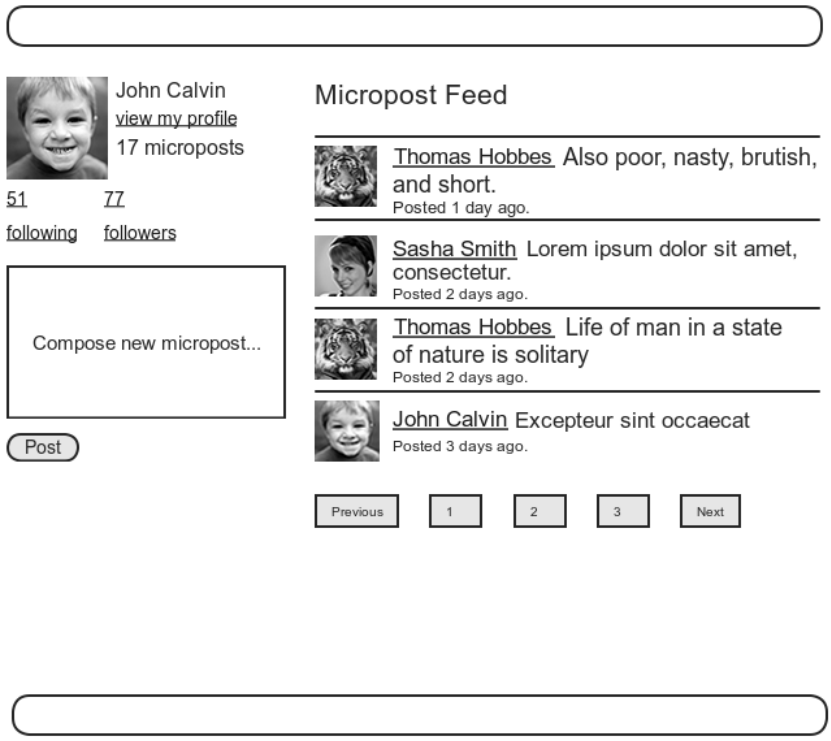


Figure 11.18 A mockup of a user's Home page with a status feed.

11.3.1 Motivation and Strategy

The basic idea behind the feed is simple. Figure 11.19 shows a sample **`microposts`** database table and the resulting feed. The purpose of a feed is to pull out the microposts whose user ids correspond to the users being followed by the current user (and the current user itself), as indicated by the arrows in the diagram.

Since we need a way to find all the microposts from users followed by a given user, we'll plan on implementing a method called **`from_users_followed_by`**, which we will use as follows:

```
Micropost.from_users_followed_by(user)
```

Although we don't yet know how to implement it, we can already write tests for for its functionality. The key is to check all three requirements for the feed: microposts for followed users and the user itself should be included in the feed, but a post from an

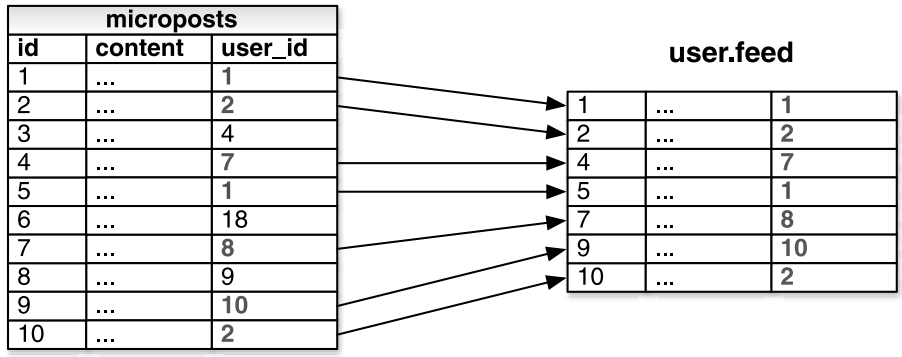


Figure 11.19 The feed for a user (id 1) following users 2, 7, 8, and 10.

unfollowed user should not be included. Two of these requirements already appear in our tests: Listing 10.38 verifies that a user's own microposts appear in the feed, while the micropost from an unfollowed user doesn't appear. Now that we know how to follow users, we can add a third type of test, this time checking that the microposts of a followed user appear in the feed, as shown in Listing 11.41.

Listing 11.41 The final tests for the status feed.
spec/models/user_spec.rb

```
require 'spec_helper'

describe User do
  .
  .
  .
  describe "micropost associations" do
    before { @user.save }
    let!(:older_micropost) do
      FactoryGirl.create(:micropost, user: @user, created_at: 1.day.ago)
    end
    let!(:newer_micropost) do
      FactoryGirl.create(:micropost, user: @user, created_at: 1.hour.ago)
    end
    .
    .
    .
    describe "status" do
      let(:unfollowed_post) do
        FactoryGirl.create(:micropost, user: FactoryGirl.create(:user))
      end
      let(:followed_user) { FactoryGirl.create(:user) }
```

```
      before do
        @user.follow!(followed_user)
        3.times { followed_user.microposts.create!(content: "Lorem ipsum") }
      end

      its(:feed) { should include(newer_micropost) }
      its(:feed) { should include(older_micropost) }
      its(:feed) { should_not include(unfollowed_post) }
      its(:feed) do
        followed_user.microposts.each do |micropost|
          should include(micropost)
        end
      end
    end
  end
end
```

Implementing the feed simply defers the hard work to **`Micropost.from_users_followed_by`**, as shown in Listing 11.42.

Listing 11.42 Adding the completed feed to the User model.
`app/models/user.rb`

```
class User < ActiveRecord::Base
  .
  .
  .
  def feed
    Micropost.from_users_followed_by(self)
  end
  .
  .
  .
end
```

11.3.2 A First Feed Implementation

Now it's time to implement **`Micropost.from_users_followed_by`**, which for simplicity we'll just refer to as "the feed." Since the final result is rather intricate, we'll build up to the final feed implementation by introducing one piece at a time.

The first step is to think of the kind of query we'll need. What we want to do is select from the **microposts** table all the microposts with ids corresponding to the users being followed by a given user (or the user itself). We might write this schematically as follows:

```
SELECT * FROM microposts
WHERE user_id IN (<list of ids>) OR user_id = <user id>
```

In writing this code, we've guessed that SQL supports an **IN** keyword that allows us to test for set inclusion. (Happily, it does.)

Recall from the proto-feed in Section 10.3.3 that Active Record uses the **where** method to accomplish the kind of select shown above, as illustrated in Listing 10.39. There, our select was very simple; we just picked out all the microposts with user id corresponding to the current user:

```
Micropost.where("user_id = ?", id)
```

Here, we expect it to be more complicated, something like

```
where("user_id in (?) OR user_id = ?", following_ids, user)
```

(Here we've used the Rails convention of **user** instead of **user.id** in the condition; Rails automatically uses the **id**. We've also omitted the leading **Micropost.** since we expect this method to live in the Micropost model itself.)

We see from these conditions that we'll need an array of ids corresponding to the users being followed. One way to do this is to use Ruby's **map** method, available on any "enumerable" object, i.e., any object (such as an Array or a Hash) that consists of a collection of elements.[11] We saw an example of this method in Section 4.3.2; it works like this:

```
$ rails console
>> [1, 2, 3, 4].map { |i| i.to_s }
=> ["1", "2", "3", "4"]
```

11. The main requirement is that enumerable objects must implement an **each** method to iterate through the collection.

Situations like the one illustrated above, where the same method (e.g., **to_s**) gets called on each element, are common enough that there's a shorthand notation using an *ampersand* **&** and a symbol corresponding to the method:[12]

```
>> [1, 2, 3, 4].map(&:to_s)
=> ["1", "2", "3", "4"]
```

Using the **join** method (Section 4.3.1), we can create a string composed of the ids by joining them on comma-space :

```
>> [1, 2, 3, 4].map(&:to_s).join(', ')
=> "1, 2, 3, 4"
```

We can use the above method to construct the necessary array of followed user ids by calling **id** on each element in **user.followed_users**. For example, for the first user in the database this array appears as follows:

```
>> User.first.followed_users.map(&:id)
=> [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51]
```

In fact, because this sort of construction is so useful, Active Record provides it by default:

```
>> User.first.followed_user_ids
=> [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51]
```

Here the **followed_user_ids** method is synthesized by Active Record based on the **has_many :followed_users** association (Listing 11.10); the result is that we need only append **_ids** to the association name to get the ids corresponding to the **user.followed_users** collection. A string of followed user ids then appears as follows:

```
>> User.first.followed_user_ids.join(', ')
=> "4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51"
```

12. This notation actually started as an extension Rails made to the core Ruby language; it was so useful that it has now been incorporated into Ruby itself. How cool is that?

When inserting into an SQL string, though, you don't need to do this; the `?` interpolation takes care of it for you (and in fact eliminates some database-dependent incompatibilities). This means we can use

```
user.followed_user_ids
```

by itself.

At this point, you might guess that code like

```
Micropost.from_users_followed_by(user)
```

will involve a class method in the **`Micropost`** class (a construction mentioned briefly in Section 4.4.1). A proposed implementation along these lines appears in Listing 11.43.

Listing 11.43 A first cut at the **`from_users_followed_by`** method.
`app/models/micropost.rb`

```
class Micropost < ActiveRecord::Base
  .
  .
  .
  def self.from_users_followed_by(user)
    followed_user_ids = user.followed_user_ids
    where("user_id IN (?) OR user_id = ?", followed_user_ids, user)
  end
end
```

Although the discussion leading up to Listing 11.43 was couched in hypothetical terms, it actually works! You can verify this yourself by running the test suite, which should pass:

```
$ bundle exec rspec spec/
```

In some applications, this initial implementation might be good enough for most practical purposes. But it's not the final implementation; see if you can make a guess about why not before moving on to the next section. (*Hint*: What if a user is following 5,000 other users?)

11.3.3 Subselects

As hinted at in the last section, the feed implementation in Section 11.3.2 doesn't scale well when the number of microposts in the feed is large, as would likely happen if a user

were following, say, 5000 other users. In this section, we'll reimplement the status feed in a way that scales better with the number of followed users.

The problem with the code in Section 11.3.2 is that

```
followed_user_ids = user.followed_user_ids
```

pulls *all* the followed users' ids into memory and creates an array the full length of the followed users array. Since the condition in Listing 11.43 actually just checks inclusion in a set, there must be a more efficient way to do this, and indeed SQL is optimized for just such set operations. The solution involves pushing the finding of followed user ids into the database using a *subselect*.

We'll start by refactoring the feed with the slightly modified code in Listing 11.44

Listing 11.44 Improving **`from_users_followed_by`**.
app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
  .
  .
  .
  # Returns microposts from the users being followed by the given user.
  def self.from_users_followed_by(user)
    followed_user_ids = user.followed_user_ids
    where("user_id IN (:followed_user_ids) OR user_id = :user_id",
          followed_user_ids: followed_user_ids, user_id: user)
  end
end
```

As preparation for the next step, we have replaced

```
where("user_id IN (?) OR user_id = ?", followed_user_ids, user)
```

with the equivalent

```
where("user_id IN (:followed_user_ids) OR user_id = :user_id",
      followed_user_ids: followed_user_ids, user_id: user)
```

The question mark syntax is fine, but when we want the *same* variable inserted in more than one place the second syntax is more convenient.

The above discussion mentions that we will be adding a *second* occurrence of **user_id** in the SQL query. In particular, we can replace the Ruby code

```
followed_user_ids = user.followed_user_ids
```

with the SQL snippet

```
followed_user_ids = "SELECT followed_id FROM relationships
                     WHERE follower_id = :user_id"
```

This code contains a SQL *subselect*, and internally the entire select for user 1 would look something like this:

```
SELECT * FROM microposts
WHERE user_id IN (SELECT followed_id FROM relationships
                  WHERE follower_id = 1)
      OR user_id = 1
```

This subselect arranges for all the set logic to be pushed into the database, which is more efficient.[13]

With this foundation, we are ready for an efficient feed implementation, as seen in Listing 11.45. Note that, because it is now raw SQL, **followed_user_ids** is *interpolated*, not escaped. (It actually works either way, but logically it makes more sense to interpolate in this context.)

Listing 11.45 The final implementation of **from_users_followed_by**.
app/models/micropost.rb

```
class Micropost < ActiveRecord::Base
   attr_accessible :content
   belongs_to :user

   validates :user_id, presence: true
   validates :content, presence: true, length: { maximum: 140 }

  default_scope order: 'microposts.created_at DESC'
```

13. For a more advanced way to create the necessary subselect, see the blog post "Hacking a subselect in ActiveRecord."

```
  def self.from_users_followed_by(user)
    followed_user_ids = "SELECT followed_id FROM relationships
                         WHERE follower_id = :user_id"
    where("user_id IN (#{followed_user_ids}) OR user_id = :user_id",
          user_id: user.id)
  end
end
```

This code has a formidable combination of Rails, Ruby, and SQL, but it does the job, and does it well. (Of course, even the subselect won't scale forever. For bigger sites, you would probably need to generate the feed asynchronously using a background job. Such scaling subtleties are beyond the scope of this tutorial, but the Scaling Rails screencasts are a good place to start.)

11.3.4 The New Status Feed

With the code in Listing 11.45, our status feed is complete. As a reminder, the code for the Home page appears in Listing 11.46; this code creates a paginated feed of the relevant microposts for use in the view, as seen in Figure 11.20.[14] Note that the **paginate** method actually reaches all the way into the Micropost model method in Listing 11.45, arranging to pull out only 30 microposts at a time from the database. (You can verify this by examining the SQL statements in the development server log file.)

Listing 11.46 The **home** action with a paginated feed.
app/controllers/static_pages_controller.rb

```
class StaticPagesController < ApplicationController

  def home
    if signed_in?
      @micropost  = current_user.microposts.build
      @feed_items = current_user.feed.paginate(page: params[:page])
    end
  end
  .
  .
  .
end
```

14. In order to make a prettier feed for Figure 11.20, I've added a few extra microposts by hand using the Rails console.

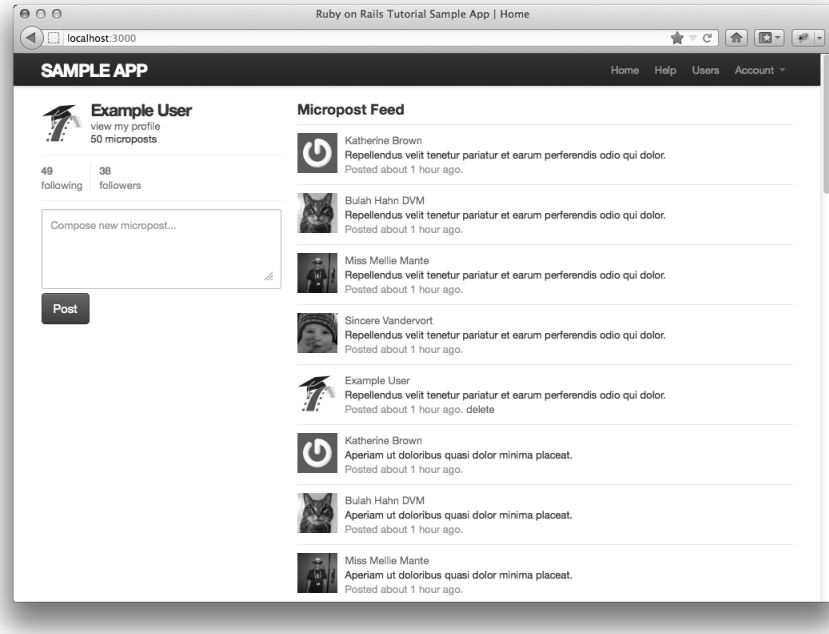


Figure 11.20 The Home page with a working status feed.

11.4 Conclusion

With the addition of the status feed, we've finished the core sample application for the *Rails Tutorial*. This application includes examples of all the major features of Rails, including models, views, controllers, templates, partials, filters, validations, callbacks, **`has_many`**/**`belongs_to`** and **`has_many through`** associations, security, testing, and deployment. Despite this impressive list, there is still much to learn about Rails. As a first step in this process, this section contains some suggested extensions to the core application, as well as suggestions for further learning.

Before moving on to tackle any of the application extensions, it's a good idea to merge in your changes:

```
$ git add .
$ git commit -m "Add user following"
$ git checkout master
$ git merge following-users
```

As usual, you can also push the code and deploy the application if you want:

```
$ git push
$ git push heroku
$ heroku pg:reset SHARED_DATABASE --confirm <name-heroku-gave-to-your-app>
$ heroku run rake db:migrate
$ heroku run rake db:populate
```

11.4.1 Extensions to the Sample Application

The proposed extensions in this section are mostly inspired either by general features common to web applications, such as password reminders and email confirmation, or features specific to our type of sample application, such as search, replies, and messaging. Implementing one or more of these application extensions will help you make the transition from following a tutorial to writing original applications of your own.

Don't be surprised if it's tough going at first; the blank slate of a new feature can be quite intimidating. To help get you started, I can give two pieces of general advice. First, before adding any feature to a Rails application, take a look at the RailsCasts archive to see if Ryan Bates has already covered the subject.[15] If he has, watching the relevant RailsCast first will often save you a ton of time. Second, always do extensive Google searches on your proposed feature to find relevant blog posts and tutorials. Web application development is hard, and it helps to learn from the experience (and mistakes) of others.

Many of the following features are quite challenging, and I have given some hints about the tools you might need to implement them. Even with hints, they are *much* more difficult than the book's end-of-chapter exercises, so don't be discouraged if you can't solve them without considerable effort. Due to time constraints, I am not available for one-on-one assistance, but if there is sufficient interest I might release standalone article/screencast bundles on some of these extensions in the future; go to the main Rails Tutorial website at http://railstutorial.org and subscribe to the news feed to get the latest updates.

15. Note that RailsCasts usually omit the tests, which is probably necessary to keep the episodes nice and short, but you could get the wrong idea about the importance of testing. Once you've watched the relevant RailsCast to get a basic idea of how to proceed, I suggest writing the new feature using test-driven development. (In this context, I recommend taking a look at the RailsCast on "How I test." You'll see that Ryan Bates himself often uses TDD for real-life development, and in fact his testing style is similar to style used in this tutorial.)

Replies

Twitter allows users to make "@replies", which are microposts whose first characters are the user's login preceded by the `@` sign. These posts only appear in the feed of the user in question or users following that user. Implement a simplified version of this, restricting @replies to appear only in the feeds of the recipient and the sender. This might involve adding an **`in_reply_to`** column in the **`microposts`** table and an extra **`including_replies`** scope to the Micropost model.

Since our application lacks unique user logins, you will also have to decide on a way to represent users. One option is to use a combination of the id and the name, such as **`@1-michael-hartl`**. Another is to *add* a unique username to the signup process and then use it in @replies.

Messaging

Twitter supports direct (private) messaging by prefixing a micropost with the letter "d." Implement this feature for the sample application. The solution will probably involve a Message model and a regular expression match on new microposts.

Follower Notifications

Implement a feature to send each user an email notification when they gain a new follower. Then make the notification optional, so that users can opt out if desired. Among other things, adding this feature requires learning how to send mail with Rails. To get started, I suggest viewing the RailsCast on Action Mailer in Rails 3.

Password Reminders

Currently, if our application's users forget their passwords, they have no way to retrieve them. Because of the one-way secure password hashing in Chapter 6, our application can't email the user's password, but it can send a link to a reset form. Follow the steps in the RailsCast on Remember Me & Reset Password to fix this omission.

Signup Confirmation

Apart from an email regular expression, the sample application currently has no way to verify the validity of a user's email address. Add an email address verification step to confirm a user's signup. The new feature should create users in an inactive state, email the user an activation URI, then change the user to an active state when the URI gets hit. You might want to read up on state machines in Rails to help you with the inactive/active transition.

RSS Feed

For each user, implement an RSS feed for their microposts. Then implement an RSS feed for each status feed, optionally restricting access to that feed using an authentication scheme. The RailsCast on generating RSS feeds will help get you started.

REST API

Many websites expose an Application Programmer Interface (API) so that third-party applications can get, post, put, and delete the application's resources. Implement such a REST API for the sample application. The solution will involve adding `respond_to` blocks (Section 11.2.5) to many of the application's controller actions; these should respond to requests for XML. Be careful about security; the API should only be accessible to authorized users.

Search

Currently, there is no way for users to find each other, apart from paging through the user index or viewing the feeds of other users. Implement a search feature to remedy this. Then add another search feature for microposts. The RailsCast on simple search forms will help get you started. If you deploy using a shared host or a dedicated server, I suggest using Thinking Sphinx (following the RailsCast on Thinking Sphinx). If you deploy on Heroku, you should follow the Heroku full text search instructions.

11.4.2 Guide to Further Resources

There are a wealth of Rails resources in stores and on the web—indeed, the supply is so rich that it can be overwhelming. The good news is that, having gotten this far, you're ready for almost anything else out there. Here are some suggestions for further learning:

- The *Rails Tutorial* screencasts: I have prepared a full-length screencast course based on this book. In addition to covering all the material in the book, the screencasts are filled with tips, tricks, and the kind of see-how-it's-done demos that are hard to capture in print. They are available for purchase through the *Rails Tutorial* website. (*Note*: The screencasts for the second edition are currently in preparation. They will be a paid upgrade, but current customers will receive a substantial discount.)

- RailsCasts: It's hard to overemphasize what a great resource the RailsCasts are. I suggest starting by visiting the RailsCasts episode archive and clicking on subjects that catch your eye.
- Scaling Rails: One topic we've hardly covered in the *Rails Tutorial* book is performance, optimization, and scaling. Luckily, most sites will never run into serious scaling issues, and using anything beyond plain Rails is probably premature optimization. If you do run into performance issues, the Scaling Rails series from Gregg Pollack of Envy Labs is a good place to start. I also recommend investigating the site monitoring applications Scout and New Relic.[16] And, as you might suspect by now, there are RailsCasts on many scaling subjects, including profiling, caching, and background jobs.
- Ruby and Rails books: As mentioned in Chapter 1, I recommend *Beginning Ruby* by Peter Cooper, *The Well-Grounded Rubyist* by David A. Black, and *The Ruby Way* by Hal Fulton for further Ruby learning, and *The Rails 3 Way* by Obie Fernandez and *Rails 3 in Action* (wait for the second edition) by Ryan Bigg and Yehuda Katz for more about Rails.
- PeepCode and Code School: The screencasts at PeepCode and interactive courses at Code School are consistently high-quality, and I warmly recommend them.

11.5 Exercises

1. Add tests for destroying relationships associated with a given user (i.e., as implemented by `dependent :destroy` in Listing 11.4 and Listing 11.16). *Hint*: Follow the example in Listing 10.15.
2. The `respond_to` method seen in Listing 11.38 can actually be hoisted out of the actions into the Relationships controller itself, and the `respond_to` blocks can be replaced with a Rails method called `respond_with`. Prove that the resulting code, shown in Listing 11.47, is correct by verifying that the test suite still passes. (For details on this method, do a Google search on "rails respond_with".)
3. Refactor Listing 11.31 by adding partials for the code common to the following/followers pages, the Home page, and the user show page.
4. Following the model in Listing 11.19, write tests for the stats on the profile page.

16. In addition to being a clever phrase—*new relic* being a contradiction in terms—New Relic is also an anagram for the name of the company's founder, Lew Cirne.

Listing 11.47 A compact refactoring of Listing 11.38.

```
class RelationshipsController < ApplicationController
  before_filter :signed_in_user

  respond_to :html, :js

  def create
    @user = User.find(params[:relationship][:followed_id])
    current_user.follow!(@user)
    respond_with @user
  end

  def destroy
    @user = Relationship.find(params[:id]).followed
    current_user.unfollow!(@user)
    respond_with @user
  end
end
```

Index

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

Symbols

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

A

About page
  about route, adding (Listing 3.14), 100–101
  about view, adding, 101–102
  adding, 99–103
  adding code to test contents of (Listing 3.13), 99
  code for (Listing 3.16), 102
  footer partial with links for (Listing 5.25), 206
  with HTML structure (Listing 3.21), 108
  with HTML structure removed (Listing 3.28), 113
  new, *102*
  refactoring, 103
  StaticPages controller with added about action (Listing 3.15), 101
  tests for static pages (Listing 5.27), 210
  view for, with Embedded Ruby title (Listing 3.24), 110–111
  writing a failing test for, 99–100
abstraction layers, 226n4
access control, 456–459
access control in manipulating Microposts, 456–459
accessible attributes and first validation, 432–433
accessible attributes in model file, 230
actions, 85–86
Active Record, 222
  callback, 253
  count method, 295
  creating user objects, 230–233
  finding user objects, 233–235
  updating user objects, 235–236
  See also Validations
adding files, in Git, 30–31
administrative users, 413–417
  attr_accessible, 416–417
  attr_accessible attributes for User model without :admin attribute (Listing 9.42), 417
  deleting, 413–417
  migration to add boolean admin attribute to users (Listing 9.40), 415

administrative users (*continued*)
sample data populator code with admin user (Listing 9.41), 416
tests for admin attribute (Listing 9.39), 414
user delete links (viewable by admins) (Listing 9.45), 419
User model with admin boolean attribute, *415*
administrative users, deleting, 413–417
Ajax
follow button with, 524–529
form for following a user using (Listing 11.35), 525
form for unfollowing a user using (Listing 11.36), 525
JavaScript Embedded Ruby to create following relationship (Listing 11.39), 529
problem solved by, 524
Ruby JavaScript to destroy following relationship (Listing 11.40), 529
Ajax requests, responding to, 525–529
JS-ERb, 528–529
in Relationships controller (Listing 11.38), 527
tests for Relationships controller (Listing 11.37), 526
ampersand (&), 534
anchor tag, 97
annotate, 229–230
annotated User model (Listing 6.5), 229–230
ApplicationController class with inheritance (Listing 2.16), 72
Application Programmer Interface (API), 542
application root, 9, 28–29, 125, 161
Architectural Styles and the Design of Network-based Software Architectures (Fielding), 60n4
arrays, in Ruby data structures, 142–145
asset directory in asset pipeline, 187–188
asset pipeline, Sass and, 187–190
asset directory, 187–188
efficiency in production, 189–190
manifest files, 188–189
preprocessor engines, 189
assignment, 352
See also Mass assignment
associations
Micropost resource, 68–70
user/micropost, 433–438
user/relationship, 491–494
associative arrays, 148
asynchronous JavaScript and XML. *See* Ajax
attr_accessible
administrative users, 416–417
attributes for User model without :admin attribute (Listing 9.42), 417
making name and email attributes accessible (Listing 6.6), 230
to prevent mass assignment, 230, 416–417
attribute accessors, 162
authenticate method
has_secure_password, 264, 338
moving the authenticate method into the Sessions helper (Listing 10.27), 457–458
test for (Listing 6.29), 262–263
authentication, 260–263
adding authentication to Microposts controller actions (Listing 10.28), 458
vs. authorization, 385
sessions and, 325–326
signin failure, 325–343
signin success, 343–363
See also Authenticate method
authenticity token, 301
authorization, 385–396
vs. authentication, 385
of following and followers pages, tests for (Listing 11.28), 516–517
friendly forwarding, 392–396
for relationships controller, tests for (Listing 11.33), 522–523
requiring right user, 390–392
requiring signed-in users, 386–389
automated testing, 77
Automattic, 286
avatar, 286n7

B

Bates, Ryan, 6, 7, 540
BCrypt cost factor in test environment, redefining (Listing 7.11), 286
before filters, 373
adding a signed_in_user before filter (Listing 9.12), 387
applied to every action in controller, 387
correct_user before filter in microposts, 477–478

correct_user before filter to protect edit/update pages (Listing 9.15), 391
current_user boolean method, 391–392
in requiring right user, 390–392
restricting destroy action to admins (Listing 9.48), 422
Beginning Ruby (Cooper), 4, 5, 129, 543
Black, David A., 6, 543
blocks, in Ruby data structures, 146–148
Booleans, 133, 138–139, 142
Bootstrap
adding bootstrap-sass gem to (Listing 5.3), 175
adding to application.js (Listing 8.25), 358
and custom CSS in layout structure, 175–186
framework, 176, 317
browsers, 11–12
built-in Ruby classes, modifying, 158–159
bundle exec, eliminating, 118–119
binstubs, 119
RVM Bundler integration, 118–119
Bundler, 19–23
business logic, 25

C

callback, 253, 346–348
Capybara, 79
in Cucumber step files, 367
integration tests, 93
signin tests, 330
signup tests, 294
syntax for CSS id, 471
in test-driven development, 94–95
test for destroying microposts, 477
tests for user update action (Listing 9.9), 383
cascading style sheets (CSS), 152–153, 190–197
asset directory, 187–188
Bootstrap framework, 176, 317
Capybara syntax for CSS id, 471
custom CSS, 175–186
efficiency in production, 189–190
HTML source produced by CSS includes (Listing 4.7), 153
layout links, 197–211
manifest files, 188–189
for microposts (Listing 10.24), 452–453
mixins, 274–275
nesting, 190–192
partials, 181–186
preprocessor engines, 189
in Ruby data structures, 152–153
Sass, 187–197
site navigation, 169–175
structure, adding, 167–186
for styling error messages (Listing 7.24), 311
for user index (Listing 9.26), 400
user signup, 211–215
variables, 193–197
Celadon Cedar Stack, 40
chaining, 139, 421
checkout command, 28, 32
Chrome, 11–12, 103, 170
classes, 153–163
built-in, modifying, 158–159
code for example user (Listing 4.9), 161
constructor, 153–154
container class, 172
controller, 159–161
defining Word class in console (Listing 4.8), 156
inheritance, 155–157
user, 161–163
class methods, 154–155
class name converted to id, 493n5
Code School, 6, 543
command lines, 10, *11*
comments, 134–135
commit command, in Git, 31
config directory, 9, 88, *89*
constructor classes, 153–154
Contact page
action for (Listing 5.18), 199
adding, 197–199
adding route for (Listing 5.17), 199
footer partial with links for (Listing 5.25), 206
for sample app, 114–117
tests for (Listing 5.16), 198
tests for static pages (Listing 5.27), 210
view for (Listing 5.19), 199
container class, 172
content validations, Micropost model, 443–444
controller classes, 159–161
cookies, 349–351
expiring 20 years in the future, 350
remember token added to, 379

cookies (*continued*)
remember token removed from, 363
used as a hash, 349–351
Cooper, Peter, 4, 5, 543
correct_user before filter
in microposts, 477–478
to protect edit/update pages (Listing 9.15), 391
counting columns, 105n12
count method, 295
create action
adding (empty) @feed_items instance variable to (Listing 10.45), 474–475
completed, 313
completed Sessions controller create action (not yet working) (Listing 8.13), 343
handling signup failure (but not success) (Listing 7.21), 305
for microposts, 461
Microposts controller create action (Listing 10.30), 461
preliminary version of sessions create action (Listing 8.9), 337
for Sessions controller, 326, 336–338, 343, 395
Sessions create action with friendly forwarding (Listing 9.20), 395
in signup failure, 304–305
strategy for using, 304
tests for post-save behavior in (Listing 7.32), 323
user create action with save and redirect (Listing 7.25), 314
for Users controller, 425, 459
creating microposts, 459–467
adding micropost instance variable to home action (Listing 10.34), 463
adding microposts creation to Home page (Listing 10.31), 461
form partial for creating microposts (Listing 10.33), 463
Microposts controller create action (Listing 10.30), 461
partial for user info sidebar (Listing 10.32), 462
tests for (Listing 10.29), 460
updating error-messages partial from Listing 7.23 to work with other objects (Listing 10.35), 464
updating errors for editing users (Listing 10.37), 465
updating rendering of user signup errors (Listing 10.36), 465
cross-site request forgery (CSRF), 301
cross-site scripting attack, 481
CSS. *See* Cascading style sheets (CSS)
CSS: The Missing Manual (McFarland), 5
Cucumber, 363–371
adding cucumber-rails gem to Gemfile (Listing 8.31), 364
adding helper method and custom RSpec matcher (Listing 8.34), 371
features and steps, 365–368
features to test user signin (Listing 8.32), 366
installation and setup, 364–365
RSpec examples, equivalent, 368–371
signin tests using, 363–371
steps needed to get signin features to pass (Listing 8.33), 368
current user, 351–355
defining assignment to (Listing 8.20), 352
definition for (Listing 8.21), 353
finding, using remember_token (Listing 8.22), 353
non-nil, 356
in signin success, 351–355
current_user? boolean method, 391–392

D

database indices, 254
database migration. *See* Migration
data model
defined, 47
micropost, 48–49
user, 47–48
debug
adding code for debug box, including Sass mixin (Listing 7.2), 274
information, adding to site layout (Listing 7.1), 273–274
information, restricting to development environment, 276
information in sign up, 271–276
output, *275*
in Rails environments, 276
default Rails directory structure, 19*t*

default Rails page, *24*
  with the app environment, *25*
default scope in Micropost model refinements, 440–441
demo app, 45–75
  conclusion, 74–75
  Microposts resource, 63–74
  planning the application, 45–49
  Users resource, 49–63
demo app, deploying, 73–74
dependent refinements in Micropost model, 441–443
deploying Rails, 39–42
destroy action
  adding factory for administrative users (Listing 9.43), 417–418
  adding working destroy action (Listing 9.46), 420–421
  in deleting users, 417–422
  before filter restricting destroy action to admins (Listing 9.48), 422
  test for protecting destroy action (Listing 9.47), 421–422
  tests for delete links (Listing 9.44), 418–419
  user index /users with delete links, *420*
destroying microposts
  ensuring that user's microposts are destroyed along with user (Listing 10.16), 443
  feed item partial with added delete link (Listing 10.47), 476
  Microposts controller destroy action (Listing 10.49), 477–478
  mockup of proto-feed with micropost delete links, *476*
  testing that microposts are destroyed when users are (Listing 10.15), 442
  tests for Microposts controller destroy action (Listing 10.48), 477
  user home page after deleting second-most-recent micropost, *479*
development environment, 9–27
  browsers, 11–12
  command lines, 10, *11*
  IDEs, 10
  terminals, 11
  text editors, 10, *11*
  time learning tools, 12
development log, 231–232, 450n4
directories
  standard directory and file structure, *18*
  summary of default Rails directory structure, 19*t*
div tags, 171
doctype, 84
Document Object Model (DOM), 528
domain logic, 25
domain-specific language (DSL), 3, 94, 283
drb option, 125
duplication, eliminating, 103, 111–113
dynamic pages. *See* Slightly dynamic pages

E

each method, 146, 151, 245, 399, 533n11
edit form, in updating users, 374–380
edits in updating users, successful, 382–384
edits in updating users, unsuccessful, 380–382
Emacs, 29
Embedded Ruby
  instance variables and, 162
  JavaScript, to create following relationship (Listing 11.39), 529
  slightly dynamic pages, 108–111
Embedded Ruby title
  view for About page with (Listing 3.24), 110–111
  view for Help page with (Listing 3.23), 110
  view for Home page with (Listing 3.22), 109
empty? method, 138, 139, 310
encrypted passwords, 255–257
Engine Yard, 13, 16
Engine Yard Cloud, 39
environment loading, adding to Spork.prefork block (Listing 3.36), 124
equality comparison operator, 144
ERb. *See* Embedded Ruby
error messages, signup, 308–312
  code to display error messages on signup form (Listing 7.22), 309
  CSS for styling error messages (Listing 7.24), 311
  failed signup with error messages, *312*
  partial for displaying form submission error messages (Listing 7.23), 309
exceptions, 234n8

F

factories
  complete factory file, including new factory for microposts (Listing 10.12), 439
  to simulate User model objects (Listing 7.8), 284
  test for user show page (Listing 7.9), 285
  testing user show page with, 282–286
Factory Girl, 283–286
  adding to Gemfile (Listing 7.7), 284
  in micropost refinements, 439–440
  sequence, defining (Listing 9.32), 407
  sequence, solving problems in, 406
  slow nature of running, 285–286
Faker gem, 403
  adding to Gemfile (Listing 9.29), 403
  lorem ipsum text, 450–451, 451n5
feed, 429
  proto-, 467–475
  RSS, 542
  status, 529–539
Fernandez, Obie, 6, 142n5, 543
Fielding, Roy, 60
Files
  standard directory and file structure, *18*
  summary of default Rails directory structure, 19*t*
Firebug, 12, 301
Firefox, 11–12, 89, 170
first feed implementation, 532–535
flash, 315–317
  adding contents of flash variable to site layout (Listing 7.26), 315–316
  adding flash message to user signup (Listing 7.27), 317
  ERb in site layout using content tag (Listing 7.33), 324
  vs. flash.now, 316n11
  message for failed signin, 339–343, *340*
flash.now, 316n11, 342
follow and unfollow buttons, 519–529
  with Ajax, 524–529
  current user's followers, *520*
  profile of user to follow, with follow button, *486*
  profile with unfollow button and incremented followers count, *487*
  Relationships controller (Listing 11.34), 523–524
  tests for (Listing 11.32), 521–522
  tests for relationships controller authorization (Listing 11.33), 522–523
  user profile with follow button, *514*
  users being followed by current user, *520*
  working follow button, 519–524
followed users in relationship model, 495–500
follower notifications, 541
followers, 500–503
  implementing user.followers using reverse relationships (Listing 11.16), 502
  model for user followers using reverse Relationship model, *500*
  testing for reverse relationships (Listing 11.15), 501
followers relationship model, 500–503
follow form, 505–514
  adding followed_users and followers actions to Users controller (Listing 11.18), 506
  adding follow form and follower stats to user profile page (Listing 11.27), 513
  for following a user using (Listing 11.35), 525
  form for following user (Listing 11.25), 512
  form for unfollowing user (Listing 11.26), 512
  partial for follow/unfollow form (Listing 11.23), 511
  RESTful routes provided by custom rules in resource, 506*t*
  routes added for user relationships (Listing 11.24), 512
  for unfollowing a user using (Listing 11.36), 525
following
  adding following/follower relationships to sample data (Listing 11.17), 503–504
  following? and follow! utility methods (Listing 11.12), 498
  problem with the data model (and a solution), 485–490
  relationship model, 484–503
  sample following data, 503–505
  user/relationship associations, 491–494

users, 503–544
utility methods, tests for (Listing 11.11), 497
following and followers pages, 515–519
following and followers actions (Listing 11.30), 518
mockups of, *515–516*
show_follow view used to render following and followers (Listing 11.31), 519
test for followed_users and followers pages (Listing 11.29), 517–518
tests for authorization of (Listing 11.28), 516–517
following data, sample, 503–505
following? method, 497–500
follow! method, 497–500
forgery, 112
format, validating, 245–248
form_for, 297–300
form tag, 303, 334, 372
forward slash (/), 8
Fowler, Martin, 222n1
friendly forwarding, 392–396
adding store_location to signed-in user before filter (Listing 9.19), 394–395
code to implement (Listing 9.18), 394
Sessions create action with (Listing 9.20), 395
test for friendly forwarding (Listing 9.17), 393
full-table scan, 254
Fulton, Hal, 6, 543
functions, 91

G

gem configuration file
creating (Listing 1.1), 16
suppressing ri and rdoc documentation in (Listing 1.2), 16
Gemfile
adding annotate gem to (Listing 6.4), 229
adding bcrypt-ruby to (Listing 6.24), 255
adding bootstrap-sass gem to (Listing 5.3), 175
adding cucumber-rails gem to (Listing 8.31), 364
adding Factory Girl to (Listing 7.7), 284
adding Faker gem to (Listing 9.29), 403
default, in the first_app directory (Listing 1.4), 20
default Gemfile in the first app directory (Listing 1.4), 20
for demo app (Listing 2.1), 46
with explicit version of each Ruby gem (Listing 1.5), 21–22
including will paginate in (Listing 9.31), 405
needed to use PostgreSQL instead of SQLite (Listing 3.31), 117
for sample app (Listing 3.1), 78
for sample app (Listing 3.35), 123
for sample app, final (Listing 9.49), 423–424
for sample app including Guard (Listing 3.33), 120
gems, 14
gemsets, 14–15
generated code, scaffolding and, 3
generate script, 49, 86, 94
GET, 89–90
Git, 27–39
adding files in, 30–31
benefit of using, 31–32
branches, 35–36
commit command, 31
committing, 36–37
editing, 36
first-time repository setup, 28–30
first-time setup, 27–28
installing, 13
merging, 37–38
pushing, 38–39
README file, 34–35, *35*
README file, README.md (Listing 1.8), 36
README file formatted with Markdown, *39*
status command, 30
GitHub, 32–34
creating first app repository at, *33*
creating sample app repository at, *81*
initial README file for project at, *35*
repository page, *34*
.gitignore
augmented .gitignore file (Listing 1.7), 29–30
default .gitignore created by rails command (Listing 1.6), 29
Goia, Mircea, 14

Gravatar, 286–291
  adding sidebar to user show view (Listing 7.14), 290
  defining gravatar_for helper method (Listing 7.13), 288
  editing, 382–384
  SCSS for styling user show page, including sidebar (Listing 7.15), 290–291
  in sign up, 286–291
  user profile page /users/1 with default Gravatar, *289*
  user show page /users/1 with sidebar and CSS, *291*
  user show page with custom Gravatar, *289*
  user show view with name and (Listing 7.12), 287
Guard
  automated tests with, 120–122
  Gemfile for sample app including (Listing 3.33), 120
  Spork with Guard, 126–127
gVim, 28

H

Hansson, David Heinemeier, 2, 4
hashes, 337
  nested (Listing 4.6), 151
  in Ruby data structures, 148–152
hash symbol, 21
has_secure_password
  authenticate method, 264, 338
  User, 263–265
have_selector method, 104
Head First HTML, 5
Help page
  code added to test (Listing 3.11), 98
  generated view for (Listing 3.8), 92
  with HTML structure (Listing 3.20), 107
  with HTML structure removed (Listing 3.27), 113
  tests for static pages (Listing 5.27), 210
  view for, with Embedded Ruby title (Listing 3.23), 110
Heroku
  commands, 41–42
  creating a new application at (Listing 1.9), 40
  deployment, 40–41
  setup, 39–40
hierarchies, inheritance, 70–73, *73*
Home page
  adding follower stats to (Listing 11.21), 509
  adding microposts creation to (Listing 10.31), 461–462
  with follow stats, *511*
  generated view for (Listing 3.7), 92
  with HTML structure (Listing 3.19), 107
  with HTML structure removed (Listing 3.26), 113
  with link to signup page (Listing 5.2), 173
  mockup with form for creating microposts, *459*
  mockup with proto-feed, *467*
  SCSS for Home page sidebar (Listing 11.22), 510
  with status feed, mockup of, *530*
  with status feed and incremented following count, *488*
  testing following/follower statistics on (Listing 11.19), 507
  view for, with Embedded Ruby title (Listing 3.22), 109
  view for, with HTML structure (Listing 3.19), 107
  with working status feed, *539*
href, 97
HTML
  About page with HTML structure removed (Listing 3.28), 113
  About page with structure (Listing 3.21), 108
  code for signin form (Listing 8.7), 334
  for edit form defined in Listing 9.3 and shown in Figure 9.2. (Listing 9.4), 377
  for form in Figure 7.12 (Listing 7.20), 301
  initial user edit page with pre-filled name and email, *377*
  produced by CSS includes (Listing 4.7), 153
  for signin form produced by Listing 8.7, 335
  signup form, 301–303, *335*
  for signup form/signup for new users, *300*
  typical, with friendly greeting (Listing 3.3), 84
  user edit action (Listing 9.2), 375
  for user edit form (Listing 9.2), 377
HTTP, 89–90
HTTP verbs, 89, 90
hypertext reference (href), 97
hypertext transfer protocol. *See* HTTP

I

IDEs, 10
implicit return, 141
index, user. *See* User index
index action, simplified for demo app (Listing 2.4), 61
index.html file, 82–84, *83*
index page
  with 100 sample users, *405*
  correspondence between pages and URIs for Users resource, 52*t*
  initial, for Users resource, *52*
  micropost, *68*
  with one user, *402*
  with second user, *55*
  tests for (Listing 9.23), 398–399
  users with pagination, *410*
inheritance
  ApplicationController class with (Listing 2.16), 72
  class, 155–157
  classes, 155–157
  hierarchies, 70–73, *73*
  Micropost class with (Listing 2.13), 71
  MicropostsController class with (Listing 2.15), 72
  User class with (Listing 2.12), 71
  UsersController class with (Listing 2.14), 72
inheritance class, 155–157
inheritance hierarchies, 70–73, *73*
initialization hash, 231
inspect method, 151
installing Rails, 16–17
instance variables, 61, 162
  adding (empty) @feed items to create action (Listing10.45), 474–475
  adding to home action (Listing 10.41), 471
  adding to the home action (Listing 10.34), 463
  adding to user show action (Listing 10.22), 449
integrated development environments (IDEs), 10
integration tests, 93
  See also Tests
interpolation, 136–137
  string, 115, 133, 142, 162, 209
IRC client, 14n10
iTerm, 11

J

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

K

Katz, Yehuda, 364, 543

L

layout, filling in, 167–219
  adding structure, 167–186
  asset pipeline, Sass and, 187–190
  conclusion, 215–216
  exercises, 217–219
  layout links, 197–211
  stylesheets and, improving with Sass, 190–197
  user signup, 211–215
layout files
  duplication eliminated with, 103, 111–113
  sample application site layout (Listing 3.25), 112
  sample application site layout (Listing 4.1), 130
  sample application site layout (Listing 4.3), 132
  site layout with added structure (Listing 5.1), 169
layout links, 197–211
  changing for signed-in users (Listing 8.24), 357–358
  named routes, 205–207
  Rails routes, 202–205
  route tests, 200–202
  RSpec, 207–211
  test for links on layout (Listing 5.36), 218
layout links, changing, 355–359
  adding Bootstrap JavaScript library to application.js (Listing 8.25), 358

layout links, changing (*continued*)
the signed in? helper method (Listing 8.23), 356
for signed-in users (Listing 8.24), 357–358
signin success and, 355–359
length validations, 243–244
adding for name attribute (Listing 6.15), 244
constraining micropost characters (Listing 2.9), 67
test name for (Listing 6.14), 244
Linux, 13–14
Linux Mint, 14
Linux Ubuntu, 14
lists, unordered, 172
literal constructor, 153–154
literal strings, 135
Loeffler, David, 10
log, development, 231–232
log files, ignoring, 29

M

Macintosh OS X, 11
MacVim, 28
magic columns, 226, 232
manifest files in asset pipeline, 188–189
map method, 147–148, 533
mapping for site links, 198*t*
mass assignment
attr_accessible used to prevent, 230, 416
invalid, ensuring Rails throws errors on (Listing 10.6), 436
memoization, 354n7
Merb, merger with Rails, 4
message passing in Ruby, objects and, 138–141
messaging, 541
method chaining, 139, 421
method definitions, 141
micropost associations, 433–438
Micropost class with inheritance (Listing 2.13), 71
micropost data model, 48–49
micropost migration (Listing 10.1), 430
Micropost model, 429–444, *431*
accessible attributes and first validation, 432–433
basic model, 430–432
content validations, 443–444
initial Micropost spec (Listing 10.2), 431
micropost migration (Listing 10.1), 430
refinements, 439–443
tests for (Listing 10.17), 443–444
tests for validity of new micropost (Listing 10.3), 432
user has many microposts (Listing 10.11), 438
user/micropost associations, 433–438
validation for user (Listing 10.4), 433
validations (Listing 10.18), 444
microposts
adding to sample data (Listing 10.23), 451
CSS for (Listing 10.24), 452–453
destroying along with user (Listing 10.16), 443
form partial for creating (Listing 10.33), 463
ordering with default scope (Listing 10.14), 441
partial for showing single micropost (Listing 10.21), 449
sample microposts, 450–454
summary of user/micropost association methods, 434*t*
testing that microposts are destroyed when users are (Listing 10.15), 442
testing the order of a user's microposts (Listing 10.13), 440–441
Microposts, manipulating, 454–479
access control, 456–459
creating microposts, 459–467
destroying microposts, 475–479
micropost pagination links, *455*
proto-feed, 467–475
Microposts, showing, 445–454
profile page with microposts, mockup of, *445*
sample microposts, 450–454
user show page, augmenting, 446–450
Microposts controller
create action (Listing 10.30), 461
destroy action (Listing 10.49), 477–478
in schematic form (Listing 2.8), 65–66
tests for destroy action (Listing 10.48), 477
MicropostsController class with inheritance (Listing 2.15), 72
Microposts resource, 63–75
access control, 456–459
associations, 68–70
demo app, deploying, 73–74
error messages for failed micropost creation, *69*
inheritance hierarchies, 70–73, *73*

length validations, 243–244
micropost belonging to user (Listing 2.11), 69
between microposts and users, *70*
microtour, 63–66
Rails routes with new rule (Listing 2.7), 65
RESTful routes provided by, 65*t*
routes for (Listing 10.25), 455
user has many microposts (Listing 2.10), 68
validations, 66–68
microtour, 63–66
migration
to add boolean admin attribute to users (Listing 9.40), 415
micropost (Listing 10.1), 430
password, 256–257
Rake used in, 50
user model, 223–228
for User model (to create users table) (Listing 6.2), 225
mockups, 167–168
model annotation in model file, 228–230
model file, 228–230
accessible attributes, 230
model annotation, 228–230
modeling demo microposts, 48–49
modeling demo users, 47–48
modeling users, 221–269
conclusion, 267
exercises, 268–269
passwords, 254–267
user model, 222–236
user validations, 236–254
model-view-controller (MVC), 25–27
in action, 56–62
in Rails, diagram of, *57*
schematic representation of, *26*
motivation
in Ruby, 129–133
in status feed, 529–532
MVC. *See* Model-view-controller (MVC)

N

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

O

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

P

PagesController. *See* StaticPages controller
pagination, for showing all users, 404–410
paginating users in index action (Listing 9.35), 409
tests for pagination (Listing 9.33), 407–408
pagination links, micropost, *455*
palindrome? method, 155–156, 158
Paperclip gem, 287n8
partial refactoring, for showing all users, 410–412
partials, 181–186
adding CSS for site footer (Listing 5.13), 185–186
for displaying form submission error messages (Listing 7.23), 309
for HTML shim (Listing 5.9), 183
for the site footer (Listing 5.11), 184
for the site header (Listing 5.10), 184
site layout with footer partial (Listing 5.12), 185
site layout with partials for stylesheets and header (Listing 5.8), 182–183
updating error-messages (Listing 10.35), 464
passwords, 254–267
adding bcrypt-ruby to Gemfile (Listing 6.24), 255
and confirmation, 257–260
creating a user, 265–267
encrypted, 255–257
ensuring that User object has password_digest column (Listing 6.25), 256

passwords (*continued*)
migration, 256–257
migration to add password_digest column to users table (Listing 6.26), 256
reminders, 540, 541
secure, adding, 254–260
test for password and password confirmation (Listing 6.28), 259–260
testing for password and password_confirmation attributes (Listing 6.27), 257
user authentication, 260–263
user has secure password, 263–265
User model with added password_digest attribute, *255*
See also Authenticate method
Patterns of Enterprise Application Architecture (Fowler), 222n1
PeepCode, 6, 543
pending spec, 237
persistence, 223
Phusion Passenger, 39
Pik project, 13
pluralize text helper, 310
PostgreSQLn, 46–47, 115, 117, 223, 253n15
pound sign. *See* Hash symbol
preprocessor engines in asset pipeline, 189
presence, validating, 239–243
Preston-Werner, Tom, 286
private keyword, 348
production in asset pipeline, efficiency in, 189–190
profile images, 286, 382
profile links, 332
protected page
mockup of, *385*
signin form after trying to access, *388*
proto-feed, 467–475
adding feed instance variable to home action (Listing 10.41), 471
adding (empty) @feed items instance variable to create action (Listing10.45), 474–475
adding status feed to Home page (Listing 10.44), 473
Home page after creating micropost, *474*
Home page with, *473*
mockup of Home page with, *467*
preliminary implementation for micropost status feed (Listing 10.39), 469
single feed item partial (Listing 10.43), 472
status feed partial (Listing 10.42), 472
test for rendering feed on Home page (Listing 10.40), 470–471
tests for (Listing 10.38), 468
public/index.html file, *83*
puts method, 136

R

Rails
approach to learning, 4–6
deploying, 39–42
development environment setup, 9–27
environments, 276–277
intermediate-to-advanced resources, 6–7
introduction, 3–9
Merb merger and, 4
Ruby and, importance of, 129–165 (*See also* Ruby)
running to generate new application (Listing 1.3), 17–18
scaling, 7
version control with Git, 27–39
Rails, installing
Git, installing, 13
Rails, installing (Windows), 13, 16–17
Ruby, installing, 13–15
RubyGems, installing, 15–16
The Rails 3 Way (Fernandez), 6, 82n5, 142n5, 543
RailsCasts, 6, 7, 540, 543
Rails command, 17–19
default .gitignore created by (Listing 1.6), 29
to generate new application (Listing 1.3), 17–18
Rails console, 134
Rails Guides, 6, 189, 202, 228, 506, 543
Rails Machine, 39
Rails root, 8–9
Rails routes, 202–205
adding mapping for the (Listing 5.23), 204
adding Users resource to (Listing 7.3), 279
commented-out hint for defining (Listing 5.22), 204
with new rule for Microposts resource (Listing 2.7), 65

with rule for Users resource (Listing 2.2), 58
for static pages (Listing 5.21), 202
Rails server, 23–25
The Rails 3 Way (Fernandez), 6, 142n5, 543
Rails Tutorial help page, 9n6
Rake, 50, 51
task for populating database with sample users (Listing 9.30), 403–404
ranges, 145–146
README file
Git, 34–35, *35*
improved, formatted with Markdown (Listing), *39*
improved, for sample app (Listing 3.2), 80
initial, for project at GitHub, *35*
new README file, README.md (Listing 1.8), 36
updating, 80
Red, Green, Refactor, 94
Green, 100–102
Red, 99–100
Refactor, 103
refactoring
in adding static pages, 103
compact, of Listing 11.38 (Listing 11.47), 544
first attempt at index view (Listing 9.36), 411
partial, 410–412
refactored following and followers actions (Listing 11.30), 518
refinements in Micropost model, 439–443
default scope, 440–441
dependent: destroy, 441–443
regular expression (regex), 246
relationship model, 484–503, *491*
adding belongs to associations to (Listing 11.6), 494
adding indices for relationships table (Listing 11.1), 490
adding User model followed_users association (Listing 11.10), 496
followed users, 495–500
of followed users through user relationships, *489*
followers, 500–503
following? and follow! utility methods (Listing 11.12), 498
implementing user.followers using reverse relationships (Listing 11.16), 502
implementing user/relationships has_many association (Listing 11.4), 493
problem with, 485–491
for reverse relationships, 500–503
test for unfollowing a user (Listing 11.12), 499
test for user.followed_users attribute (Listing 11.9), 496
testing for reverse relationships (Listing 11.15), 501
testing for user.relationships attribute (Listing 11.3), 492
testing Relationship creation and attributes (Listing 11.2), 491–492
testing Relationship model validations (Listing 11.7), 495
testing user/relationships belongs_to association (Listing 11.5), 494
tests for "following" utility methods (Listing 11.11), 497
unfollowing user by destroying user relationship (Listing 11.14), 499–500
for user followers using reverse relationship model, *500*
user/relationship associations, 491–494
validations, 495
relationships attribute, 492
Relationships controller
Ajax requests in, responding to (Listing 11.38), 527
follow and unfollow buttons (Listing 11.34), 523–524
responses to Ajax requests, tests for (Listing 11.37), 526
reload method, 383
remember token, 344
added to cookies, 379
before_save callback to create (Listing 8.18), 348–349
cookie in local browser, *360*
current user found by using (Listing 8.22), 353
first test for (Listing 8.15), 345
migration to add to users table (Listing 8.16), 346
removed from cookies, 363
test for valid (nonblank) (Listing 8.17), 347

remember token (*continued*)
User model with added remember_token attribute, *345*
render, 183
replies, 541
repository setup, 28–30
request specs. *See* Tests
resources
advanced Rails, 4, 6
guide to further, 542–543
REST API, 542
REST architecture, 45, 59, 65, 86, 90
RESTful routes
provided by Microposts resource, 65*t*
provided by Users resource, 65*t*
reverse relationships, 500–503
followers using reverse relationship model, *500*
implementing user.followers using reverse relationships (Listing 11.16), 502
testing for reverse relationships (Listing 11.15), 501
root, 8–9
routes in layout links
named, 205–207
Rails, 202–205
tests, 200–202
RSpec
adding helper method and custom RSpec matcher (Listing 8.34), 371
Cucumber equivalent, 368–371
custom matchers, 368–371
layout links, 207–211
request specs, 93, 368
RSS feed, 542
Rubular, 247, *248*
Ruby, 129–165
comments, 134–135
conclusion, 164
exercises, 164–165
gems, 14
gemsets, 14–15
installing, 13–15
method defintions, 141
motivation, 129–133
objects and message passing, 138–141
strings, 135–138
title helper, 142
Ruby classes. *See* Classes
Ruby data structures, 142–153
arrays, 142–145
blocks, 146–148
cascading style sheets, 152–153
hashes and symbols, 148–152
ranges, 145–146
RubyGems, installing, 15–16
Ruby JavaScript (RJS)
to create following relationship (Listing 11.39), 529
to destroy following relationship (Listing 11.40), 529
RubyMine, 10
Ruby on Rails. *See* Rails
Ruby Version Manager (RVM), 8, 13, 118
The Ruby Way (Fulton), 6, 129, 543

S

Safari, 11–12, 89, 170
sample application, extensions to, 540–542
follower notifications, 541
messaging, 541
password reminders, 541
replies, 541
REST API, 542
RSS feed, 542
search, 542
signup confirmation, 541
sample users, showing all, 403–404
sandbox, 231, 252, 265
Sass, 187–197
asset pipeline and, 187–190
improving stylesheets with, 190–197
save!, 497
scaffolding, 2–3
scaling Rails, 7
scope, 440–441
screencasts, 538, 542
SCSS
converting to CSS, 192
error messages styled with, 311
for Home page sidebar (Listing 11.22), 510
initial SCSS file converted to use nesting and variables (Listing 5.15), 195–197
rewriting, 193–194
Sass supported by, 190

for styling user show page, including sidebar (Listing 7.15), 290–291
search, 542
Secure Sockets Layer (SSL), 318
deploying production with, in signup success, 317–321
Seguin, Wayne E., 13, 14
self, 157, 348
session hijacking attack, 318, 351
sessions
authentication and, 325–326
defined, 325–326
destroying a session (user signout) (Listing 8.29), 362
preliminary version of sessions create action (Listing 8.9), 337
sessions create action with friendly forwarding (Listing 9.20), 395
signin failure and, 325–326
sign out method in Sessions helper module (Listing 8.30), 363
Sessions controller
adding resource to get standard RESTful actions for sessions (Listing 8.2), 328
completed Sessions controller create action (not yet working) (Listing 8.13), 343
create action for, 326, 336–338, 343, 395
signin failure and, 326–329
tests for new session action and view (Listing 8.1), 327
short-circuit evaluation, 355
showing microposts. *See* Microposts, showing
sidebar
partial for the user info sidebar (Listing 10.32), 462
SCSS for Home page (Listing 11.22), 510
in SCSS for styling user show page (Listing 7.15), 290–291
in sign up, 288–291
signed in? helper method (Listing 8.23), 356
signed-in users
authorization of, 386–389
requiring, 386–389
sign in, 325–372
conclusion, 371–372
Cucumber, signin tests using, 363–371
exercises, 372
signin failure, 325–343
flash message, rendering with, 339–343
reviewing from submission, 336–338
sessions, 325–326
Sessions controller, 326–329
signin form, 333–336, *335*
signin tests, 330–333
signin form, 333–336, *335*
code for (Listing 8.7), 334
HTML for signin form produced by Listing 8.7 (Listing 8.8), 335
initial failed signin, with create as in Listing 8.9., *336*
signin failure and, 333–336
signing out, 361–363
destroying a session (user signout) (Listing 8.29), 362
sign out method in Sessions helper module (Listing 8.30), 363
sign_in method, signin success and, 349–351
signin success, 343–363
current user, 351–355
layout links, changing, 355–359
remembering user signin status, 343–349
signing out, 361–363
sign_in method, 349–351
signin upon signup, 359–361
signin tests
signin failure and, 330–333
using Cucumber, 363–371
signin upon signup, 359–361
sign up, 271–324
conclusion, 321
exercises, 321–324
failure in (*See* Signup failure)
Rails environments in, 276–277
showing users, 271–291
success in (*See* Signup success)
signup confirmation, 541
signup failure, 303–312, *306*
apartial for displaying form submission error messages (Listing 7.23), 309
code to display error messages on signup form (Listing 7.23), 309
create action that can handle (but not success) (Listing 7.21), 305

signup failure (*continued*)
CSS for styling error messages (Listing 7.24), 311
debug information, *307*
mockup of signup failure page, *304*
signup error messages, 308–312, *312*
working form, 303–308
signup form, 292–303
adding @user variable to the new action (Listing 7.18), 299
CSS for (Listing 7.19), 300
filled-in form with text and password fields, *302*
form_for, using, 297–300
form to sign up new users (Listing 7.17), 298
HTML, 301–303
HTML for form in figure 7.12 (Listing 7.20), 301
for new users, *300*
tests for signing up users (Listing 7.16), 296–297
tests for user signup, 293–297
using form_for, 297–300
signup page
initial (stub) (Listing 5.33), 214
linking the button to (Listing 5.34), 215
route for (Listing 5.32), 214
signing in user upon signup (Listing 8.27), 361
signin upon signup, 359–361
testing that newly signed-up users are also signed in (Listing 8.26), 360–361
Users controller, 212
signup success, 312–321
deploying production with SSL, 317–321
finished signup form, 313–315
first signup, 317
flash, 315–319
mockup of, *314*
signup URI, in user signup, 213–215
site navigation in filling in layout, 169–175
Home page with link to signup page (Listing 5.2), 173
site layout with added structure (Listing 5.1), 169
skeleton for a shuffle method attached to the String class (Listing 4.11), 165
skeleton for a string shuffle function (Listing 4.10), 164
slightly dynamic pages, 103–113
duplication, eliminating with layouts, 103, 111–113
Embedded Ruby, 108–111
instance variables and Embedded Ruby, 162
passing title tests, 106–108
testing a title change, 103–107
testing title page, 103–106
spike, 93
split method, 143
Spork, 123–127
adding environment loading to Spork.prefork block (Listing 3.36), 124
configuring RSpec to automatically use (Listing 3.37), 125
Gemfile for sample app (Listing 3.35), 123
Guardfile updated for Spork (Listing 3.38), 126
Guard with Spork, 126–127
speeding up tests with, 123–127
SQL injection, 470
SQLite Database Browser, 226, *227,* 266
Stack Overflow, 301, 492n4
staging area, 30
static pages, 77–128
conclusion, 114
exercises, 114–117
test-driven development, 93–99
testing, 93–103
See also Slightly dynamic pages
static pages, adding, 99–103
green, 100–102
red, 99–100
refactor, 103
static pages, advanced setup, 117–128
bundle exec, eliminating, 118–119
Guard, automated tests with, 120–122
Spork, speeding up tests with, 123–127
Sublime Text, tests inside, 127–128
static pages, making, 82–92
with Rails, 85–92
truly static pages, 82–85
undoing things, 87–88
StaticPages controller
with about action (Listing 3.15), 101
generating (Listing 3.4), 86
inheritance hierarchy for, *160*

made by Listing 3.4 (Listing 3.6), 91
routes for home and help actions in (Listing 3.5), 88
spec with base title (Listing 3.29), 115–116
spec with title tests (Listing 3.18), 105
stats, 505–514
adding follower stats to Home page (Listing 11.21), 509
adding follow form and follower stats to user profile page (Listing 11.27), 513
Home page with follow stats, *511*
mockup of stats partial, *505*
a partial for displaying follower stats (Listing 11.20), 508
SCSS for Home page sidebar (Listing 11.22), 510
testing following/follower statistics on the Home page (Listing 11.19), 507
stats form, 505–514
status command, in Git, 30
status feed, 529–539
adding completed feed to User model (Listing 11.42), 532
final implementation of from_users_followed_by (Listing 11.45), 537–538
final tests for (Listing 11.41), 531–532
first cut at from_users_followed_by (Listing 11.43), 535
first feed implementation, 532–535
home action with paginated feed (Listing 11.46), 538
Home page with working status feed, *539*
improving from_users_followed_by (Listing 11.44), 536
mockup of a user's Home page with, *530*
motivation and strategy, 529–532
new, 538–539
partial for a single feed item (Listing 10.43), 472
preliminary implementation for micropost (Listing 10.39), 469
subselects, 535–538
for user following users, *531*
strategy in status feed, 529–532
string interpolation, 115, 133, 142, 162, 209
string literals, 135
strings
double-quoted, 137–138
printing, 136–137
in Ruby, 135–138
single-quoted, 137–138
structure in filling in layout, 167–186
bootstrap and custom CSS, 175–186
partials, 181–186
site navigation, 169–175
stylesheets. *See* Cascading style sheets (CSS)
Sublime Text, tests inside, 127–128
Sublime Text 2, 10, 16, 127
subselects in status feed, 535–538
sudo, 8
superclass method, 155
symbols, 148–152
system setups, 27

T

TDD. *See* Test-driven development (TDD)
terminals, 11
ternary operator, 481, 482
test-driven development (TDD), 5
Green, 100–102
Red, 99–100
Red, Green, Refactor, 94
Refactor, 103
Spork, 123–127
in testing static pages, 93–99
testing tools, 93
tests
for admin attribute (Listing 9.39), 414
for authorization of following and followers pages (Listing 11.28), 516–517
automated tests with Guard, 120–122
for Contact page (Listing 5.16), 198
for creating microposts (Listing 10.29), 460
for delete links (Listing 9.44), 418–419
for destroy action in Microposts controller (Listing 10.48), 477
for email format validation (Listing 6.16), 245–246
for follow and unfollow buttons (Listing 11.32), 521–522
for "following" utility methods (Listing 11.11), 497
for friendly forwarding (Listing 9.17), 393

tests (*continued*)
for full_title helper (Listing 5.37), 219
Guard, automated tests with, 120–122
for index page (Listing 9.23), 398–399
integration tests, 93
for Micropost model (Listing 10.17), 443–444
for Microposts controller destroy action (Listing 10.48), 477
for micropost's user association (Listing 10.8), 437
for new session action and view (Listing 8.1), 327
for pagination (Listing 9.33), 407–408
passing title, 106–108
for post-save behavior in (Listing 7.32), 323
for proto-feed (Listing 10.38), 468
for Relationships controller (Listing 11.37), 526
for relationships controller authorization (Listing 11.33), 522–523
for Relationships controller authorization (Listing 11.33), 522–523
for responses to Ajax requests (Listing 11.37), 526
for reverse relationships (Listing 11.15), 501
for routes in layout links, 200–202
for showing microposts on user show page (Listing 10.19), 446
signin, using Cucumber, 363–371
for signin failure, 330–333
for signing up users (Listing 7.16), 296–297
signin tests using Capybara, 294, 330
signin tests using Cucumber, 363–371
spec with title tests (Listing 3.18), 105
speeding up with Spork, 123–127
static pages (Listing 5.27), 210
for static pages, 93–99
for static pages (Listing 5.27), 210
for status feed, final (Listing 11.41), 531–532
Sublime Text, tests inside, 127–128
for title change, 103–106
title test (Listing 3.17), 104
user, initial, 236–239
for user index page (Listing 9.23), 398–399
for user show page (Listing 7.9), 285
for user signup, 293–297
for user's microposts attribute (Listing 10.9), 437–438
for user update action (Listing 9.9), 383
for user validations, initial, 236–239
for utility methods, (Listing 11.11), 497
for validity of new micropost (Listing 10.3), 432
text editors, 10, *11*
TextMate, 10, 28, 105n12
time helpers, 350
timestamps, 225
title change
passing title tests, 106–107
testing, 103–106
title helper, 142
tests for full_title helper (Listing 5.37), 219
title test (Listing 3.17), 104
toggle method, 414
tools, learning, 12
Torvalds, Linus, 27

U

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

URLs
  correspondence between pages and Users resource, 52*t*
  defined, 2n1
user
  administrative, 413–417
  creating, 265–267
  current user? method (Listing 9.16), 392
  destroying, 499–500
  has secure password, 263–265
  new user view with partial (Listing 9.51), 425
  paginating, 404–410
  requiring signed-in users, 386–389
  requiring the right user, 390–392
  sample users, 403–404
  showing, 271–291
  signin status, remembering, 343–349
  stub view for showing user information (Listing 7.4), 280
  summary of user/micropost association methods/updating, 434*t*
  tests, initial, 236–239
user authentication. *See* Authentication
user authorization. *See* Authorization
user class, 161–163
User class with inheritance (Listing 2.12), 71
user data model, 47–48
user edit form
  adding test for Settings link (Listing 9.5), 378
  HTML for (Listing 9.2), 377
  mockup of, *374*
  partial for new and edit form fields (Listing 9.50), 425
  tests for user update action (Listing 9.9), 383
  updating error-messages partial from Listing 7.23 to work with other objects (Listing 10.35), 464
  updating trendering of user signup errors (Listing 10.36), 465
  user edit action (Listing 9.2), 375
  user edit view (Listing 9.3), 376
  user update action (Listing 9.10), 384
user.followers method, 500
user has_many microposts (Listing 10.11), 438
  micropost belongs to user (Listing 2.11), 69
  relationship between a user and its microposts, *434*
user index, 396–403
  adding URI to users link (Listing 9.28), 401–402
  CSS for (Listing 9.26), 400
  first refactoring attempt at index view (Listing 9.36), 411
  including will paginate in Gemfile (Listing 9.31), 405
  mockup of, *397*
  paginating users in index action (Listing 9.35), 409
  pagination, 404–410
  with pagination (Listing 9.34), 408
  partial refactoring, 410–412
  partial to render single user (Listing 9.37), 412
  refactored (Listing 9.38), 412
  requiring signed-in user for index action (Listing 9.22), 398
  for showing all users, 396–403
  test for "Users" link URI (Listing 9.27), 401
  testing that index action is protected (Listing 9.21), 396–397
  tests for pagination (Listing 9.33), 407–408
  user index action (Listing 9.24), 399
  user index view (Listing 9.25), 400
  view (Listing 9.25), 400
user index page
  page 2 of, *411*
  tests for (Listing 9.23), 398–399
  users with 100 sample users, *405*
  users with only one user, *402*
  users with pagination, *410*
user info sidebar, partial for (Listing 10.32), 462
user/micropost associations, 433–438
User microposts, 429–482
  conclusion, 479–480
  exercises, 480–482
  manipulating, 454–479
  model, 429–444, *431*
  resources, 63–74
  showing, 445–454
User model, 222–236
  accessible attributes, 230
  with added password_digest attribute, *255*
  adding annotate gem to Gemfile (Listing 6.4), 229

User model (*continued*)
annotated User model (Listing 6.5), 229–230
brand new (Listing 6.3), 228
for demo application (Listing 2.5), 61
generating (Listing 6.1), 224
making name and email attributes accessible (Listing 6.6), 230
migration for (to create a users table) (Listing 6.2), 225
migrations, 223–228
model file, 228–230
user objects, 230–236
user objects
creating, 230–233
finding, 233–235
updating, 235–236
user profile page, mockup of, *445*
user/relationship associations, 491–494
implementing has_many association (Listing 11.4), 493
See also Relationship model
users, deleting, 413–422
administrative users, 413–417
destroy action, 417–422
users, following, 483–544
conclusion, 539–543
current user's profile, *484*
exercises, 543–544
finding a user to follow, *485*
Home page with status feed and incremented following count, *488*
implementation of user following, *488*
model of followed users through user relationships, *489*
profile of user to follow, with follow button, *486*
profile with unfollow button and incremented followers count, *487*
resources, guide to further, 542–543
sample application, extensions to, 540–542
status feed, 529–539
test for unfollowing (Listing 11.13), 499
web interface for, 503–529
See also Relationship model
users, showing all, 396–412
pagination, 404–410
partial refactoring, 410–412
sample users, 403–404
user index, 396–403
users, showing in sign up, 271–291
debug information, 272–276
Gravatar, 286–291
Rails environments, 276–277
sidebar, 288–291
user show page, testing, 282–286
Users resource, 278–281
users, updating, 373–385
edit form, 374–380
successful edits, 382–384
unsuccessful edits, 380–382
Users controller, 212
adding followed_users and followers actions to Users controller (Listing 11.18), 506
class with inheritance (Listing 2.14), 72
create action for, 425, 459
initial, with new action (Listing 5.29), 212
in schematic form (Listing 2.3), 58
with show action (Listing 7.5), 281
testing the user show page with factories, 282–286
in user signup, 212
user show page, *53,* 282–286
adding sidebar to user show view (Listing 7.14), 290
adding title and heading for user profile page (Listing 7.10), 285
defining gravatar_for helper method (Listing 7.13), 288
factories to simulate User model objects (Listing 7.8), 284
Factory Girl added to Gemfile (Listing 7.7), 284
in Microposts, augmenting, 446–450
recap of initial User pages spec (Listing 7.6), 282–283
redefining BCrypt cost factor in test environment (Listing 7.11), 286
SCSS for styling, including sidebar (Listing 7.15), 290–291
tests for (Listing 7.9), 285
user profile page /users/1 with default Gravatar, *289*
at /users/1 after adding Users resource, *282*

Users controller with show action (Listing 7.5), 281
user show page /users/1 with sidebar and CSS, *291*
user show page with custom Gravatar, *289*
user show view with name and (Listing 7.12), 287
user signup, 211–215
adding flash message to (Listing 7.27), 317
errors, updating rendering of (Listing 10.36), 465
signup URI, 213–215
tests for, 293–297
users controller, 212
Users resource, 49–63
adding to the routes file (Listing 7.3), 279
correspondence between pages and URLs, 52*t*
MVC in action, 56–62
Rails routes with rule for (Listing 2.2), 58
RESTful routes provided by, 65*t*
in sign up, 278–281
weaknesses of, 62–63
Users resource tour, 51–56
user update action (Listing 9.10), 384
initial (Listing 9.8), 381
tests for (Listing 9.9), 383
user validations, 236–254
format, 245–248
length, 243–244
presence, 239–243
uniqueness, 249–254
user tests, initial, 236–239

V

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

W

web interface for following users, 503–529
adding following/follower relationships to sample data (Listing 11.17), 503–504
follow button with Ajax, working, 524–529

web interface for following users (*continued*)
follow form, 505–514
following and followers pages, 515–519
follow/unfollow buttons, working, 519–524
sample following data, 503–505
stats, 505–514
Webrat, 79n1
The Well-Grounded Rubyist (Black), 6, 129, 543
will paginate method, 408
Windows, 11
wireframes, 167
wrapping long words, helper for (Listing 10.50), 481

Y

YAML, 276n3

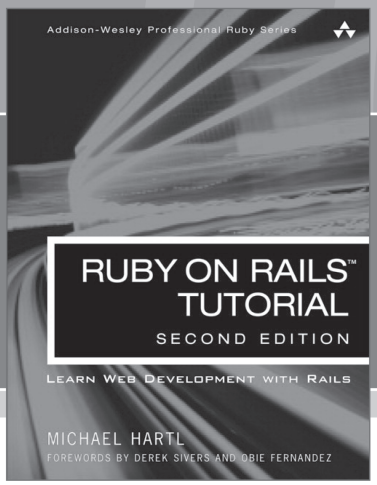
Z

zero-offset, 143

From Hartl/Ruby on Rails™ Tutorial, Second Edition
ISBN-10 0321832051 (ISBN-13 9780321832054).

From Hartl/Ruby on Rails™ Tutorial, Second Edition
ISBN-10 0321832051 (ISBN-13 9780321832054).

From Hartl/Ruby on Rails™ Tutorial, Second Edition
ISBN-10 0321832051 (ISBN-13 9780321832054).



Safari Books Online

FREE Online Edition

Your purchase of ***Ruby on Rails™ Tutorial, Second Edition,*** includes access to a free online edition for 45 days through the **Safari Books Online** subscription service. Nearly every Addison-Wesley Professional book is available online through **Safari Books Online**, along with thousands of books and videos from publishers such as Cisco Press, Exam Cram, IBM Press, O'Reilly Media, Prentice Hall, Que, Sams, and VMware Press.

Safari Books Online is a digital library providing searchable, on-demand access to thousands of technology, digital media, and professional development books and videos from leading publishers. With one monthly or yearly subscription price, you get unlimited access to learning tools and information on topics including mobile app and software development, tips and tricks on using your favorite gadgets, networking, project management, graphic design, and much more.

Activate your FREE Online Edition at informit.com/safarifree

STEP 1: Enter the coupon code: VDRAOVH.

STEP 2: New Safari users, complete the brief registration form. Safari subscribers, just log in.

If you have difficulty registering on Safari or accessing the online edition, please e-mail customer-service@safaribooksonline.com

From Hartl/Ruby on Rails™ Tutorial, Second Edition
ISBN-10 0321832051 (ISBN-13 9780321832054).