require 'act\_as\_page\_extractor/version'

require 'active\_record'

require 'awesome\_print'

require 'filesize'

require 'total\_compressor'

require 'docsplit'

require 'pdf\_utils'

require 'prawn'

require 'pdf-reader'

require 'act\_as\_page\_extractor/modules/tools.rb'

require 'act\_as\_page\_extractor/modules/validating.rb'

require 'act\_as\_page\_extractor/modules/unzipping.rb'

require 'act\_as\_page\_extractor/modules/extracting.rb'

require 'act\_as\_page\_extractor/modules/saving.rb'

require 'act\_as\_page\_extractor/modules/interface'

module ActAsPageExtractor

extend ActiveSupport::Concern

included do

before\_create { self.page\_extraction\_state = EXTRACTING\_STATES[:new] }

before\_destroy :remove\_files

end

# attr\_reader :options

module ClassMethods

def act\_as\_page\_extractor(options: {})

define\_method(:save\_as\_pdf){|\*args| options[:save\_as\_pdf] }

define\_method(:extracted\_filename){|\*args| self.send(options[:filename].to\_sym) }

ActAsPageExtractor.define\_singleton\_method(:extracted\_filename) {|\*args| options[:filename] }

ActAsPageExtractor.define\_singleton\_method(:document\_class) {|\*args| options[:document\_class].constantize }

define\_method(:extracted\_document\_id){|\*args| options[:document\_id] }

define\_method(:additional\_fields){|\*args| options[:additional\_fields] }

end

end

EXTRACTING\_STATES = {

new: 'new',

extracting: 'extracting',

extracted: 'extracted',

'error.extraction': 'error.extraction'

}.freeze

TMP\_EXTRACTION\_FILE\_STORAGE = "#{Dir.pwd}/tmp/page\_extraction".freeze

FILE\_STORAGE = "#{Dir.pwd}/public".freeze

PDF\_STORAGE = "#{FILE\_STORAGE}/uploads/extracted/pdf".freeze

def initialized

# add all need callbacks

#on destroy remove pdf

#Add to Readme!!

#rails g act\_as\_page\_extractor:migration Document category\_id user\_id

# add to [Document] model:

# has\_many :extracted\_pages, dependent: :destroy

create\_pdf\_dir

end

def page\_extract!

initialized

cleanup\_pages

create\_tmp\_dir

begin

copy\_document

# debug\_info

unzip\_document

if valid\_document

extract\_pages

save\_to\_db

end

ensure

update\_state

save\_pdf

finish

end

end

def create\_pdf\_dir

if save\_as\_pdf

FileUtils::mkdir\_p(PDF\_STORAGE) unless File.exists?(PDF\_STORAGE)

end

end

def create\_tmp\_dir

@tmp\_dir = "#{TMP\_EXTRACTION\_FILE\_STORAGE}/#{SecureRandom.hex(6)}"

FileUtils::mkdir\_p(@tmp\_dir) unless File.exists?(@tmp\_dir)

end

def copy\_document

@origin\_document\_path = "#{FILE\_STORAGE}#{self.send(:extracted\_filename).url.to\_s}"

ap @origin\_document\_path

FileUtils.cp(@origin\_document\_path, @tmp\_dir)

@copy\_document\_path = "#{@tmp\_dir}/#{@origin\_document\_path.split("/").last}"

@document\_filename = @origin\_document\_path.split("/").last

end

def finish

remove\_tmp\_dir

end

def remove\_tmp\_dir

FileUtils.rm\_rf(@tmp\_dir) if @tmp\_dir =~ /\/tmp\//

end

end

# rails g model ExtractedPage page:text document\_id:integer category\_id:integer page\_number:integer

# Rails 4 way

# 9.2.7.1 Multiple Callback Methods in One Class

# 258 page

# class ActiveRecord::Base

# def self.acts\_as\_page\_extractor(document\_field=:filename)

# auditor = Auditor.new(audit\_log)

# after\_create auditor

# after\_update auditor

# after\_destroy auditor

# end

# end