Listing 13.51: Setting an explicit controller and action.
app/views/shared/_feed.html.erb

```
<% if @feed_items.any? %>
  <ol class="microposts">
    <%= render @feed_items %>
  </ol>
  <%= will_paginate @feed_items,
                    params: { controller: :static_pages, action: :home } %>
<% end %>
```

Now clicking on either of the pagination links in Figure 13.17 yields the expected second page, as shown in Figure 13.19.

Exercises

Solutions to the exercises are available to all Rails Tutorial purchasers here.

To see other people's answers and to record your own, subscribe to the Rails Tutorial course or to the Learn Enough All Access Bundle.

1. Use the newly created micropost UI to create the first real micropost. What are the contents of the `INSERT` command in the server log?
2. In the console, set `user` to the first user in the database. Confirm that the values of `Micropost.where("user_id = ?", user.id)`, `user.microposts`, and `user.feed` are all the same. *Hint*: It's probably easiest to compare directly using `==`.

13.3.4 Destroying microposts

The last piece of functionality to add to the Microposts resource is the ability to destroy posts. As with user deletion (Section 10.4.2), we accomplish this with "delete" links, as mocked up in Figure 13.20. Unlike that case, which restricted user destruction to admin users, the delete links will work only for microposts created by the current user.

Our first step is to add a delete link to the micropost partial as in Listing 13.22. The result appears in Listing 13.52.

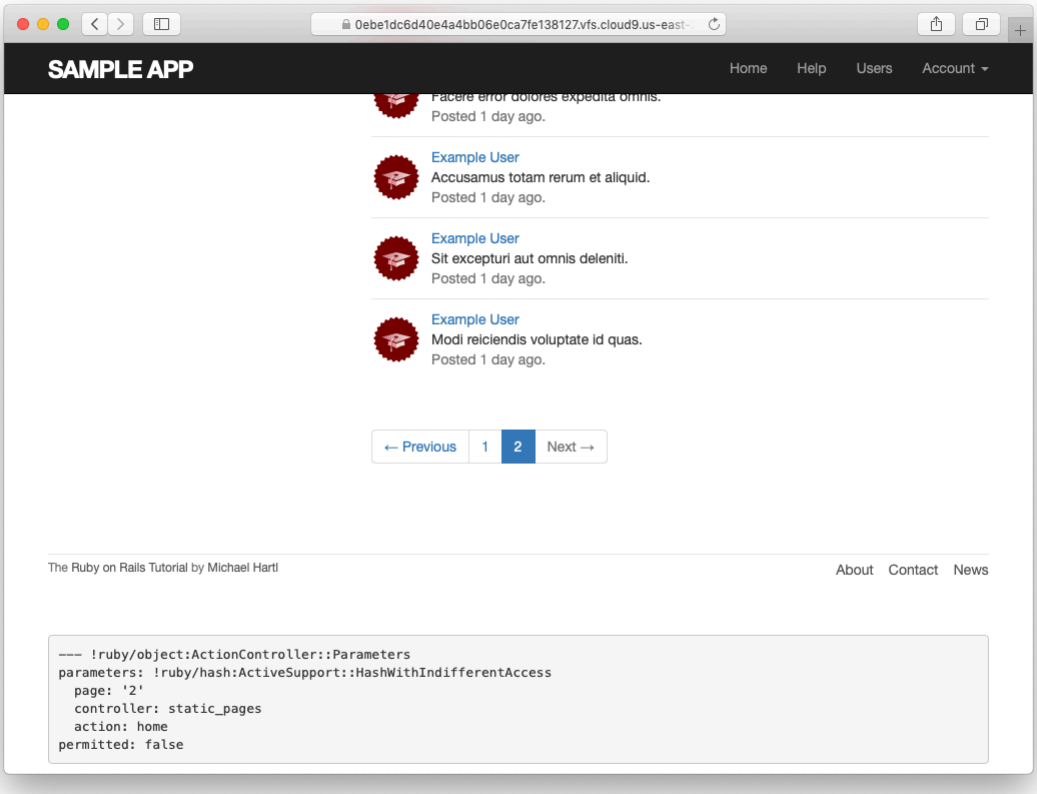

Figure 13.19: The result of a working pagination link to the second page.

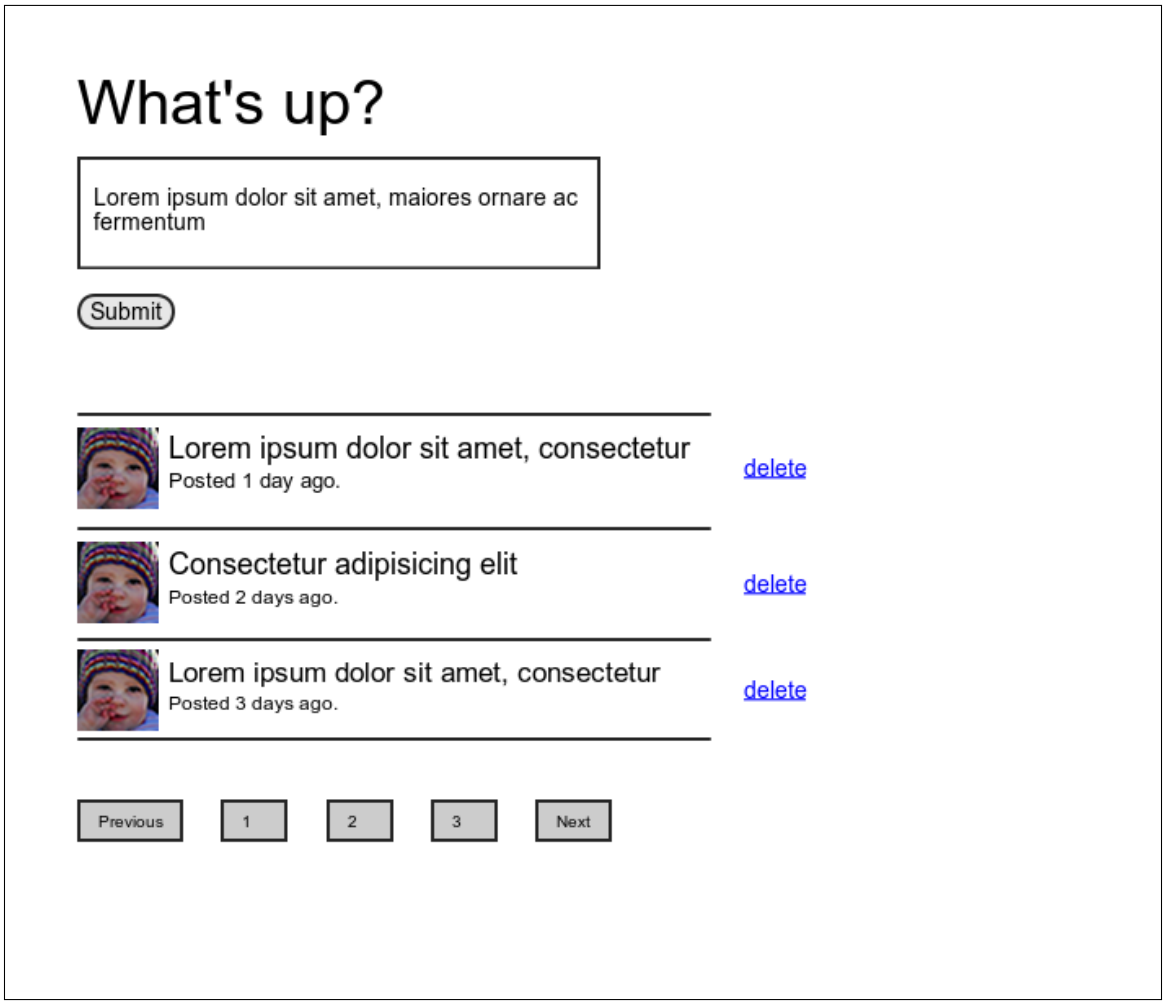

Figure 13.20: A mockup of the proto-feed with micropost delete links.

Listing 13.52: Adding a delete link to the micropost partial.
app/views/microposts/_micropost.html.erb

```
<li id="micropost-<%= micropost.id %>">
  <%= link_to gravatar_for(micropost.user, size: 50), micropost.user %>
  <span class="user"><%= link_to micropost.user.name, micropost.user %></span>
  <span class="content"><%= micropost.content %></span>
  <span class="timestamp">
    Posted <%= time_ago_in_words(micropost.created_at) %> ago.
    <% if current_user?(micropost.user) %>
      <%= link_to "delete", micropost, method: :delete,
                                       data: { confirm: "You sure?" } %>
    <% end %>
  </span>
</li>
```

The next step is to define a **destroy** action in the Microposts controller, which is analogous to the user case in Listing 10.59. The main difference is that, rather than using an **@user** variable with an **admin_user** before filter, we'll find the micropost through the association, which will automatically fail if a user tries to delete another user's micropost. We'll put the resulting **find** inside a **correct_user** before filter, which checks that the current user actually has a micropost with the given id. The result appears in Listing 13.53.

Listing 13.53: The Microposts controller **destroy** action.
app/controllers/microposts_controller.rb

```
class MicropostsController < ApplicationController
  before_action :logged_in_user, only: [:create, :destroy]
  before_action :correct_user,   only: :destroy
  .
  .
  .
  def destroy
    @micropost.destroy
    flash[:success] = "Micropost deleted"
    redirect_to request.referrer || root_url
  end

  private

    def micropost_params
      params.require(:micropost).permit(:content)
```

```
    end

    def correct_user
      @micropost = current_user.microposts.find_by(id: params[:id])
      redirect_to root_url if @micropost.nil?
    end
end
```

Note that the **destroy** method in Listing 13.53 redirects to the URL

```
request.referrer || root_url
```

This uses the **request.referrer** method,[14] which is related to the **request.original_url** variable used in friendly forwarding (Section 10.2.3), and is just the previous URL (in this case, the Home page).[15] This is convenient because microposts appear on both the Home page and on the user's profile page, so by using **request.referrer** we arrange to redirect back to the page issuing the delete request in both cases. If the referring URL is **nil** (as is the case inside some tests), Listing 13.53 sets the **root_url** as the default using the **||** operator. (Compare to the default options defined in Listing 9.24.)

With the code as above, the Home page has working delete links (Figure 13.21), which you can verify by deleting, e.g., the second post (Figure 13.22).

Exercises

Solutions to the exercises are available to all Rails Tutorial purchasers here.

To see other people's answers and to record your own, subscribe to the Rails Tutorial course or to the Learn Enough All Access Bundle.

1. Create a new micropost and then delete it. What are the contents of the **DELETE** command in the server log?

[14] This corresponds to `HTTP_REFERER`, as defined by the specification for HTTP. Note that "referer" is not a typo—the word is actually misspelled in the spec. Rails corrects this error by writing "referrer" instead.

[15] I didn't remember offhand how to get this URL inside a Rails application, so I Googled "rails request previous url" and found a Stack Overflow thread with the answer.

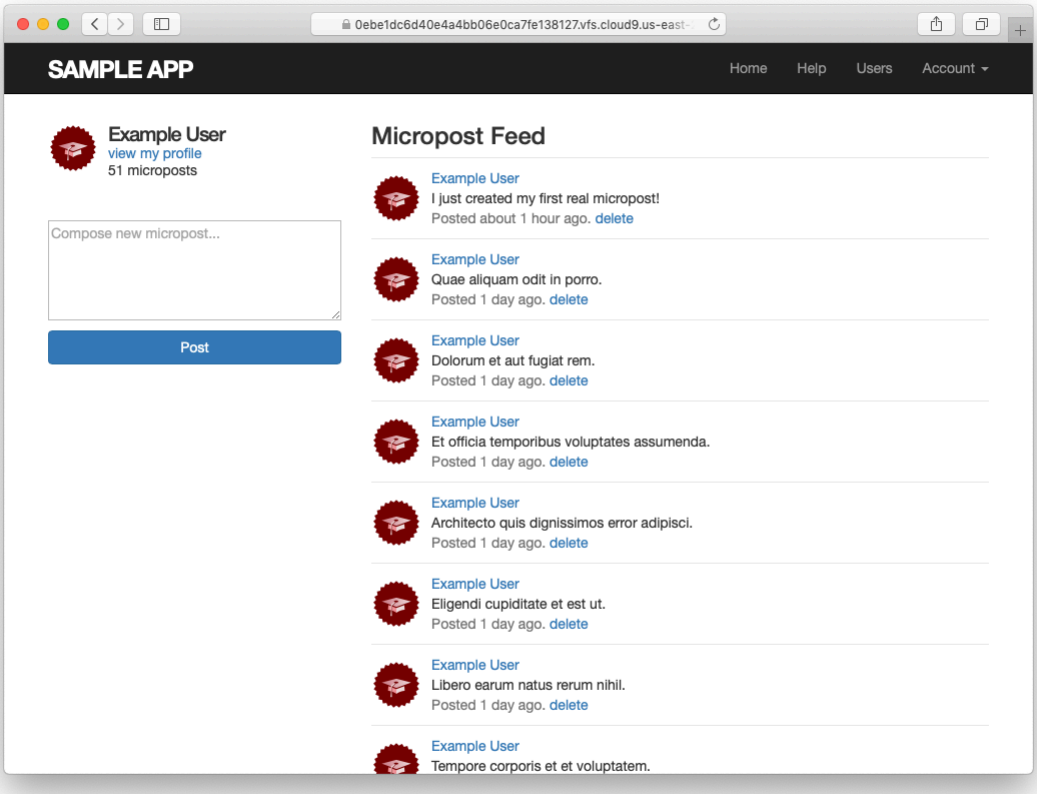

Figure 13.21: The Home page with delete links.

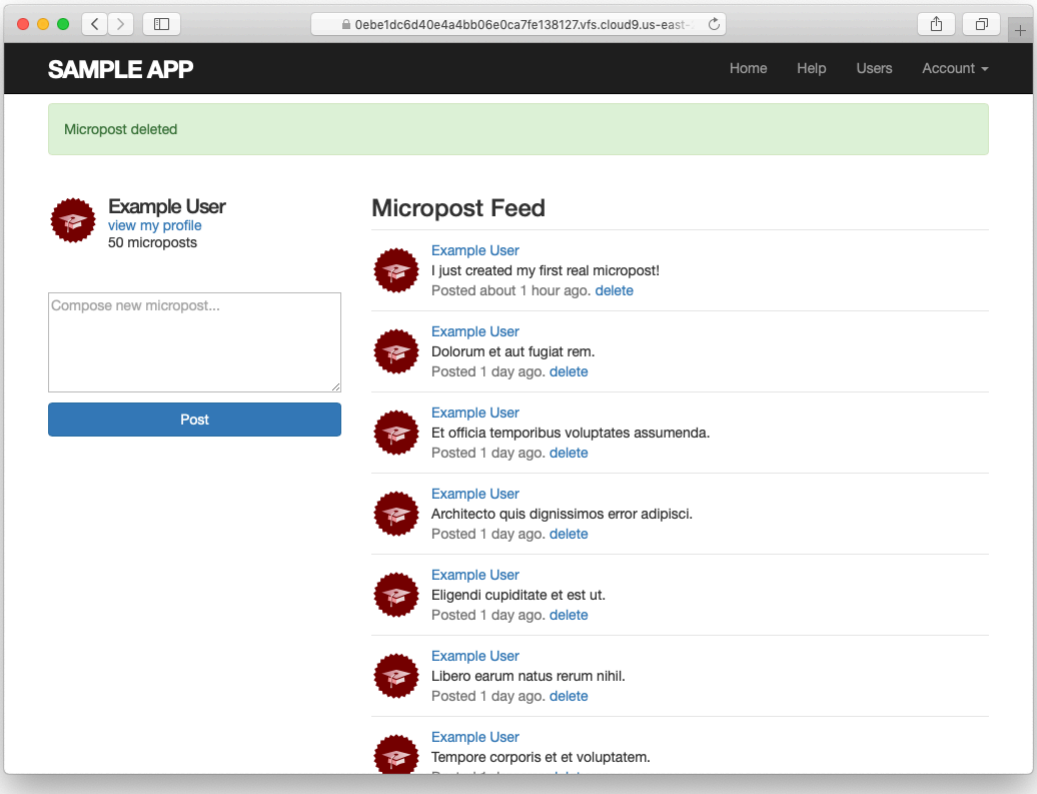

Figure 13.22: The result of deleting the second post.

2. Confirm directly in the browser that the line `redirect_to request.referrer || root_url` can be replaced with the line `redirect_back(fallback_location: root_url)`. (This method was added in Rails 5.)

13.3.5 Micropost tests

With the code in Section 13.3.4, the Micropost model and interface are complete. All that's left is writing a short Microposts controller test to check authorization and a micropost integration test to tie it all together.

We'll start by adding a few microposts with different owners to the micropost fixtures, as shown in Listing 13.54. (We'll be using only one for now, but we've put in the others for future reference.)

Listing 13.54: Adding a micropost with a different owner.
test/fixtures/microposts.yml

```
.
.
.
ants:
  content: "Oh, is that what you want? Because that's how you get ants!"
  created_at: <%= 2.years.ago %>
  user: archer

zone:
  content: "Danger zone!"
  created_at: <%= 3.days.ago %>
  user: archer

tone:
  content: "I'm sorry. Your words made sense, but your sarcastic tone did not."
  created_at: <%= 10.minutes.ago %>
  user: lana

van:
  content: "Dude, this van's, like, rolling probable cause."
  created_at: <%= 4.hours.ago %>
  user: lana
```

We next write a short test to make sure one user can't delete the microposts of a different user, and we also check for the proper redirect, as seen in

Listing 13.55.

Listing 13.55: Testing micropost deletion with a user mismatch. GREEN
test/controllers/microposts_controller_test.rb

```
require 'test_helper'

class MicropostsControllerTest < ActionDispatch::IntegrationTest

  def setup
    @micropost = microposts(:orange)
  end

  test "should redirect create when not logged in" do
    assert_no_difference 'Micropost.count' do
      post microposts_path, params: { micropost: { content: "Lorem ipsum" } }
    end
    assert_redirected_to login_url
  end

  test "should redirect destroy when not logged in" do
    assert_no_difference 'Micropost.count' do
      delete micropost_path(@micropost)
    end
    assert_redirected_to login_url
  end

  test "should redirect destroy for wrong micropost" do
    log_in_as(users(:michael))
    micropost = microposts(:ants)
    assert_no_difference 'Micropost.count' do
      delete micropost_path(micropost)
    end
    assert_redirected_to root_url
  end
end
```

Finally, we'll write an integration test to log in, check the micropost pagination, make an invalid submission, make a valid submission, delete a post, and then visit a second user's page to make sure there are no "delete" links. We start by generating a test as usual:

```
$ rails generate integration_test microposts_interface
      invoke  test_unit
      create    test/integration/microposts_interface_test.rb
```

The test appears in Listing 13.56. See if you can connect the lines in Listing 13.12 to the steps mentioned above.

Listing 13.56: An integration test for the micropost interface. GREEN
test/integration/microposts_interface_test.rb

```
require 'test_helper'

class MicropostsInterfaceTest < ActionDispatch::IntegrationTest

  def setup
    @user = users(:michael)
  end

  test "micropost interface" do
    log_in_as(@user)
    get root_path
    assert_select 'div.pagination'
    # Invalid submission
    assert_no_difference 'Micropost.count' do
      post microposts_path, params: { micropost: { content: "" } }
    end
    assert_select 'div#error_explanation'
    assert_select 'a[href=?]', '/?page=2'  # Correct pagination link
    # Valid submission
    content = "This micropost really ties the room together"
    assert_difference 'Micropost.count', 1 do
      post microposts_path, params: { micropost: { content: content } }
    end
    assert_redirected_to root_url
    follow_redirect!
    assert_match content, response.body
    # Delete post
    assert_select 'a', text: 'delete'
    first_micropost = @user.microposts.paginate(page: 1).first
    assert_difference 'Micropost.count', -1 do
      delete micropost_path(first_micropost)
    end
    # Visit different user (no delete links)
    get user_path(users(:archer))
    assert_select 'a', text: 'delete', count: 0
  end
end
```

Because we wrote working application code first, the test suite should be GREEN:

Listing 13.57: GREEN

```
$ rails test
```

Exercises

Solutions to the exercises are available to all Rails Tutorial purchasers here.

To see other people's answers and to record your own, subscribe to the Rails Tutorial course or to the Learn Enough All Access Bundle.

1. For each of the four scenarios indicated by comments in Listing 13.56 (starting with "Invalid submission"), comment out application code to get the corresponding test to RED, then uncomment to get back to GREEN.
2. Add tests for the sidebar micropost count (including proper pluralization). Listing 13.58 will help get you started.

Listing 13.58: A template for the sidebar micropost count test.
test/integration/microposts_interface_test.rb

```
require 'test_helper'

class MicropostInterfaceTest < ActionDispatch::IntegrationTest

  def setup
    @user = users(:michael)
  end
  .
  .
  .
  test "micropost sidebar count" do
    log_in_as(@user)
    get root_path
    assert_match "#{FILL_IN} microposts", response.body
    # User with zero microposts
    other_user = users(:malory)
    log_in_as(other_user)
    get root_path
    assert_match "0 microposts", response.body
    other_user.microposts.create!(content: "A micropost")
    get root_path
```