Лекция 9

План

- Роутинг
- Контроллеры
- Шаблонизаторы

config/database.yml

```yaml
default: &default
  adapter: postgresql
  encoding: unicode

development:
  <<: *default
  database: r6_development
  username: username
  password: password

production:
  <<: *default
  database: r6_production
  username: r6
  password: <% ENV['R6_DATABASE_PASSWORD'] %>
```

rake для БД

```
db: create
db: create: all
db: drop
db: drop: all
```

Запуск сервера

- `rails s`
- `rails server`

Роутинг

```ruby
Rails.application.routes.draw do
  # For details on the DSL available within this file,
  #see https://guides.rubyonrails.org/routing.html
end
```

```ruby
Rails.application.routes.draw do
  get 'home/index'
  root 'home#index'
end
```

HTTP Verb	Path	Controller#Action	Used for
GET	/photos	photos#index	display a list of all photos
GET	/photos/new	photos#new	return an HTML form for creating a new photo
POST	/photos	photos#create	create a new photo
GET	/photos/:id	photos#show	display a specific photo
GET	/photos/:id/edit	photos#edit	return an HTML form for editing a photo
PATCH/PUT	/photos/:id	photos#update	update a specific photo
DELETE	/photos/:id	photos#destroy	delete a specific photo

Resources

```
resources :photos, :books, :videos
```

```
resources :photos
resources :books
resources :videos
```

Одиночные ресурсы

```
get 'profile', to: 'users#show'

get 'profile', action: :show,
               controller: 'users'
```

resource :geocoder

HTTP Verb	Path	Controller#Action	Used for
GET	/geocoder/new	geocoders#new	return an HTML form for creating the geocoder
POST	/geocoder	geocoders#create	create the new geocoder
GET	/geocoder	geocoders#show	display the one and only geocoder resource
GET	/geocoder/edit	geocoders#edit	return an HTML form for editing the geocoder
PATCH/PUT	/geocoder	geocoders#update	update the one and only geocoder resource
DELETE	/geocoder	geocoders#destroy	delete the geocoder resource

Пространства имен

/admin/articles

```ruby
namespace :admin do
  resources :articles, :comments
end
```

/articles

```ruby
scope module: 'admin' do
  resources :articles, :comments
end
```

Вложенные ресурсы

```
resources :magazines do
  resources :ads
end
```

```
resources :publishers do
  resources :magazines do
    resources :photos
  end
end
```

HTTP Verb	Path	Controller#Action	Used for
GET	/magazines /:magazine_id/ads	ads#index	display a list of all ads for a specific magazine
GET	/magazines /:magazine_id/ads/new	ads#new	return an HTML form for creating a new ad belonging to a specific magazine
POST	/magazines /:magazine_id/ads	ads#create	create a new ad belonging to a specific magazine

Ограничение методов

```
resources :photos, except: :destroy
resources :photos, only: [:index, :show]
```

Мелкие ресурсы (а не глубокие)

```
resources :articles do
  resources :comments,
                    only: [:index, :new, :create]
end

resources :comments, only:
                  [:show, :edit, :update, :destroy]
```

DSL для мелких

```
resources :articles do
  resources :comments, shallow: true
end

resources :articles, shallow: true do
  resources :comments
  resources :quotes
  resources :drafts
end
```

Повторяющиеся ресурсы

```
resources :messages do
  resources :comments
end

resources :articles do
  resources :comments
  resources :images, only: :index
end
```

Concerns

```
concern :commentable do
  resources :comments
end

concern :image_attachable do
  resources :images, only: :index
end
```

Concerns

```
resources :messages,
                concerns: :commentable

resources :articles,
    concerns: [:commentable, :image_attachable]
```

Больше методов

```
resources :photos do
  collection do
    get 'search'
  end
end
```

```
resources :photos do
  member do
    get 'preview'
  end
end
```

Еще больше

```
resources :comments do
  get 'preview', on: :new
end
```

Статические и динамические сегменты

```
get 'photos/:id/with_user/:user_id’,
                                to: 'photos#show'

get 'photos/:id/:user_id', to: 'photos#show'
```

Defaults

```
get 'photos/:id’,
        to: 'photos#show’,
                defaults: { format: 'jpg' }
```

Helpers

```
articles_path #для себя
new_article_path
article_path(:id)
article_comments_path
article_comments_url #для остальных
```

Переименование ресурсов

```
resources :photos, as: 'images'
#сработает только для helper-а

resources :magazines do
  resources :ads, as: 'periodical_ads'
end
```

Контроллеры

```ruby
class ClientsController < ApplicationController
  def new
  end
end


def new
  @client = Client.new
end
```

```ruby
class ClientsController < ApplicationController

  def index
    if params[:status] == "activated"
      @clients = Client.activated
    else
      @clients = Client.inactivated
    end
  end
end
```

Параметры POST

```
def create
    @client = Client.new(params[:client])
    if @client.save
      redirect_to @client
    else
      render "new"
    end
  end
```

Хэши параметров

```
{ "company":
    { "name": "acme",
      "address": "123 Carrot Street"
    }
}
```

```
params[:company]# => {name: "acme", …}
```

config.wrap_parameters

```
{ name: "acme",
  address: "123 Carrot Street",
  company:
    { name: "acme",
      address: "123 Carrot Street"
    }
}
```

Strong Params

- Пользователи присылают все подряд
- Если писать массовую обработку, то есть риск передать лишнее
- Хочется использовать белые списки

```ruby
class PeopleController < ActionController::Base
  def create
    Person.create(params[:person])
  end

  def update
    person = current_account.people.find(params[:id])
    person.update!(person_params)
    redirect_to person
  end

  private

  def person_params
    params.require(:person).permit(:name, :age)
  end
end
```

Разрешенные скаляры

String,
Symbol,
NilClass,
Numeric,
TrueClass,
FalseClass,
Date,
Time,
DateTime,
StringIO,
IO,
ActionDispatch*::*Http*::*UploadedFile,
Rack*::*Test*::*UploadedFile

Параметры с вложенными объектами

```
params.permit(:name, { emails: [] },
              friends: [ :name,
                         { family: [ :name ], hobbies: [] }])
```

Обязательные параметры

```
params.require(:log_entry)
```

```
params.require(:log_entry).permit!
```

Сессии

```
# на клиенте
ActionDispatch::Session::CookieStore
# в приложении
ActionDispatch::Session::CacheStore
# в БД
ActionDispatch::Session::ActiveRecordStore
```

Находим текущего пользователя

```
class ApplicationController < ActionController::Base

  private
  def current_user
    @_current_user ||= session[:current_user_id] &&
        User.find_by(id: session[:current_user_id])
  end
end
```

Пишем в сессию

```
class LoginsController < ApplicationController
  def create
    if user = User.authenticate(params[:username],
                                params[:password])
      session[:current_user_id] = user.id
      redirect_to root_url
    end
  end
end
```

Удаление из сессии

```
class LoginsController < ApplicationController
  def destroy
    session.delete(:current_user_id)
    @_current_user = nil
    redirect_to root_url
  end
end
```

Базовые уведомления

```
class LoginsController < ApplicationController
  def destroy
    session.delete(:current_user_id)
    flash[:notice] = "You have successfully logged out."
    redirect_to root_url
  end
end
```

Базовые уведомления на редиректах

```
redirect_to root_url, notice:
                    "You have successfully logged out."

redirect_to root_url, alert: "You're stuck here!"

redirect_to root_url, flash: { referral_code: 1234 }
```

Уведомления на этом же запросе

```
class ClientsController < ApplicationController
  def create
    @client = Client.new(params[:client])
    if @client.save
      # ...
    else
      flash.now[:error] = "Could not save client"
      render action: "new"
    end
  end
end
```

Cookie

```ruby
class CookiesController < ApplicationController
  def set_cookie
    cookies[:just_cookie] = 'wow, that is a cookie'
    cookies.encrypted[:expiration_date] = Date.tomorrow
    redirect_to action: 'read_cookie'
  end

  def read_cookie
    cookies.encrypted[:expiration_date]
  end
end
```

Возвращаем разные форматы

```
class UsersController < ApplicationController
  def index
    @users = User.all
    respond_to do |format|
      format.html
      format.xml  { render xml: @users }
      format.json { render json: @users }
    end
  end
end
```

Фильтры/callbacks

- before
- around
- after

Before

```
class ApplicationController < ActionController::Base
  before_action :require_login

  private

  def require_login
    unless logged_in?
      flash[:error] = "You must be logged in to access this section"
      redirect_to new_login_url
    end
  end
end
```

Skip

```
class LoginsController < ApplicationController
  skip_before_action :require_login, only: [:new, :create]
end
```

Around ???

```
class ChangesController < ApplicationController
  around_action :wrap_in_transaction, only: :show

  private

  def wrap_in_transaction
    ActiveRecord::Base.transaction do
      begin
        yield
      ensure
        raise ActiveRecord::Rollback
      end
    end
  end
end
```

Request

Property of `request`	Purpose
host	The hostname used for this request.
domain(n=2)	The hostname's first `n` segments, starting from the right (the TLD).
format	The content type requested by the client.
method	The HTTP method used for the request.
get?, post?, patch?, put?, delete?, head?	Returns true if the HTTP method is GET/POST/PATCH /PUT/DELETE/HEAD.
headers	Returns a hash containing the headers associated with the request.
port	The port number (integer) used for the request.
protocol	Returns a string containing the protocol used plus "://", for example "http://".
query_string	The query string part of the URL, i.e., everything after "?".
remote_ip	The IP address of the client.
url	The entire URL used for the request.

Response

Property of response	Purpose
body	This is the string of data being sent back to the client. This is most often HTML.
status	The HTTP status code for the response, like 200 for a successful request or 404 for file not found.
location	The URL the client is being redirected to, if any.
content_type	The content type of the response.
charset	The character set being used for the response. Default is "utf-8".
headers	Headers used for the response.

View

```
$ rails generate scaffold article
      [...]
      invoke  scaffold_controller
      create    app/controllers/articles_controller.rb
      invoke    erb
      create      app/views/articles
      create      app/views/articles/index.html.erb
      create      app/views/articles/edit.html.erb
      create      app/views/articles/show.html.erb
      create      app/views/articles/new.html.erb
      create      app/views/articles/_form.html.erb
      [...]
```

Шаблонизаторы

HTML

```
<!DOCTYPE html>
<html>
    <body>

    <h1>My First Heading</h1>
    <p>My first paragraph.</p>

    </body>
</html>
```

Динамическая генерация

В основе – работа со строками

Хочется выполнять код на языке бэкенда

Быстро (вроде бы)

ERB (erubi, embedded ruby)

```
<h1>
  Hello <%= name %>!
</h1>
```

if - else

```
<% if @favorite_food == "chocolate" %>
  Are you a chocolate lover? Here are some of our best PREMIUM chocolate bars!
<% else %>
  Here are our top 10 snacks that people bought this month.
<% end %>
```

Циклы

```
<% @books.each do |book| %>
  <%= book.title %>
  <%= book.author %>
  <br>
<% end %>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>BoardGameStats</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>

    <%= stylesheet_link_tag    'application’,
                media: 'all', 'data-turbolinks-track': 'reload' %>
    <%= javascript_include_tag 'application’,
                               'data-turbolinks-track': 'reload' %>
  </head>

  <body>
    <%= yield %>
  </body>
</html>
```

Формы

```
<%= form_with(url: "/search", method: "get") do %>
  <%= label_tag(:q, "Search for:") %>
  <%= text_field_tag(:q) %>
  <%= submit_tag("Search") %>
<% end %>
```

Helpers - Чекбокс

```
<%= check_box_tag(:pet_dog) %>
<%= label_tag(:pet_dog, "I own a dog") %>
<%= check_box_tag(:pet_cat) %>
<%= label_tag(:pet_cat, "I own a cat") %>
```

Helpers - Чекбокс

```
<%= radio_button_tag(:age, "child") %>
<%= label_tag(:age_child, "I am younger than 21") %>
<%= radio_button_tag(:age, "adult") %>
<%= label_tag(:age_adult, "I am over 21") %>
```

```
<%= text_area_tag(:message, "Hi, nice site", size: "24x6") %>
<%= password_field_tag(:password) %>
<%= hidden_field_tag(:parent_id, "5") %>
<%= search_field(:user, :name) %>
<%= telephone_field(:user, :phone) %>
<%= date_field(:user, :born_on) %>
<%= datetime_local_field(:user, :graduation_day) %>
<%= month_field(:user, :birthday_month) %>
<%= week_field(:user, :birthday_week) %>
<%= url_field(:user, :homepage) %>
<%= email_field(:user, :address) %>
<%= color_field(:user, :favorite_color) %>
<%= time_field(:task, :started_at) %>
<%= number_field(:product, :price, in: 1.0..20.0, step: 0.5) %>
<%= range_field(:product, :discount, in: 1..100) %>
```

Вне Rails

```ruby
require 'erb'
Book     = Struct.new(:title, :author)
template = ERB.new(File.read('template.erb'))
template.result_with_hash(books: [Book.new("test"), Book.new("abc")])
```

HAML

```
%head
  %title Title
  = 123 * 2
  %body
    - @collection.each do |x|
      %p
        = x
```

Slim

```
.col-lg-4
  .features-icons-item.mx-auto.mb-5.mb-lg-0.mb-lg-3
    .features-icons-icon.d-flex
      i.icon-calendar.m-auto.theme-red
    h3
      ' Статистика с
      =@data[:min_year]
      '  по
      =@data[:max_year]
      '  год
    p.lead.mb-0
```

Что выбрать?

- ERB быстрее остальных
- HAML и Slim лучше читаются, легче поддерживаются
- View не содержит логики
- JS работает с готовой страницей
- Мешать логику JS и шаблонизаторов не стоит
- Мешать фреймворк и шаблонизаторы не стоит

Ссылки

https://guides.rubyonrails.org/routing.html

https://www.rubyguides.com/2018/11/ruby-erb-haml-slim/