

Iron Brush Tattoo *Case Jewelry*

AnderSon

Tue May 3 10:45:24 CDT 2016

Contents

1	Config	2
1.1	Gemfile	2
1.2	Gems	2
1.3	Environments	2
1.3.1	Development	2
1.3.2	Production	2
2	First steps	5
3	Project	5
3.1	User Story	5
3.2	File Upload	5
3.2.1	Dragonfly	7
3.3	Authentication	12
3.4	Views	12
3.4.1	Routes	12
3.4.2	Static Pages	14
3.5	Controllors	14
3.5.1	Pages	14
3.5.2	Spreadsheets	15
3.6	Models	15
3.6.1	Page	15
3.6.2	Spreadsheet	15
3.7	TODO	15
	<i>Rails 4.2.6</i>	
	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'
```

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

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

1.2 Gems

3.2.1

1.3 Environments

1.3.1 Development

1.3.2 Production

./config/environments/production.rb

```
Rails.application.configure do
```

```

# Settings specified here will take precedence over those in config/application.rb.

# Code is not reloaded between requests.
config.cache_classes = true

# Eager load code on boot. This eager loads most of Rails and
# your application in memory, allowing both threaded web servers
# and those relying on copy on write to perform better.
# Rake tasks automatically ignore this option for performance.
config.eager_load = true

# Full error reports are disabled and caching is turned on.
config.consider_all_requests_local       = false
config.action_controller.perform_caching = true

# Enable Rack::Cache to put a simple HTTP cache in front of your application
# Add 'rack-cache' to your Gemfile before enabling this.
# For large-scale production use, consider using a caching reverse proxy like
# NGINX, varnish or squid.
config.action_dispatch.rack_cache = true

# Disable serving static files from the '/public' folder by default since
# Apache or NGINX already handles this.
config.serve_static_files = ENV['RAILS_SERVE_STATIC_FILES'].present?

# Compress JavaScripts and CSS.
config.assets.js_compressor = :uglifier
# config.assets.css_compressor = :sass

# Do not fallback to assets pipeline if a precompiled asset is missed.
config.assets.compile = false

# Asset digests allow you to set far-future HTTP expiration dates on all assets,
# yet still be able to expire them through the digest params.
config.assets.digest = true

# 'config.assets.precompile' and 'config.assets.version' have moved to config/initializers/assets.rb

# Specifies the header that your server uses for sending files.
# config.action_dispatch.x_sendfile_header = 'X-Sendfile' # for Apache

```

```

# config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for NGINX

# Force all access to the app over SSL, use Strict-Transport-Security, and use secure cookies.
# config.force_ssl = true

# Use the lowest log level to ensure availability of diagnostic information
# when problems arise.
config.log_level = :debug

# Prepend all log lines with the following tags.
# config.log_tags = [ :subdomain, :uuid ]

# Use a different logger for distributed setups.
# config.logger = ActiveSupport::TaggedLogging.new(SyslogLogger.new)

# Use a different cache store in production.
# config.cache_store = :mem_cache_store

# Enable serving of images, stylesheets, and JavaScripts from an asset server.
# config.action_controller.asset_host = 'http://assets.example.com'

# Ignore bad email addresses and do not raise email delivery errors.
# Set this to true and configure the email server for immediate delivery to raise delivery errors.
# config.action_mailer.raise_delivery_errors = false

# Enable locale fallbacks for I18n (makes lookups for any locale fall back to
# the I18n.default_locale when a translation cannot be found).
config.i18n.fallbacks = true

# Send deprecation notices to registered listeners.
config.active_support.deprecation = :notify

# Use default logging formatter so that PID and timestamp are not suppressed.
config.log_formatter = ::Logger::Formatter.new

# Do not dump schema after migrations.
config.active_record.dump_schema_after_migration = false
end

```

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 (3.3)
 - 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*

3.6.2

– ☒ 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
  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 created' }
        format.json { render :show, status: :created, location: @spreadsheet }
      else
        format.html { render :new }
        format.json { render json: @spreadsheet.errors, status: :unprocessable_entity }
      end
    end
  end
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: :unprocessable_entity }
          end
        end
      end

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

```

– ☒ 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 1.3.2

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

4. Custom Endpoints

3.4.1

- ☐ 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 Authentication

3.4 Views

3.4.1 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

3.4.2 Static Pages

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

3.5.1

1. Home

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

3.5 Controllers

3.5.1 Pages

Static pages controller

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

3.5.2 Spreadsheets

3.6 Models

3.6.1 Page

3.6.2 Spreadsheet

```
./app/models/spreadsheet.rb
```

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

3.7 TODO

- ☐ sidekiq
 - ☐ background processes for creating pdfs
- ☐ requirements
 - ☐ roo
 - ☐ chronic
- ☐ pdflatex
- ☐ migrate code from cj-parser
- ☐ user authentication
- ☐ file upload
 - ☐ only xlsx file?
 - ☐ AWS
- ☐ file storage
 - ☐ archival api?