thebucknerlife / **authentication_with_bcrypt_in_rails_4.md**
Last active 2 days ago

Simple Authentication in Rail 4 Using Bcrypt

authentication_with_bcrypt_in_rails_4.md

Simple Authentication with Bcrypt

This tutorial is for adding authentication to a vanilla Ruby on Rails app using Bcrypt and has_secure_password.

The steps below are based on Ryan Bates's approach from Railscast #250 Authentication from Scratch (revised).

You can see the final source code here: repo. I began with a stock rails app using `rails new gif_vault`

Steps

1. Create a user model with a name, email and password_digest (all strings) by entering the following command into the command line: `rails generate model user name email password_digest` .

   Note: If you already have a user model or you're going to use a different model for authentication, that model must have an attribute names password_digest and some kind of attribute to identify the user (like an email or a username).

2. Run `rake db:migrate` in the command line to migrate the database.

3. Add these routes below to your routes.rb file. Notice I also deleted all the comments inside that file. Don't forget to leave the trailing `end` , though.

```
# config/routes.rb

GifVault::Application.routes.draw do

    # This route sends requests to our naked url to the *cool* action in the *gif* controller.
    root to: 'gif#cool'

    # I've created a gif controller so I have a page I can secure later.
    # This is optional (as is the root to: above).
    get '/cool' => 'gif#cool'
    get '/sweet' => 'gif#sweet'

    # These routes will be for signup. The first renders a form in the browse, the second will
    # receive the form and create a user in our database using the data given to us by the user.
    get '/signup' => 'users#new'
    post '/users' => 'users#create'

end
```

4. Create a users controller:

```
# app/controllers/users_controller.rb

class UsersController < ApplicationController

end
```

5. Add a **new** action (for rendering the signup form) and a **create** action (for receiving the form and creating a user with the form's parameters.):

```
# app/controllers/users_controller.rb

class UsersController < ApplicationController

    def new
    end

    def create
    end

end
```

6. Now create the view file where we put the signup form.

```
<!-- app/views/users/new.html.erb -->

<h1>Signup!</h1>

<%= form_for :user, url: '/users' do |f| %>

  Name: <%= f.text_field :name %>
  Email: <%= f.text_field :email %>
  Password: <%= f.password_field :password %>
  Password Confirmation: <%= f.password_field :password_confirmation %>
  <%= f.submit "Submit" %>

<% end %>
```

A note on Rail's conventions: This view file is for the **new** action of the **users controller**. As a result, we save the file here: `/app/views/users/new.html.erb` . The file is called **new**.html.erb and it is saved inside the views folder, in a folder we created called **users**.

That's the convention: view files are inside a folder with the same name as the controller and are named for the action they render.

7. Add logic to **create** action and add the private `user_params` method to sanitize the input from the form (this is a new Rails 4 thing and it's required). You might need to adjust the parameters inside the `.permit()` method based on how you setup your User model.

```
class UsersController < ApplicationController

  def new
  end

  def create
    user = User.new(user_params)
    if user.save
      session[:user_id] = user.id
      redirect_to '/'
    else
      redirect_to '/signup'
    end
  end

private

  def user_params
    params.require(:user).permit(:name, :email, :password, :password_confirmation)
  end
end
```

8. Go to your Gemfile and uncomment the 'bcrypt' gem. We need bcrypt to securely store passwords in our database.

```
source 'https://rubygems.org'

# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '4.0.4'

# Use sqlite3 as the database for Active Record
gem 'sqlite3'

...

# Use ActiveModel has_secure_password
gem 'bcrypt', '~> 3.1.7'

...
```

9. Go to the User model file and add `has_secure_password` . This is the line of code that gives our User model authentication methods via bcrypt.

```
# app/models/user.rb

class User < ActiveRecord::Base

  has_secure_password

end
```

10. Run `bundle install` from the terminal then restart your rails server.

    Note: Windows users might have issues with bcrypt. If so, copy the error into Google and look for answers on Stack Overflow. There is documentation online for how to fix Windows so the bcrypt works.

11. Create a sessions controller. This is where we create (aka login) and destroy (aka logout) sessions.

```
# app/controllers/sessions_controller.rb

class SessionsController < ApplicationController

  def new
  end

  def create
  end

  def destroy
  end

end
```

12. Create a form for user's to login with.

```
<!-- app/views/sessions/new.html.erb -->

<h1>Login</h1>

<%= form_tag '/login' do %>

  Email: <%= text_field_tag :email %>
  Password: <%= password_field_tag :password %>
  <%= submit_tag "Submit" %>

<% end %>
```

13. Update your routes file to include new routes for the sessions controller.

```
GifVault::Application.routes.draw do

  root to: 'gif#cool'

  # these routes are for showing users a login form, logging them in, and logging them out.
  get '/login' => 'sessions#new'
  post '/login' => 'sessions#create'
  get '/logout' => 'sessions#destroy'

  get '/signup' => 'users#new'
  post '/users' => 'users#create'

end
```

14. Update the sessions_controller with the logic to log users in and out.

```
  # app/controllers/sessions_controller.rb

  def create
    user = User.find_by_email(params[:email])
    # If the user exists AND the password entered is correct.
    if user && user.authenticate(params[:password])
      # Save the user id inside the browser cookie. This is how we keep the user
      # logged in when they navigate around our website.
      session[:user_id] = user.id
      redirect_to '/'
    else
    # If user's login doesn't work, send them back to the login form.
      redirect_to '/login'
    end
  end

  def destroy
    session[:user_id] = nil
    redirect_to '/login'
  end
```

15. Update the application controller with new methods to look up the user, if they're logged in, and save their user object to a variable called @current_user. The `helper_method` line below current_user allows us to use `@current_user` in our view files. Authorize is for sending someone to the login page if they aren't logged in - this is how we keep certain pages our site secure... user's have to login before seeing them.

```
# app/controllers/application_controller.rb

class ApplicationController < ActionController::Base
  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

  def current_user
    @current_user ||= User.find(session[:user_id]) if session[:user_id]
  end
  helper_method :current_user

  def authorize
    redirect_to '/login' unless current_user
  end

end
```

16. Add a `before_filter` to any controller that you want to secure. This will force user's to login before they can see the actions in this controller. I've created a gif controller below which I'm going to secure. The routes for this controller were added to the routes.rb in the beginning of this tutorial.

```
# app/controllers/gif_controller.rb

class GifController < ApplicationController

  before_filter :authorize

  def cool
  end

  def free
  end

end
```

17. You can update your application layout file to show the user's name if they're logged in and some contextual links.

```
<!-- app/views/layout/application.html.erb -->

<!DOCTYPE html>
<html>
<head>
  <title>GifVault</title>
  <%= stylesheet_link_tag    "application", media: "all", "data-turbolinks-track" => true %>
  <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  <%= csrf_meta_tags %>
</head>
<body>

# added these lines.
<% if current_user %>
  Signed in as <%= current_user.name %> | <%= link_to "Logout", '/logout' %>
<% else %>
  <%= link_to 'Login', '/login' %> | <%= link_to 'Signup', '/signup' %>
<% end %>

<%= yield %>

</body>
</html>
```

Things Missing

- Adding flash messages would be simple and provide feedback to the user if things go wrong.

All done! Feel free to fork and update this. Reach me at @thebucknerlife on Twitter.

dickeyxxx commented on Apr 8, 2014

instead of

```
current_user ||= User.find(session[:user_id]) if session[:user_id]
```

you want to do

```
@current_user ||= User.find(session[:user_id]) if session[:user_id]
```

if it is in a local variable it won't be saved

thebucknerlife commented on Apr 8, 2014 gist owner

Done. Thanks **@dickeyxxx**

gxespino commented on Oct 26, 2014

Thanks a lot for this!

BrunoCanongia commented on Nov 3, 2014

So simple! Thank you.

dimarymar commented on Nov 16, 2014

Thanks man!

Johnsalzarulo commented on Dec 9, 2014

This guide is gold. I spent hours playing with devise for authentication. This is dead simple. Thanks!

jd-gray commented on Jan 8, 2015

Thank you for the help! I was wondering how to accomplish that before_filter :authorize

lilijreey commented on Jan 18, 2015

Thanks for your help! clear talk.

oakfield commented on Jan 19, 2015

Thanks, I found this very helpful.

myerspa commented on Feb 5, 2015

Little bit late since this has been around since last April...but this is awesome! Straight forward implementation of a concept that pretty much every app will need.

Nice!

efecarranza commented on Feb 11, 2015

awesome job!
what i never understood was why we needed password_digest, thanks!

riliwanrabo commented on Feb 27, 2015

Awesome Guide!!!

EricFries commented on Mar 14, 2015

Great guide. Thanks!

AugustoOdy commented on Mar 31, 2015

Thanks for the help, awesome job!

ashok4 commented on Apr 8, 2015

Great guide. but please mention where to put validation&error messages
can you please help me...!

Epigene commented on Apr 24, 2015

Excellent writeup, though one could wish the author had used rails route helpers instead of path literals.

prwhitehead commented on May 4, 2015

Hello,

Firstly thank you very much for this, its been really useful in getting tro grips with Rails Auth.

If I may, I would like to make one suggestion. In your `UserController` the `create` method redirects back to the `new` method; meaning all validation information is lost. It took me a while to figure out what was going on - so for others that are new to the paradigms laid out by Rails, the following will rerender the form and display your error messages for the `signup` form:

```
def create
  @user = User.new(user_params)

  if @user.save
    session[:user_id] = @user.id
    redirect_to :home, notice: 'Account created successfully'
  else
    flash[:error] = 'An error occured!'
    render 'new'
  end
end
```

lamh85 commented on May 19, 2015

Hello all

First, thank you very much for this guide. It is very invaluable for people who want to learn about authentication.

Just a caution about #14. Perhaps I coded something differently, but the step does not work because my params hash is different:

```
{"utf8"=>"✓", "authenticity_token"=>"i4mIRjS+H3/V2y5LZ8/H+lSymYOotuDW16BDWKemxDMHAyEMrC2VklQoyyQZx0868AQvJnsr+ZsMGxUjiLuq5
"login"=>{"email"=>"mike@smith.com", "password"=>"[FILTERED]"}, "commit"=>"Log in"}
```

In other words, my hash does not have a key called "email", as mentioned in step 14. Calling it would require params[:login][:email] instead of just params[:email]. The same is for calling the password.

Just a heads up in case #14 doesn't work for you.

Thanks again for this guide!

kevlarr commented on Jul 2, 2015

This guide was just phenomenal.

iMikie commented on Jul 9, 2015

Boy did this ever fill an incredible need. I wish everyone would write as clearly as you.
Thank you so much.

ijunaid8088 commented on Aug 7, 2015

where is that method Authenticate?

Corstiaan84 commented on Aug 11, 2015

Great primer on basic Rails auth. Thanks!

iscott commented on Aug 26, 2015

This is awesome! Thanks for posting.

sabrams86 commented on Aug 28, 2015

Awesome! I'm just getting into rails and this is perfect for introducing me to everything needed to use has_secure_password

edenasevic1 commented on Sep 9, 2015

First of all great guide, I really enjoyed reading it. But, I think there is one mistake.

We generated User model with 3 attributes, name, email and password_digest, but in user_params we take all 4 values from form fields and try to save it in database, which causes error, because we have one parameter more than we need.

thomascarterx commented on Sep 14, 2015

This really is something everyone really needs. I too wrestled around with devise and cannot believe how easy and fast it was to do it like this. Great tutorial. Thanks so much for putting this on here. It really helped a lot. You should consider doing more tutorials, if you haven't already, and you have the time. Your communication style is very efficient.

lstanard commented on Oct 14, 2015

Thank you! This is an incredibly handy reference!

sadanandkenganal commented on Oct 18, 2015

Its very simple and nice. Please explain validation also for signp and login form.

andim27 commented on Nov 9, 2015

Super solution for windows users:
Note: Windows users might have issues with bcrypt. If so, copy the error into Google and look for answers on Stack Overflow.
This gem doesn't work on win 7 64 bit/ruby 2.2 rails 4.2/
!! Please fix problem for windows users !!

lskd commented on Nov 9, 2015

Awesome walk thru, thanks for sharing.

cloudsben commented on Jan 16

Awesome, really simple authentication

juancpgo commented on Feb 4

Great! Thanks. What about password resetting (when user looses it)? Does anybody suggest an approach?

AndrewHendrie commented on Feb 9

Typo in the first code block:

These routes will be for signup. The first renders a form in the browse, the second will

receive the form and create a user in our database using the data given to us by the user.

`browse` should be `browser`

oklaiss commented on Mar 20

Very helpful, thanks for this.

shahfaizanali commented on Mar 24

How to disable password requirement. I have Employee < User. I don't need authentication for employees.

cdekker commented on Mar 25

@shahfaizanali best practice is to make a bare virtual class 'VirtualUser' and have 2 classes inherit from that:

- Employee < VirtualUser
- User < VirtualUser

Then you can implement the authentication stuff from this guide only in the User class and not be bothered by it in the Employee class.

However, all employees will have the unused Password_digest column, which is somewhat of a design flaw. Maybe rethink your model structure? :)

ManojDatt commented on Mar 26

These steps has really worked for me..Thanks for help..

khalidh12 commented on Mar 31

wow really superb....
thnx.... keep it up bro...

it is very very useful to beginners...

andrewdotcudzilo commented on Apr 27

Excellent bare-metals guide

SeptivianaSavitri commented on Jun 26

wow, its amazingly simple,thanks for the tutorial.
But I wonder if we can use this method to combine with omniauth?
Thanks :)

FottyM commented on Jul 2

```
awesome
```

oquidave commented on Jul 29

Worked fine for me. Thanks. Although I would to know what goes on behind the scenes.

Yuvisiva12 commented on Aug 5

Good Tutorial. Thanks for this..

silverdr commented on Aug 10

One of the unfortunately scarce pieces of good documentation for Rails. TNX.

jamesaduke commented on Aug 10

really great tutorial

martinbeentjes commented on Aug 23

Great straight-to-the-point tutorial! Thanks for this!

cloudsben commented on Sep 6

Great Tutorial !

aaossa commented on Sep 19

If you're on Windows, using Ruby 2.3 and bcrypt gives you problems, this comment was the solution for me

levanlinh1995 commented on Oct 2

tks, nice