Modern image uploads

AT HLIDACKY.CZ

About me

Name: Ondřej Žádník

Ruby on Rails programmer at PrimeHammer

Currently working on project Hlidacky.cz

Computer science student

Hlídačky.cz

- #1 marketplace for babysitters and cleaners in Czech Republic
- 73 330 registered users, over 5400 active babysitters

Motivation

- Profile photos of babysitters/parents and image galleries
- S3 uploads
- Direct upload to S3
- Cropping(ImageMagick)
- Create versions
- Processing in background jobs

Available options

- Carrierwave – not multiple direct S3 uploads, old gems(cw backgrounder), unsecure
- Paperclip – doesn‘t have S3 direct uploads
- Dragonfly – no background jobs support
- Refile – no background jobs support
- Filestack – great solution but too expensive for startup

New way? Shrine!

- Ruby library for handling image uploads
- Good design (storages, plugins)
- Upload options
- Inspired by Refile but with upfront processing
- Inspired by CarrierWave's uploaders

Simplicity&Flexibility

- Plugin system – small core with essential functionality, other features as plugins
- Shrine ships with over 25 plugins
- not complex DSL
- Simple validations
- CW, Refile and Paperclip 9-12 depedencies, Shrine only 1 dependency for downloading file

Simplicity&Flexibility

```ruby
Shrine.plugin :activerecord # :activerecord
Shrine.plugin :cached_attachment_data # for forms
Shrine.plugin :direct_upload, presign: true
Shrine.plugin :remote_url, max_size: 6*1024*1024
Shrine.plugin :pretty_location
Shrine.plugin :determine_mime_type
Shrine.plugin :store_dimensions

# we do not want to reveal validations to the user
Shrine.plugin :validation_helpers, default_messages: {
  max_size: ->(max) { I18n.t("errors.file.max_size") },
  mime_type_inclusion: ->(whitelist) { I18n.t("errors.file.mime_type_inclusion") },
  mime_type_exclusion: ->(blacklist) { I18n.t("errors.file.mime_type_exclusion") },
  extension_inclusion: ->(list) { I18n.t("errors.file.extension_inclusion") },
}
Shrine.plugin :delete_raw, storages: [:cache]

Shrine.plugin :backgrounding
Shrine.plugin :processing
Shrine.plugin :versions
Shrine.plugin :hooks

Shrine::Attacher.promote { |data| Resque.enqueue(PromoteJob, data)}
Shrine::Attacher.delete { |data| Resque.enqueue(ShrineDeleteJob, data)}
```

Performance&Security

- moving file instead of copying them for large files
- parallelize plugin
- designed for backgrounding
- support for direct uploads(jquery file upload)
- processing before storing after validation
- CW processing *before* validation, security issue(DoS upload large files)

Performance&Security

```ruby
class PromoteJob
  extend Resque::Plugins::Logger
  @queue = :shrine

  def self.perform(data)
    require "pusher"
    ActiveRecord::Base.connection_pool.with_connection do
      Shrine::Attacher.promote(data)
    end
  end
end
```

```ruby
class Image < ActiveRecord::Base
  include ImageUploaderShrine[:image]

  acts_as_paranoid

  belongs_to :imageable, polymorphic: true

  MIN_SIZE = 1024
  MAX_SIZE = 6*1024*1024

end
```

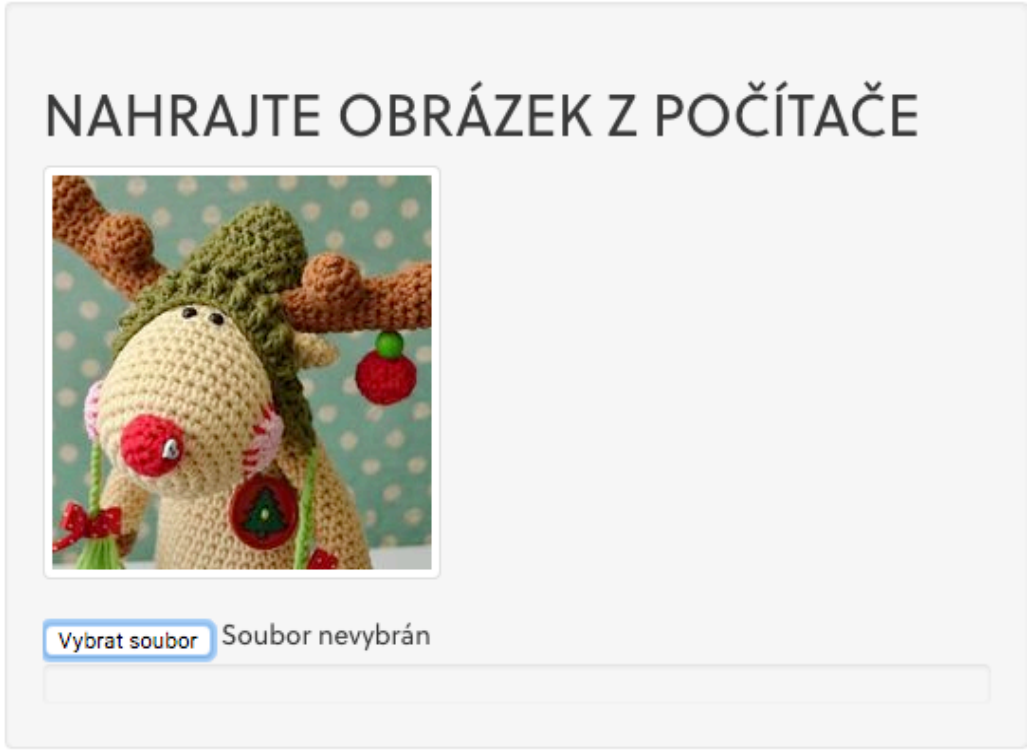
Uploader

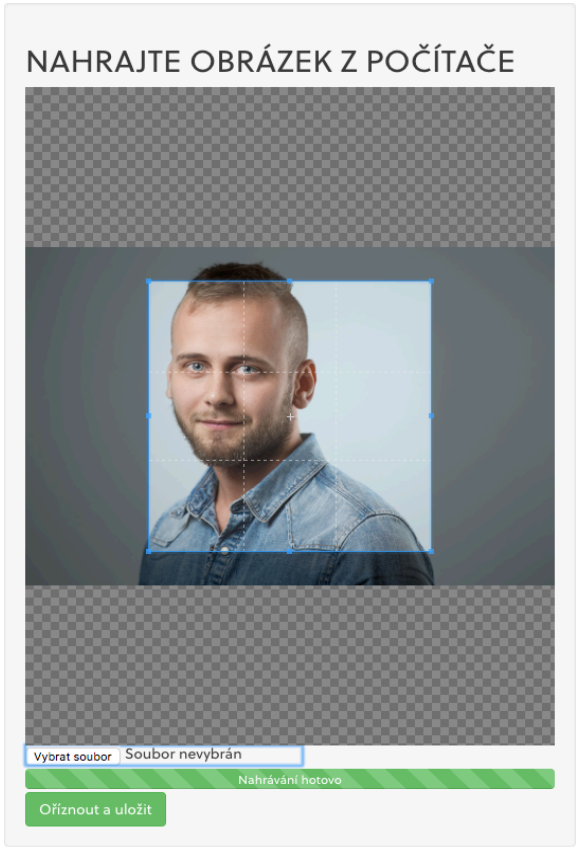
```ruby
class ImageUploaderShrine < Shrine

  Attacher.validate do
    validate_max_size 8.megabytes
    validate_mime_type_inclusion ['image/jpg', 'image/jpeg', 'image/png']
    validate_min_height 400
    validate_min_width 400
  end

  process(:store) do |io, context|
    require "image_processing/mini_magick"

    converted = ImageProcessing::MiniMagick.convert(io.download, "jpg")
    cropping_args = io.metadata['cropping'].values_at('width', 'height', 'x', 'y')

    cropped = ImageProcessing::MiniMagick.crop(converted, *cropping_args, gravity: "NorthWest")
    size_400 = ImageProcessing::MiniMagick.resize_to_fit(cropped,    400, 400)
    size_200 = ImageProcessing::MiniMagick.resize_to_fit(size_400,    200, 200)

    {original: converted, cropped_original: cropped, medium: size_400, small: size_200}
  end

  def after_upload(io, context)
    begin
      Pusher.trigger("processing_complete_signal_#{context[:record][:imageable_id]}", 'processing_complete', { message: 'ok' })
    rescue Pusher::Error
      Rollbar.error('Failed to send callback after image processing with Pusher.')
    end
  end
end
```

Hlidacky.cz image upload flow

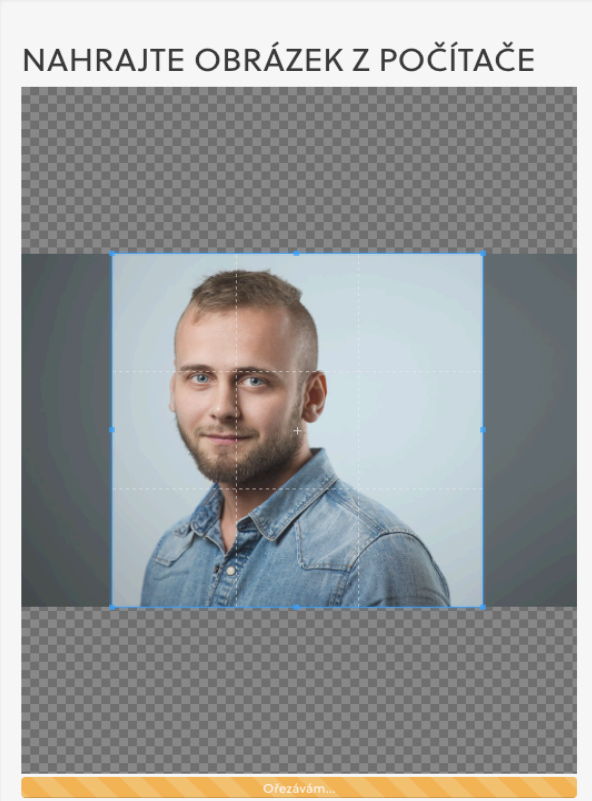
- Direct upload to S3 -> return url of uploaded image(jquery file upload)
- Download image from url -> load cropper.js on it
- Crop -> processed versions -> stored on S3
- Pusher receives callback from uploader and redirect

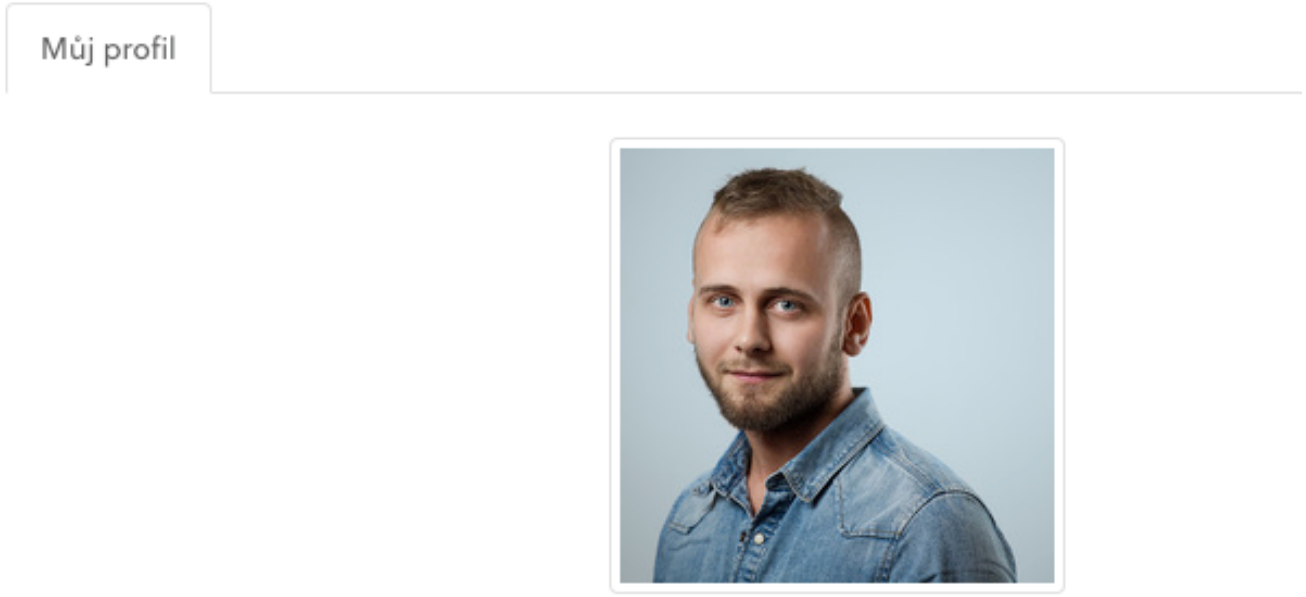
Hlidacky.cz image upload flow





Hlidacky.cz image upload flow





References

http://shrinerb.com/

https://twin.github.io/shrine-2-0-released/

https://twin.github.io/file-uploads-asynchronous-world/

Thank you