Ruby on Rails: Authorization

So far we've seen how to build an authentication system that lets users sign up, log in, and log out.

In addition to authentication, many web apps have a way to give specific users permission to access certain parts of the site. For example, a blog would give only its authors permission to access the editing and publishing parts of the site. Permissions are defined with an authorization system.

Let's create an authorization system for a recipe website built with Rails.

**1.** We've provided a Rails app to get you started. Let's take a look - first install the gems in the **Gemfile**.

**2.** Next start a Rails development server.

**3.** Then preview the app by visiting[http://localhost:8000](http://localhost:8000/).

**4.** Click on "View Recipes" to see recipes inside a cuisine.

How can we add an authorization system to this app? Click Next to learn more.

Using the [request/response cycle](https://www.codecademy.com/articles/request-response-cycle-dynamic) as a guide, here's how authorization fits in:

1. The browser makes a request for a URL
2. The request hits the Rails router
3. Before the router sends the request on to the controller action, the app determines whether the user has access permission by looking at the user's *role*.

What is a role? A role is a way to manage what parts of a site a user has access to. A user's role is specified in the database.

**5.** Let's begin building an authorization system by adding a role column to the users table.

Open the migration file for the users table in **db/migrate/**, and add the following column:

* a string column called role

**6.** Run the migration to update the database with the users table.

Great! In the users table, we now have a column named role that we can use to assign different roles to users, such as "editor" or "admin".

**7.** Let's add a method to the User model that will help us use the role column in our application. Within the class User, beneath the has\_secure\_passwordmethod, type

def editor?

self.role == 'editor'

end

We can use this method to determine if a user has the role of editor.

**8.** Open **db/seeds.rb**. We've added a user named Mateo with the role 'editor'. Seed the database with this data

$ rake db:seed

**9.** We should now be able to use the editor? method to check whether a user has an editor role. First, start the Rails console by running

$ rails console

Then check whether Mateo is an editor

> mateo = User.find\_by(email: 'mateo@email.com')

> mateo.editor?

We created a method named editor? that checks whether a user's role is "editor", and returns trueor false. The method uses self to refer to the current instance of a User object.

Now that we can determine whether a user has an editor role on the site, let's add a few methods to the Application controller (**app/controllers/application\_controller.rb**) to make sure that users with the editor role can access specific parts of the site.

**10.**

In the Application controller (**app/controllers/application\_controller.rb**), below require\_user, add another method named require\_editor

def require\_editor

redirect\_to '/' unless current\_user.editor?

end

**11.** Next, in the Recipes controller, use another before action that usesrequire\_editor to permit only users with an editor role to access the show andedit actions

before\_action :require\_editor, only: [:show, :edit]

**12.** Then in **app/views/recipes/show.html**, use the editor? method to display an edit link only if a user is an editor

<% if current\_user && current\_user.editor? %>

<p class="recipe-edit">

<%= link\_to "Edit Recipe", edit\_recipe\_path(@recipe.id) %>

</p>

<% end %>

**13.** Try it out - first log in to the app as a user without a role. Looking at **db/seeds.rb**, Julian doesn't have an editor role, so use his email julian@email.com and password Julian1 to log in. Then visit<http://localhost:8000/recipes/1>. You shouldn't see the Edit link on this page.

**14.** Log out of the app, and then log back in as an editor. Looking at **db/seeds.rb**, Mateo has an editor role, so use his email mateo@email.com and passwordMateo1 to log in to the app. Then visit<http://localhost:8000/recipes/1>. You should see the Edit link.

Great work! The role-based authorization system is working. Users with an editor role have permissions to see the edit page, while users without that role do not.

**15.** In the User model, add a method named admin? that determines whether a user has an admin role on the site.

**16.** Open **db/seeds.rb**. We've added a user named Freida with the role admin.

We should now be able to use theadmin? method to check whether a user has an editor role. Enter the Rails console and type:

> freida = User.find\_by(email: 'freida@email.com')

> freida.admin?

Great job! We now have a way to determine whether a user has an admin role on the site. Let's add a few methods to the Application controller to make sure that users with the admin role can access specific parts of the site.

**17.** In the Application controller, create a method named require\_admin.

**18.** In the Recipes controller, create a before action that calls the require\_adminmethod before running the destroymethod.

**19.** In **app/views/recipes/show.html.erb**, use the admin? method to display a delete link only if a user is an admin:

<% if current\_user && current\_user.admin? %> <p class="recipe-delete"><%= link\_to "Delete", recipe\_path(@recipe), method: "delete" %><p> <% end %>

**20.** Try it out. Sign in as Freida with the email freida@email.com and password Freida1. Then visit <http://localhost:8000/recipes/1>. You should see your delete link.

Congratulations! You built a authorization system from scratch.

1. The *role* column in the User model specifies a users' role
2. A method like def admin? and def editor?is created for business logic
3. The require\_editor and require\_adminmethods redirect to () if the current user is not an editor or admin.
4. The before action acts a filter, callingrequire\_editor or require\_admin before excuting controller actions.
5. The current\_user method can be used in the views to display links based on the signed in user's role.