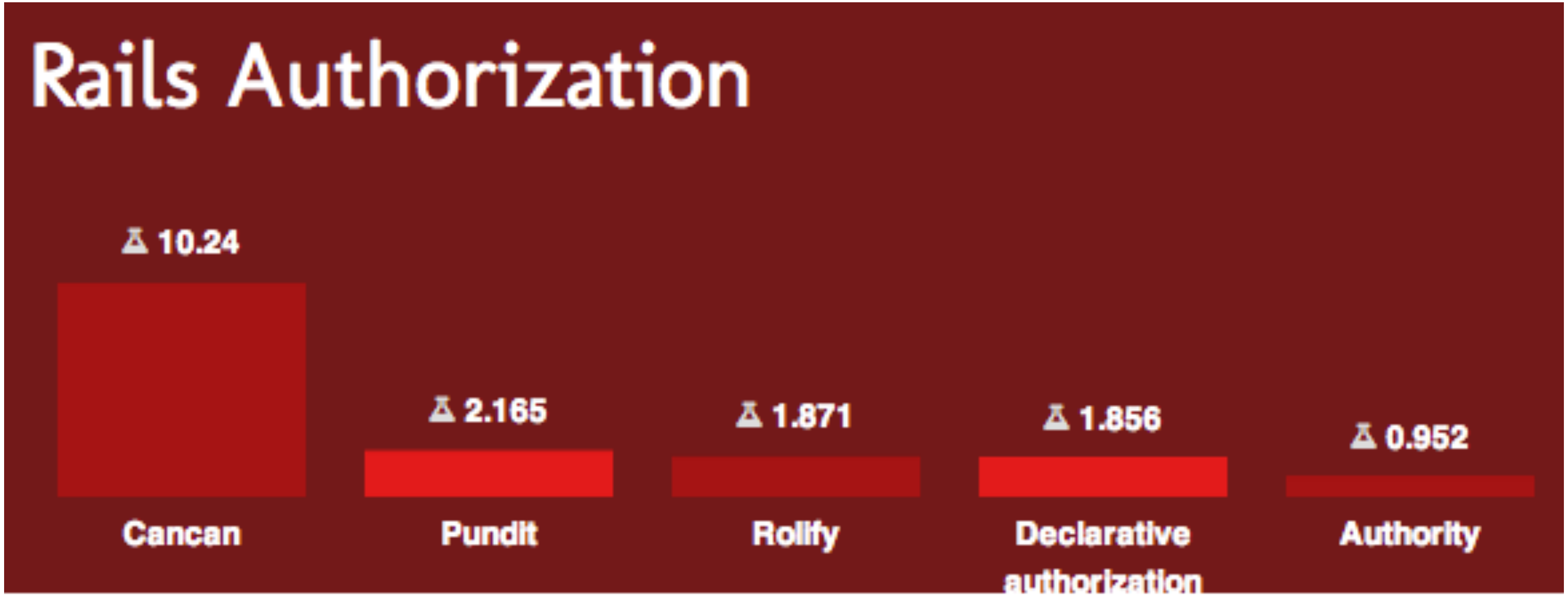
Authorization

CanCan and Pundit

Authorization

Authorization is allowing/restricting or constraining some set of behavior or operations based on the "role" being played by a user. Typically, this is done using Roles as in Role Based Access Control, (RBAC).

Popular Options



CanCan

CanCan is an authorization library for Ruby on Rails which restricts what resources a given user is allowed to access. All permissions are defined in a single location (the Ability class) and not duplicated across controllers, views, and database queries.

CanCan Example

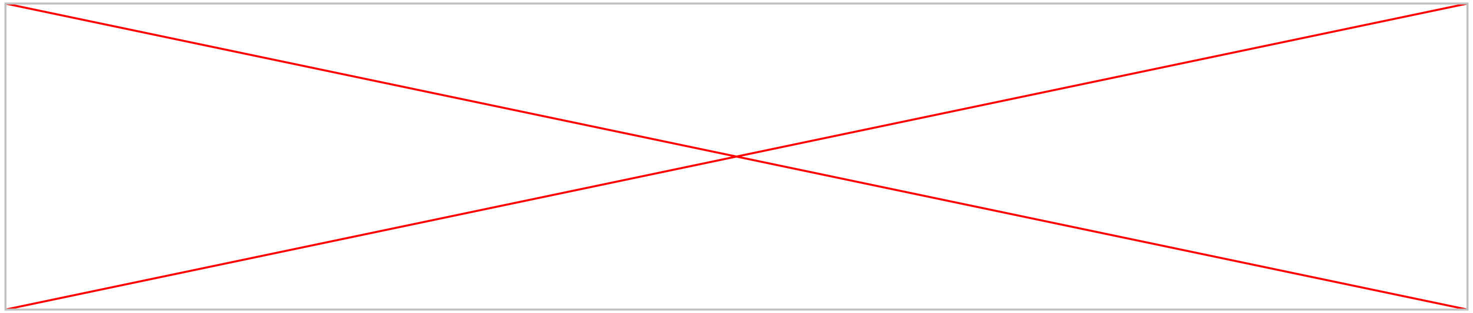
Getting Started

CanCan expects a current_user method to exist in the controller. First, set up some authentication (such as Devise).

CanCan Example

Define Abilities

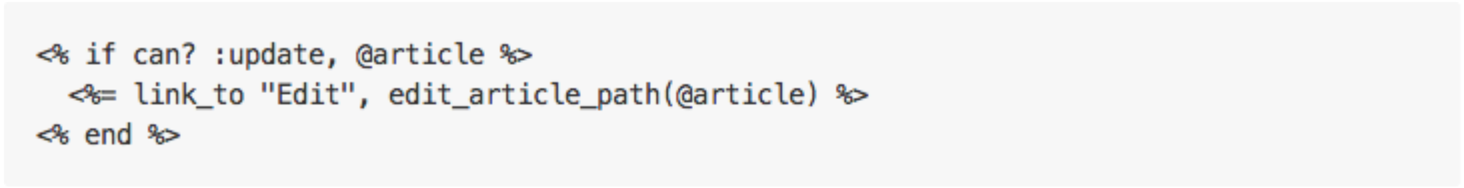
User permissions are defined in an Ability class. CanCan 1.5 includes a Rails generator for creating this class.



CanCan Example

Check Abilities

The current user's permissions can then be checked using the can? and cannot? methods in the view and controller.

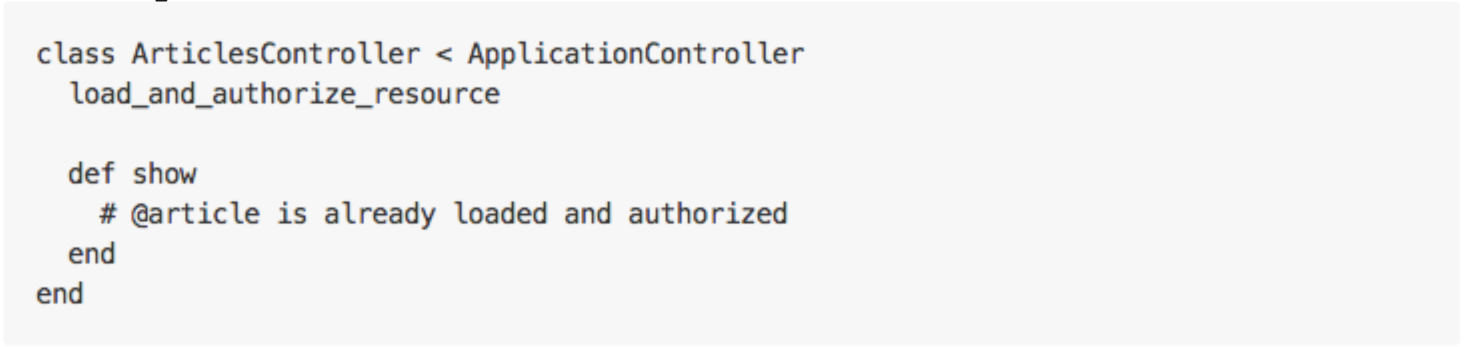


```
<% if can? :update, @article %>
  <%= link_to "Edit", edit_article_path(@article) %>
<% end %>
```

CanCan Example

Load & Authorize

load_and_authorize_resource method is provided to automatically authorize all actions. It will use a before filter to load the resource into an instance variable and authorize it for every action.



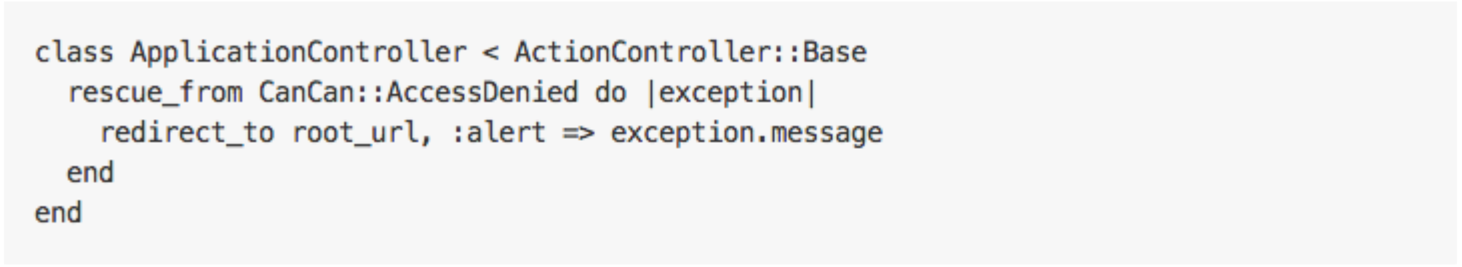
```
class ArticlesController < ApplicationController
  load_and_authorize_resource

  def show
    # @article is already loaded and authorized
  end
end
```

CanCan Example

Handle Unauthorized Access

If the user authorization fails, a CanCan::AccessDenied exception will be raised. You can catch this and modify its behavior in the ApplicationController.



```
class ApplicationController < ActionController::Base
  rescue_from CanCan::AccessDenied do |exception|
    redirect_to root_url, :alert => exception.message
  end
end
```

CanCan Strengths

- Simple to implement for small apps

CanCan Weaknesses

- It has not been updated in over a year
- Ability Class quickly becomes unwieldy/ inefficient once authorization logic becomes more complex
- CanCan has complex DSL, so it can be hard to force it to fit a specific application's needs

Pundit

Pundit provides a set of helpers which guide you in leveraging regular Ruby classes and object oriented design patterns to build a simple, robust and **scaleable** authorization system.

Pundit Example

Policies

Pundit is focused around the notion of policy classes. We suggest that you put these classes in app/policies.

```ruby
class PostPolicy
  attr_reader :user, :post

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

  def update?
    user.admin? or not post.published?
  end
end
```

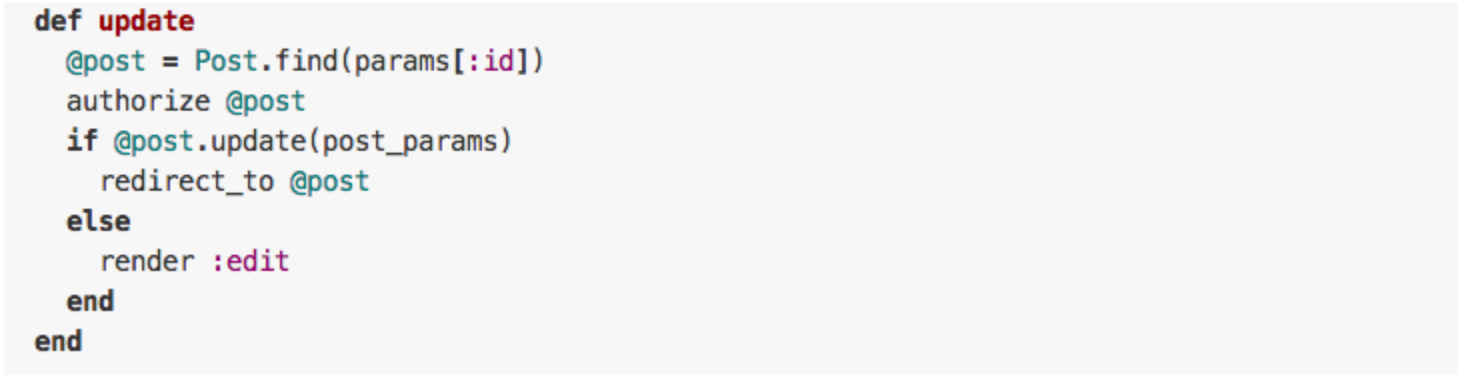
Pundit Example

- The policy class should have the same name as some kind of model class, only suffixed with the word "Policy".
- The first argument is a user.
- The second argument is some kind of model object, whose authorization you want to check.
- The class implements some kind of query method, in this case update?. Usually, this will map to the name of a particular controller action.

Pundit Example

Controllers

Supposing that you have an instance of class Post, Pundit now lets you do this in your controller:



```
def update
  @post = Post.find(params[:id])
  authorize @post
  if @post.update(post_params)
    redirect_to @post
  else
    render :edit
  end
end
```

Pundit Example

The authorize method automatically infers that Post will have a matching PostPolicy class, and instantiates this class, handing in the current user and the given record. It then infers from the action name, that it should call update? on this instance of the policy.

Pundit Example

Views

You can easily get a hold of an instance of the policy through the policy method in both the view and controller. This is especially useful for conditionally showing links or buttons in the view:

```
<% if policy(@post).update? %>
  <%= link_to "Edit post", edit_post_path(@post) %>
<% end %>
```

Pundit Strength

- It's a small library, so it is simple to get started and understand
- Pundit policies allows the user to break down authorization logic into smaller more maintainable chunks
- No complicated DSL

Pundit Weaknesses

- Authorization logic is not all in one place, so it is a little more complicated at first

Overall Comparison

CanCan uses a simple Abilities class to store all authorization logic, whereas Pundit breaks this logic into multiple Policy classes

Resources

The Ruby Toolbox - Rails Authorization

CanCan RailsCast

CanCan GitHub Page

Pundit GitHub Page