command-line prompt, 6

Bash command-line interface, 13
cd (change directory) command, 13
common commands, 14*t*
Conquering the Command Line (Bates), 13
cp (copy files) command, 13
mkdir (create directory) command, 13
mv (move files) command, 13
overview, 13
Unobtrusive JavaScript, 655
Unordered list (ul:) tag, 185
Unsuccessful sign-up. *See* Sign-up failure
upcase method (Listing 6.25), 256
update action
password reset (Listing 10.51), 525–528
updating users (Listings 9.1 to 9.5, 9.9), 417–419, 422–423
Updating
remote servers. *See* PATCH request
User objects, 240–241
Updating users
adding a URL to the settings link (Listing 9.4), 416–417
edit action (Listings 9.1 to 9.4), 412–417
edit form (Listing 9.3), *415*
edit form mockups, *415*
edit view (Listing 9.2), 414
strong parameters, 459
update action (Listings 9.1 to 9.4), 417–419
Updating users, successful edits with TDD
acceptance tests (Listing 9.8), 421
allowing blank passwords (Listing 9.10), 423
Gravatar image cropping, *421*
overview, 420–424
successful mockup, *424*
update action (Listing 9.9), 422–423
Updating users, unsuccessful edits
edit form mockup, *419*
overview (Listing 9.5), 417–419
testing (Listing 9.6), 419–420
update action (Listing 9.5), 417–419
Uploading images
CarrierWave uploader (Listings 11.57 to 11.60, 11.66), 594–597, 603–606
micropost (Listings 11.57 to 11.60), 594–597
URI (Uniform Resource Identifier), 3n
URL (Uniform Resource Locator)
correspondence between pages, 58*t*
definition, 3n
url_encode method, 488
User activation methods, testing (Listing 10.32), 504
User class (Listing 4.13), 172–174
User class inheritance, 235
User data, showing. *See* Displaying user data
User email addresses, displaying, 284
User info sidebar (Listing 11.37), 574
User interface, Cloud9, *10*
User model. *See also* Databases
with account activation attributes (Listing 10.3), *475*
database migrations, 229–234
email addresses as user names, 230n
generating (Listing 6.1), 231
model file, 235
naming conventions, 231
overview, 228
persistence, 229
sample, *230*, *233*
sample code (Listing 6.3), 235
sketch of, *231*
viewing the structure of, 234
User model, User objects
creating, 235–238
current date and time, 237
destroying, 238
finding, 238
initialization hash, 236
saving to the database, 236
timestamps, 237
updating, 240–241
validity. *See* Validating user data
User names
displaying, 284
email addresses as, 230n
User objects
creating, 235–238
current date and time, 237
destroying, 238
finding, 238–240
initialization hash, 236

User objects (*continued*)
  saving to the database, 236
  timestamps, 237
User page, toy application, 58, *60*
User profile page
  adding a Users resource (Listing 7.3), 283
  creating, 282–286
  Gravatars (globally recognized avatars) (Listings 7.7, 7.8), 288–293
  logging in, mockup, *358*
  mockups, *276*, *324*
  RESTful URLs, 283–284, 284*t*
  server log errors, *283*
  sidebar (Listings 7.9, 7.10), 291–293
  stub view for user information (Listing 7.4), 284
  test for valid submission (Listing 7.26), 323–325
User profile page mockups
  basic page, *276*
  code for microposts, *560*
  CSS, *292*
  custom Gravatar, *291*
  default Gravatar, *290*
  with microposts, *556*, *564*, 565–568
  with pagination links, *566*
  sidebar, *292*
  unfollow button, *641*
  with unstyled microposts, *562*
User profile page mockups, following-users
  basic page, *614*
  finding a user to follow, *615*
  follow button, *616*, *640*
  incremented followers count, *617*
  unfollow button, *617*, *641*
User sign-up test (Listing 10.30), 502–503
user.followers method (Listings 12.12, 12.13), 628–630
user.following method (Listings 12.8 to 12.10), 625–628
Users
  adding to the fixture file (Listing 9.20), 431
  deleting. *See* Deleting, users
  displaying. *See* Showing all users
  editing. *See* Updating users
  updating. *See* Updating users
  validating. *See* Validating user data
Users controller
  creating a sign-up page (Listing 5.28), 220
  with a show action (Listing 7.5), 285
  toy application (Listing 2.4), 64–66
Users index with delete links, mockup, *457*, *461*
Users resource, adding to a user profile page (Listing 7.3), 283. *See also* Toy application, Users resource
UsersController class (Listing 2.15), 80
UTC (Coordinated Universal Time), 237n

valid? method, 236
Validating
  images, CarrierWave uploader, 597–601
  Micropost models (Listings 11.2 to 11.8), 544–545
  Relationship model (Listings 12.4 to 12.6), 623–625
  test suites (Listing 3.12), 105
  user login (Listing 8.20), 365–366
Validating user data
  case sensitivity (Listings 6.25, 6.26), 255–256
  email addresses (Listings 6.18 to 6.19, 6.21, 6.23), 249–261
  format (Listings 6.18 to 6.19, 6.21), 249–254
  length (Listings 6.14, 6.16), 247–249
  presence (Listings 6.7, 6.9, 6.11 to 6.12), 243–247
  with regular expressions (Listing 6.21), 251–254, 252*t*
  validity testing (Listings 6.4 to 6.5), 242–243
Variable names, Sass style sheets, 207
Variables, Sass style sheets (Listing 5.15), 206–210
vendor/assets directory, 200
Version control
  application source code. *See* Git
  definition, 29
  free resources online, 29
  online resources, 29
  specifying a version range (Listings 1.1, 1.3, 1.4, 1.5), 11, 13–21

Vertical bars ( )
  block variable indicator, 156
  or operator, 148
Vertical bars, equal sign (=), or equals
    assignment operator, 355–356
Vertical dots in code listings, 8
Views
  cleaned up (Listings 3.33 to 3.35), 121–122
  corresponding actions (Listings 3.7, 3.8),
    99–100
  embedded Ruby (Listings 3.28 to 3.31),
    117–119
  full_title helper function (Listings 4.2, 4.3,
    4.9), 139, 151
Virtual attributes, 262
Visscher, Sjoerd, 183n3

w[ ] technique, 249
WEBrick webserver, 327
where method (Listing 11.45), 581–582
Whitespace sensitivity, HTML, 112, 185n
will_paginate method
  showing all users (Listings 9.40, 9.41),
    448–450
  showing microposts (Listing 11.23), 558–559
Wireframes. *See* Mockups
–without production option, 89
Workspaces
  creating on Cloud9, *11*
  Rails Tutorial Cloud9, 9

YAML (YAML Ain't Markup Language), 282n
yield method, 117, 185
You have not declared a Ruby version..., error
    message, 329

Z
ero-offset arrays, 152