Ruby On Rails

.

.

- ruby '2.5.1', rails '5.2.3' .
- gem 'carrierwave' .

1

- gem install carrierwave

2

- Gemfile gem 'carrierwave'
- bundle install

, , '/upload' API, '/import' API .

synapeditor.config.js

```
{
    'editor.import.api': '/import',
    'editor.upload.image.api': '/upload',
    'editor.upload.video.api': '/upload',
    'editor.upload.file.api': '/upload',
}
```

1.

```
> rails generate uploader File
```

./app/uploaders/file_uploader.rb

```
Class FileUploader < CarrierWave::Uploader::Base
# Include RMagick or MiniMagick support:
  # include CarrierWave::RMagick
  # include CarrierWave::MiniMagick


  # Choose what kind of storage to use for this uploader:
  storage :file
  # storage :fog

  # Override the directory where uploaded files will be stored.
  # This is a sensible default for uploaders that are meant to be mounted:
  def store_dir
    "uploads/"
  end

  # Provide a default URL as a default if there hasn't been a file uploaded:
  # def default_url(*args)
  #   # For Rails 3.1+ asset pipeline compatibility:
  #   # ActionController::Base.helpers.asset_path("fallback/" + [version_name, "default.png"].compact.join
('_'))
  #
  #   "/images/fallback/" + [version_name, "default.png"].compact.join('_')
  # end

  # Process files as they are uploaded:
  # process scale: [200, 300]
  #
  # def scale(width, height)
  #   # do something
  # end

  # Create different versions of your uploaded files:
  # version :thumb do
  #   process resize_to_fit: [50, 50]
  # end

  # Add a white list of extensions which are allowed to be uploaded.
  # For images you might use something like this:
  # def extension_whitelist
  #   %w(jpg jpeg gif png)
  # end

  # Override the filename of the uploaded files:
  # Avoid using model.id or version_name here, see uploader/store.rb for details.
  def filename
    "#{secure_token}.#{file.extension}" if original_filename.present?
  end

  protected
  def secure_token
    var = :"@#{mounted_as}_secure_token"
    model.instance_variable_get(var) or model.instance_variable_set(var, SecureRandom.uuid)
  end
end
```

2.

```
> rails generate model UploadFile
```

./app/models/upload_file.rb

```
class UploadFile < ApplicationRecord
        mount_uploader :file, FileUploader
end
```

3.

```
> rails generate controller UploadFile
```

./app/controller/upload_file_controller.rb

```
class UploadFileController < ApplicationController
        skip_before_action :verify_authenticity_token # csrf_token

        def upload
        uploaded = UploadFile.create(file: params[:file])


        render json:{
            "uploadPath": uploaded.file.url
        }
    end

    def import
        root_path = Rails.root

        uploaded = UploadFile.create(file: params[:file])

        # 1.
        input_file_path = uploaded.file.path
        input_file_name = File.basename(input_file_path)
        output_dir_name = File.basename(input_file_name, File.extname(input_file_name))
        converter_path = "%s/sedocConverter/sedocConverter_exe" % [root_path]
        font_dir_path = "%s/fonts" % [root_path]
        output_dir_path = "%s/public/output/%s" % [root_path, output_dir_name]
        tmp_dir_path = "%s/tmp" % [root_path]

        system("%s -pz -f %s %s %s %s" % [converter_path, font_dir_path, input_file_path, output_dir_path,
tmp_dir_path])

        # 2. PB
        pb_file_path = "%s/document.pb" % [output_dir_path]
        aFile = File.open(pb_file_path, "r")
        aSerialized = Array.new

        if aFile
            aFile.sysread(16)
            aFile.each_byte do |byte|
                aSerialized.push(byte & 0xFF)
            end
        end

        # 3.
        File.delete(input_file_path) if File.exist?(input_file_path)
        File.delete(pb_file_path) if File.exist?(pb_file_path)

        render json: {
            "importPath": "/output/%s" % [File.basename(output_dir_path)],
            "serializedData": aSerialized
        }
    end
end
```

4.

```
# UploadFile   'file'
> rails generate migration AddFileToUploadFile file:string
> rake db:migrate
```

5.

./config/routes.rb

```
Rails.application.routes.draw do
        post 'import' => 'upload_file#import'
        post 'upload' => 'upload_file#upload'
end
```

- Java Spring Framework
- Java Servlet
- ASP.NET (C#)
- ASP(Classic)
- PHP
- PHP4
- Django
- Ruby On Rails
- Wordpress plugin