Web Application Architectures

Module 5: Middleware
Lecture 6: Rails Controllers–Request Handling

THE UNIVERSITY of NEW MEXICO

Action Pack

- We've already considered how models (the 'M' in MVC) are supported by the Ruby `ActiveRecord` module in Rails.

- Views and controllers in Rails are supported by Action Pack, which consists of three Ruby modules: `ActionDispatch`, `ActionController` and `ActionView`.
  We'll consider the first two of these in this module, and `ActionView` in the next module.

- Controllers are the heart of a Rails application – when a user connects to your Rails application, they do it by asking the application to execute a controller action.

  How does Rails determine the controller action it will execute?

Dispatcher

THE UNIVERSITY of NEW MEXICO

- When an HTTP request is made to a Rails application, the `ActionDispatch` module is used to map that request to a particular controller action.

- Requests are mapped to controller actions via the routes defined in `./config/routes.rb` file. We already seen that you can view the routes defined in your application by executing:

```
$ rake routes
```

- To connect a request to a controller action, you add a route to `./config/routes.rb`.
  Examples of the various ways you can do this are provided as comments in that file.

Dispatcher–Routes

- Ex. Adding the following to `./config/routes.rb`:

```
get 'products/:id' => 'catalog#view'
```

  will map a GET request that uses the URL:

```
http://localhost:3000/products/10
```

  to the `view` method in the `CatalogContoller` class, assigning the value 10 to the `params[:id]` hash that will be made available to all methods in that class.

- This doesn't look like the syntax used by the scaffold generator to create the routes for our `posts` and `comments`:

```
resources :posts
resources :comments
```

RESTful Controllers

- By default, Rails controllers are RESTful and therefore use resource routing.
- REST stands for Representational State Transfer.
- The fundamental philosophy behind REST is that clients should communicate with servers through stateless connections, where:
  - Long term state is kept on the server side by maintaining a set of identifiable resources, `posts` and `comments` in our case.
  - The client can access these resources (perform CRUD operations on them) through a highly limited but uniform interface (a set of URLs in our case).
  - Computation proceeds by identifying the resource and the CRUD operation you'd like to perform on it.

RESTful Controllers

- A REST-based web application can be contrasted to a RPC-based web application.
  - In RPC-based applications, clients send requests to servers, asking them to execute a specified procedure (available on the server) using the supplied parameters. The server must advertise the services it offers. SOAP is a protocol, developed by Microsoft, that supports this approach.
  - REST assumes a simple set of verbs (controller actions/methods) that can operate over a rich set of nouns (resources).
  - RPC allows for arbitrary complexity on the server side.
- The constraints imposed by REST can lead to web applications that easier-to-write and maintain. Rather than implementing remotely accessible services, a simple interface for performing CRUD operations on resources is provided.
- The Programmable Web, which treats the WWW as a vast collection of addressable resources, is greatly facilitated by REST.

Resourceful Routes

THE UNIVERSITY of NEW MEXICO

The statement in `./config/routes.rb`:

```
resources :posts
```

produces seven different routes in your application, all mapping to methods in the `PostsController` class:

HTTP Verb	Path	Method	Purpose
GET	/posts	index	dispaly all posts
GET	/posts/new	new	return form for creating a post
POST	/posts	create	create a new post
GET	/posts/:id	show	display a specific post
GET	/posts/:id/edit	edit	return form for editing a post
PATCH/PUT	/posts/:id	update	update a specific post
DELETE	/posts/:id	destroy	delete a specific post