Web Application Architectures

Module 5: Middleware
Lecture 7: Rails Controllers–Response

THE UNIVERSITY *of* NEW MEXICO

Rails Controllers — Request Processing

- In the last lecture we say how the dispatcher routes a request to a particular controller action (method).
  Ex. The HTTP GET request

  `http://localhost:3000/posts/1`

  will route to the `show` method in the `PostsController` class, passing `params[:id]` with a value of 1 to the controller. Note: this class is defined in the file:
  `./app/contollers/posts_controller.rb`.
- Next, the `show` method will use the `ActiveRecord#find` method to retrieve the post with id=1 from the database, and assign it to the instance variable `@post`.
- Finally, the controller will pass `@post` to the view, i.e., to the template file:
  `./app/views/posts/show.html.erb`
  and this will be used to create the HTML that will be sent to the browser.

Filters

- The PostsController#show method is defined as follows:

  ```
  # GET /posts/1
  # GET /posts/1.json
  def show
  end
  ```

  It doesn't retrieve the post!

- The desired post is actually retrieved from the database using a filter called set_post.

- Filters allow controllers to run shared pre and post processing code over their methods.

Filters

Sessions

- In general, the "state" of an application which needs to persist across requests should be stored in the database. E.g., posts and comments are persisted in the database.
- There are times when data needs to be persisted differently. E.g., the current contents of a shopping cart.
- Whenever a user connects to a Rails application, a session is created.
- Session data is stored in Rails using a hash structure that persists across requests, and can be accessed by controllers.
  Ex. `session[:current_user] = user.id`
- A flash hash is part of the session that is cleared with each request (its value is made to the next request). A controller can use this to send a message that can be displayed to the user on the next request.
  Ex.
  lstinline$flash[: notice] =' Postwassuccessfullycreated.'$

Response – Rendering HTML or JSON

- The request:

  ```
  http://localhost:3000/posts/1
  ```

  assumes that HTML will be returned. I.e., it's the same as:

  ```
  http://localhost:3000/posts/1.html
  ```

- Rails can return other formats, e.g., JSON capabilities are also provided by default.

  Ex. The following request, will be routed to the same controller method as before:

  ```
  http://localhost:3000/posts/1.json
  ```

  However, it will be rendered using the file:
  ```
  ./app/views/posts/show.json.builder
  ```
  and JSON will be returned to the client.

Response – Rendering HTML or JSON

Response Format

One last look at rake routes:

```
Prefix    Verb          URI Pattern              Controller#Action

posts     GET    /posts(.:format)                posts#index
          POST   /posts(.:format)                posts#create
new_post  GET    /posts/new(.:format)            posts#new
edit_post GET    /posts/:id/edit(.:format)       posts#edit
post      GET    /posts/:id(.:format)            posts#show
          PATCH  /posts/:id(.:format)            posts#update
          PUT    /posts/:id(.:format)            posts#update
          DELETE /posts/:id(.:format)            posts#destroy
```

Redirects

- Rails may also respond to an HTTP request using the `redirect_to` method.
- This method actually tells the browser to send a new request for a different URL.
  Ex. `redirect_to 'www.example.com'`
- Rails has shortcuts for URLs within your application – they're the prefix listed when you view routes:
  Ex. `redirect_to posts_url`
  will redirect to the `index` method in the `PostsController`.

  Ex. You can assign a flash message as a part of a redirection:

```
redirect_to @post, notice: 'Post was successfully created.'
```