```
    redirect_to root_url and return unless FILL_IN
  end
  .
  .
  .
end
```

11.4 Email in production

Now that we've got account activations working in development, in this section we'll configure our application so that it can actually send email in production. We'll first get set up with a free service to send email, and then configure and deploy our application.

To send email in production, we'll use SendGrid, which is available as an add-on at Heroku for verified accounts. (This requires adding credit card information to your Heroku account, but there is no charge when verifying an account.) For our purposes, the "starter" tier (which as of this writing is limited to 400 emails a day but costs nothing) is the best fit. We can add it to our app as follows:

```
$ heroku addons:create sendgrid:starter
```

(This might fail on systems with an older version of Heroku's command-line interface. In this case, either upgrade to the latest Heroku toolbelt or try the older syntax **`heroku addons:add sendgrid:starter`**.)

To configure our application to use SendGrid, we need to fill out the SMTP settings for our production environment. As shown in Listing 11.41, you will also have to define a **`host`** variable with the address of your production website.

Listing 11.41: Configuring Rails to use SendGrid in production.
`config/environments/production.rb`

```
Rails.application.configure do
  .
```

```
  .
  .
  config.action_mailer.raise_delivery_errors = true
  config.action_mailer.delivery_method = :smtp
  host = '<your heroku app>.herokuapp.com'
  config.action_mailer.default_url_options = { host: host }
  ActionMailer::Base.smtp_settings = {
    :address        => 'smtp.sendgrid.net',
    :port           => '587',
    :authentication => :plain,
    :user_name      => ENV['SENDGRID_USERNAME'],
    :password       => ENV['SENDGRID_PASSWORD'],
    :domain         => 'heroku.com',
    :enable_starttls_auto => true
  }
  .
  .
  .
end
```

The email configuration in Listing 11.41 includes the **user_name** and **password** of the SendGrid account, but note that they are accessed via the **ENV** environment variable instead of being hard-coded. This is a best practice for production applications, which for security reasons should never expose sensitive information such as raw passwords in source code. In the present case, these variables are configured automatically via the SendGrid add-on, but we'll see an example in Section 13.4.4 where we'll have to define them ourselves. In case you're curious, you can view the environment variables used in Listing 11.41 as follows:

```
$ heroku config:get SENDGRID_USERNAME
$ heroku config:get SENDGRID_PASSWORD
```

At this point, you should merge the topic branch into master:

```
$ rails test
$ git add -A
$ git commit -m "Add account activation"
$ git checkout master
$ git merge account-activation
```

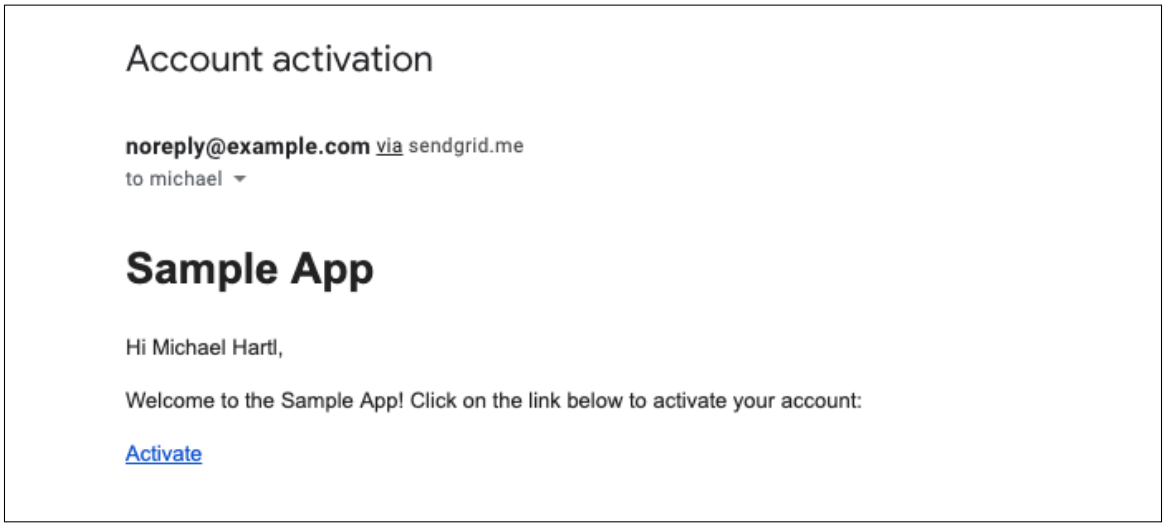

Figure 11.8: An account activation email sent in production.

Then push up to the remote repository and deploy to Heroku:

```
$ rails test
$ git push && git push heroku
$ heroku run rails db:migrate
```

Once the Heroku deploy has finished, try signing up for the sample application in production using an email address you control. You should get an activation email as implemented in Section 11.2, as shown in Figure 11.8. Clicking on the link should activate the account as promised, as shown in Figure 11.9.

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. Sign up for a new account in production. Did you get the email?

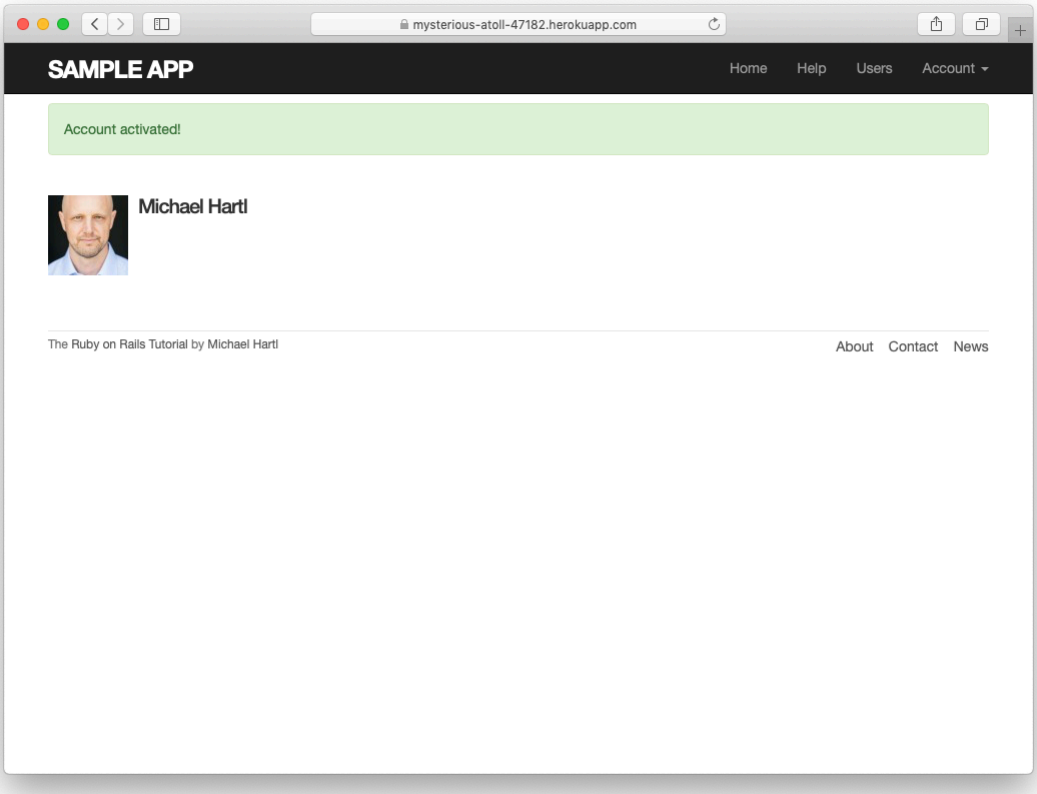

Figure 11.9: Successful account activation in production.

2. Click on the link in the activation email to confirm that it works. What is the corresponding entry in the server log? *Hint*: Run `heroku logs` at the command line.

11.5 Conclusion

With the added account activation, our sample application's sign up, log in, and log out machinery is nearly complete. The only significant feature left is allowing users to reset their passwords if they forget them. As we'll see in Chapter 12, password reset shares many features with account activation, which means that we'll be able to put the knowledge we've gained in this chapter to good use.

11.5.1 What we learned in this chapter

- Like sessions, account activations can be modeled as a resource despite not being Active Record objects.
- Rails can generate Action Mailer actions and views to send email.
- Action Mailer supports both plain-text and HTML mail.
- As with ordinary actions and views, instance variables defined in mailer actions are available in mailer views.
- Account activations use a generated token to create a unique URL for activating users.
- Account activations use a hashed activation digest to securely identify valid activation requests.
- Both mailer tests and integration tests are useful for verifying the behavior of the User mailer.
- We can send email in production using SendGrid.