ActionController

The “C” in MVC

Enable Labs

Controllers

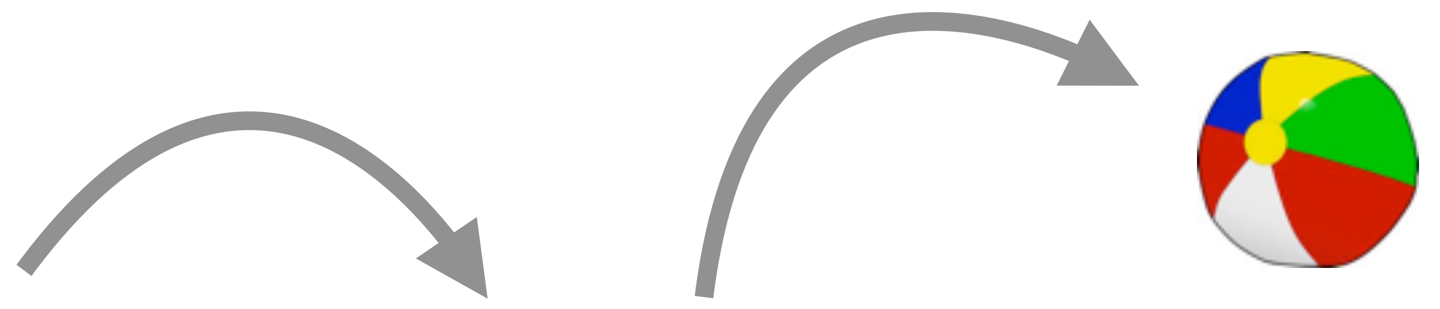
Web requests are “routed” to controller actions

http://localhost:3000/cars/9 CarsController#show

Enable Labs

...and...

```
▷ car_dealership
  ▷ app
    ▷ assets
    ▷ controllers
       cars_controller.rb
    ▷ mailers
    ▷ models
    ▷ views
      ▷ cars
          index.html.erb
          show.html.erb
      ▷ layouts
  ▷ config
...
```



...controllers render views

Enable Labs

A Simple Controller

<Rails.root>/config/routes.rb

```ruby
CarDealership::Application.routes.draw do
  resources :cars,
    :only => [
      :show,
      :new,
      :create
    ]

  root :to => 'pages#home'
end
```

<Rails.root>/app/controllers/cars_controller.rb

```ruby
class CarsController < ApplicationController
  def show
    @car = Car.find(params[:id])
  end

  def new
    @car = Car.new
  end

  def create
    @car = Car.new(params[:car])
    if @car.save
      flash[:notice] = '...successfully created.'
      redirect_to account_url
    else
      flash[:error] = '..cannot be created.'
      render :action => "new"
    end
  end
end
```

```
$ rake routes
cars     GET    /cars(.:format)               cars#index
         POST   /cars(.:format)               cars#create
new_car  GET    /cars/new(.:format)           cars#new
edit_car GET    /cars/:id/edit(.:format)      cars#edit
     car GET    /cars/:id(.:format)           cars#show
         PUT    /cars/:id(.:format)           cars#update
         DELETE /cars/:id(.:format)           cars#destroy
```

Controllers - REpresentational State Transfer

Life cycle of a model correspond to HTTP verbs

HTTP Verb →	route →	Controller Action
GET	/cars	index
GET	/car/1	show
GET	/cars/new	new
GET	/cars/1/edit	edit
POST	/cars	create
PUT	/cars/1	update
DELETE	/cars/1	delete

```
resources :cars
```

routes.rb

Enable Labs

Controllers

- Defines public methods known as “Actions”
- ApplicationController
  - Created for you during “rails new”
  - extends ActionController::Base
  - Your application’s controllers usually extends ApplicationController
    - inherits base functionality of ApplicationController
      - Example: adding authorization/authentication functionality to subclasses controllers
- Response to differently formatted requests, i.e. HTML, JSON, XML

Enable Labs

Controller Actions

- Provide a wealth of accessible data for you!
  - params hash - GET/POST/PUT parameters in hash
  - request instance
    - .headers, .xhr?, .env
- Ultimate responsibility of a controller...
  - render
    - automatically renders a view with the action name if render is not called
    - can output different formats, i.e. HTML, JSON, XML based on the incoming request
    - dont' "double render" a response!
  - redirect
    - send an HTTP redirect so that the client requests a different page

```
class CarsController < ApplicationController
  def show
    @car = Car.find(params[:id])
    if @car
      if request.xhr?
        render :json => @car
      else
        render :show
      end
    else
      redirect_to :index,
         :notice => 'No Car Found'
    end
  end
end
```

Enable Labs

Let’s Not talk about MIME Types

When you want to force the MIME Type

```
render :text => “I’m very boring, and plain text”
```

```
render :json => @car
```

```
render :json => @car.errors, :status => :unprocessable_entity
```

When you want to let the request dictate the MIME Type

```
respond_to do |format|
  format.html  #app/views/cars/show.html.erb
  format.json { render json: @car }
  format.xml   #app/views/cars/show.xml.builder
end
```

http://localhost:3000/cars/1
http://localhost:3000/cars/1.json
http://localhost:3000/cars/1   **accept: application/json**
http://localhost:3000/cars/1.xml

Enable Labs

Controllers: Some guidance

- Business logic goes in the Models and Modules... keep it out of the Controller
  - "Skinny controller, fat model."

- “Concern” yourself with the responsibility and size of the controller
  - Consider adding a friendships_controller instead of a “link_friend” action in your friends_controller.rb
  - Think RESTfully

- Keeping it DRY with Filters

```
class FriendsController < ApplicationController
  before_filter :fetch_friend, :except => [:new, :create]
  ...
end
```

Enable Labs

Namespaced Resources

http://localhost/admin/cars/9

<Rails.root>/config/routes.rb

```
CarDealership::Application.routes.draw do
  resources :cars, :only => [:index, :show]
  namespace :admin do
    resources :cars
  end
end
```

http://localhost/admin/cars/:id

<Rails.root>/app/controllers/admin/cars_controller.rb

```
class Admin::CarsController < ApplicationController
  def new ...
  def create ...
  def edit ...
  def update ...
  def destroy ...
end
```

```
▸ car_dealership
  ▸ app
    ▸ assets
    ▸ controllers
      ▸ admin
          cars_controller.rb
        cars_controller.rb
    ▸ mailers
    ▸ models
    ▸ views
      ▸ layouts
  ▸ config
  ...
```

Nested Resources

http://localhost/dealerships/11/cars/9

<Rails.root>/config/routes.rb

```ruby
CarDealership::Application.routes.draw do
  resources :dealerships do
    resources :cars
  end
end
```

http://localhost/dealerships/:dealership_id/cars/:id

<Rails.root>/app/controllers/admin/cars_controller.rb

```ruby
class CarsController < ApplicationController
  before_filter :fetch_dealership

  def fetch_dealership
    @dealership = Dealership.find(params[:dealership_id])
  end
end
```

```
car_dealership
  app
    assets
    controllers
      cars_controller.rb
      dealerships_controller.rb
    mailers
    models
    views
      layouts
  config
  ...
```

Cookies & Sessions

- Both are hashes available in the controller
- Avoid storing complex Ruby objects, instead put id:s in the session and keep data in the database, i.e. use session[:user_id] rather than session[:user]
- Sessions
  - Can be store on the server or in browser cookie
    - Application.config.session_store :cookie_store, key: '_my_session'
  - Rails uses a cookie or request parameter, _session_id, to keep track of your session
- Cookies
  - stored on the client and sent with each request
  - can be cryptographically signed

```
cookies[:lang] = "en”
cookies.delete(:lang)
```

```
session[:user_id] = @user.id
session[:user_id] = nil
reset_session
```

Enable Labs

ActionView

Our Face to the World

Enable Labs

ActionView

- Rendering a response to the client
  - Layouts
    - <Rails.root>/app/views/layouts/application.html.erb
    - layout ‘admin’
  - Templates
  - Partials
- Templates use helper methods to generate links, forms, and JavaScript, and to format text

Template

```
render :show
```

```
<Rails.root>/app/views/cars/show.html.erb
```

MIME Type ↗ ↖ Rendering Engine

Partial

```
render “car_accessories”
```

```
<Rails.root>/app/views/cars/_car_accessories.html.erb
```

Enable Labs

Example View

<Rails.root>/app/dealerships/show.html.erb

```
<p>
  <b>Name:</b>
  <%=h @dealership.name %>
</p>

<%= link_to 'Edit', edit_dealership_path(@dealership) %>
<%= link_to 'Back', dealerships_path %>
<%= link_to "Delete", dealership_path(@dealership), :confirm => "Are you sure?", :method=>:delete %>
```

- Templates use "helper methods" to render:
  - Links
  - Forms
  - JavaScript
  - Format text (dates, numbers, currency, ...)

Enable Labs

Example View: Forms

<Rails.root>/app/dealerships/new.html.erb

```
<% form_for @dealership do |f| %>
  <%= f.label :name %>
  <%= f.text_field :name %>
  <%= f.submit ‘save’ %>
<% end %>
```

Does the ‘edit’ form look similar to the ‘new’ form?

- FormHelper
  - hidden_field
  - label
  - password_field
  - radio_button
  - select
    - options_for_select
  - text_area
  - text_field

*_tag vs *_field

Enable Labs

Example View: Forms for Nested Resources

<Rails.root>/app/cars/new.html.erb

```
<% form_for [@dealership, @car] do |f| %>
  <%= f.label :model %>
  <%= f.text_field :model %>
  <%= f.submit ‘save’ %>
<% end %>
```

```
<form accept-charset="UTF-8" action="/dealerships/1/cars" class="new_car" id="new_car"
method="post">
  <div style="margin:0;padding:0;display:inline">
    <input name="utf8" type="hidden" value="&#x2713;" />
    <input name="authenticity_token" type="hidden" value="DCM7qs6Wh7d1t0XB5GkiD82H5v35iHvKXLTCK0zeINw=" />
  </div>
  <label for="car_model">Model</label>
  <input id="car_model" name="car[model]" size="30" type="text" />
  <input name="commit" type="submit" value="Save" />
</form>
```

Where do templates live?

```
▹ car_dealership
  ▹ app
    ▹ assets
    ▹ controllers
    ▹ helpers
    ▹ mailers
    ▹ models
    ▹ views
      ▹ layouts
  ▹ config
  ...
```

- app/views/<plural resource name>/*
- Templates that belong to a certain controller typically live under app/view/controller_name, i.e. templates for Admin::UsersController would live under app/views/admin/users
- Templates shared across controllers are put under app/views/shared. You can render them with render :template => ‘shared/my_template’
- You can have templates shared across Rails applications and render them with render :file => ‘path/to/template’

Enable Labs

Template Environment

- Templates have access to the controller object's flash, headers, logger, params, request, response, and session
- Instance variables (i.e. @variable) in the controller are available in templates
- The current controller is available as the attribute "controller"
- Default templating language is Embedded Ruby (erb)
  - <%= ruby code here %> - Evaluates the Ruby code and prints the last evaluated value to the page
  - <% ruby code here %> - Evaluates Ruby code without outputting anything to the page

```
<% dealership.cars.order(:make, :model, :year).each do |car| %>
  <p><%= car.make %> - <%= car.make %> - <%= car.make %></p>
<% end %>
```

Enable Labs

Haml

```
#profile
  .left.column
    #date= print_date
    #address= current_user.address
  .right.column
    #email= current_user.email
    #bio= current_user.bio
```

Haml

vs

HTML with ERB

```
<div id="profile">
  <div class="left column">
    <div id="date"><%= print_date %></div>
    <div id="address"><%= current_user.address %></div>
  </div>
  <div class="right column">
    <div id="email"><%= current_user.email %></div>
    <div id="bio"><%= current_user.bio %></div>
  </div>
</div>
```

Enable Labs

Useful Lab Session Commands

Cloning the Lesson Project

```
$ git clone https://github.com/EnableLabs/rails_training_feb_2013.git
$ cd ./rails_training_feb_2013/week3/car_dealership
 # if asked, please ‘trust’ the .rvmrc file
$ bundle install
```

Useful rake commands

```
$ rake db:migrate
```

```
$ rake db:test:prepare
```

Running tests with rspec

```
$ rspec
```

Let the test failure guide your next step

Enable Labs

Shameless Self Promotion

We do Ruby on Rails Development and Training

Enable Labs

Ruby and Rails Training

- One day to five day programs.
- Introduction to Ruby
- Advanced Ruby
- Introduction to Rails
- Advanced Rails
- Test Driven Development
- Behavior Driven Development
- Test Anything with Cucumber
- Advanced Domain Modeling with ActiveRecord
- Domain Driven Development with Rails

Enable Labs

Ruby on Rails Development

- Full Life Cycle Project Development
  - Inception
  - Implementation
  - Deployment
  - Long Term Support
- Ruby on Rails Mentoring
  - Get your team up to speed using Rails
- Project Rescue

Enable Labs