**Rails Authorisation Lesson**

1. Rails new twitter --databae=postgresql
2. Rails db:create
3. Rails s
4. Rails g scaffold tweet title content user:references

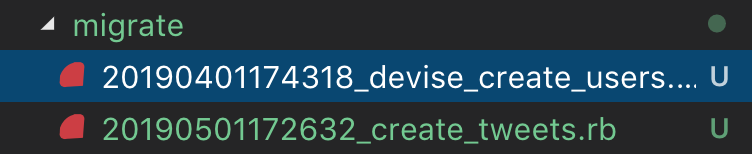
* Tweets owned by specific user
* Can’t db:migrate until created user

1. Authenticate users with devise 🡪 bundle add devise
2. Rails g devise:install
3. Paste into development.rb
4. config.action\_mailer.default\_url\_options = { host: 'localhost', port: 3000 }
5. Rails g controller home index
6. To routes.rb
7. root "home#index"
8. To application.html.erb
9. <p *class*="notice"><%= notice %></p>
10. <p *class*="alert"><%= alert %></p>

Above <%= yield %>

1. Rails g devise:views
2. Create user model using devise for authentication 🡪 Rails g devise user
3. Migration fails because create user needs to happen before create tweets

Rename the files so that create users is earlier than create tweets



1. Rails db:migrate
2. Rails s
3. <http://localhost:3000/users/sign_up> - sign up an account
4. While logged in, add to application.html.erb under notice and error
5. <p>You are logged in as<%= current\_user.inspect %></p>
6. Change to current\_user.email
7. Once logged in, when creating a new user, we don’t’ need to pass along the user field

* Go to tweets/new.html.erb, remove the
* <div *class*="field">
* <%= form.label :user\_id %>
* <%= form.text\_field :user\_id %>
* </div>

1. Has\_many :tweets added to User.rb
2. Add to def create in tweets\_controller.rb
3. @tweet.user = current\_user
4. For security, remove :user from def tweet\_params in tweets\_controller.rb

* We don’t’ want to accept user id as parameter when creating new tweet

1. In tweets/index.html.erb change tweet.user to tweet.user.email
2. In tweets/show.html.erb change tweet.user to tweet.user.email
3. In application.html.erb in body,
4. <p>You are logged in as <%= current\_user.email %> (<%= link\_to "Sign out", destroy\_user\_session\_path, method: :delete %>)</p>
5. To fix the error when logging out, modify application.html.erb in step 33 to

<% if user\_signed\_in? %>

<p>You are logged in as <%= current\_user.email %> (<%= link\_to "Sign out", destroy\_user\_session\_path, method: :delete %>)</p>

<% end %>

1. To hide all tweets before logging in

In tweets controller, add to top

before\_action :authenticate\_user!

Before you can do any CRUD function in tweets controller, authenticate user

1. Add sign up and log in links to home page home/index.html.erb
2. <p><%= link\_to "Sign up", new\_user\_registration\_path %></p>
3. <p><%= link\_to "Log in", new\_user\_session\_path %></p>
4. To prevent editing or delete someone else’s tweets that are not yours

Use rolify gem to give users one or more roles and decide what they can or can not do based on the roles

Bundle add rolify

1. Rails g rolify Role User

Specifies two things, what you want to call the model and the second parameter is what is the name of the user model

1. Rails db:migrate
2. Rails s
3. Specify which model uses the role system. Change tweet model to the below
4. class Tweet < ApplicationRecord
5. resourcify
6. belongs\_to :user
7. end

This means Turn this model into a resource to apply roles to

1. To prevent editing or delete someone else’s tweets that are not yours  
   edit, destroy, update functions in tweets controller affected
2. To prevent editing, in tweet controller, change
3. def edit
4. # if curretn user is NOT owner of tweet
5. if current\_user != @tweet.user
6. flash[:alert] = "You cannot edit that tweet!"
7. redirect\_to(request.referrer)
8. end
9. end

if not the owner of tweet, flash an alert and redirect back to wherever they were before

1. Instead of doing 46 for destroy, update add this to tweet controller under private

def check\_owner

if current\_user != @tweet.user

flash[:alert] = "You cannot edit that tweet!"

redirect\_to(request.referrer)

end

end

1. At top of tweets controller , add
2. before\_action :check\_owner, only: [:edit, :destroy, :update]

check\_owner runs before it even starts an action

1. To hide the edit and destroy links when it is not the owner of the tweet

Tweets index.html.erb modify as

<% if tweet.user == current\_user %>

<td><%= link\_to 'Edit', edit\_tweet\_path(tweet) %></td>

<td><%= link\_to 'Destroy', tweet, method: :delete, data: { confirm: 'Are you sure?' } %></td>

<% end %>

1. To hide the edit button in show when it is not the owner of the tweet

Tweets show.html.erb modify as

<% if @tweet.user == current\_user %>

<%= link\_to 'Edit', edit\_tweet\_path(@tweet) %> |

<% end %>

**Second half**

Creating Moderator and Admin roles

Moderator can’t change, only delete

Admin can do everything

Any other models that use rolify roles with, need to add resourcify in the model (i.e. tweet model). It tells rolify to use roles in connection with tweets model.

1. Rails c
2. user.all
3. user = User.find(2)
4. user.add\_role :admin
5. user.roles displays the roles a person has
6. user.has\_role? :admin (queries whether the user has a particular role)
7. Change check\_owner in tweets controller to make sure that if it is not the owner of the tweet AND they are not admin, they are not allowed to do edit, update, destroy functions

def check\_owner

if current\_user != @tweet.user && !current\_user.has\_role?(:admin)

flash[:alert] = "You cannot edit that tweet!"

redirect\_to(request.referrer)

end

end

1. If current user is owner or admin, then you can show edit, destroy buttons in tweets index.html.erb

<% if tweet.user == current\_user || current\_user.has\_role?(:admin) %>

<td><%= link\_to 'Edit', edit\_tweet\_path(tweet) %></td>

<td><%= link\_to 'Destroy', tweet, method: :delete, data: { confirm: 'Are you sure?' } %></td>

<% end %>

1. 7 & 8 steps result in code that is not very DRY and error prone because it is doing the same thing in two different places in the whole project.

Abstract it separately and return the logic as true or false

Add to tweet.rb

Model does nto have access to current\_user (devise). So need to pass in user. This way the method is not dependent on devise specifically. This below code will work irrespective of what authorisation system you use (loose couple and generic)

def can\_change?(user)

return user == *self*.user || user.has\_role?(:admin)

end

if the user is the owner of tweet or is an admin, then they can edit, update, destroy

1. Back to tweets controller, in def check\_owner

Replace

if current\_user != @tweet.user && !current\_user.has\_role?(:admin)

with

if !@tweet.can\_change?(current\_user)

1. In tweets index.html.erb

Replace

<% if tweet.user == current\_user || current\_user.has\_role?(:admin) %>

With

<% if tweet.can\_change?(current\_user) %>

1. Create a new account called moderator

* Delete but not edit

1. Rails c
2. User.all (turn the latest one into a moderator)
3. user = User.find(3)
4. user.add\_role :moderator
5. user.roles (to check)
6. update tweet.rb model with the new helper methods

def can\_edit?(user)

return user == *self*.user || user.has\_role?(:admin)

end

def can\_destroy?(user)

return user == *self*.user || user.has\_role?(:admin) || user.has\_role?(:moderator)

end

1. in tweets controller, change def check\_owner to
2. def check\_edit
3. if !@tweet.can\_edit?(current\_user)
4. # if current\_user != @tweet.user && !current\_user.has\_role?(:admin)
5. flash[:alert] = "You cannot edit that tweet!"
6. redirect\_to(request.referrer)
7. end
8. end
9. def check\_destroy
10. if !@tweet.can\_destroy?(current\_user)
11. # if current\_user != @tweet.user && !current\_user.has\_role?(:admin)
12. flash[:alert] = "You cannot delete that tweet!"
13. redirect\_to(request.referrer)
14. end
15. end
16. In tweets controller, change before\_action :check\_owner at the top to
17. before\_action :check\_edit, only: [:edit, :update]
18. before\_action :check\_destroy, only: [:edit, :update]
19. In tweets index.html.erb, change

From

<% if tweet.can\_change?(current\_user) %>

<td><%= link\_to 'Edit', edit\_tweet\_path(tweet) %></td>

<td><%= link\_to 'Destroy', tweet, method: :delete, data: { confirm: 'Are you sure?' } %></td>

<% end %>

to

<% if tweet.can\_edit?(current\_user) %>

<td><%= link\_to 'Edit', edit\_tweet\_path(tweet) %></td>

<% end %>

<% if tweet.can\_destroy?(current\_user) %>

<td><%= link\_to 'Destroy', tweet, method: :delete, data: { confirm: 'Are you sure?' } %></td>

<% end %>

1. Change the edit link in tweets show.html.erb to
2. <% if @tweet.can\_edit?(current\_user) %>
3. <%= link\_to 'Edit', edit\_tweet\_path(@tweet) %> |
4. <% end %>

So edit link is only visible to ppl who CAN EDIT (moderator, admin, owner)

Edit link is not visible to ppl who can’t edit (ppl who didn’t own the tweet)