

# Growing Rails Apps

with ❤️ from datarockets

# Easy to start

- Requirements are simple
- First steps are simple





# Hard to maintain

- Edge cases start to appear
- New features are more complicated
- Codebase become bigger and harder to maintain



# We use the wrong tools

- Callbacks in models and controllers and suppressing them
- Fat models
- Default scopes
- Custom actions in controllers









# We have better tools

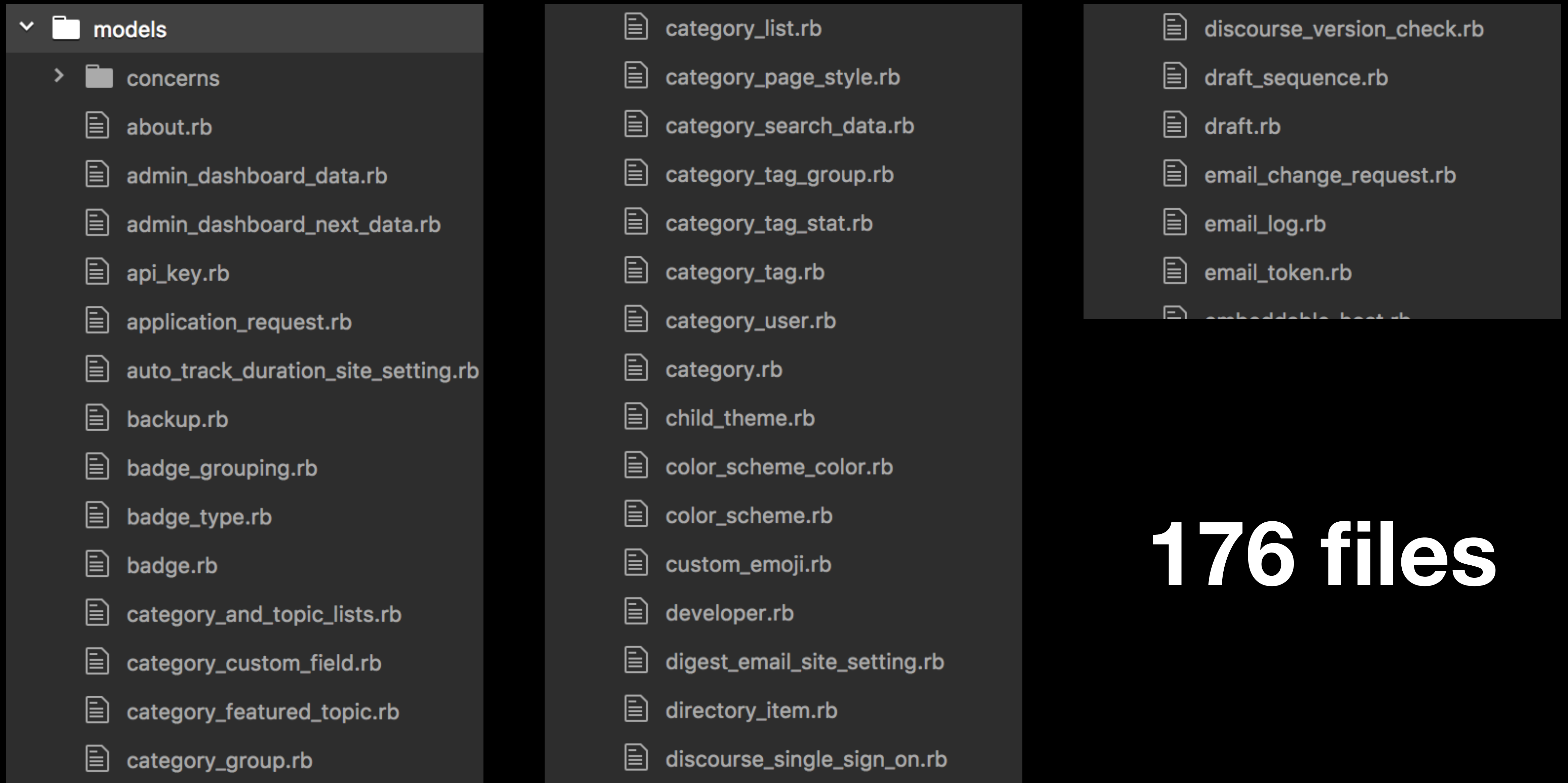
- Form Objects
- Service Objects
- Trailblazer
- DRY framework
- ROM



# Code Organization




# Code Organization



**176 files**





176?



# Flat Models Structure

```
post_action_type.rb  
post_action.rb  
post_analyzer.rb  
post_custom_fields.rb  
post_detail.rb  
post_mover.rb  
post_reply.rb  
post_revision.rb  
post_search_data.rb  
post_stat.rb  
post_timing.rb  
post_upload.rb  
post.rb
```



# Flat Models Structure

```
post_action_type.rb
post_action.rb
post_analyzer.rb
post_custom_fields.rb
post_detail.rb
post_mover.rb
post_reply.rb
post_revision.rb
post_search_data.rb
post_stat.rb
post_timing.rb
post_upload.rb
post.rb
```

```
class Post < ActiveRecord::Base
  has_many :post_uploads
  has_many :uploads, through: :post_
  has_many :post_details
  has_many :post_revisions
  # ...
end
```

# Flat Models Structure

```
post_action_type.rb
post_action.rb
post_analyzer.rb
post_custom_fields.rb
post_detail.rb
post_mover.rb
post_reply.rb
post_revision.rb
post_search_data.rb
post_stat.rb
post_timing.rb
post_upload.rb
post.rb
```

```
class Post < ActiveRecord::Base
  has_many :post_uploads
  has_many :uploads, through: :post_
  has_many :post_details
  has_many :post_revisions
  # ...
end
```

```
post.post_details
```

```
post.post_revisions
```

# Namespace Your Models

```
class Post
  has_many :details
  has_many :uploads, class_name: "Post::Upload"
end
```

```
class Post::Detail # post/detail.rb
  belongs_to :post
end
```

```
class Post::Upload # post/upload.rb
  belongs_to :post
  belongs_to :upload, class_name: "::Upload"
end
```



# Namespace Your Models

```
post_action_type.rb
post_action.rb
post_analyzer.rb
post_custom_fields.rb
post_detail.rb
post_mover.rb
post_reply.rb
post_revision.rb
post_search_data.rb
post_stat.rb
post_timing.rb
post_upload.rb
post.rb
```



```
post/
  action/
    type.rb
  action.rb
  analyzer.rb
  custom_fields.rb
  detail.rb
  mover.rb
  reply.rb
  revision.rb
  search_data.rb
  stat.rb
  timing.rb
  upload.rb
post.rb
```

# Namespace Your Models

```
post/  
post.rb
```

# Controllers

```
$ cat app/controllers/users_controller.rb | wc -l
```

```
1097
```



1097?





# Custom Actions

```
resources :users do
  put "suspend"
  put "unsuspend"

  put "grant_moderation"
  put "revoke_moderation"

  # ...
end
```

# Resourceful Routes

```
resources :users do  
  put "suspend"  
  put "unsuspend"  
end
```



```
resources :users do  
  resource :suspension,  
    only: [:create, :destroy]  
end
```

# Resourceful Routes

```
resources :users do  
  put "suspend"  
  put "unsuspend"  
end
```




```
resources :users do  
  resource :suspension,  
    only: [:create, :destroy]  
end
```

```
DELETE /users/:user_id/suspension suspensions#destroy  
POST   /users/:user_id/suspension suspensions#create
```



# Resourceful Routes

```
resources :users do  
  put "suspend"  
  put "unsuspend"  
end
```



```
resources :users do  
  resource :suspension,  
    only: [:create, :destroy]  
end
```

```
DELETE /users/:user_id/suspension suspensions#destroy  
POST   /users/:user_id/suspension suspensions#create
```

```
app/controllers/  
  users_controller.rb  
  suspensions_controller.rb
```

# Resourceful Routes

```
resources :users do  
  put "suspend"  
  put "unsuspend"  
end
```



```
resources :users do  
  resource :suspension,  
    only: [:create, :destroy]  
end
```

```
DELETE /users/:user_id/suspension suspensions#destroy  
POST   /users/:user_id/suspension suspensions#create
```

```
app/controllers/  
  users_controller.rb  
  suspensions_controller.rb
```



# Resourceful Routes

```
resources :users do
  scope module: :users do
    resource :suspension, only: [:create, :destroy]
    resource :moderation_permission, only: [:create, :destroy]
  end
end
```

```
app/controllers/
  users/
    suspensions_controller.rb
    moderator_permissions_controller.rb
  users_controller.rb
```

# Long Actions

```
def index
  @rss_posts = posts.map do |post|
    { description: rss_description_for(post), title: post.title }
  end
end

def rss_description_for(post)
  post.body.truncate(256) + info_about_likes(post)
end

def info_about_likes(post)
end
```

# Long Actions

```
def index  
  @rss_posts = RssPosts.new(posts)  
end
```



```
class RssPosts
  def each
    @posts.each do |post|
      yield { description: rss_description_for(post), title: p
    end
  end

  private

  def rss_description_for(post)
    post.body.truncate(256) + info_about_likes(post)
  end

  def info_about_likes(post)
  end
end
```

```
class RssPosts
  def each
    @posts.each do |post|
      yield { description: rss_description_for(post), title: p
    end
  end

  private

  def rss_description_for(post)
    post.body.truncate(256) + info_about_likes(post)
  end

  def info_about_likes(post)
  end
end
```





```
class RssPosts # rss_posts.rb
  def each
    @posts.each do |post|
      yield Post.new(post)
    end
  end
end
```

```
class RssPosts::Post # rss_posts/post.rb
  def description
    @post.body + info_about_likes
  end

  def info_about_likes
  end
end
```



# Organize Your Code

```
rss_posts/  
  post.rb  
rss_posts.rb
```



**“It makes no sense to create one more class and one more file instead of creating one private method”**

*– Your Coworker*

# Sandi Metz Rules

1. Classes can be no longer than one hundred lines of code.
2. Methods can be no longer than five lines of code.
3. Pass no more than four parameters into a method. Hash options are parameters.
4. Controllers can instantiate only one object. Therefore view can only know about one instance variable and views should only send messages to that object.



# Dima's Rule

Don't send parameters to private methods. Consider creating a separate class.



# POODR





# Quick Recap

- Namespace models
- Namespace controllers and keep them small
- Namespace service classes and keep them clean



# NAMESPACE





# Thank you!

[twitter.com/dzhlobo](https://twitter.com/dzhlobo)

[fb.com/dima.zhlobo](https://fb.com/dima.zhlobo)

[vk.com/datarockets](https://vk.com/datarockets)

with ❤️ from datarockets