Rails Best Practices

Владимир Пенкин
@mindwork
http://shell.github.com

План

- Что такое хороший код
- Распределение кода
- RESTful best practices
- Модель
- Контролер
- Представление

Перед применением

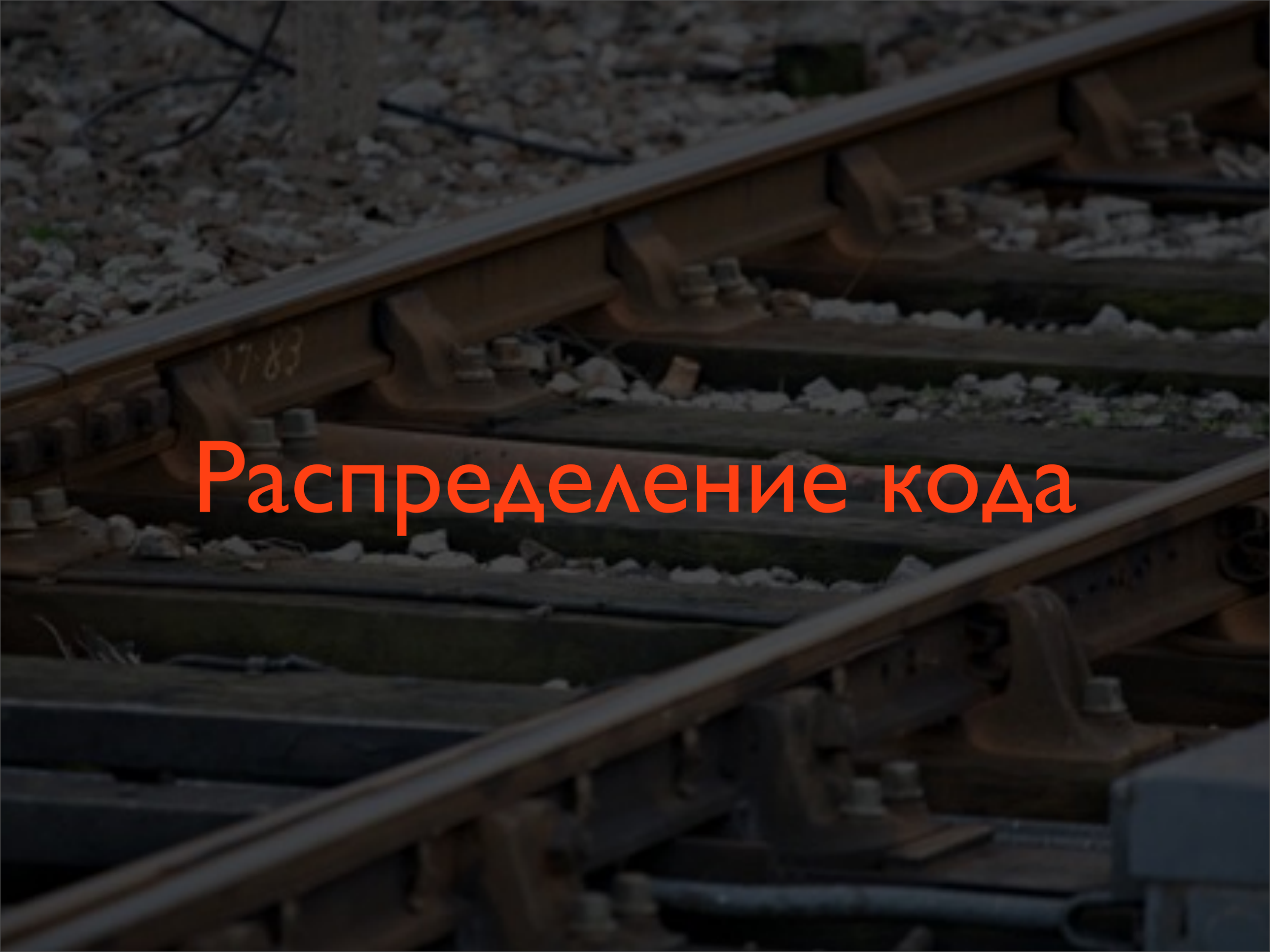
- Внимание! Тестируйте перед тем как изменять!

Зачем всё это?

- Большие приложения
- Командная разработка, разные стили написания кода

Что такое хороший код?

- Читабельный
- Гибкий
- Эффективный
- Поддерживаемый
- Согласованный
- Тестируемый



Распределение кода

Применение named_scope

Плохо!

```
class PostsController < ApplicationController
  def index
    @public_posts = Post.find(:all, :conditions => { :state => 'public' },
                              :limit => 10, :order => 'created_at desc')
    @draft_posts = Post.find(:all, :conditions => { :state => 'draft' },
                              :limit => 10, :order => 'created_at desc')
  end
end
```

Применение named_scope

Хорошо!

```
class UsersController < ApplicationController
  def index
    @published_post = Post.published
    @draft_post = Post.draft
  end
end

class Post < ActiveRecord::Base
  named_scope :published, :conditions => { :state => 'published' },
                          :limit => 10, :order => 'created_at desc')
  named_scope :draft, :conditions => { :state => 'draft' },
                          :limit => 10, :order => 'created_at desc')
end
```

Виртуальные аттрибуты

Плохо!

```
<% form_for @user do |f| %>
  <%= text_filed_tag :full_name %>
<% end %>
```

```
class UsersController < ApplicationController
  def create
    @user = User.new(params[:user])
    @user.first_name = params[:full_name].split(' ', 2).first
    @user.last_name = params[:full_name].split(' ', 2).last
    @user.save
  end
end
```

Виртуальные аттрибуты

Хорошо!

```
class User < ActiveRecord::Base
  def full_name
    [first_name, last_name].join(' ')
  end
  def full_name=(name)
    split = name.split(' ', 2)
    self.first_name = split.first
    self.last_name = split.last
  end
end
```

Логика модели

Плохо!

```
class PostController < ApplicationController
  def publish
    @post = Post.find(params[:id])
    @post.update_attribute(:is_published, true)
    @post.approved_by = current_user
    if @post.create_at > Time.now - 7.days
      @post.popular = 100
    else
      @post.popular = 0
    end
    redirect_to post_url(@post)
  end
end
```

Логика модели

Хорошо!

```
class Post < ActiveRecord::Base
  def publish
    self.is_published = true
    self.approved_by = current_user
    if self.create_at > Time.now-7.days
      self.popular = 100
    else
      self.popular = 0
    end
  end
end
```

Формы вложенных моделей

Плохо!

```
class Product < ActiveRecord::Base
  has_one :detail
end

class Detail < ActiveRecord::Base
  belongs_to :product
end

<% form_for :product do |f| %>
  <%= f.text_field :title %>
  <% fields_for :detail do |detail| %>
    <%= detail.text_field :manufacturer %>
  <% end %>
<% end %>
```

Формы вложенных моделей

Плохо!

```
class Product < ApplicationController
  def create
    @product = Product.new(params[:product])
    @details = Detail.new(params[:detail])
    Product.transaction do @product.save!
      @details.product = @product
      @details.save!
    end
  end
end
```

Вложенные формы

Хорошо!

```
class Product < ActiveRecord::Base
  has_one :detail
  accepts_nested_attributes_for :detail
end

<% form_for :product do |f| %>
  <%= f.text_field :title %>
  <% f.fields_for :detail do |detail| %>
    <%= detail.text_field :manufacturer %>
  <% end %>
<% end %>

class Product < ApplicationController
  def create
    @product = Product.new(params[:product])
    @product.save
end end
```

RESTful

- Помогает организовать контроллеры, действия и пути стандартным образом

Контроллер

Плохо!

```
class EventsController < ApplicationController


  def index
  end

  def show
  end

  def create
  end

  def update
  end

  def destroy
  end

  def feeds
  end

  def add_comment
  end

  def show_comment
  end

  def destroy_comment
  end

  def edit_comment
  end

  def approve_comment
  end

  def white_member_list
  end

  def black_member_list
  end

  def deny_user
  end

  def allow_user
  end

  def edit_managers
  end

  def set_user_as_manager
  end

  def set_user_as_member
  end

  def watch_list
  end

  def add_favorite
  end

  def invite
  end

  def join
  end

  def leave
  end


end
```

Контроллер

Хорошо !

```
class EventsController < ApplicationController
  def index; end
  def show; end
end
class CommentsControlers < ApplicationController
  def index; end
  def create; end
  def destroy; end
end
def FavoriteControllers < ApplicationController
  def create; end
  def destroy; end
end
class EventMembershipsControlers < ApplicationController
  def create; end
  def destroy; end
end
```

Routing

Плохо!

```
match '/terms'    => 'pages#terms'
match '/company'  => 'pages#company'
match '/faq'      => 'pages#faq'
match '/news'     => 'pages#news'
match '/static'   => 'pages#static'
match '/static'   => 'pages#static'
```

Routing

Хорошо !

```
PagesController.action_methods.each do |action|
 hash = { "/#{action}" => "pages##{action}"}
 match hash
end
```

Модель

named_scope

Плохо!

```
class PostController < ApplicationController
  def search
    conditions = { :title => "%#{params[:title]}%" } if params[:title]
    conditions.merge!{ :content => "%#{params[:content]}%" } if
params[:content]
    case params[:order]
      when "title" : order = "title desc"
      when "created_at" : order = "created_at"
    end
    if params[:is_published]
      conditions.merge!{ :is_published => true }
    end
    @posts = Post.find(:all, :conditions => conditions, :order => order,
                       :limit => params[:limit])
  end
end
```

named_scope

Хорошо!

```
class Post < ActiveRecord::Base
  named_scope :matching, lambda { |column, value|
    return {} if value.blank?
    { :conditions => ["#{column} like ?", "%#{value}%"] }
  }
  named_scope :order, lambda { |order|
    { :order => case order
      when "title" : "title desc"
      when "created_at" : "created_at"
    end }
  }
end
```

named_scope

Хорошо !

```
class PostController < ApplicationController
  def search
    @posts = Post.matching(:title, params[:title])
                 .matching(:content, params[:content])
                 .order(params[:order])
  end
end
```

Метапрограммирование

Плохо!

```ruby
class Post < ActiveRecord::Base
  validate_inclusion_of :status, :in => ['draft', 'published', 'spam']
  def self.all_published
    find(:all, :conditions => { :status => 'published' }
  end
  def self.all_spam
    find(:all, :conditions => { :status => 'spam' }
  end
  def published?
    self.stats == 'published'
  end
  def spam?
    self.stats == 'spam'
  end
end
```

Метапрограммирование

Хорошо !

```ruby
class Post < ActiveRecord::Base
  STATUSES = ['draft', 'published', 'spam']
  validate_inclusion_of :status, :in => STATUSES
  class << self
    STATUSES.each do |status_name|
      define_method "all_#{status}" do
        find(:all, :conditions => { :status => status_name }
      end
    end
  end
  STATUSES.each do |status_name|
    define_method "#{status_name}?" do
      self.status == status_name
    end
  end
end
```

Упрощение модели

Модули

Плохо!

```
class User < ActiveRecord::Base
  validates_presence_of :cellphone
  before_save :parse_cellphone
  def parse_cellphone
    # do something
  end
end
```

Модули

Хорошо !

```
class User < ActiveRecord::Base
  Include HasCellphone
end
```

Observer

Плохо!

```
class Project < ActiveRecord::Base
  after_create :send_create_notifications
  private
    def send_create_notifications
      self.members.each do |member|
        ProjectNotifier.deliver_notification(self, member)
      end
    end
end
```

Observer

Хорошо !

```
class Project < ActiveRecord::Base # nothing here
end

# app/observers/project_notification_observer.rb
class ProjectNotificationObserver < ActiveRecord::Observer
  observe Project
  def after_create(project)
    project.members.each do |member|

      ProjectMailer.deliver_notice(project, member)
    end
  end
end
```

Controller

Фильтры

Плохо!

```
class PostController < ApplicationController
  def show
    @post = current_user.posts.find(params[:id]
  end

  def edit
    @post = current_user.posts.find(params[:id]
  end

  def update
    @post = current_user.posts.find(params[:id]
    @post.update_attributes(params[:post])
  end
end
```

Фильтры

Хорошо, но не очень

```
class PostController < ApplicationController
  before_filter :find_post, :only => [:show, :edit, :update, :destroy]

  def update
    @post.update_attributes(params[:post])
  end

  def destroy
    @post.destroy
  end

  protected
    def find_post
      @post = current_user.posts.find(params[:id])
    end
end
```

Фильтры

Лучше

```ruby
class PostController < ApplicationController
  before_filter :authenticate

  def update
    find_post
    @post.update_attributes(params[:post])
  end

  def destroy
    find_post
    @post.destroy
  end

  protected
    def find_post
      @post = current_user.posts.find(params[:id])
    end
end
```

Представление

Instance variable vs local variable

Плохо!

```
class Post < ApplicationController def show
  @post = Post.find(params[:id) end
end

<%= render :partial => "sidebar" %>
```

Хорошо!

```
<%= render :partial => "sidebar", :locals => { :post => @post } %>
```

Tweaks

Hash to Object

Плохо!

```
class ::Hash

  # add keys to hash
  def to_obj
    self.each do |k,v|
      if v.kind_of? Hash
        v.to_obj
      end
      k=k.gsub(/\.|\s|-|\/|\'/, '_').downcase.to_sym
      self.instance_variable_set("@#{k}", v)
      self.class.send(:define_method, k,
        proc{self.instance_variable_get("@#{k}")})
      self.class.send(:define_method, "#{k}=", proc{|v|
        self.instance_variable_set("@#{k}", v)})
    end
    return self
  end
end
```

Hash to Object

Хорошо !

```ruby
require 'ostruct'

hash = { "country" => "Australia", :population => 20_000_000 }
data = OpenStruct.new(hash)

p data          # -> <OpenStruct country="Australia" population=20000000>
```

Ссылки

- http://blog.davidchelimsky.net/wp-content/uploads/2010/11/duplication.pdf
- http://railscasts.com
- http://rails-bestpractices.com
- http://refactormycode.com/

Спасибо за внимание

http://bit.ly/shell_rbp

(осторожно рекурсия!)

Владимир Пенкин

@mindwork

http://shell.github.com