CHAPTER 2

# Rails Routing

# The Rails Router

The Rails router recognises HTTP Verbs and URLs and dispatches them to a controller's action.

It can also generate paths and URLs, avoiding the need to hardcode strings in your views.

## Connecting URLs to Code

When your Rails application receives an incoming request for:

|  |
| --- |
| GET /user/1 |

It 'asks' the router to match it to a controller action.

If the first matching route is

|  |
| --- |
| get '/user/:id', to: 'user#show' |

The request is dispatched to the patients controller's show action with { id: '1' } in params.

## Generating Paths and URLs from Code

You can also generate paths and URLs. If the route above is modified to be:

|  |
| --- |
| get '/user/:id', to: 'user#show', as: 'user' |

and your application contains this code in the controller:

|  |
| --- |
| @user = User.find(1) |

and this in the corresponding view:

|  |
| --- |
| <%= link\_to 'user Record', user\_path(@user) %> |

then the router will generate the path /user/1.

This makes your code easier to understand.

Note that the id does not need to be specified in the route helper.

# Resource Routing

## The Rails Default

Resource routing allows you to quickly declare all the common routes for a given resourceful controller.

Instead of declaring separate routes for your index, show, new, edit, create, update and destroy actions, a resourceful route declares them in a single line of code.

## Resources on the Web

Browsers request pages from Rails by making a request for a URL using a specific HTTP method, such as GET, POST, PATCH, PUT and DELETE.

Each method is a request to perform an operation on the resource. A resource route maps several related requests to actions in a single controller.

When your Rails application receives an incoming request:

DELETE /user/1

It asks the router to map it to a controller action. If the first matching route is:

resources :user

Rails would send the request to the destroy method on the user controller with { id: '1' } in params.

## CRUD, Verbs, and Actions

In Rails, a resourceful route provides a mapping between HTTP verbs and URLs to controller actions.

By convention, each action also maps to particular CRUD operations in a database.

A single entry in the routing file, such as

**resources :users**

This will automatically create seven different routes in your application, all mapping to the User controller: These can be viewed by running - **rails routes** – at the command line.

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

The router uses the HTTP verb and the URL to determine the matched action.

Routes are matched in a 'top down' style this is an important consideration when ‘adding’ your own routes to an existing routes.rb file.

If you have a resources entry before a get entry the controller will never 'see' the get entry as the match will occur at the resource level.

## Path and URL Helpers

Creating a resourceful route will also expose a number of helpers to the controllers in your application. In the case of

**Resources :users**

users\_path returns /users

new\_user\_path returns /users/new

edit\_user\_path(:id) returns /users/:id/edit

user\_path(:id) returns /users/:id

Each of these helpers has a corresponding \_url helper (such as users\_url) which returns the same path prefixed with the current host, port and path prefix.

## Defining Multiple Resources at the Same Time

If you need to create routes for more than one resource, you can save a bit of typing by defining them all with a single call to **resources**

resources :departments, :employees

This will produce exactly the same result as

resources :departments

resources :employees

## Singular Resources

Sometimes, you have a resource that clients always look up without referencing an ID.

For example, you would like /profile to always show the profile of the currently logged in user.

In this case, you can use a singular resource to map /profile (rather than /profile/:id) to the show action:

Get 'profile', to: users#show

Passing a String to match will expect a controller#action format, while passing a Symbol will map directly to an action.

Get 'profile', to: :show

## Controller Namespaces and Routing

You may wish to organize groups of controllers under a single namespace.

Most commonly, you might group a number of administrative controllers under an Admin:: namespace.

You would place these controllers under the app/controllers/admin directory, and you can group them together in your router

|  |
| --- |
| namespace :admin do    resources :posts, :comments  end |

This will create a number of routes for each of the posts and comments controller.

For Admin::PostsController, Rails will create the following routes

|  |  |  |  |
| --- | --- | --- | --- |
| **HTTP Verb** | **Path** | **Action** | **Used for** |
| GET | /admin/posts | index | admin\_posts\_path |
| GET | /admin/posts/new | new | new\_admin\_post\_path |
| POST | /admin/posts | create | admin\_posts\_path |
| GET | /admin/posts/:id | show | admin\_post\_path(:id) |
| GET | /admin/posts/:id/edit | edit | edit\_admin\_post\_path(:id) |
| PATCH/PUT | /admin/posts/:id | update | admin\_post\_path(:id) |
| DELETE | /admin/posts/:id | destroy | admin\_post\_path(:id) |

## Adding RESTful Actions

You are not limited to the seven routes that RESTful routing creates by default.

If you like, you may add additional routes that apply to the collection or individual members of the collection.

## Adding Member Routes

To add a member route, just add a member block into the resource block.

|  |
| --- |
| resources :users do    member do      get 'preview'    end  end |

This will recognize /users/1/preview with GET, and route to the preview action of UsersController, with the resource id value passed in params[:id]. It will also create the preview\_user\_url and preview\_user\_path helpers.

Within the block of member routes, each route name specifies the HTTP verb that it will recognize. You can use get, patch, put, post, or delete here. If you don't have multiple member routes, you can also pass :on to a route, eliminating the block:

|  |
| --- |
| resources :users do    get 'preview', on: :member  end |

# Inspecting and Testing Routes

To get a complete list of the available routes in your application, put

http://localhost/rails/info/routes

into your browser while your server is running in the **development** environment.

You can also execute the rails routes command in your terminal to produce the same output.

Both methods will list all of your routes, in the same order that they appear in routes.rb. For each route, you'll see:

The route name (if any)

The HTTP verb used (if the route doesn't respond to all verbs)

The URL pattern to match

The routing parameters for the route

For example, here's a small section of the rake routes output for a RESTful route:

|  |
| --- |
| users GET    /users(.:format)          users#index            POST   /users(.:format)          users#create   new\_user GET    /users/new(.:format)      users#new  edit\_user GET    /users/:id/edit(.:format) users#edit |

You may restrict the listing to the routes that map to a particular controller setting the CONTROLLER environment variable

CONTOLLER=users rails routes

To get a ‘slightly’ more human readable version of the routes display you can add the expanded option to the rails routes command:

rails routes –expanded