Rails Controllers / Views

controllers, views, routes, modules, view helpers

Devpoint Labs - Dave Jungst / Jake Sorce

Skinny Controllers / Fat Models

Views should be reduced to primarily HTML by putting as much logic as we can in models and controllers.

Controllers should be reduced to primarily directing traffic.

Models should contain all logic that acts on a record.

Controllers

Controllers are where our CRUD actions are contained.

- create => POST
- read => GET
- update => PUT
- delete => DELETE

Controllers (cont)

Basic rails crud methods in a controller

resources :person

- index => @people = Person.all
- new => @person = Person.new
- create => Person.create(person_params)
- edit => @person = Person.find(params[:id])
- update => Person.find(params[:id]).update(person_params)
- destroy => Person.find(params[:id]).destroy

Controllers (cont)

A controller must return HTML, JSON, XML, or redirect to a method that does

```
def index
  @people = Person.all      => There is a matching view so Rails knows to render the HTML
end

def list
  render ‘layouts/list’     => This returns an HTML partial
end
```

Controllers (cont)

A controller must return HTML, JSON, XML, or redirect to a method that does

```
def person
  Person.find(params[:id]).to_json
end
```

=> This will return the Person object as JSON (Think hash)

```
def update
    Person.find(params[:id).update(...)
    head :no_content
end
```

=> Since we are not rendering anything we send the status code as a json object { status: 204 }

Controllers Callbacks

Controller callbacks allow you to perform logic before or after a controller method is called

- before_action => Called before entering method
- after_action   => Called after method has run
- skip_before_action => skips before actions
- skip_after_action   => skips after actions

filter :method_to_call, (only/except): [:my, :route, :methods]

Controllers Callback Example

```
class PeopleController < ApplicationController
before_action :find_person, only: [:edit, :update, :destroy]

private
  def find_person
    @person = Person.find_by(id: params[:id])
  end
end
```

```ruby
class PeopleController < ApplicationController
  before_action :set_person, only: [:show, :edit, :update, :destroy]
  before_action :validate_paid, except: [:destroy]
  after_action :more_cowbell
  skip_after_action :more_cowbell, only: [:new]

  def index
    @people = Person.all
  end

  def show
  end

  def new
    @person = Person.new
  end

  def edit
  end

  def update
    if @person.update(person_params)
      redirect_to @person
    else
      render :edit
    end
  end

  def destroy
    @person.destroy
  end

  private
    def set_person
      @person = Person.find(params[:id])
    end

    def validate_paid
      @person.paid?
    end

    def more_cowbell
      Rails.logger.info "DING DING DING DING"
    end
end
```

Views

The view should be reduced to primarily HTML a few conditionals and loops are acceptable but adding more logic to the view is considered bad practice.

Partials

A partial is a snippet of HTML code that can be rendered and reused by other views.

Partials are named beginning with an underscore. This is part of rails convention over configuration.
_shopping_list.html.erb

app/views/person/_shopping_list.html.erb

```
<h2>Shopping List</h2>
<ul>
  <% @person.items.each do |item| %>
    <li> <%= item %> </li>
  <% end %>
</ul>
```

app/views/person/show.html.erb

```
<h2><%= @person.name %></h2>
<h3>Shopping List</h3>
<%= render partial: ‘shopping_list’ %>
```

Partials

When you are sharing a view between multiple controllers you can pass in locals

app/views/person/_shopping_list.html.erb

```
<h2>Shopping List</h2>
<ul>
  <% person.items.each do |item| %>
    <li> <%= item %> </li>
  <% end %>
</ul>
```

```
def generate_list
  person = Person.find(params[:id])
  render partial: ‘shopping_list’, locals: { person: person }
end
```

```
<h2><%= @person.name %></h2>
<h3>Shopping List</h3>
<%= render partial: ‘shopping_list’, locals: { person: @person %>
```

View Helpers

Shared code between views

app/helpers/state_helper.rb

```
def valid_states
 [“Washington”, “Colorado”, “Oregon”, “Alaska”]
end
```

app/views/some_view.rb

```
<h2>Valid States</h2>
<% valid_states.each do |state| %>
  <p><%= state %></p>
<% end %>
```

Routes

Routes are found in the config/routes.rb file

There are many ways to define the routes for your application here are a few:

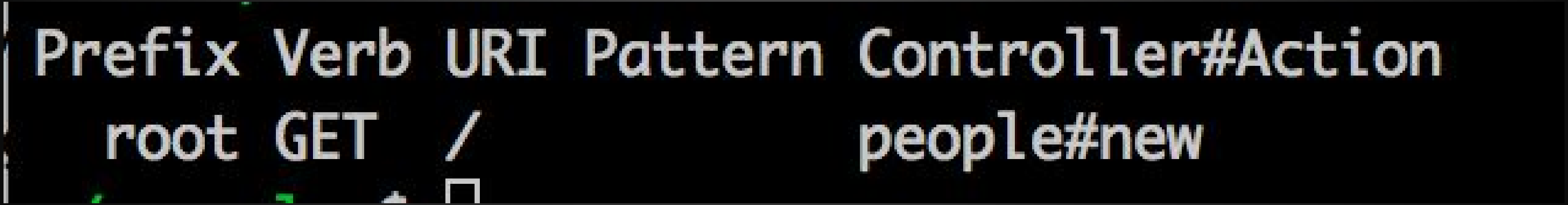
- resources
- get ‘people#new’
- root ‘people#index’

Routes - Root Syntax

The Root Syntax:

in config/routes.rb - root ‘people#new

in your terminal - rake routes



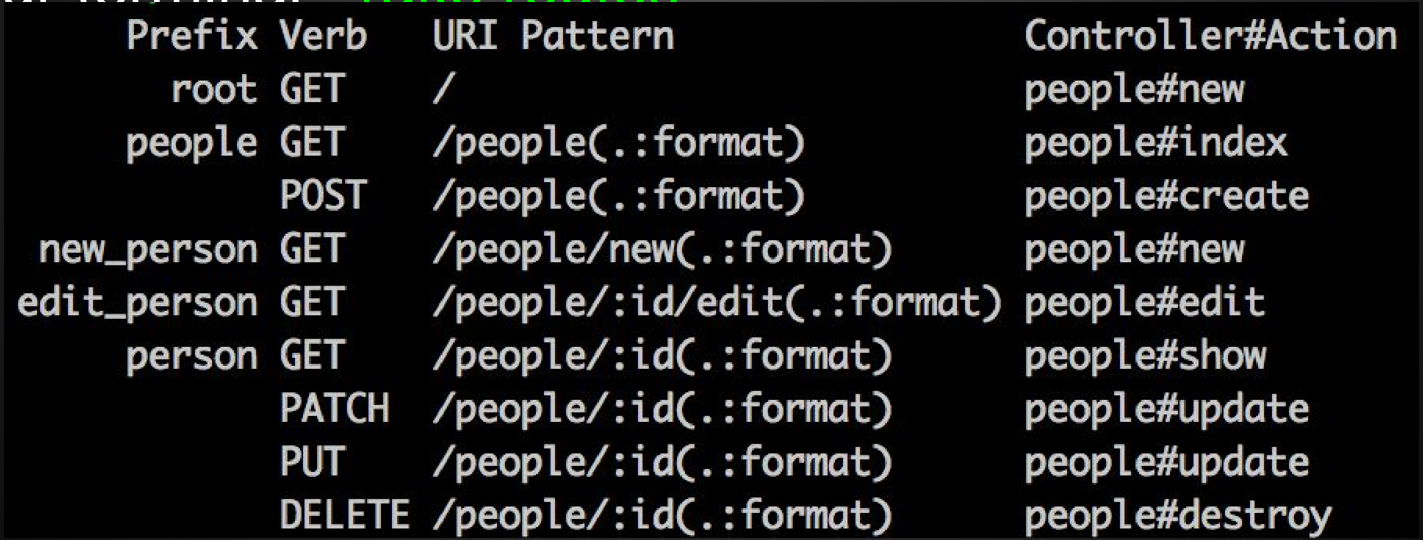
```
Prefix Verb URI Pattern Controller#Action
  root GET  /           people#new
```

Routes - Resources Syntax

The Resources Syntax:

in config/routes.rb - resources :people

in your terminal - rake routes



```
      Prefix Verb   URI Pattern                     Controller#Action
        root GET    /                               people#new
      people GET    /people(.:format)               people#index
             POST   /people(.:format)               people#create
  new_person GET    /people/new(.:format)           people#new
 edit_person GET    /people/:id/edit(.:format)      people#edit
      person GET    /people/:id(.:format)           people#show
             PATCH  /people/:id(.:format)           people#update
             PUT    /people/:id(.:format)           people#update
             DELETE /people/:id(.:format)           people#destroy
```

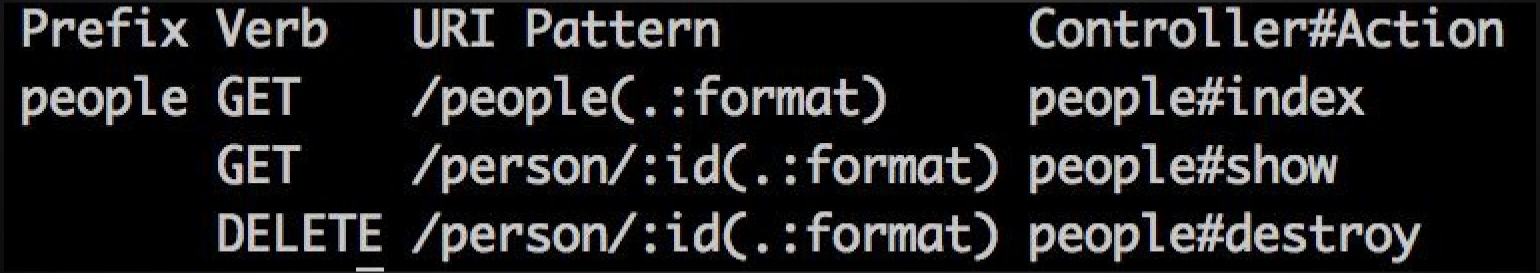
Routes - Regular Routes

The Regular Route Syntax:

in config/routes.rb -

```
  get '/people', to: 'people#index'
  get '/person/:id', to: 'people#show'
  delete '/person/:id', to: 'people#destroy'
```

in your terminal - rake routes



```
Prefix Verb   URI Pattern                Controller#Action
people GET    /people(.:format)          people#index
       GET    /person/:id(.:format)      people#show
       DELETE /person/:id(.:format)      people#destroy
```

Routes - Nested Routes

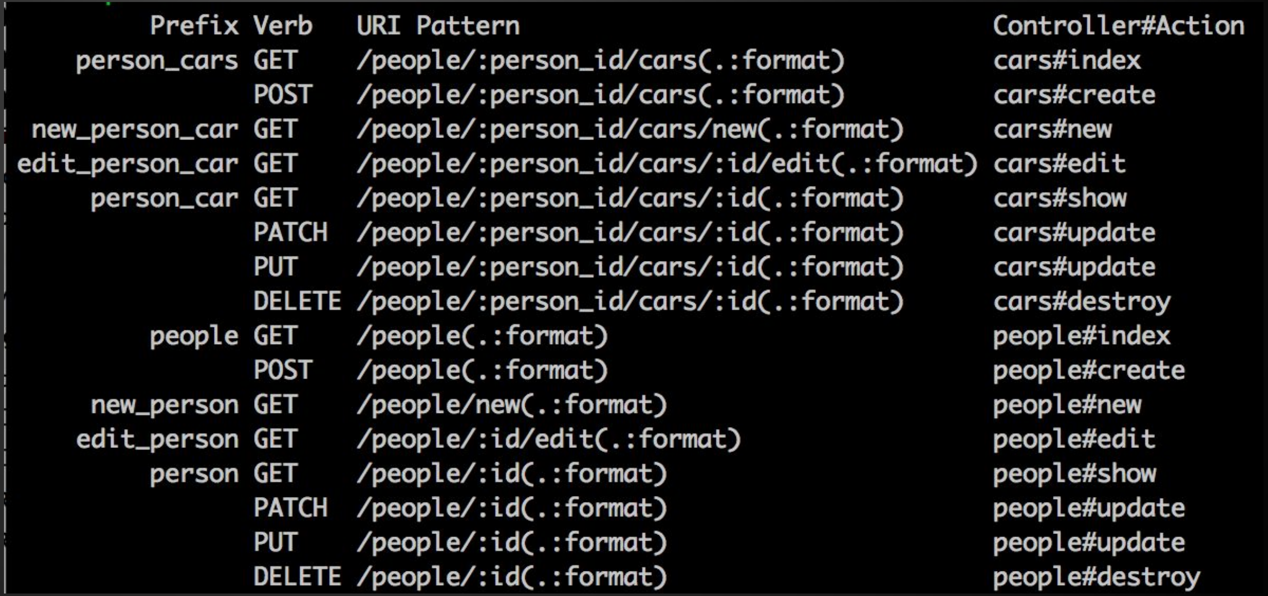
The Nested Routes Syntax:

in config/routes.rb -

```
resources :people do
  resources :cars
end
```

in your terminal -

```
rake routes
```



```
          Prefix Verb   URI Pattern                                   Controller#Action
     person_cars GET    /people/:person_id/cars(.:format)             cars#index
                 POST   /people/:person_id/cars(.:format)             cars#create
  new_person_car GET    /people/:person_id/cars/new(.:format)         cars#new
 edit_person_car GET    /people/:person_id/cars/:id/edit(.:format)    cars#edit
      person_car GET    /people/:person_id/cars/:id(.:format)         cars#show
                 PATCH  /people/:person_id/cars/:id(.:format)         cars#update
                 PUT    /people/:person_id/cars/:id(.:format)         cars#update
                 DELETE /people/:person_id/cars/:id(.:format)         cars#destroy
          people GET    /people(.:format)                             people#index
                 POST   /people(.:format)                             people#create
      new_person GET    /people/new(.:format)                         people#new
     edit_person GET    /people/:id/edit(.:format)                    people#edit
          person GET    /people/:id(.:format)                         people#show
                 PATCH  /people/:id(.:format)                         people#update
                 PUT    /people/:id(.:format)                         people#update
                 DELETE /people/:id(.:format)                         people#destroy
```

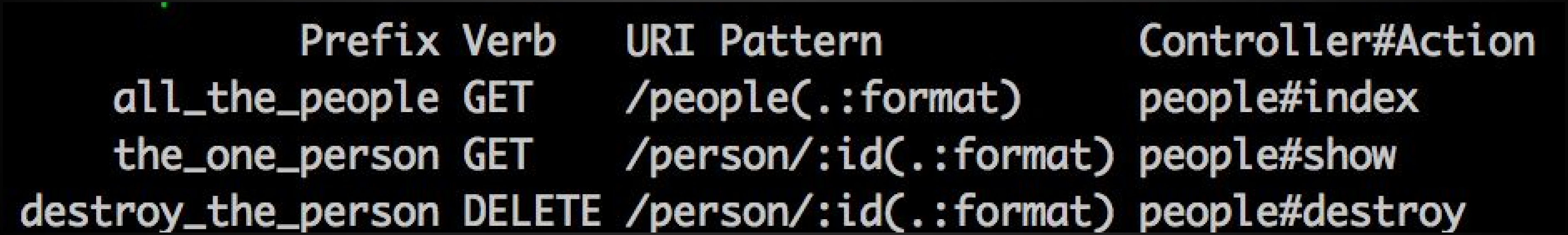
Routes - Named Routes

The Resources Syntax:

in config/routes.rb -

```
  get '/people', to: 'people#index', as: 'all_the_people'
  get '/person/:id', to: 'people#show', as: 'the_one_person'
  delete '/person/:id', to: 'people#destroy', as: 'destroy_the_person'
```

in your terminal - rake routes



```
            Prefix Verb   URI Pattern             Controller#Action
    all_the_people GET    /people(.:format)       people#index
    the_one_person GET    /person/:id(.:format)   people#show
destroy_the_person DELETE /person/:id(.:format)   people#destroy
```

Concerns

Concerns are helper methods for controllers.

Located in app/controllers/concerns

You can put code in here that can be shared between controllers and reused. This code is typically logic that is repeated across controllers

We will go deeper into this time permitting.

Working with Validation Errors

When a user submits a form to your server and the validation on the model fails we should handle this gracefully. We need to return the user back to the form they submitted and show them correct errors.

Working With Validation Errors - Methods Available

- @person.errors
  - Errors Object

    => #<ActiveModel::Errors:0x007faf2b346610>
- @person.errors.messages
  - Errors Hash

    => {:name=>["can't be blank"]}
- @person.errors.full_messages
  - String

    => Name can't be blank

Working with Validation Errors - Controller

The Controller:

```
class PeopleController < ApplicationController
  def create
    @person = User.new(person_params)
    if @person.save
      redirect_to users_path, flash: {notice: ‘Person Created Successfully.’}
    else
      render :new, flash: {error: ‘Something went wrong’ }
    end
  end
end
```

Working with Validation Errors - View

The View:

```
<% if @person.errors.any? %>
<ul>
  <% @person.errors.full_messages.each do |msg| %>
    <li><%= msg %></li>
  <% end %>
</ul>
<% end %>

<%= form_for @person do |f| %>
<% end %>
```