

Laboratorio de Computación IV

Clase 17

Andrés Fortier

Repaso



- Roles
 - Definición y asignación.
 - Control antes de llevar adelante una acción (hoy).
 - Gemas
 - rolify - <https://github.com/RolifyCommunity/rolify>
 - pundit - <https://github.com/elabs/pundit>
- Bootstrap y layouts

Roles

- Requerimientos:
 - Página de listado de posts público.
 - Los últimos 10 posts (por fecha de creación).
 - Sólo pueden crear posts los usuarios logueados.
 - Contenido de un post público.
 - Sólo lo puede editar/borrar el autor.

Listado de posts público

- Modificar el seed para que nos genere mas de 10 posts

```
db/seeds.rb
```

```
for i in 0..15
  Article.create!({
    title: "Post number #{i}",
    text: "My #{i} post!",
    author: joeUser
  });
end
```

```
Article.create!(title: 'Hi everybody!', text: "This is Jane's first post", author: janeUser);
```

Listado de posts público

- Modificar el controller

```
/app/controllers/articles_controller.rb
```

```
class ArticlesController < ApplicationController
  skip_before_action :authenticate_user!, only: [:index]

  def index
    @articles = Article
      .all
      .limit(10)
      .order(created_at: :desc)
  end
  ...
end
```

Listado de posts público

- Sólo mostrar el link de crear post si el usuario está logueado.

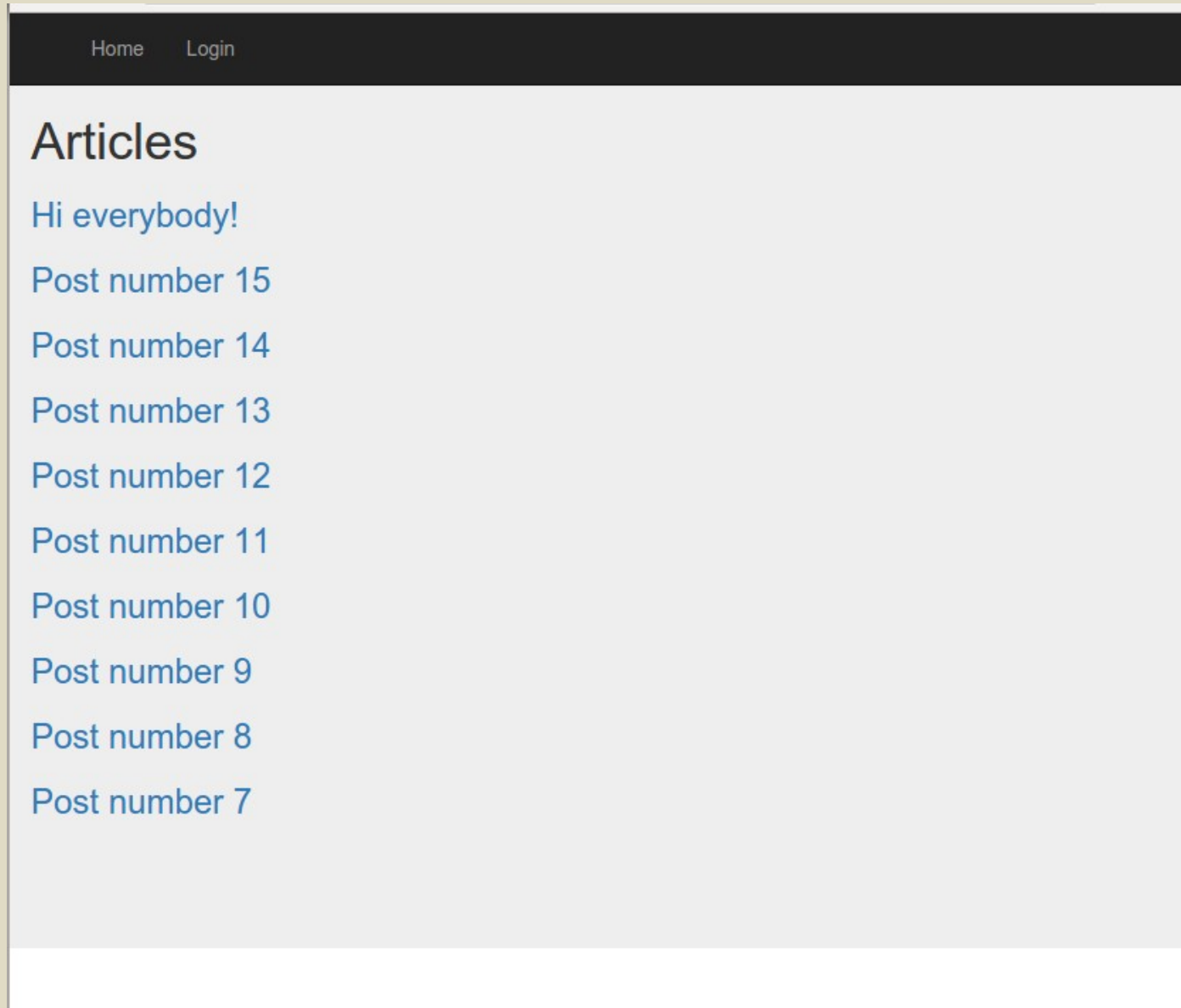
```
/app/views/articles/index.html.erb
```

```
...
```

```
<% if current_user.present? %>  
  <%= link_to 'New article', new_article_path %>  
<% end %>
```

- Prueben ir al index con usuario logueado y sin loguear.

Listado de posts público



Listado de posts público

- ¿Es necesario cambiar algo mas?

Contenido de un post público

- Eliminar la autenticación del show.
- Validar el usuario para mostrar los links
 - Si no hay usuario, sólo el “Back to index”.
 - Si es el autor, mostramos el borrar y eliminar.

Contenido de un post público

```
/app/controllers/articles_controller.rb
```

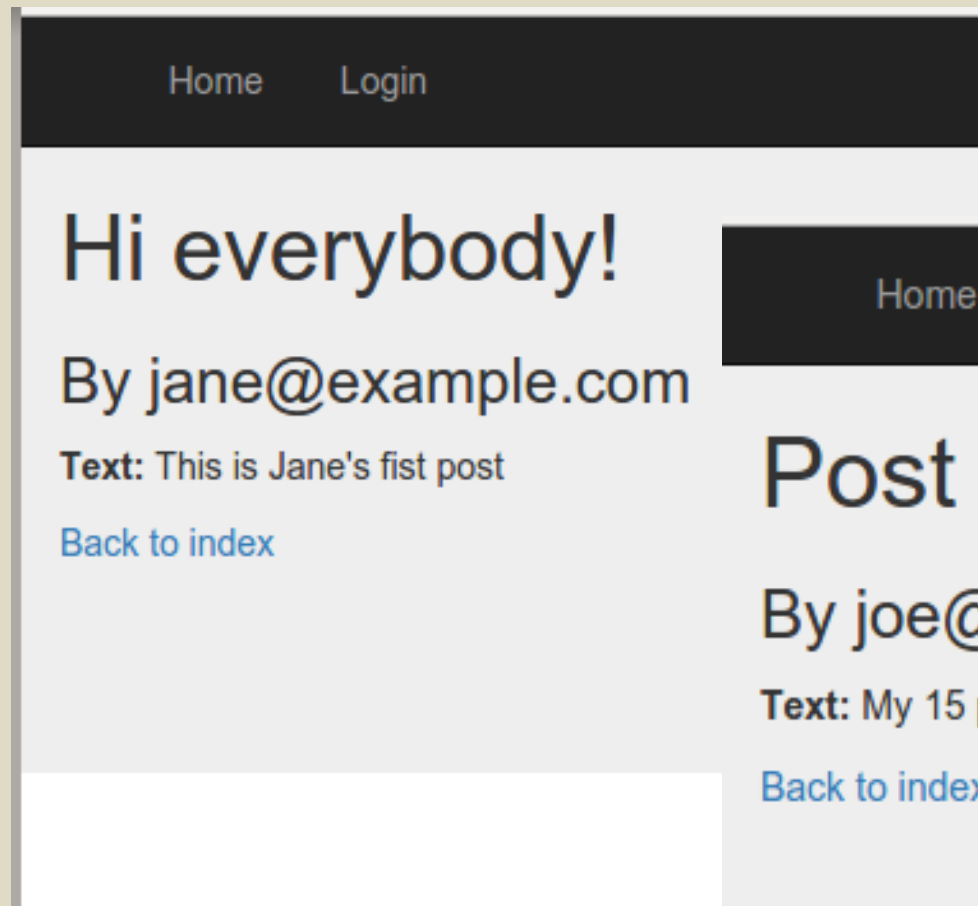
```
class ArticlesController < ApplicationController
  skip_before_action :authenticate_user!,
    only: [:index, :show]
  ...
end
```

```
/app/views/articles/show.html.erb
```

```
<% if current_user.present? && current_user ==
@article.author%>
  <%= link_to 'Edit', edit_article_path(@article) %>
  <%= link_to 'Delete', article_path(@article), method:
:delete %>
<% end %>
```

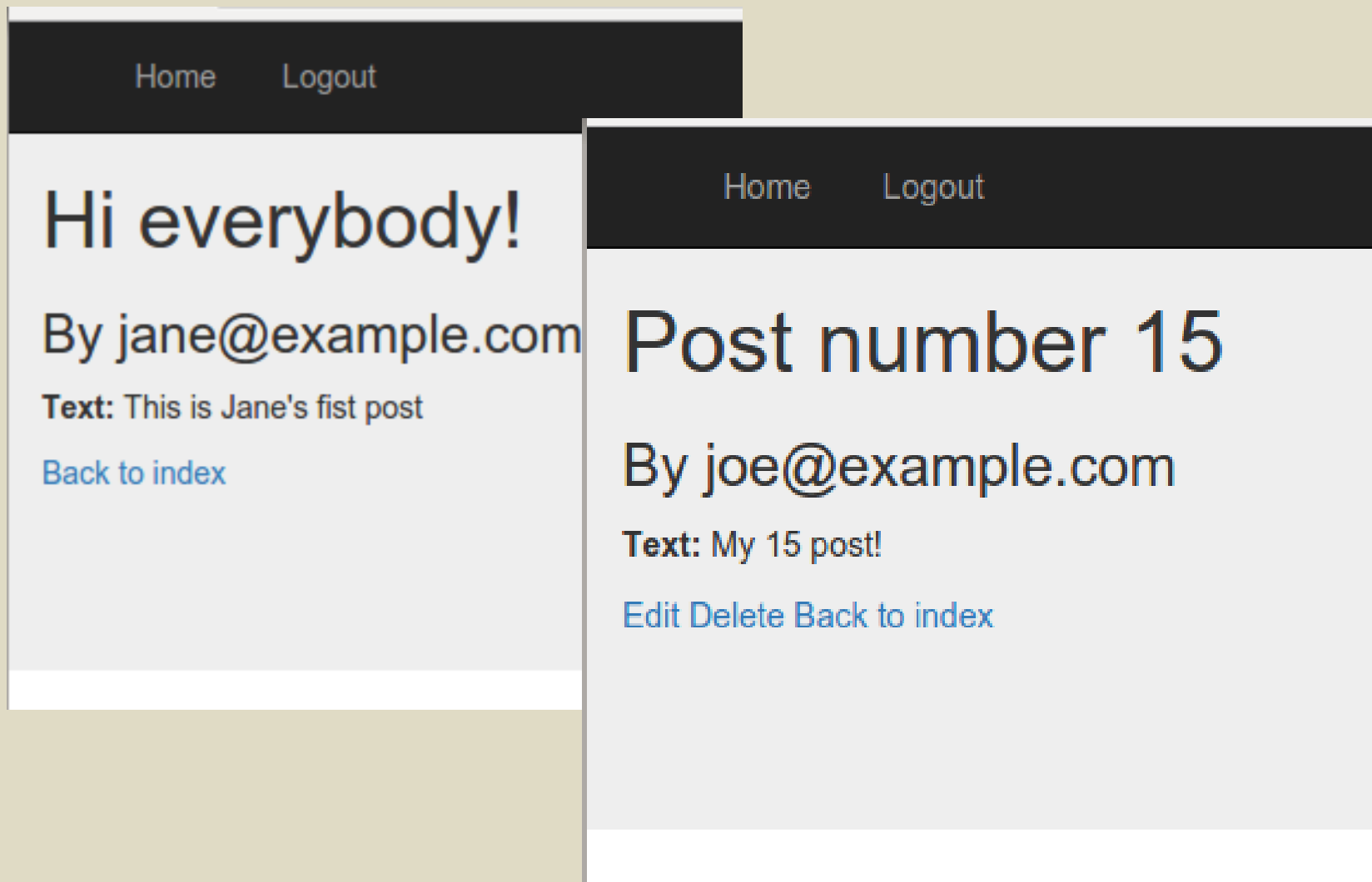
Contenido de un post público

- Usuario no logueado



Contenido de un post público

- Logueado como joe@example.com



Contenido de un post público

- ¿Es necesario chequear algo mas?
 - <http://localhost:3000/articles/17/edit>

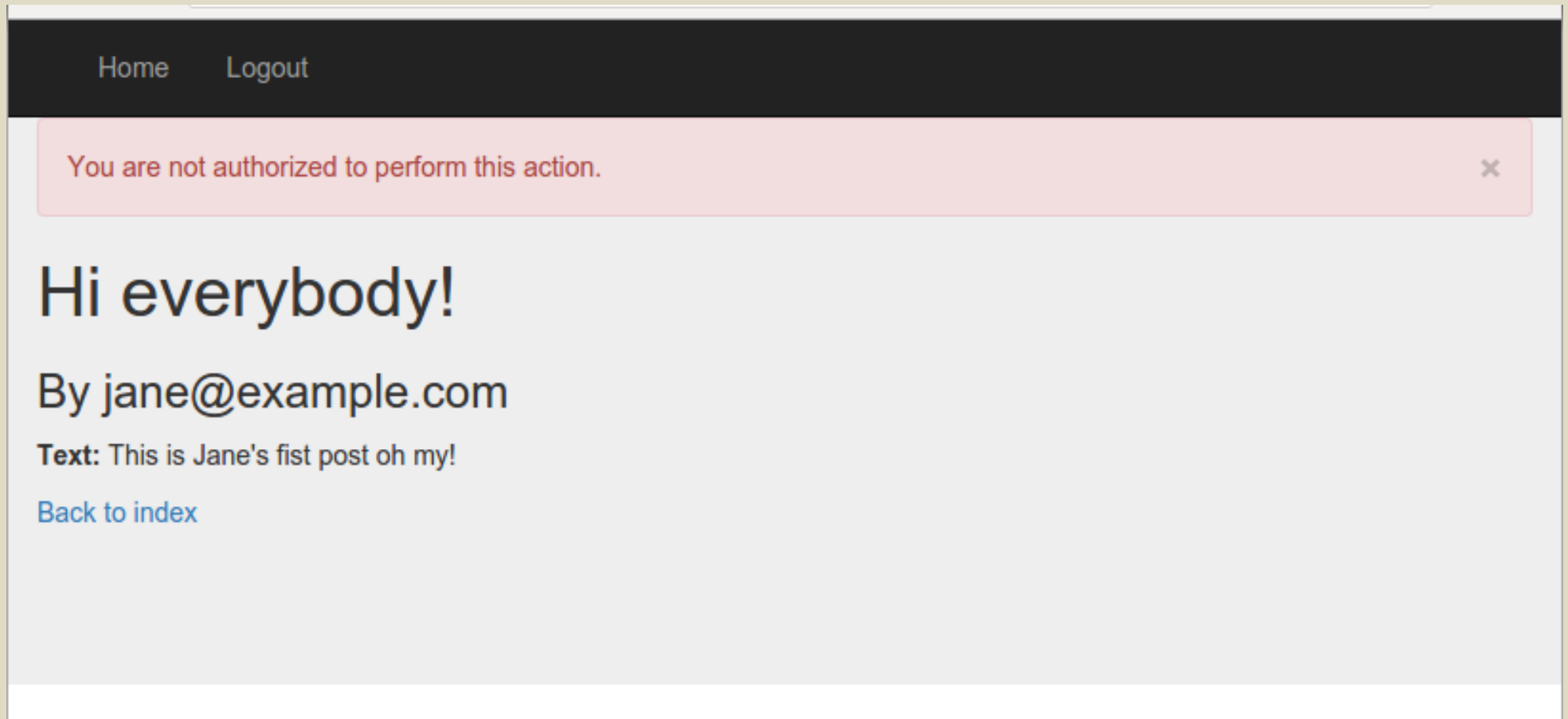
```
$ bin/rake routes | grep articles
GET      /                  articles#index
GET      /articles(.:format) articles#index
POST     /articles(.:format) articles#create
GET      /articles/new(.:format) articles#new
GET      /articles/:id/edit(.:format) articles#edit
GET      /articles/:id(.:format) articles#show
PATCH   /articles/:id(.:format) articles#update
DELETE  /articles/:id(.:format) articles#destroy
```

Contenido de un post público

```
/app/controllers/articles_controller.rb
class ArticlesController < ApplicationController
  ...
  def edit
    @article = Article.find(params[:id])
    if @article.author != current_user
      flash[:alert] = 'You are not authorized to perform
this action.'
      redirect_to @article
    end
  end
  ...
end
```

Contenido de un post público

- Intentemos nuevamente
<http://localhost:3000/articles/17/edit>



Contenido de un post público

- Factoricemos el código

```
/app/controllers/articles_controller.rb
```

```
private

  def redirect_if_not_author(article)
    if article.author != current_user
      flash[:alert] = 'You are not authorized to perform
this action.'
      redirect_to @article
    end
  end

  ...
```


Contenido de un post público

```
/app/controllers/articles_controller.rb
```

```
def edit
  @article = Article.find(params[:id])
  redirect_if_not_author(@article)
end
```

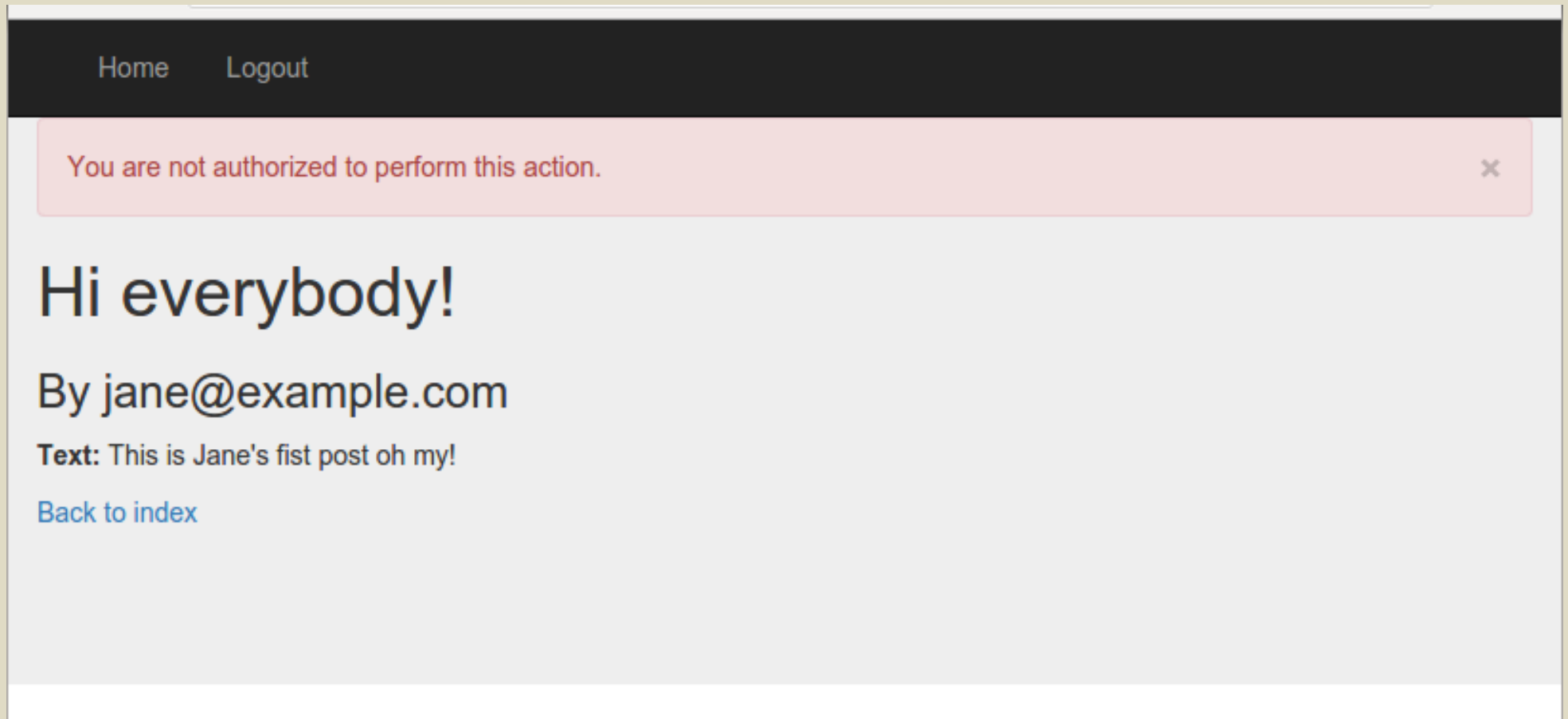
```
def destroy
  @article = Article.find(params[:id])
  redirect_if_not_author(@article)
  ...
end
```

```
def update
  @article = Article.find(params[:id])
  redirect_if_not_author(@article)
  ...
end
```

Contenido de un post público

- Verifiquemos

<http://localhost:3000/articles/17/edit>



The screenshot shows a web application interface. At the top, a dark navigation bar contains links for 'Home' and 'Logout'. Below this, a light pink error message box states 'You are not authorized to perform this action.' with a close button. The main content area displays a public post with the text 'Hi everybody!' followed by 'By jane@example.com'. Below the post text, it says 'Text: This is Jane's fist post oh my!'. At the bottom of the post content, there is a blue link that says 'Back to index'.

Home Logout

You are not authorized to perform this action. x

Hi everybody!

By jane@example.com

Text: This is Jane's fist post oh my!

[Back to index](#)

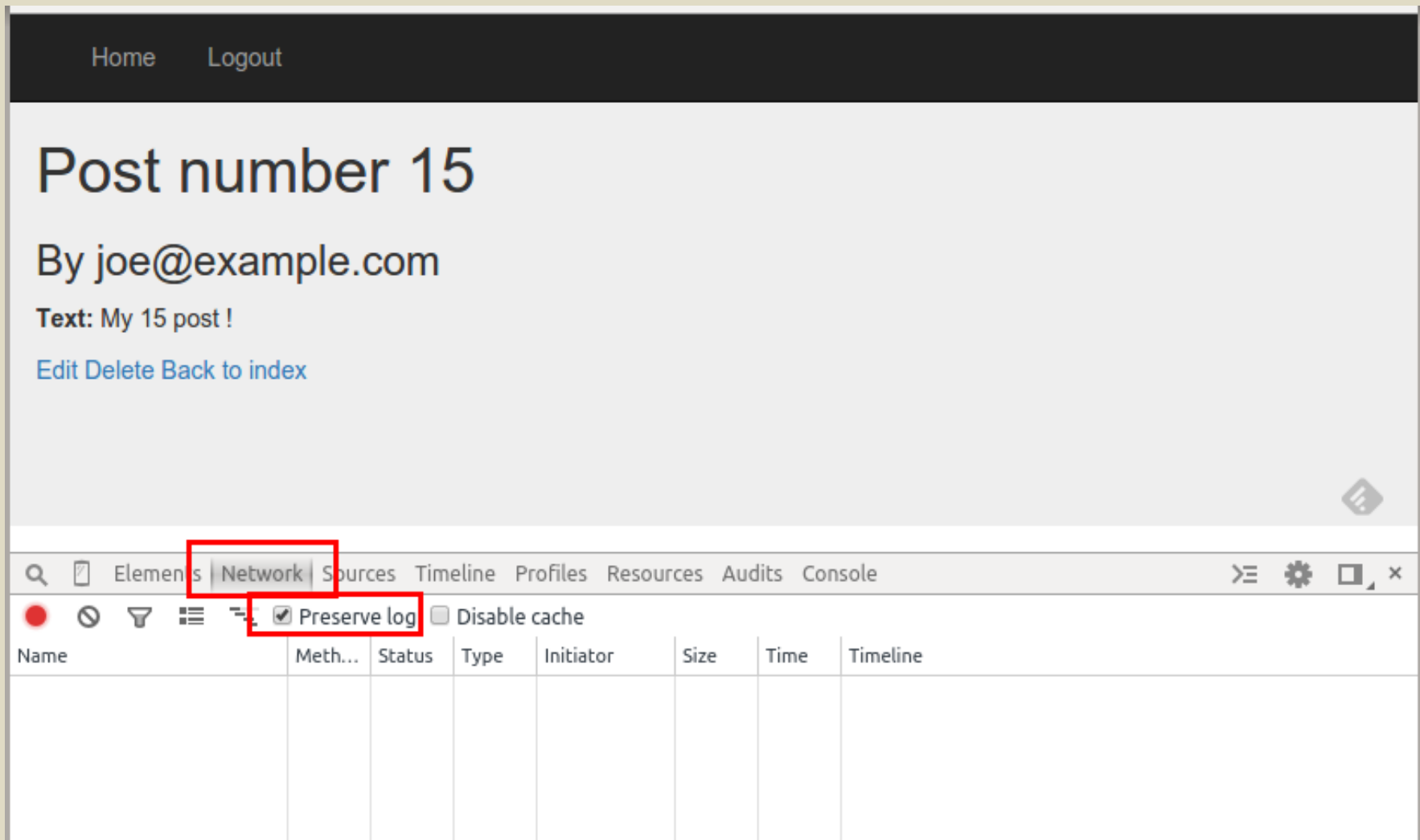
Contenido de un post público



- ¿Cómo verificamos el borrado o edición de un artículo?
- Idea
 - Borrar un post al que tenemos acceso para generar un request válido.
 - Modificar el request para que borre otro post.
 - Validar que no tenemos acceso.

Contenido de un post público

- Herramientas del navegador



Contenido de un post público

- Borremos el post

The screenshot shows a web browser at `localhost:3000/articles`. The page has a dark header with 'Home' and 'Logout' links. Below the header, the title 'Articles' is followed by a list of links: 'Hi everybody!', 'Post number 14', 'Post number 13', and 'Post number 12'. At the bottom, the Chrome DevTools Network tab is open, showing a list of requests. The first request, a POST to `jquery.js?bo...`, is highlighted with a red box. An arrow points from the text 'Botón derecho → Copy as cURL' to the right-click context menu of this request.

Articles

Hi everybody!

Post number 14

Post number 13

Post number 12

Botón derecho → Copy as cURL

Name	Meth...	Status	Type	Initiator	Size	Time	Timeline
16	POST	302	text/...	jquery.js?bo...	921 B	169 ms	
articles	GET	200	text/...	http://local...	4.4 KB	296 ms	
framework_and_overrid...	GET	200	text/...	articles:7	(from...	0 ms	

Contenido de un post público



- En una consola
 - Peguen el contenido
 - Cambien el 16 por 17

```
$ curl 'http://localhost:3000/articles/17' -H 'Cookie:  
_ejemplorails_session= ... --compressed
```

Contenido de un post público

- Veamos el log del servidor

```
Started DELETE "/articles/17" for 127.0.0.1 at 2015-05-17
16:54:13 -0300
Processing by ArticlesController#destroy as HTML
  Parameters:
{"authenticity_token"=>"qvu9qGw36Nl+0LVBFX2p3TBzhVirGfsmNkLIK
73KSuU=", "id"=>"17"}
  User Load (0.2ms)  SELECT  "users".* FROM "users"  WHERE
"users"."id" = 1  ORDER BY "users"."id" ASC LIMIT 1
  Article Load (0.1ms)  SELECT  "articles".* FROM "articles"
WHERE "articles"."id" = ? LIMIT 1  [["id", 17]]
  User Load (0.1ms)  SELECT  "users".* FROM "users"  WHERE
"users"."id" = ? LIMIT 1  [["id", 2]]
Redirected to http://localhost:3000/articles/17
  (0.1ms)  begin transaction
  SQL (0.4ms)  DELETE FROM "articles" WHERE "articles"."id" =
?  [["id", 17]]
  (150.1ms)  commit transaction
```

Contenido de un post público

```
...
Redirected to
Completed 500 Internal Server Error in 159ms

AbstractController::DoubleRenderError (Render and/or redirect
were called multiple times in this action. Please note that
you may only call render OR redirect, and at most once per
action. Also note that neither redirect nor render terminate
execution of the action, so if you want to exit an action
after redirecting, you need to do something like
"redirect_to(...) and return".):
  app/controllers/articles_controller.rb:46:in `destroy'
...
```

- ¿Qué sucedió?

Contenido de un post público

```
/app/controllers/articles_controller.rb
```

```
def destroy
  @article = Article.find(params[:id])
  redirect_if_not_author(@article)
  return if performed?

  @article.destroy
  redirect_to articles_path
end

def update
  @article = Article.find(params[:id])
  redirect_if_not_author(@article)
  return if performed?
  ...
end
```

Contenido de un post público



- db:reset.
- Reiniciar rails server.
- Borrar el post con id 16 de joe.
- Copiar la acción como cURL.
- Pegar en terminal y cambiar por id 17.
- Ver la consola del servidor

Contenido de un post público

- Veamos el log del servidor

```
Started DELETE "/articles/17" for 127.0.0.1 at 2015-05-17
17:02:40 -0300
Processing by ArticlesController#destroy as HTML
  Parameters:
  {"authenticity_token"=>"nkSZMZQchevk9rk8z70l7RS+2gbOvJqgctaHa
G5ocdQ=", "id"=>"17"}
    User Load (0.2ms)  SELECT  "users".* FROM "users"  WHERE
"users"."id" = 1  ORDER BY "users"."id" ASC LIMIT 1
    Article Load (0.1ms)  SELECT  "articles".* FROM "articles"
WHERE "articles"."id" = ? LIMIT 1  [["id", 17]]
    User Load (0.1ms)  SELECT  "users".* FROM "users"  WHERE
"users"."id" = ? LIMIT 1  [["id", 2]]
Redirected to http://localhost:3000/articles/17
Completed 302 Found in 5ms (ActiveRecord: 0.4ms)
```

Problemas con el enfoque



- Estamos definiendo la política de acceso a un recurso en el controller.
 - Complejidad y SRP.
 - Reutilización.
 - Tests.
- Fácil de introducir bugs de seguridad
 - ``return if performed?``

pundit



- <https://github.com/elabs/pundit>
- Modela los permisos de acceso a los recursos en clases
 - SRP.
 - Reutilización.
 - Tests.
- Provee helpers para integrar los chequeos con los controllers y vistas

pundit

```
/Gemfile
```

```
..  
gem 'pundit', '~> 1.1.0'  
...
```

```
$ bundle install
```

```
/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  
  before_action :authenticate_user!  
  include Pundit  
end
```

pundit

```
$ rails g pundit:install
      create  app/policies/application_policy.rb
$ cat app/policies/application_policy.rb
class ApplicationPolicy
  attr_reader :user, :record

  def initialize(user, record)
    @user = user
    @record = record
  end

  def index?
    false
  end

  ...

end
```

- Definamos una política para los artículos

```
/app/policies/article_policy.rb
```

```
class ArticlePolicy < ApplicationPolicy
```

```
  def index?
```

```
    true
```

```
  end
```

```
  def create?
```

```
    record.author == user
```

```
  end
```

```
  def new?
```

```
    user.present?
```

```
  end
```

```
... 
```


- Definamos una política para los artículos

```
/app/policies/article_policy.rb
```

```
...
```

```
def update?  
  record.author == user  
end
```

```
def destroy?  
  record.author == user  
end
```

```
end
```

pundit

- Utilicemos la política para autorizar
 - Borrar el código viejo

```
/app/controllers/articles_controller.rb
```

```
def edit
  @article = Article.find(params[:id])
  authorize @article
end
```

pundit

- Saquemos por un momento el `if` de los links

```
/app/views/articles/show.html.erb
```

```
...  
<%= link_to 'Edit', edit_article_path(@article) %>  
<%= link_to 'Delete', article_path(@article), method:  
:delete %>  
...
```

- Intenten editar el post 17

pundit

Pundit::NotAuthorizedError in ArticlesController#edit

not allowed to edit? this #<Article id: 17, title: "Hi everybody!", text: "This is Jane's fist post", created_at: "2015-05-17 20:32:11", updated_at: "2015-05-17 20:32:11", author_id: 2>

Extracted source (around line #38):

```
36   def edit
37     @article = Article.find(params[:id])
38     authorize @article
39   end
40
```

pundit

- Modifiquemos el application controller

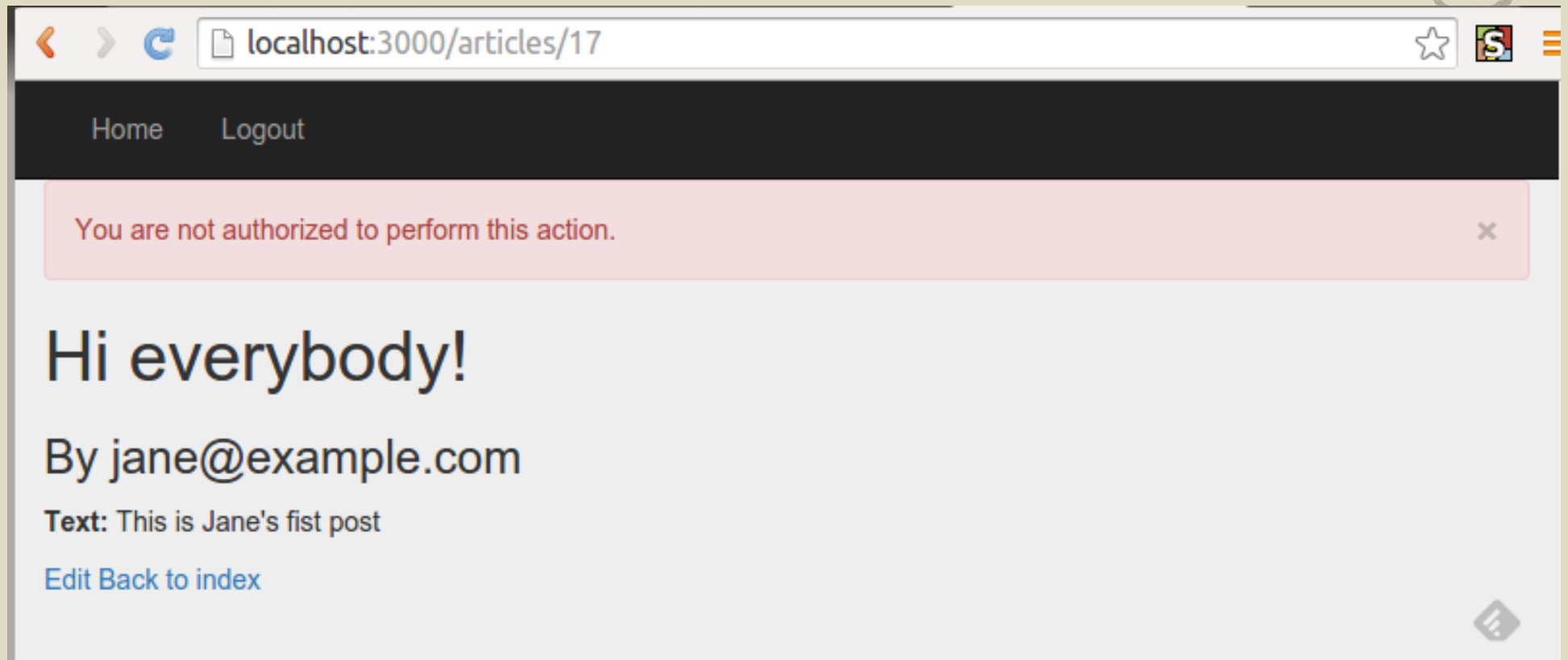
```
/app/controllers/application_controller.rb
class ApplicationController < ActionController::Base
  ...
  include Pundit

  rescue_from Pundit::NotAuthorizedError, with:
:  user_not_authorized

  private

  def user_not_authorized
    flash[:alert] = "You are not authorized to perform
this action."
    redirect_to(request.referrer || root_path)
  end
end
```

pundit



- Volvamos a nuestra vista del artículo

```
/app/views/articles/show.html.erb
```

```
...  
<% if policy(@article).edit? %>  
  <%= link_to 'Edit', edit_article_path(@article) %>  
<% end %>  
<% if policy(@article).destroy? %>  
  <%= link_to 'Delete', article_path(@article), method:  
:delete %>  
<% end %>  
<%= link_to 'Back to index', articles_path %>
```

- Pequeña modificación al índice

```
/app/views/articles/index.html.erb
```

```
...  
<% if policy(Article).new? %>  
  <%= link_to 'New article', new_article_path %>  
<% end %>
```


pundit

- Resumen

```
/app/controllers/articles_controller.rb
```

```
class ArticlesController < ApplicationController
  skip_before_action :authenticate_user!, only:
[:index, :show]
```

```
  def index
    authorize Article
    @articles = Article
      .all
      .limit(10)
      .order(created_at: :desc)
  end
```

```
  ...
```

pundit

/app/controllers/articles_controller.rb

...

def show

@article = Article.find(params[:id])

authorize @article

end

def new

authorize Article

@article = Article.new

end

...

pundit

/app/controllers/articles_controller.rb

...

def create

@article = Article.new(article_params)

@article.author = current_user

authorize @article

...

def edit

@article = Article.find(params[:id])

authorize @article

end

...

pundit

```
/app/controllers/articles_controller.rb
```

```
def destroy
  @article = Article.find(params[:id])
  authorize @article
```

```
  @article.destroy
  redirect_to articles_path
end
```

```
def update
  @article = Article.find(params[:id])
  authorize @article
```

```
  ...
end
```

```
...
```

- Permitamos que un admin pueda modificar artículos de otros autores

```
/app/policies/article_policy.rb
```

```
class ArticlePolicy < ApplicationPolicy

  ...

  def update?
    (record.author == user) || (user.has_role? :admin)
  end

  def destroy?
    (record.author == user) || (user.has_role? :admin)
  end

  ...
end
```

pundit

- Intenten ahora editar el post con id 17.

- Considerar
 - Solapamiento entre Autenticación y Autorización.
 - Solapamiento entre roles y atributos
 - ¿Rol “autor”?
 - ¿Cómo sabemos que estamos autorizando todas las rutas?

better_errors

- https://github.com/charliesome/better_errors
- Página de error mejorada para debuggear.

RuntimeError at /
boom

Application Frames

All Frames

A

HomeController#index

app/controllers/home_controller.rb, line 3

HomeController#send_action

actionpack (4.0.0) lib/action_controller/metal/implicit_render.rb, line 4

HomeController#process_action

actionpack (4.0.0) lib/abstract_controller/base.rb, line 189

HomeController#process_action

actionpack (4.0.0) lib/action_controller/metal/rendering.rb, line 10

block in HomeController#process_action

actionpack (4.0.0) lib/abstract_controller/callbacks.rb, line 18

HomeController#_run__4018816283576875081__process_action__callbacks

activesupport (4.0.0) lib/active_support/callbacks.rb, line 383

HomeController#run_callbacks

activesupport (4.0.0) lib/active_support/callbacks.rb, line 80

HomeController#process_action

actionpack (4.0.0) lib/abstract_controller/callbacks.rb, line 17

app/controllers/home_controller.rb

```
1 class HomeController < ApplicationController
2   def index
3     raise "boom"
4   end
5 end
```

```
>> def foo
..   "better_errors".upcase
.. end
=> nil
>>
```

▲ This is a live shell. Type in here.

Request info

Request parameters {"controller"=>"home", "action"=>"index"}

Rack session #<ActionDispatch::Request::Session:0x7f8623453728 not yet loaded>

better_errors

- Soporta
 - Stack trace
 - Código fuente con syntax-highlighting.
 - Inspeccionar variables
 - Un shell para evaluar código.

Importante!

/Gemfile

```
..  
group :development do  
  gem "better_errors"  
end  
...
```

Tarea para el hogar



- Instalar better_errors
- Ver <http://blog.arkency.com/2014/07/4-ways-to-early-return-from-a-rails-controller/>
- ¿Cómo sabemos que estamos autorizando todas las rutas?
 - <https://github.com/elabs/pundit#ensuring-policies-are-used>