Iron Brush Tattoo *Case Jewelry*

AnderSon

Tue May 3 10:45:24 CDT 2016

Contents

https://github.com/IronBrushTattoo/cj_rails.git

1 Config

1.1 Gemfile

```
./Gemfile

source 'https://rubygems.org'

gem 'rails', '4.2.6'
gem 'pg', '~> 0.15'
gem 'sass-rails', '~> 5.0'
gem 'uglifier', '>= 1.3.0'
gem 'coffee-rails', '~> 4.1.0'
gem 'jquery-rails'
gem 'turbolinks'
gem 'jbuilder', '~> 2.0'
gem 'sdoc', '~> 0.4.0', group: :doc
gem 'dragonfly', '~> 1.0.12'
gem 'rack-cache', :require => 'rack/cache'
#gem 'prawn', '~> 2.1'
gem 'prawn'

group :development, :test do
  gem 'byebug'
end

group :development do
  gem 'web-console', '~> 2.0'
  gem 'spring'
end
```

1.2 Gems

3.2.1 1

2 First steps

```
rake db:migrate
rake db:setup
```

3 Project

The purpose of this application is to produce several pdf files from an xlsx file, as a re-implementation of `https://github.com/IronBrushTattoo/cj` as a web application.

3.1 User Story

- user logs in (3.4)
  - users will be piercers
- chooses xlsx file for upload 3.2
- selects number of days back to make labels from
- submits
  - BACKGROUND
    - cj-parser.rb does what it does...
      - ☐ rewrite in rails?
- downloads sheets(pdf files)

3.2 File Upload

- possible gems `https://www.ruby-toolbox.com/categories/rails_file_uploads`
  - paperclip
    - nb
      - used paperclip before
      - seemed to be designed specifically for image files
      - always worked well
  - carrierwave
    - nb
      - used before, but not thoroughly
      - i kind of remember having issues with it

- dragonfly `https://github.com/markevans/dragonfly` `http://markevans.github.io/dragonfly/` `http://markevans.github.io/dragonfly/rails/`

  Dragonfly is a highly customizable ruby gem for handling images and other attachments and is already in use on thousands of websites

  3.2.1

  * nb
    - used briefly before
    - i remember it being an easy configuration
- attachment fu `https://github.com/technoweenie/attachment_fu`

  Treat an ActiveRecord model as a file attachment, storing its patch, size, content type, etc. `http://weblog.technoweenie.net`

  * nb
    - has not been maintained since Apr 25, 2009
- refile
  * nb
    - was my next choice when previously working with file uploads
- jquery.fileupload-rails
- imagery
- attached
- papermill
- fileuploader-rails
- filecip
- simple-image-uploader

3.2.1 Dragonfly

`http://markevans.github.io/dragonfly/rails/`

1. Setup

   - ☒ gem 'dragonfly', '~> 1.0.12'

     - ☒ modify **??**
     - ☒ bundle install

   - ☒ rails g dragonfly

     generates config/initializers/dragonfly.rb

     `./config/initializers/dragonfly.rb`

     ```
     require 'dragonfly'

     # Configure
     Dragonfly.app.configure do
       plugin :imagemagick

       secret "72245c7371d66ccca6f9356779fa16e3104e6676c1e57af987e9e3f92130dca5"

       url_format "/media/:job/:name"

       datastore :file,
         root_path: Rails.root.join('public/system/dragonfly', Rails.env),
         server_root: Rails.root.join('public')
     end

     # Logger
     Dragonfly.logger = Rails.logger

     # Mount as middleware
     Rails.application.middleware.use Dragonfly::Middleware

     # Add model functionality
     if defined?(ActiveRecord::Base)
     ```

```
  ActiveRecord::Base.extend Dragonfly::Model
  ActiveRecord::Base.extend Dragonfly::Model::Validations
end
```

2. Handling attachments

   - example (replace Photo model with Spreadsheet)

     Model: *Photo*

     – add *image* attribute to Photo

     ```
     class Photo < ActiveRecord::Base
       dragonfly_accessor :image  # defines a reader/writer for image
       # ...
     end
     ```

     – needs *image*$_{uid}$ column, create migration with

     ```
     rails g migration
     ```

     ```
     add_column :photos, :image_uid, :string
     add_column :photos, :image_name, :string  # Optional - if you want
                                               # urls to end with the
                                               # original filename
     ```

     – view for uploading

     ```
     app/views/photos/...
     ```

     ```
     <% form_for @photo do |f| %>
       ...
       <%= f.file_field :image %>
       ...
     <% end %>
     ```

     – allow parameter *image* to be accepted by the controller

     ```
     app/controllers/photos_controller.rb
     ```

```
params.require(:photo).permit(:image)
```

  - view for displaying

```
<%= image_tag @photo.image.thumb('400x200#').url if @photo.image_stored?
```

  - more can be done with models

- Spreadsheet model sketch based on above example

  Model: *Spreadsheet*

  3.7.2

  - ☒ add *file* attribute to Spreadsheet

```
class Spreadsheet < ActiveRecord::Base
  dragonfly_accessor :file  # defines a reader/writer for file
  # ...
end
```

  - ☒ needs $file_{uid}$ column, create migration with

```
rails g migration AddFileUidToSpreadsheets file_uid:string
rails g migration AddFileNameToSpreadsheets file_name:string
```

```
./db/migrate/20160504011342_add_file_uid_to_spreadsheets.
rb ./db/migrate/20160504011542_add_file_name_to_spreadsheets.
rb
```

```
add_column :spreadsheets, :file_uid, :string
add_column :spreadsheets, :file_name, :string  # Optional - if you want
                                               # urls to end with the
                                               # original filename
```

```
rake db:migrate
```

  - ☒ view for uploading

    ./app/views/spreadsheets/

```
<% form_for @spreadsheet do |f| %>
  ...
  <%= f.file_field :file %>
  ...
<% end %>
```

- ☒ allow parameter *file* to be accepted by the controller

  ./app/controllers/spreadsheets_controller.rb

```
params.require(:spreadsheet).permit(:file)
```

```
class SpreadsheetsController < ApplicationController
  before_action :set_spreadsheet, only: [:show, :edit, :update, :destroy

  def index
    @spreadsheets = Spreadsheet.all
  end

  def show
    @spreadsheet = Spreadsheet.find(params[:id])
    respond_to do |format|
      format.html
      format.pdf do
        pdf = SpreadsheetPdf.new(@spreadsheet, view_context)
        send_data pdf.render,
                  filename: "spreadsheet_#{@spreadsheet.created_at.strft
                  type: "application/pdf",
                  disposition: "inline"
      end
    end
  end

  def new
    @spreadsheet = Spreadsheet.new
  end

  def edit
  end
```

```
def create
  @spreadsheet = Spreadsheet.new(spreadsheet_params)

  respond_to do |format|
    if @spreadsheet.save
      format.html { redirect_to @spreadsheet, notice: 'Spreadsheet was
      format.json { render :show, status: :created, location: @spreads
    else
      format.html { render :new }
      format.json { render json: @spreadsheet.errors, status: :unproce
    end
  end
end

def update
  respond_to do |format|
    if @spreadsheet.update(spreadsheet_params)
      format.html { redirect_to @spreadsheet, notice: 'Spreadsheet was
      format.json { render :show, status: :ok, location: @spreadsheet
    else
      format.html { render :edit }
      format.json { render json: @spreadsheet.errors, status: :unproce
    end
  end
end

def destroy
  @spreadsheet.destroy
  respond_to do |format|
    format.html { redirect_to spreadsheets_url, notice: 'Spreadsheet w
    format.json { head :no_content }
  end
end

private
def set_spreadsheet
  @spreadsheet = Spreadsheet.find(params[:id])
end

def spreadsheet_params
```

```
    params.require(:spreadsheet).permit(:index, :file)
  end
end
```

        - nb
          1 1(c)i

    - ☒ view for displaying

      ./app/views/spreadsheets/show.html.erb ./app/views/spreadsheets/index.html.erb

      ```
      <%= @spreadsheet.file_name if @spreadsheet.file_stored? %>
      ```

    - more can be done with models

3. Caching

    - ☒ ??

      ```
      gem 'rack-cache', :require => 'rack/cache'
      ```

        - ☒ bundle install

    - ☒ uncomment in `Production`

      ```
      config.action_dispatch.rack_cache = true
      ```

4. Custom Endpoints

   3.5.1

    - ☐ text generation example

      ```
      get "text/:text" => Dragonfly.app.endpoint { |params, app|
        app.generate(:text, params[:text], 'font-size' => 42)
      }
      ```

    - ☐ endpoint callable from javascript (e.g. /image?file=egg.png&size=30x30)

      ```
      get "image" => Dragonfly.app.endpoint { |params, app|
        app.fetch_file("some/dir/#{params[:file]}").thumb(params[:size])
      }
      ```

3.3 File Conversion

3.3.1 xlsx processing

- roo

3.3.2 latex processing

1. Prawn `https://github.com/prawnpdf/prawn`

   (a) Setup

   - ☒ ??

   ```
   #gem 'prawn', '~> 2.1'
   gem 'prawn'
   ```

   - ☒ bundle install

   i. Manual

   `http://prawnpdf.org/manual.pdf`

   - ☒ clone repository

     `git clone https://github.com/prawnpdf/prawn`

   - ☐ switch to the stable branch

     `git branch stable`

   - ☒ bundle install
   - ☐ bundle exec rake manual generates *manual.pdf* in the project root

   (b) basic$_{\text{concepts}}$/creation.rb

   ```
   Prawn::Document
   Prawn::Document.generate
   ```

   (c) Tutorials

i. Creating PDF Using Prawn in Ruby on Rails

`http://www.idyllic-software.com/blog/creating-pdf-using-prawn-in-ruby-o`

- ☒ ??

  ```
  gem 'prawn'
  bundle install
  ```

- ☒ `./config/initializers/mime_types.rb`
  create a PDF Mime::Type inside $\text{mime}_{\text{types}}$.rb

  ```
  # Be sure to restart your server when you modify this file.

  # Add new mime types for use in respond_to blocks:
  # Mime::Type.register "text/richtext", :rtf

  Mime::Type.register "application/pdf", :pdf
  ```

- ☒ `./app/controllers/spreadsheets_controller.rb`
  3.6.2
  `http://www.idyllic-software.com/blog/creating-pdf-using-prawn-in-ru`

  ```
  class InvoicesController < ApplicationController

    before_filter :authenticate_customer!, :only => [:index, :show]

    def index
      @invoices = Invoice.all_invoices(current_customer)
    end

    def show
      @invoice = Invoice.find(params[:id])
      respond_to do |format|
        format.html
        format.pdf do
          pdf = InvoicePdf.new(@invoice, view_context)
          send_data pdf.render, filename:
            "invoice_#{@invoice.created_at.strftime("%d/%m/%Y")}.pdf",
            type: "application/pdf"
        end
      end
  ```

```
  end

end
```

  - ☒ open pdf in browser instead of download
    add /disposition: "inline"/ after type
- ☒ create a new class **app/pdfs/spreadsheet$_{\mathbf{pdf}}$**
  3.8.1
  3.6.2
- ☒ restart server
- ☒ 3.8.1
  set the *@spreadsheet* and *view$_{context}$*

```
def initialize(spreadsheet, view)
  super()
  @spreadsheet = spreadsheet
  @view = view
  text "Spreadsheet #{@spreadsheet.id}"
end
```

- ☐ create different methods inside 3.8.1 as per what you want to show on the pdf
  - example

```
def logo
  logopath = "#{Rails.root}/app/assets/images/logo.png"
  image logopath, :width => 197, :height => 91
end
```

A. Issues
   - ☐ NameError (uninitialized constant SpreadsheetsController::SpreadsheetPdf)

2. nb `https://rubygems.org/search?utf8=%E2%9C%93&query=latex` `http://www.sitepoint.com/hackable-pdf-typesetting-in-ruby-with-prawn/`

   `https://github.com/prawnpdf/prawn`

   1 is active and looks rad!

   - outdated but possibly useful

     `https://github.com/baierjan/rails-latex` `https://github.com/bruce/rtex`

3.4 Authentication

3.5 Views

3.5.1 Routes

./config/routes.rb

```
Rails.application.routes.draw do
  root 'pages#home'

  resources :spreadsheets

  get "spreadsheets" => "spreadsheets#new"

  # The priority is based upon order of creation: first created -> highest priority.
  # See how all your routes lay out with "rake routes".

  # You can have the root of your site routed with "root"
  # root 'welcome#index'

  # Example of regular route:
  #   get 'products/:id' => 'catalog#view'

  # Example of named route that can be invoked with purchase_url(id: product.id)
  #   get 'products/:id/purchase' => 'catalog#purchase', as: :purchase

  # Example resource route (maps HTTP verbs to controller actions automatically):
  #   resources :products

  # Example resource route with options:
  #   resources :products do
  #     member do
  #       get 'short'
  #       post 'toggle'
  #     end
  #
  #     collection do
  #       get 'sold'
  #     end
  #   end
  # Example resource route with sub-resources:
  #   resources :products do
  #     resources :comments, :sales
  #     resource :seller
  #   end

  # Example resource route with more complex sub-resources:
  #   resources :products do
  #     resources :comments
  #     resources :sales do
  #       get 'recent', on: :collection
  #     end
  #   end

  # Example resource route with concerns:
  #   concern :toggleable do
  #     post 'toggle'
  #   end
  #   resources :posts, concerns: :toggleable
  #   resources :photos, concerns: :toggleable

  # Example resource route within a namespace:
  #   namespace :admin do
  #     # Directs /admin/products/* to Admin::ProductsController
  #     # (app/controllers/admin/products_controller.rb)
  #     resources :products
  #   end
end
```

4

3.5.2 Static Pages

```
root 'pages#home'
```

3.6.1

1. Home

   `./app/views/pages/home.html.erb`

3.6 Controllers

3.6.1 Pages

Static pages controller

```
rails g controller pages --skip-assets
```

3.6.2 Spreadsheets

3.7 Models

3.7.1 Page

3.7.2 Spreadsheet

`./app/models/spreadsheet.rb`

```
class Spreadsheet < ActiveRecord::Base
  dragonfly_accessor :file  # defines a reader/writer for file
end
```

3.7.3 Label

3.8 Classes

3.8.1 SpreadsheetPdf

```
mkdir ./app/pdfs
```

`./app/pdfs/spreadsheet_pdf.rb`

```
class SpreadsheetPdf < Prawn::Document

  def initialize(spreadsheet, view)
    super()
    @spreadsheet = spreadsheet
    @view = view
    logo
    text "This is a sheet of labels for selected case jewelry, #{@spreadsheet.id}"
  end

  def logo
    logopath = "#{Rails.root}/app/assets/images/logo.png"
    image logopath, :width => 197, :height => 197
  end

end
```

- nb

  1

3.9 TODO

- ☐ Tests
- ☐ sidekiq
  - ☐ background processes for creating pdfs
- ☐ requirements
  - ☐ roo
  - ☐ chronic
- ☐ pdflatex

  1
- ☐ migrate code from cj-parser
- ☐ user authentication
- ☐ file upload
  - ☐ only xlsx file?
  - ☐ AWS
- ☐ file storage
  - ☐ archival api?