JOHN MOSESMAN

DEVELOPER AT LIFECHURCH.TV

HOW I LEVELED UP MY CODE

SCENARIO

INSTAGRAM PHOTOS SIMILAR TO MY INTERESTS.

SCALE OF 1 TO DHH

HOW WELL DO YOU KNOW RAILS?

```
rails new me
```

LEVEL 1

MINIMAL / NO STRUCTURE

```ruby
# app/controllers/photos_controller.rb
class PhotosController < ApplicationController
  def index
    @photos =
      if params[:tag].present?
        client = Instagram.setup_client("some_key", "some_secret")
        tagged_photos = client.photos_tagged(params[:tag])

        tagged_photos.select do |photo|
          # Do a lot things here:
          # Check bio, location, posts,
          # hashtags, etc.
        end
      else
        []
      end
  end
end
```

NOT GREAT
BUT HEY, IT WORKS.

LEVEL 2

FAT MODELS, SKINNY CONTROLLERS

(DONE POORLY)

I USED THE WORD USER

SO IT MUST BELONG IN THE USER MODEL?

```ruby
# app/models/user.rb
class User < ActiveRecord::Base
  before_save :do_something
  before_validation :another_thing

  devise :database_somethingable, :anotherthingable,
         :differentable, :elsable, :ableable

  has_many :something_elses
  has_many :somethings, through: :something_elses

  scope :some_scope, -> { where(scope: true) }

  def self.some_class_thing
    # ...
  end

  def something
    # ...
  end

  def something_else
    # ...
  end

  def something_different
    # ...
  end
end
```

```ruby
# app/models/user.rb
class User < ActiveRecord::Base
  before_save :do_something
  before_validation :another_thing

  devise :database_somethingable, :anotherthingable,
         :differentable, :elsable, :ableable

  has_many :something_elses
  has_many :somethings, through: :something_elses

  scope :some_scope, -> { where(scope: true) }

  def self.some_class_thing
    # ...
  end

  def something
    # ...
  end

  def something_else
    # ...
  end

  def something_different
    # ...
  end

  def similar_photos(tag)
    if tag.present?
      client = Instagram.setup_client("some_key", "some_secret")
      tagged_photos = client.photos_tagged(tag)

      tagged_photos.select do |photo|
        # Do a lot things here:
        # Check bio, location, posts,
        # hashtags, etc.
      end
    else
      []
    end
  end
end
```

```ruby
# app/controllers/photos_controller.rb
class PhotosController < ApplicationController
  def index
    @photos =
      if params[:tag].present?
        client = Instagram.setup_client("some_key", "some_secret")
        tagged_photos = client.photos_tagged(params[:tag])

        tagged_photos.select do |photo|
          # Do a lot things here:
          # Check bio, location, posts,
          # hashtags, etc.
        end
      else
        []
      end
  end
end
```

```ruby
# app/controllers/photos_controller.rb
class PhotosController < ApplicationController
  before_filter :authenticate_user!

  def index
    @photos = current_user.similar_photos(params[:tag])
  end
end
```

```ruby
# app/models/user.rb
class User < ActiveRecord::Base
  ...

  def similar_photos(tag)
    if tag.present?
      client = Instagram.setup_client("some_key", "some_secret")
      tagged_photos = client.photos_tagged(tag)

      tagged_photos.select do |photo|
        # Do a lot things here:
        # Check bio, location, posts,
        # hashtags, etc.
      end
    else
      []
    end
  end
end
```

LEVEL 3

SINGLE RESPONSIBILITY PRINCIPLE

TO PARAPHASE WIKIPEDIA:
"EACH CLASS SHOULD ONLY DO ONE THING."

```ruby
# app/models/user.rb
class User < ActiveRecord::Base
  ...

  def similar_photos(tag)
    if tag.present?
      client = Instagram.setup_client("some_key", "some_secret")
      tagged_photos = client.photos_tagged(tag)

      tagged_photos.select do |photo|
        # Do a lot things here:
        # Check bio, location, posts,
        # hashtags, etc.
      end
    else
      []
    end
  end
end
```

```
class User < ActiveRecord::Base
```

ActiveRecord::Bas

ENTER:
SERVICE OBJECTS

```ruby
# app/services/photo_service.rb
class PhotoService
  def self.similar(user, tag)
    if tag.present?
      client = Instagram.setup_client("some_key", "some_secret")
      tagged_photos = client.photos_tagged(tag)

      tagged_photos.select do |photo|
        # Do a lot things here:
        # Check bio, location, posts,
        # hashtags, etc.
      end
    else
      []
    end
  end
end
```

```ruby
class PhotosController < ApplicationController
  before_filter :authenticate_user!

  def index
    @photos = PhotoService.similar(current_user, params[:tag])
  end
end
```

LEVEL 4??

@JOHNMOSESMAN