2. Put the **`debugger`** in the User **`new`** action and hit /users/new. What is the value of **`@user`**?

7.1.4 A Gravatar image and a sidebar

Having defined a basic user page in the previous section, we'll now flesh it out a little with a profile image for each user and the first cut of the user sidebar. We'll start by adding a "globally recognized avatar", or Gravatar, to the user profile.[8] Gravatar is a free service that allows users to upload images and associate them with email addresses they control. As a result, Gravatars are a convenient way to include user profile images without going through the trouble of managing image upload, cropping, and storage; all we need to do is construct the proper Gravatar image URL using the user's email address and the corresponding Gravatar image will automatically appear. (We'll learn how to handle custom image upload in Section 13.4.)

Our plan is to define a **`gravatar_for`** helper function to return a Gravatar image for a given user, as shown in Listing 7.8.

Listing 7.8: The user show view with name and Gravatar.
`app/views/users/show.html.erb`

```
<% provide(:title, @user.name) %>
<h1>
  <%= gravatar_for @user %>
  <%= @user.name %>
</h1>
```

By default, methods defined in any helper file are automatically available in any view, but for convenience we'll put the **`gravatar_for`** method in the file for helpers associated with the Users controller. As noted in the Gravatar documentation, Gravatar URLs are based on an MD5 hash of the user's email

[8]In Hinduism, an avatar is the manifestation of a deity in human or animal form. By extension, the term *avatar* is commonly used to mean some kind of personal representation, especially in a virtual environment. (In the context of Twitter and other social media, the term *avi* has gained currency, which is likely a mutated form of *avatar*.)

address. In Ruby, the MD5 hashing algorithm is implemented using the **hexdigest** method, which is part of the **Digest** library:

```
>> email = "MHARTL@example.COM"
>> Digest::MD5::hexdigest(email.downcase)
=> "1fda4469bcbec3badf5418269ffc5968"
```

Since email addresses are case-insensitive (Section 6.2.4) but MD5 hashes are not, we've used the **downcase** method to ensure that the argument to **hexdigest** is all lower-case. (Because of the email downcasing callback in Listing 6.32, this will never make a difference in this tutorial, but it's a good practice in case the **gravatar_for** ever gets used on email addresses from other sources.) The resulting **gravatar_for** helper appears in Listing 7.9.

Listing 7.9: Defining a **gravatar_for** helper method.
app/helpers/users_helper.rb

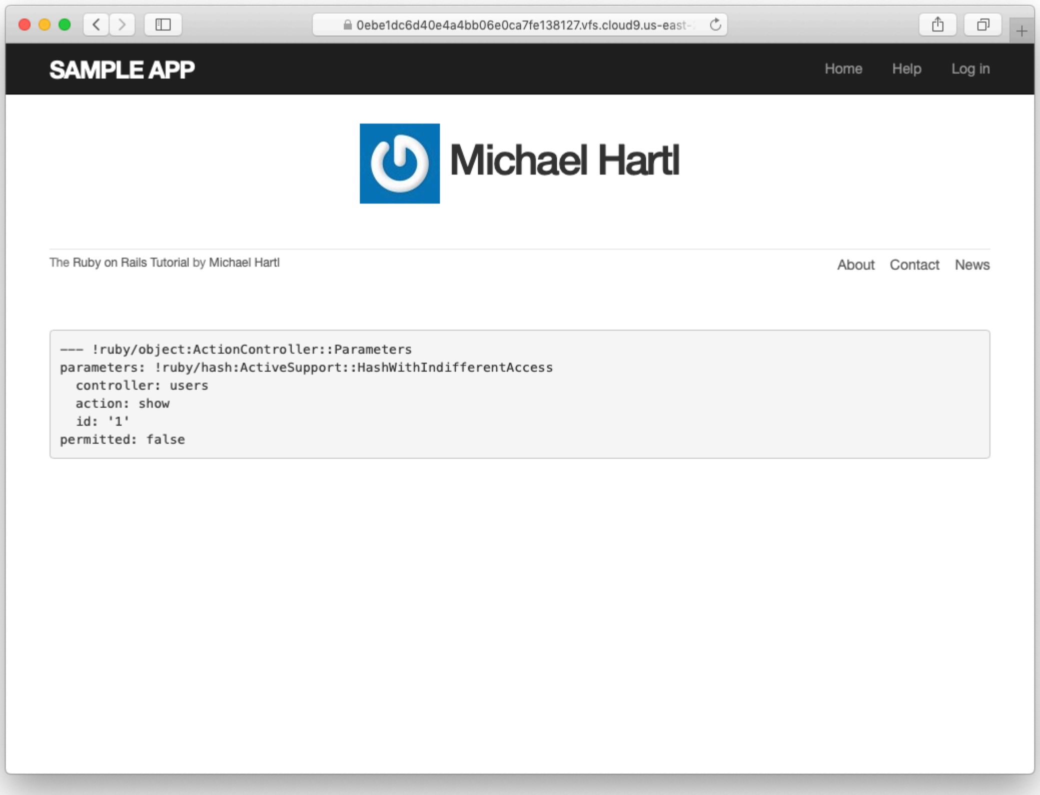
```
module UsersHelper

  # Returns the Gravatar for the given user.
  def gravatar_for(user)
    gravatar_id  = Digest::MD5::hexdigest(user.email.downcase)
    gravatar_url = "https://secure.gravatar.com/avatar/#{gravatar_id}"
    image_tag(gravatar_url, alt: user.name, class: "gravatar")
  end
end
```

The code in Listing 7.9 returns an image tag for the Gravatar with a **gravatar** CSS class and alt text equal to the user's name (which is especially convenient for visually impaired users using a screen reader).

The profile page appears as in Figure 7.8, which shows the default Gravatar image, which appears because **michael@example.com** isn't a real email address. (In fact, as you can see by visiting it, the example.com domain is reserved for examples like this one.)

To get our application to display a custom Gravatar, we'll use **update_attributes** (Section 6.1.5) to change the user's email to something I control:[9]

[9]The password confirmation isn't technically necessary here because **has_secure_password** (Section 6.3.1)


0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-
SAMPLE APP
Home
Help
Log in
Michael Hartl
The Ruby on Rails Tutorial by Michael Hartl
About
Contact
News

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  controller: users
  action: show
  id: '1'
permitted: false
```

Figure 7.8: The user profile page with the default Gravatar.

```
$ rails console
>> user = User.first
>> user.update(name: "Example User",
?>              email: "example@railstutorial.org",
?>              password: "foobar",
?>              password_confirmation: "foobar")
=> true
```

Here we've assigned the user the email address **example@railstutorial-.org**, which I've associated with the Rails Tutorial logo, as seen in Figure 7.9.

The last element needed to complete the mockup from Figure 7.1 is the initial version of the user sidebar. We'll implement it using the **aside** tag, which is used for content (such as sidebars) that complements the rest of the page but can also stand alone. We include **row** and **col-md-4** classes, which are both part of Bootstrap. The code for the modified user show page appears in Listing 7.10.

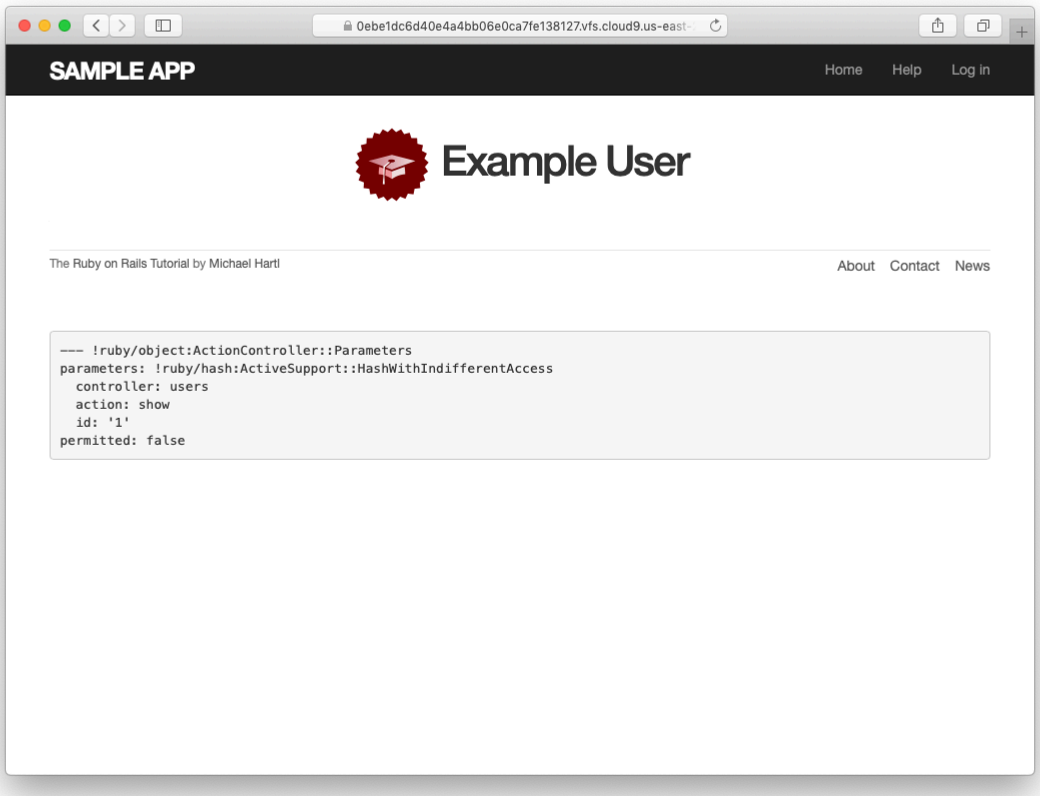
Listing 7.10: Adding a sidebar to the user **show** view.
app/views/users/show.html.erb

```
<% provide(:title, @user.name) %>
<div class="row">
  <aside class="col-md-4">
    <section class="user_info">
      <h1>
        <%= gravatar_for @user %>
        <%= @user.name %>
      </h1>
    </section>
  </aside>
</div>
```

With the HTML elements and CSS classes in place, we can style the profile page (including the sidebar and the Gravatar) with the SCSS shown in Listing 7.11.[10] (Note the nesting of the table CSS rules, which works only because of the Sass engine used by the asset pipeline.) The resulting page is shown in Figure 7.10.

actually allows the confirmation to be **nil**. The reason is so that apps that don't need password confirmation can simply omit the confirmation field. We do want a confirmation, though, so we'll include such a field in Listing 7.15.

[10]Listing 7.11 includes the **.gravatar_edit** class, which we'll put to work in Chapter 10.


0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-
SAMPLE APP
Home
Help
Log in
Example User
The Ruby on Rails Tutorial by Michael Hartl
About
Contact
News

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  controller: users
  action: show
  id: '1'
permitted: false
```

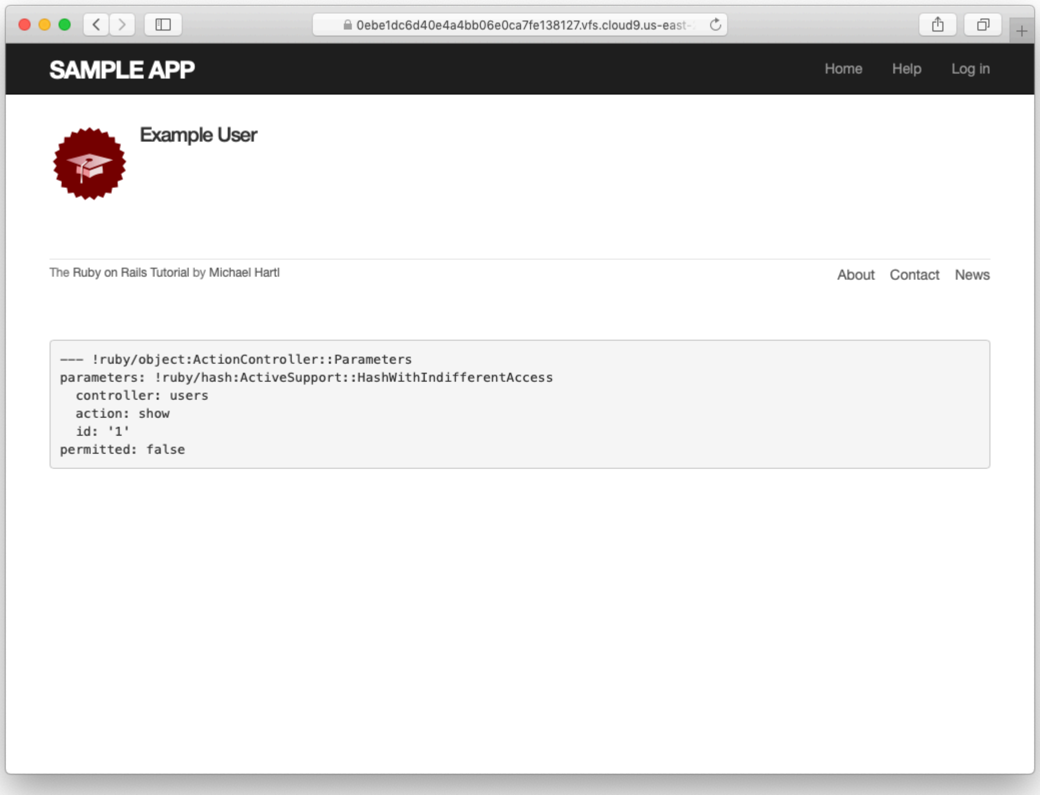
Figure 7.9: The user show page with a custom Gravatar.

Listing 7.11: SCSS for styling the user show page, including the sidebar.
app/assets/stylesheets/custom.scss

```
.
.
.
/* sidebar */

aside {
  section.user_info {
    margin-top: 20px;
  }
  section {
    padding: 10px 0;
    margin-top: 20px;
    &:first-child {
      border: 0;
      padding-top: 0;
    }
    span {
      display: block;
      margin-bottom: 3px;
      line-height: 1;
    }
    h1 {
      font-size: 1.4em;
      text-align: left;
      letter-spacing: -1px;
      margin-bottom: 3px;
      margin-top: 0px;
    }
  }
}

.gravatar {
  float: left;
  margin-right: 10px;
}

.gravatar_edit {
  margin-top: 15px;
}
```

Exercises

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


0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-
SAMPLE APP
Home
Help
Log in
Example User
The Ruby on Rails Tutorial by Michael Hartl
About
Contact
News

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  controller: users
  action: show
  id: '1'
permitted: false
```

Figure 7.10: The user show page with a sidebar and CSS.

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.

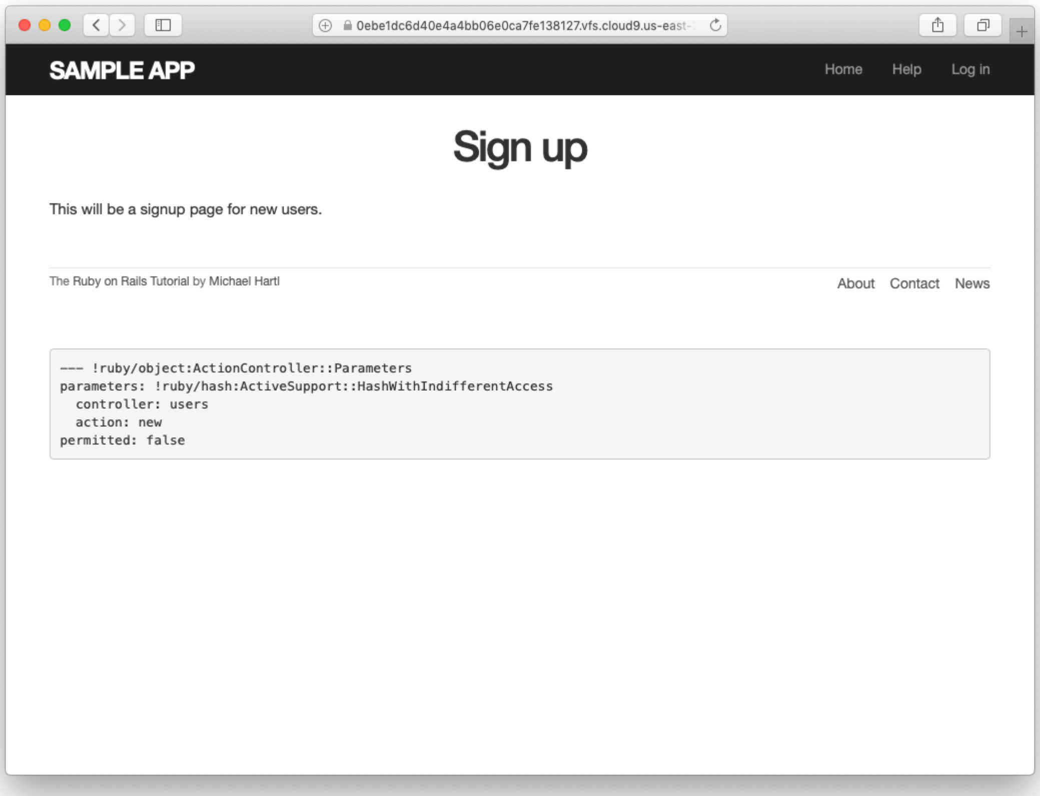
1. Associate a Gravatar with your primary email address if you haven't already. What is the MD5 hash associated with the image?
2. Verify that the code in Listing 7.12 allows the **`gravatar_for`** helper defined in Section 7.1.4 to take an optional **`size`** parameter, allowing code like **`gravatar_for user, size: 50`** in the view. (We'll put this improved helper to use in Section 10.3.1.)
3. The options hash used in the previous exercise is still commonly used, but as of Ruby 2.0 we can use *keyword arguments* instead. Confirm that the code in Listing 7.13 can be used in place of Listing 7.12. What are the diffs between the two?

Listing 7.12: Adding an options hash in the **`gravatar_for`** helper.
`app/helpers/users_helper.rb`

```ruby
module UsersHelper

  # Returns the Gravatar for the given user.
  def gravatar_for(user, options = { size: 80 })
    size         = options[:size]
    gravatar_id  = Digest::MD5::hexdigest(user.email.downcase)
    gravatar_url = "https://secure.gravatar.com/avatar/#{gravatar_id}?s=#{size}"
    image_tag(gravatar_url, alt: user.name, class: "gravatar")
  end
end
```

Listing 7.13: Using keyword arguments in the **`gravatar_for`** helper.
`app/helpers/users_helper.rb`

```ruby
module UsersHelper

  # Returns the Gravatar for the given user.
  def gravatar_for(user, size: 80)
    gravatar_id  = Digest::MD5::hexdigest(user.email.downcase)
    gravatar_url = "https://secure.gravatar.com/avatar/#{gravatar_id}?s=#{size}"
    image_tag(gravatar_url, alt: user.name, class: "gravatar")
  end
end
```



Sign up

This will be a signup page for new users.

The Ruby on Rails Tutorial by Michael Hartl

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  controller: users
  action: new
permitted: false
```

Figure 7.11: The current state of the signup page /signup.

7.2 Signup form

Now that we have a working (though not yet complete) user profile page, we're ready to make a signup form for our site. We saw in Figure 5.11 (shown again in Figure 7.11) that the signup page is currently blank: useless for signing up new users. The goal of this section is to start changing this sad state of affairs by producing the signup form mocked up in Figure 7.12.