
Ruby on Rails

Autorización

Security Helpers

- Para poder tener acceso al usuario logeado desde los views necesitamos crear un `helper`.
- Agreguemos los helpers `logged_in?` y `current_user` a `ApplicationController` y disponibilizarlos en todos los Controllers y views mediante `helper_method`.
- Haciendo esto podríamos agregar lógica al `application.html.erb` para el logout y también para desplegar información sobre el usuario logeado.

application_controller.rb x

sessions_controller.rb x

```
1 class ApplicationController < ActionController::Base
2   # Prevent CSRF attacks by raising an exception.
3   # For APIs, you may want to use :null_session instead.
4   protect_from_forgery with: :exception
5
6   before_action :ensure_login
7   helper_method :logged_in?, :current_user
8
9   protected
10  def ensure_login
11    # Always go to login page unless session contains
12    # reviewer_id
13    redirect_to login_path unless session[:reviewer_id]
14  end
15
16  def logged_in?
17    session[:reviewer_id] # nil is false
18  end
19
20  def current_user
21    @current_user ||= Reviewer.find(session[:reviewer_id])
22  end
23 end
```

views/layout/application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>IReviewed</title>
  <%= stylesheet_link_tag 'application', media: 'all', 'data-turbolinks-track' => true %>
  <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
  <%= csrf_meta_tags %>
</head>
<body>
  <% if logged_in? %>
    <div style='float: right;'>
      Logged in as <%= current_user.name %> |
      <%= link_to "Logout", logout_path, method: :delete, class: "text-danger"%>
    </div>
  <% end %>

  <% flash.each do |key, value| %>
    <div class="alert alert-dismissible alert-success" id='<%= key %>'><%= value %></div>
  <% end %>
  <%= yield %>
</body>
</html>
```

IReviewed - Google Chrome

IReviewed

VaneCanhete

localhost:3000

🔍 ☆ 🌐 🔄 ⋮

Logged in as Joe | [Logout](#)

Logged in successfully

Logged in successfully

Listing Books

| Name | Author | |
|-------------------------------|----------------|---|
| Eloquent Ruby | Russ Olsen | Show Edit Destroy |
| Beginning Ruby | Peter Cooper | Show Edit Destroy |
| Metaprogramming Ruby 2 | Paolo Perrotta | Show Edit Destroy |
| Design Patterns in Ruby | Russ Olsen | Show Edit Destroy |
| The Ruby Programming Language | David Flanagan | Show Edit Destroy |

[New Book](#)

Autorización

Hemos implementado una autenticación básica, pero aún no hicimos nada por nuestra autorización.

Cualquier persona puede editar los libros y notas de cualquier otra persona.

Solución: Editar el BooksController para enviar solo datos del `current_user` a las vistas.

```

class BooksController < ApplicationController
  before_action :set_book, only: [:show, :edit, :update, :destroy]

  # GET /books
  # GET /books.json
  def index
    @books = current_user.books.all
  end

  # GET /books/1
  # GET /books/1.json
  def show
  end

  # GET /books/new
  def new
    @book = current_user.books.new
  end

  # GET /books/1/edit
  def edit
  end

  # POST /books
  # POST /books.json
  def create
    @book = current_user.books.new(book_params)
  end
end

```

Listing Books

| Name | Author | |
|-------------------------|----------------|---|
| Metaprogramming Ruby 2 | Paolo Perrotta | Show Edit Destroy |
| Design Patterns in Ruby | Russ Olsen | Show Edit Destroy |

[New Book](#)

Entonces...

Se puede utilizar la sesión para almacenar el usuario actual o su id.

Luego se pueden observar solo los recursos asociados a ese usuario.