12.1.1 Password resets controller

Our first step is to generate a controller for the Password Resets resource, in this case making both **`new`** and **`edit`** actions per the discussion above:

```
$ rails generate controller PasswordResets new edit --no-test-framework
```

Note that we've included a flag to skip generating tests. This is because we don't need the controller tests, preferring instead to build on the integration test from Section 11.3.3.

Because we'll need forms both for creating new password resets (Figure 12.2) and for updating them by changing the password in the User model (Figure 12.3), we need routes for **`new`**, **`create`**, **`edit`**, and **`update`**. We can arrange this with the **`resources`** line shown in Listing 12.1.

Listing 12.1: Adding a resource for password resets.
`config/routes.rb`

```
Rails.application.routes.draw do
  root   'static_pages#home'
  get    '/help',    to: 'static_pages#help'
  get    '/about',   to: 'static_pages#about'
  get    '/contact', to: 'static_pages#contact'
  get    '/signup',  to: 'users#new'
  get    '/login',   to: 'sessions#new'
  post   '/login',   to: 'sessions#create'
  delete '/logout',  to: 'sessions#destroy'
  resources :users
  resources :account_activations, only: [:edit]
  resources :password_resets,     only: [:new, :create, :edit, :update]
end
```

The code in Listing 12.1 arranges for the RESTful routes shown in Table 12.1. In particular, the first route in Table 12.1 gives a link to the "forgot password" form via

HTTP request	URL	Action	Named route
`GET`	/password_resets/new	**`new`**	**`new_password_reset_path`**
`POST`	/password_resets	**`create`**	**`password_resets_path`**
`GET`	/password_resets/<token>/edit	**`edit`**	**`edit_password_reset_url(token)`**
`PATCH`	/password_resets/<token>	**`update`**	**`password_reset_path(token)`**

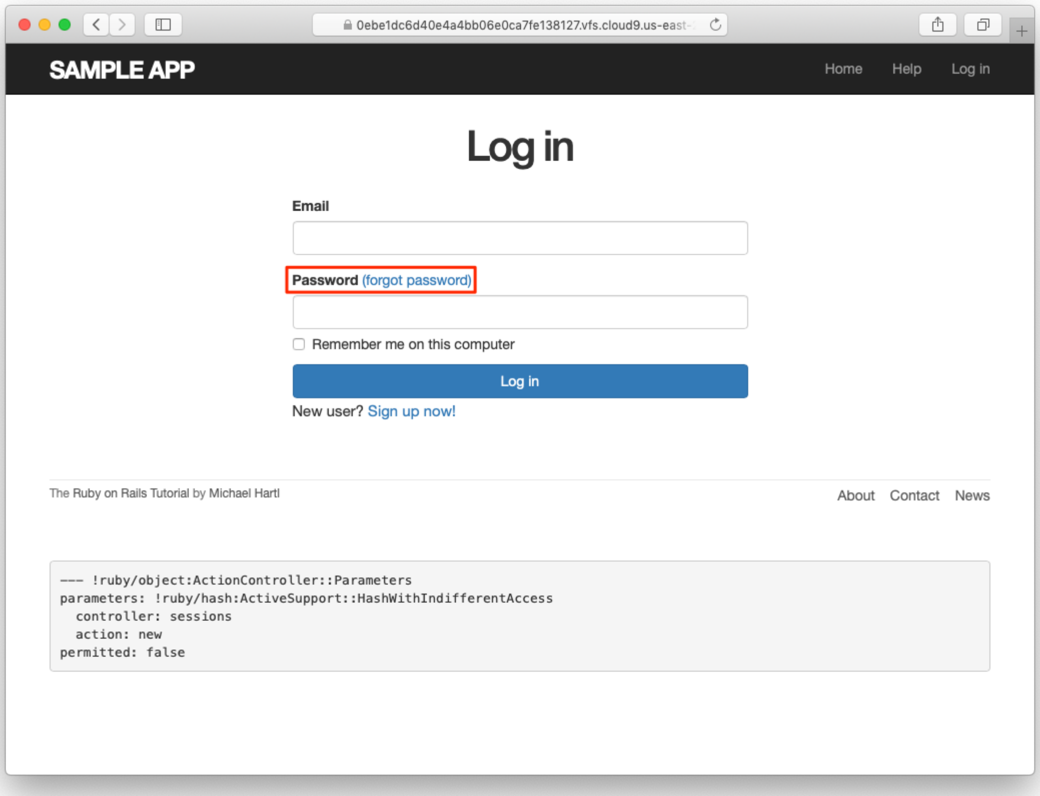
Table 12.1: RESTful routes provided by the Password Resets resource in Listing 12.1.

```
new_password_reset_path
```

as seen in Listing 12.2 and Figure 12.4.

Listing 12.2: Adding a link to password resets.
app/views/sessions/new.html.erb

```
<% provide(:title, "Log in") %>
<h1>Log in</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_with(url: login_path, scope: :session, local: true) do |f| %>

      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>

      <%= f.label :password %>
      <%= link_to "(forgot password)", new_password_reset_path %>
      <%= f.password_field :password, class: 'form-control' %>

      <%= f.label :remember_me, class: "checkbox inline" do %>
        <%= f.check_box :remember_me %>
        <span>Remember me on this computer</span>
      <% end %>

      <%= f.submit "Log in", class: "btn btn-primary" %>
    <% end %>

    <p>New user? <%= link_to "Sign up now!", signup_path %></p>
  </div>
</div>
```


0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-
SAMPLE APP
Home
Help
Log in
Log in
Email
Password (forgot password)
Remember me on this computer
Log in
New user? Sign up now!
The Ruby on Rails Tutorial by Michael Hartl
About
Contact
News

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

Figure 12.4: The login page with a "forgot password" link.

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. Verify that the test suite is still GREEN.
2. Why does Table 12.1 list the **`_url`** form of the **`edit`** named route instead of the **`_path`** form? *Hint*: The answer is the same as for the similar account activations exercise (Section 11.1.1).

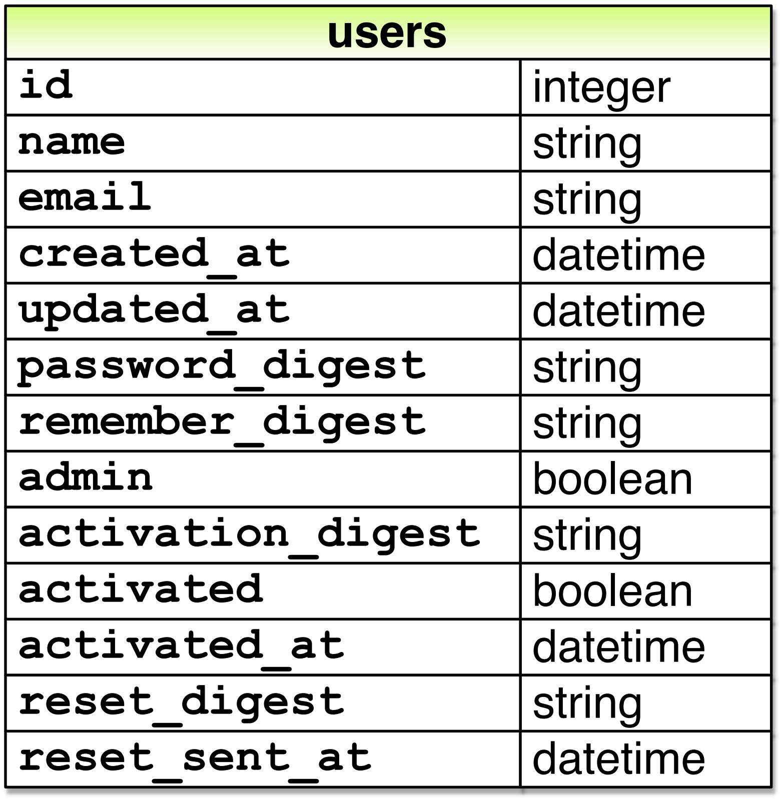
12.1.2 New password resets

To create new password resets, we first need to define the data model, which is similar to the one used for account activation (Figure 11.1). Following the pattern set by remember tokens (Chapter 9) and account activation tokens (Chapter 11), password resets will pair a virtual reset token for use in the reset email with a corresponding reset digest for retrieving the user. If we instead stored an unhashed token, an attacker with access to the database could send a reset request to the user's email address and then use the token and email to visit the corresponding password reset link, thereby gaining control of the account. Using a digest for password resets is thus essential. As an additional security precaution, we'll also plan to *expire* the reset link after a couple of hours, which requires recording the time when the reset gets sent. The resulting **`reset_digest`** and **`reset_sent_at`** attributes appear in Figure 12.5.

The migration to add the attributes from Figure 12.5 appears as follows:

```
$ rails generate migration add_reset_to_users reset_digest:string \
> reset_sent_at:datetime
```

(As in Section 11.1.2, the **`>`** on the second line is a "line continuation" character inserted automatically by the shell, and should not be typed literally.) We then migrate as usual:



users	
id	integer
name	string
email	string
created_at	datetime
updated_at	datetime
password_digest	string
remember_digest	string
admin	boolean
activation_digest	string
activated	boolean
activated_at	datetime
reset_digest	string
reset_sent_at	datetime

Figure 12.5: The User model with added password reset attributes.

```
$ rails db:migrate
```

To make the view for new password resets, we'll work in analogy with the previous form for making a new non–Active Record resource, namely, the login form (Listing 8.4) for creating a new session, shown again in Listing 12.3 for reference.

Listing 12.3: Reviewing the code for the login form.
app/views/sessions/new.html.erb

```
<% provide(:title, "Log in") %>
<h1>Log in</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_with(url: login_path, scope: :session, local: true) do |f| %>

      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>

      <%= f.label :password %>
      <%= link_to "(forgot password)", new_password_reset_path %>
      <%= f.password_field :password, class: 'form-control' %>

      <%= f.label :remember_me, class: "checkbox inline" do %>
        <%= f.check_box :remember_me %>
        <span>Remember me on this computer</span>
      <% end %>

      <%= f.submit "Log in", class: "btn btn-primary" %>
    <% end %>

    <p>New user? <%= link_to "Sign up now!", signup_path %></p>
  </div>
</div>
```

The new password resets form has a lot in common with Listing 12.3; the most important differences are the use of a different resource and URL in the call to **`form_with`** and the omission of the password attribute. The result appears in Listing 12.4 and Figure 12.6.

Listing 12.4: A new password reset view.
app/views/password_resets/new.html.erb

```
<% provide(:title, "Forgot password") %>
<h1>Forgot password</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_with(url: password_resets_path, scope: :password_reset,
                  local: true) do |f| %>
      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>

      <%= f.submit "Submit", class: "btn btn-primary" %>
    <% end %>
  </div>
</div>
```

Exercises

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

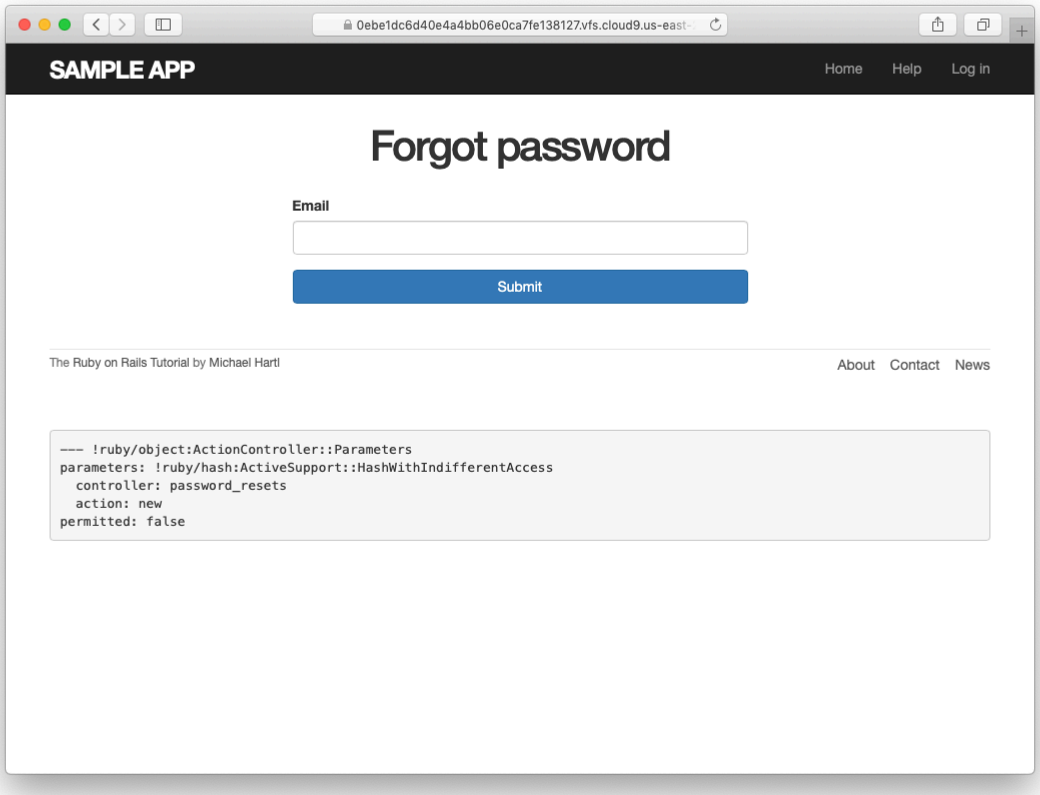
To see other people's answers and to record your own, subscribe to the Rails Tutorial course or to the Learn Enough All Access Bundle.

1. Why does the **`form_with`** in Listing 12.4 use **`:password_reset`** instead of **`@password_reset`**?

12.1.3 Password reset `create` action

Upon submitting the form in Figure 12.6, we need to find the user by email address and update its attributes with the password reset token and sent-at timestamp. We then redirect to the root URL with an informative flash message. As with login (Listing 8.11), in the case of an invalid submission we re-render the **`new`** page with a **`flash.now`** message.[2] The results appear in Listing 12.5.

[2]Security concerns about revealing the absence of the given email address are commonly misplaced. The reason is that this property is already present in every website that won't let you sign up with an email address that's already in use, which is practically all of them. Thus, indicating that a given email address isn't available for password resets gives potential attackers no additional information.


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

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

Figure 12.6: The “forgot password” form.

Listing 12.5: A **`create`** action for password resets.
`app/controllers/password_resets_controller.rb`

```ruby
class PasswordResetsController < ApplicationController

  def new
  end

  def create
    @user = User.find_by(email: params[:password_reset][:email].downcase)
    if @user
      @user.create_reset_digest
      @user.send_password_reset_email
      flash[:info] = "Email sent with password reset instructions"
      redirect_to root_url
    else
      flash.now[:danger] = "Email address not found"
      render 'new'
    end
  end

  def edit
  end
end
```

The code in the User model parallels the **`create_activation_digest`** method used in the **`before_create`** callback (Listing 11.3), as seen in Listing 12.6.

Listing 12.6: Adding password reset methods to the User model.
`app/models/user.rb`

```ruby
class User < ApplicationRecord
  attr_accessor :remember_token, :activation_token, :reset_token
  before_save   :downcase_email
  before_create :create_activation_digest
  .
  .
  .
  # Activates an account.
  def activate
    update_attribute(:activated,    true)
    update_attribute(:activated_at, Time.zone.now)
  end

  # Sends activation email.
  def send_activation_email
    UserMailer.account_activation(self).deliver_now
  end

  # Sets the password reset attributes.
  def create_reset_digest
    self.reset_token = User.new_token
    update_attribute(:reset_digest,  User.digest(reset_token))
    update_attribute(:reset_sent_at, Time.zone.now)
  end

  # Sends password reset email.
  def send_password_reset_email
    UserMailer.password_reset(self).deliver_now
  end

  private

    # Converts email to all lower-case.
    def downcase_email
      self.email = email.downcase
    end

    # Creates and assigns the activation token and digest.
    def create_activation_digest
      self.activation_token  = User.new_token
      self.activation_digest = User.digest(activation_token)
    end
end
```

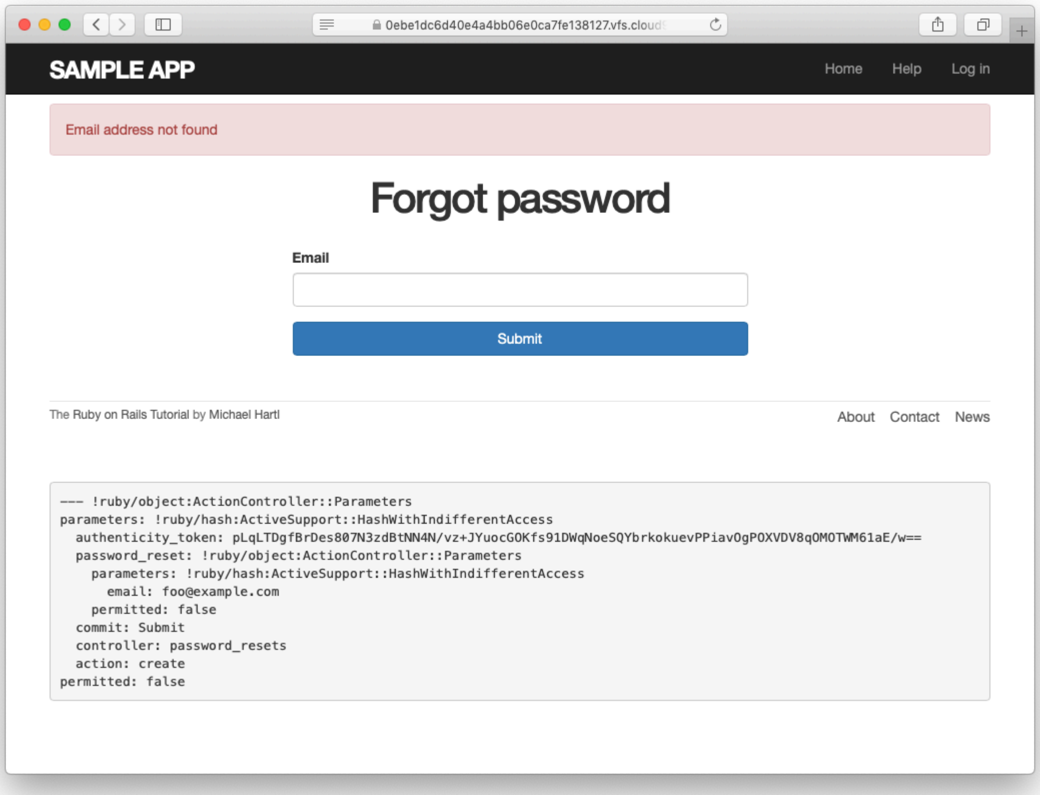
As shown in Figure 12.7, at this point the application's behavior for invalid email addresses is already working. To get the application working upon submission of a valid email address as well, we need to define a password reset mailer method.

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. Submit a valid email address to the form shown in Figure 12.6. What error message do you get?



Email address not found

Forgot password

Email

The Ruby on Rails Tutorial by Michael Hartl

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  authenticity_token: pLqLTDgfBrDes807N3zdBtNN4N/vz+JYuocGOKfs91DWqNoeSQYbrkokuevPPiavOgPOXVDV8qOMOTWM61aE/w==
  password_reset: !ruby/object:ActionController::Parameters
    parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
      email: foo@example.com
    permitted: false
  commit: Submit
  controller: password_resets
  action: create
permitted: false
```

Figure 12.7: The "forgot password" form for an invalid email address.

2. Confirm at the console that the user in the previous exercise has valid **`reset_digest`** and **`reset_sent_at`** attributes, despite the error. What are the attribute values?

12.2 Password reset emails

We left Section 12.1 with a nearly working **`create`** action in the Password Resets controller. The only thing missing is the method to deliver valid password reset emails.

If you followed Section 11.1, you already have a default **`password_reset`** method in **`app/mailers/user_mailer.rb`** as a result of the User mailer generation in Listing 11.6. If you skipped Chapter 11, you can just copy the code below (omitting the **`account_activation`** and related methods) and create the missing files as necessary.

12.2.1 Password reset mailer and templates

In Listing 12.6, we applied the design pattern implemented as a refactoring in Section 11.3.3 by putting the User mailer directly in the model (Listing 12.6):

```
UserMailer.password_reset(self).deliver_now
```

The password reset mailer method needed to get this working is nearly identical to the mailer for account activation developed in Section 11.2. We first create a **`password_reset`** method in the user mailer (Listing 12.7), and then define view templates for plain-text email (Listing 12.8) and HTML email (Listing 12.9).

Listing 12.7: Mailing the password reset link.
app/mailers/user_mailer.rb

```
class UserMailer < ApplicationMailer
```

```
  def account_activation(user)
    @user = user
    mail to: user.email, subject: "Account activation"
  end

  def password_reset(user)
    @user = user
    mail to: user.email, subject: "Password reset"
  end
end
```

Listing 12.8: The password reset plain-text email template.
app/views/user_mailer/password_reset.text.erb

```
To reset your password click the link below:

<%= edit_password_reset_url(@user.reset_token, email: @user.email) %>

This link will expire in two hours.

If you did not request your password to be reset, please ignore this email and
your password will stay as it is.
```

Listing 12.9: The password reset HTML email template.
app/views/user_mailer/password_reset.html.erb

```
<h1>Password reset</h1>

<p>To reset your password click the link below:</p>

<%= link_to "Reset password", edit_password_reset_url(@user.reset_token,
                                                      email: @user.email) %>

<p>This link will expire in two hours.</p>

<p>
If you did not request your password to be reset, please ignore this email and
your password will stay as it is.
</p>
```

As with account activation emails (Section 11.2), we can preview password reset emails using the Rails email previewer. The code is exactly analogous to Listing 11.18, as shown in Listing 12.10.

Listing 12.10: A working preview method for password reset.
test/mailers/previews/user_mailer_preview.rb

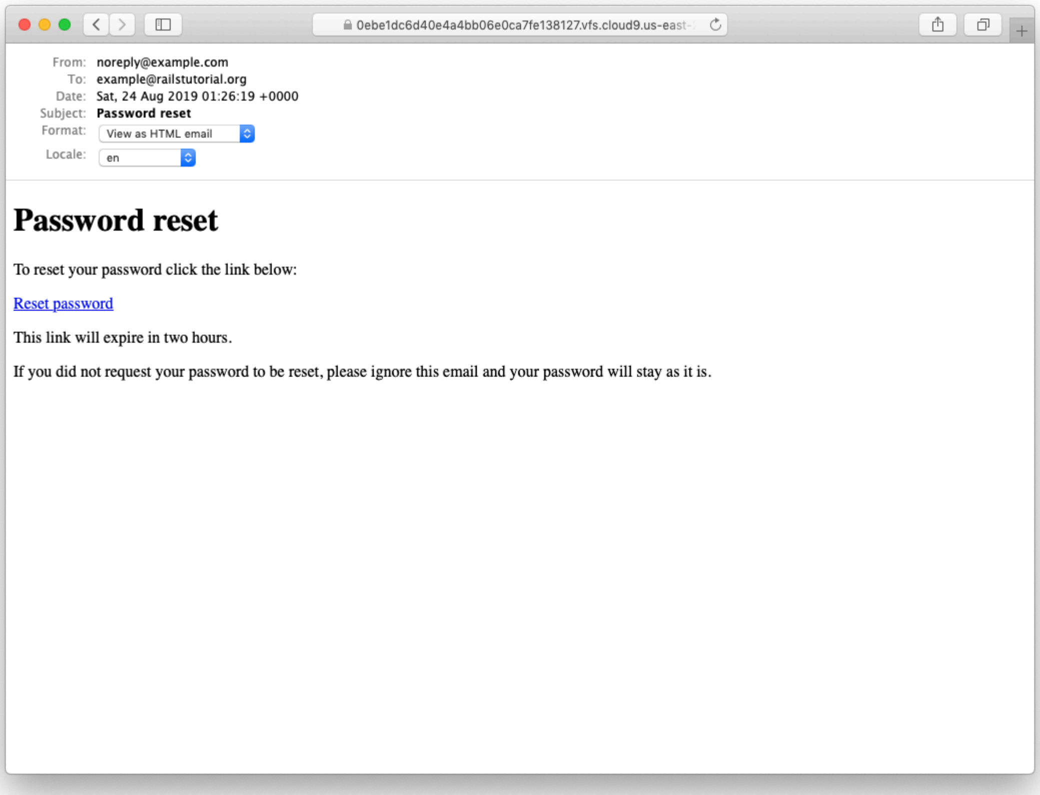
```
# Preview all emails at http://localhost:3000/rails/mailers/user_mailer
class UserMailerPreview < ActionMailer::Preview

  # Preview this email at
  # http://localhost:3000/rails/mailers/user_mailer/account_activation
  def account_activation
    user = User.first
    user.activation_token = User.new_token
    UserMailer.account_activation(user)
  end

  # Preview this email at
  # http://localhost:3000/rails/mailers/user_mailer/password_reset
  def password_reset
    user = User.first
    user.reset_token = User.new_token
    UserMailer.password_reset(user)
  end
end
```

With the code in Listing 12.10, the HTML and text email previews appear as in Figure 12.8 and Figure 12.9.

With the code in Listing 12.7, Listing 12.8, and Listing 12.9, submission of a valid email address appears as shown in Figure 12.10. The corresponding email appears in the server log and should look something like Listing 12.11.

Listing 12.11: A sample password reset email from the server log.

```
UserMailer#password_reset: processed outbound mail in 6.0ms
Delivered mail 5d609328d5d29_28872b106582ddf4886d8@ip-172-31-25-202.mail (2.8ms)
Date: Sat, 24 Aug 2019 01:30:16 +0000
From: noreply@example.com
To: michael@michaelhartl.com
Message-ID: <5d609328d5d29_28872b106582ddf4886d8@ip-172-31-25-202.mail>
Subject: Password reset
Mime-Version: 1.0
Content-Type: multipart/alternative;
 boundary="--==_mimepart_5d609328d5404_28872b106582ddf488531";
 charset=UTF-8
Content-Transfer-Encoding: 7bit
```



From: noreply@example.com
To: example@railstutorial.org
Date: Sat, 24 Aug 2019 01:26:19 +0000
Subject: **Password reset**
Format: View as HTML email
Locale: en

Password reset

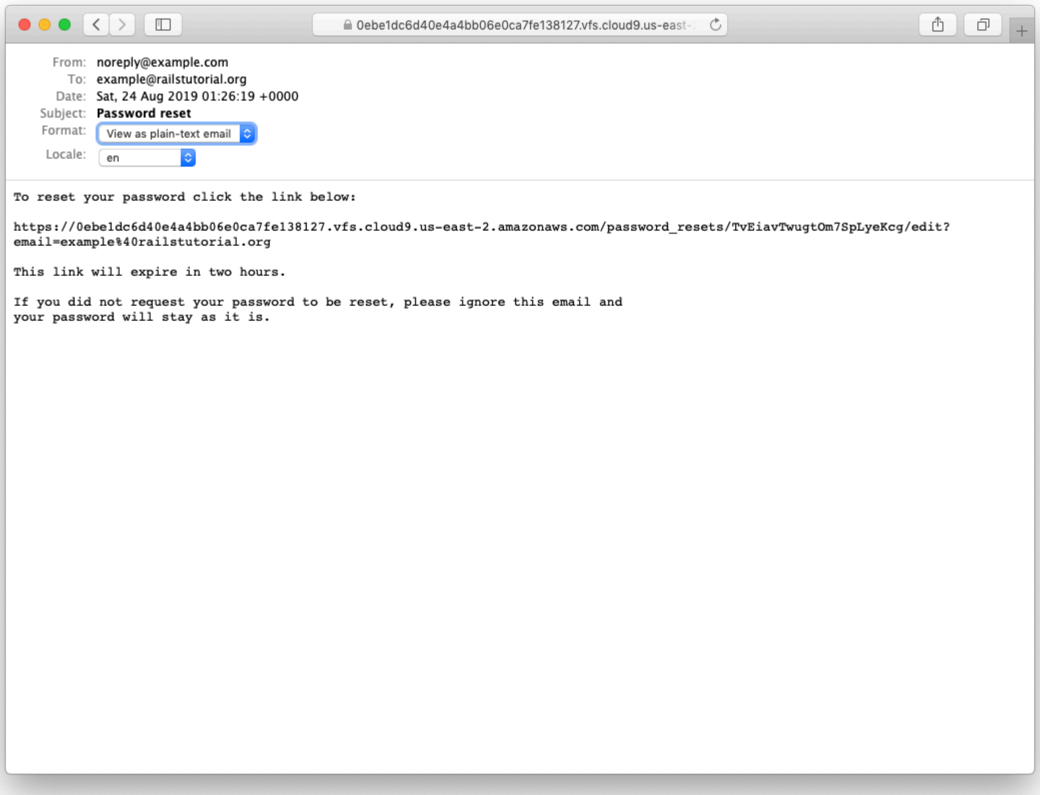
To reset your password click the link below:

Reset password

This link will expire in two hours.

If you did not request your password to be reset, please ignore this email and your password will stay as it is.

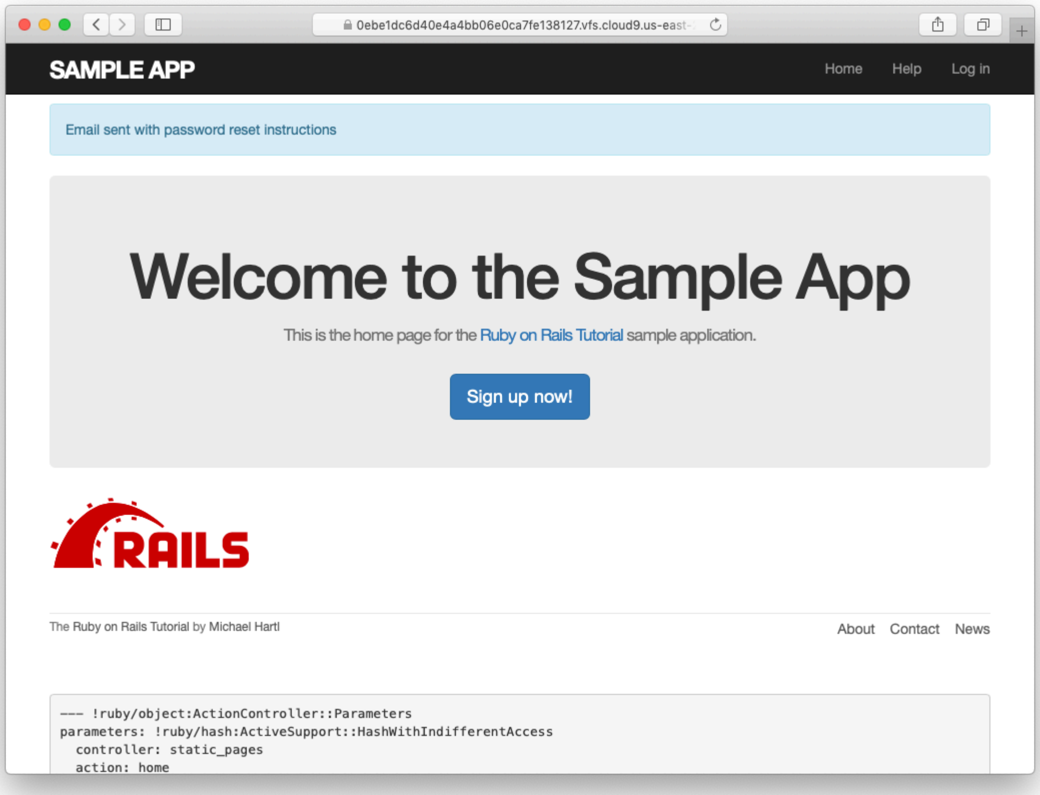
Figure 12.8: A preview of the HTML version of the password reset email.



From: noreply@example.com
To: example@railstutorial.org
Date: Sat, 24 Aug 2019 01:26:19 +0000
Subject: **Password reset**
Format: View as plain-text email
Locale: en

```
To reset your password click the link below:

https://0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-2.amazonaws.com/password_resets/TvEiavTwugtOm7SpLyeKcg/edit?
email=example%40railstutorial.org

This link will expire in two hours.

If you did not request your password to be reset, please ignore this email and
your password will stay as it is.
```

Figure 12.9: A preview of the text version of the password reset email.



Email sent with password reset instructions

Welcome to the Sample App

This is the home page for the Ruby on Rails Tutorial sample application.

The Ruby on Rails Tutorial by Michael Hartl

```
--- !ruby/object:ActionController::Parameters
parameters: !ruby/hash:ActiveSupport::HashWithIndifferentAccess
  controller: static_pages
  action: home
```

Figure 12.10: The result of submitting a valid email address.

```
----==_mimepart_5d609328d5404_28872b106582ddf488531
Content-Type: text/plain;
 charset=UTF-8
Content-Transfer-Encoding: 7bit

To reset your password click the link below:

https://0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-2.
amazonaws.com/password_resets/cT3mB4pwu7o-hrg6qEDfKg/
edit?email=michael%40michaelhartl.com

This link will expire in two hours.

If you did not request your password to be reset, please ignore this email and
your password will stay as it is.

----==_mimepart_5d609328d5404_28872b106582ddf488531
Content-Type: text/html;
 charset=UTF-8
Content-Transfer-Encoding: 7bit

<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <style>
      /* Email styles need to be inline */
    </style>
  </head>

  <body>
    <h1>Password reset</h1>

<p>To reset your password click the link below:</p>

<a href="https://0ebe1dc6d40e4a4bb06e0ca7fe138127.vfs.cloud9.us-east-2.
amazonaws.com/password_resets/cT3mB4pwu7o-hrg6qEDfKg/
edit?email=michael%40michaelhartl.com">Reset password</a>

<p>This link will expire in two hours.</p>

<p>
If you did not request your password to be reset, please ignore this email and
your password will stay as it is.
</p>
  </body>
</html>

----==_mimepart_5d609328d5404_28872b106582ddf488531--
```

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. Preview the email templates in your browser. What do the Date fields read for your previews?
2. Submit a valid email address to the new password reset form. What is the content of the generated email in the server log?
3. At the console, find the user object corresponding to the email address from the previous exercise and verify that it has valid **`reset_digest`** and **`reset_sent_at`** attributes.

12.2.2 Email tests

In analogy with the account activation mailer method test (Listing 11.20), we'll write a test of the password reset mailer method, as shown in Listing 12.12.

Listing 12.12: Adding a test of the password reset mailer method. GREEN
test/mailers/user_mailer_test.rb

```ruby
require 'test_helper'

class UserMailerTest < ActionMailer::TestCase

  test "account_activation" do
    user = users(:michael)
    user.activation_token = User.new_token
    mail = UserMailer.account_activation(user)
    assert_equal "Account activation", mail.subject
    assert_equal [user.email], mail.to
    assert_equal ["noreply@example.com"], mail.from
    assert_match user.name,               mail.body.encoded
    assert_match user.activation_token,   mail.body.encoded
    assert_match CGI.escape(user.email),  mail.body.encoded
  end

  test "password_reset" do
```

```
    user = users(:michael)
    user.reset_token = User.new_token
    mail = UserMailer.password_reset(user)
    assert_equal "Password reset", mail.subject
    assert_equal [user.email], mail.to
    assert_equal ["noreply@example.com"], mail.from
    assert_match user.reset_token,        mail.body.encoded
    assert_match CGI.escape(user.email),  mail.body.encoded
  end
end
```

At this point, the test suite should be GREEN:

Listing 12.13: 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. Run just the mailer tests. Are they GREEN?
2. Confirm that the test goes RED if you remove the second call to **`CGI.escape`** in Listing 12.12.

12.3 Resetting the password

Now that we have a correctly generated email as in Listing 12.11, we need to write the **`edit`** action in the Password Resets controller that actually resets the user's password. As in Section 11.3.3, we'll write a thorough integration test as well.