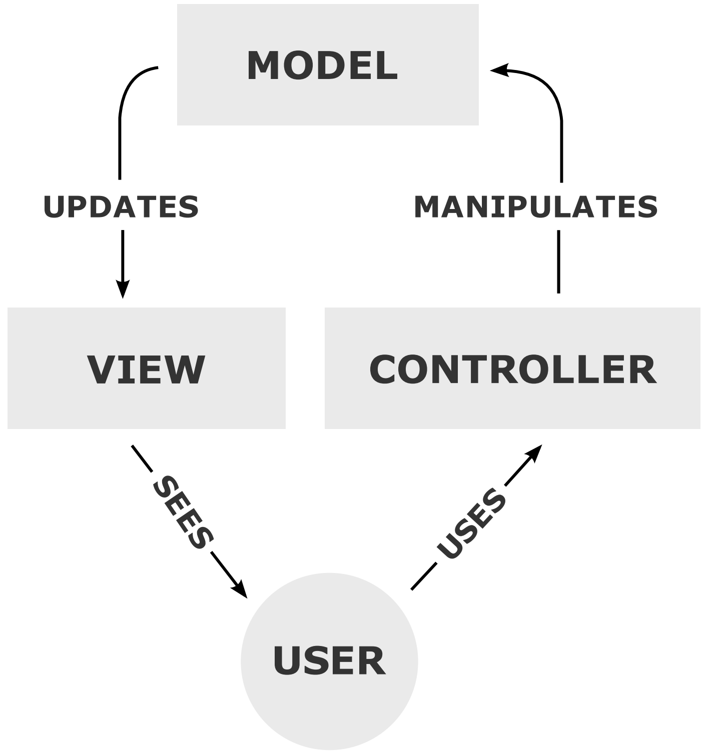
Ruby on Rails

About Ruby on Rails (Official Website)

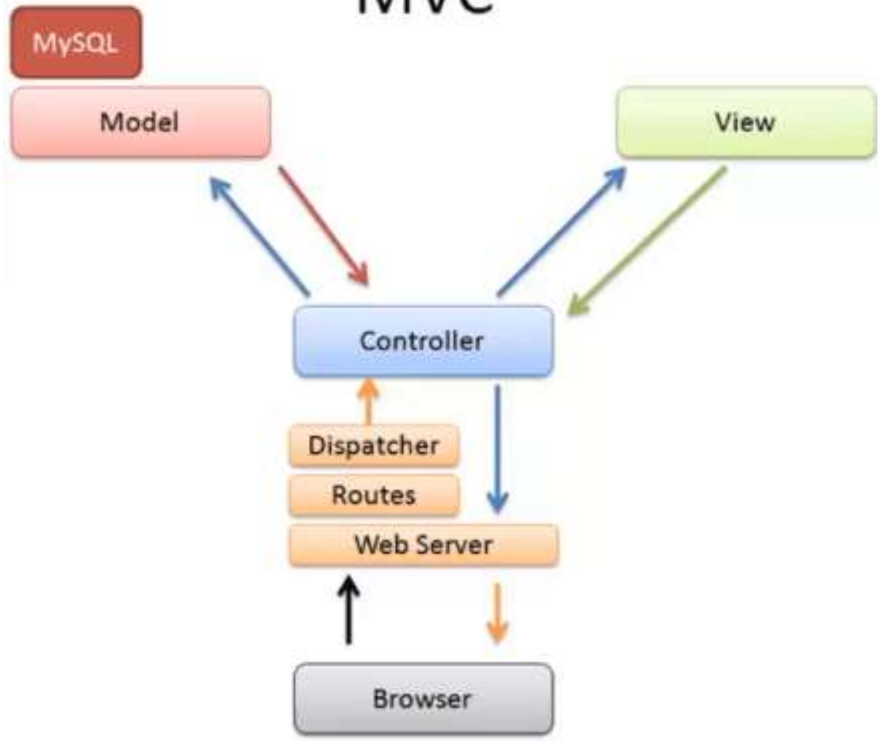
- Server-side rapid web application development framework
- Written in Ruby
- Follows the (MVC) Model-View-Controller software design pattern
- Other ideas / paradigms include:
    - (CoC) Convention over configuration
    - (DRY) Don’t repeat yourself
    - Active Record pattern (ORM system)
- A game-changer in the industry (inspired: Django [Python], Catalyst [Perl], Laravel [PHP], Sails.js [Node.js].)

Model View Controller

- **Models** – represent and interact with data.
- **Controllers** – are the "brain" of the app. Grab/sort data and format as a view. Communicates between model and view.
- **Views** – are HTML, XML, JSON, etc. sent to a client program / end-user. Templating engine (ERB.)



MVC


MySQL
Model
View
Controller
Dispatcher
Routes
Web Server
Browser

Libraries

- Rails is composed of a number of powerful Ruby libraries working together to form a cohesive rapid website application development system
  - Active Record: Data interaction layer (think User.all)
  - Action View: Templating engine (erb), provides helper functions (link_to, form_for)
  - Action Controller: Brains of the operation, business logic
  - Action Dispatch: Receives and routes incoming requests to correct controller(s)
  - Action Cable: Web sockets instead of HTTP request-response protocol in your app
  - Action Mailer: Send e-mails from application using mailer classes with views
  - and more…

Versions are Important

- With newer versions, most commands are under the “rails” command to make it more accessible and easier for newcomers
- Older versions often had more extensive commands… bin/rake bin/rails etc
- Check your versions using…
  - ruby --version
  - rails --version
  - If encountering an error, ensure you check commands and documentation for that specific version

Getting Started

- Let’s create a practice rails app (note this creates a new folder based on the project name)
  - **rails new your-project-name**
    - **rails new your-project-name --api** will remove the views, no server-side rendering
      - This is extremely useful if you’re using Rails as ONLY a backend and leaving the entirety of your front-end to something else… maybe a React app that uses JavaScript’s Fetch function or the Axios library to talk to your Rails app
        - Look into **render json** and **to_json** for use in controller methods if this is your approach!
    - **rails new your-project-name --database=postgresql** *or* **--databse=mysql**
    - Look into other generator modifiers for your projects, some things like this can be set up at the beginning in this way instead of fighting with the defaults later
  - rails s
    - Starts a Rails testing server via Puma

Gemfile

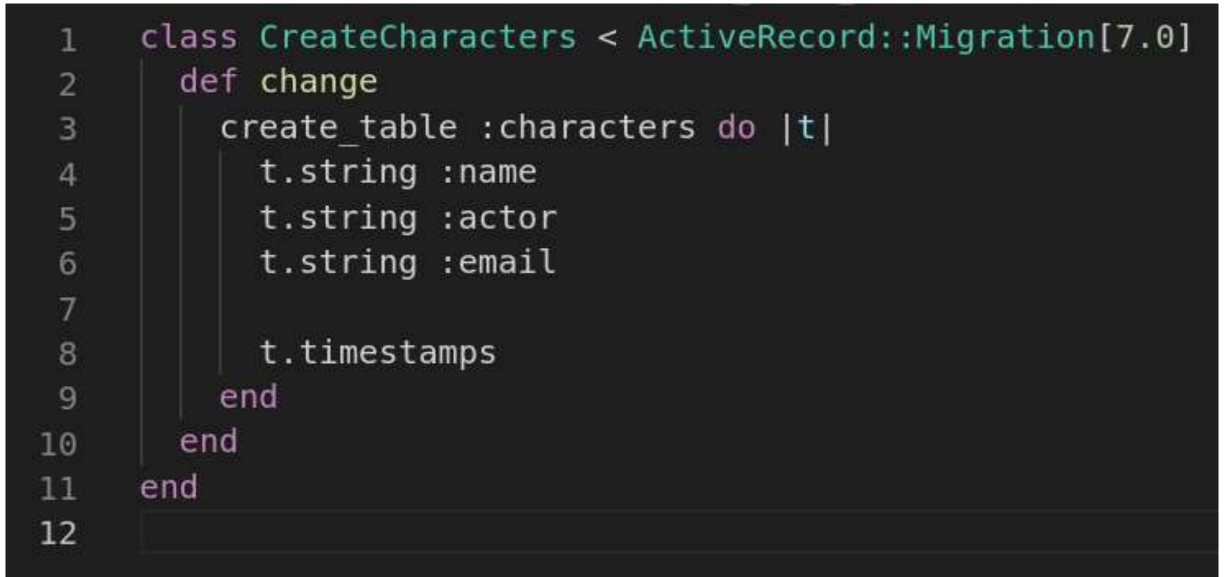
- Much like the package file included when using npm, but for gems in your Ruby [on Rails] application
- Groups are often used for separating items like “developer” or “production”
- Add to the development group of the Gemfile:
  - gem “faker”
- Equivalent to npm install is **bundle install**
  - In newer versions you can simply run **bundle**

The Console

- The Rails app can be accessed via console instead of web browser which is great for testing included classes, running commands, etc. without a formal user client
  - rails c
  - Faker
  - Faker::TvShows::TheITCrowd.character
  - exit
- If the above errors, ensure that Rails is set up correctly and that Faker is present in your project!

Generate a Model and Migration

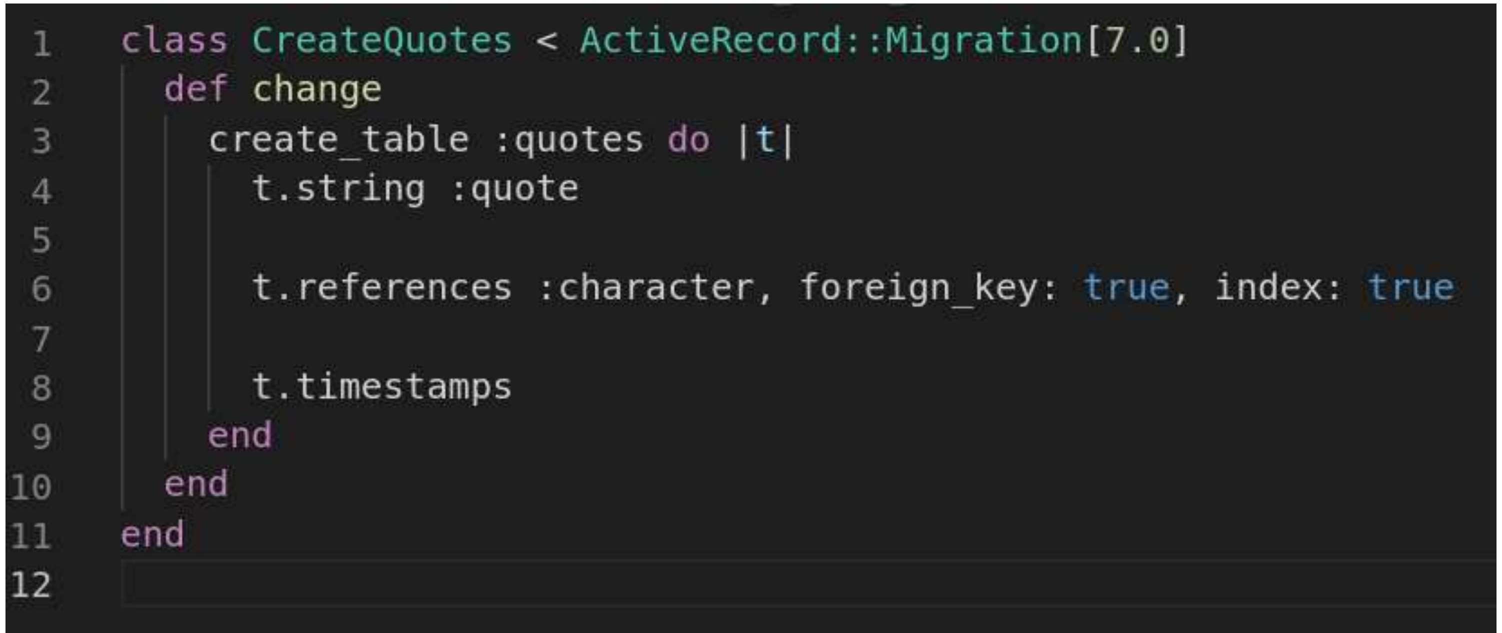
- rails g model Character
  - This creates a migration file in the db/migrate folder
  - It also generates a model file in app/models/character.rb
- Now that these are present, we can update the table structure accordingly via the migration file…



```ruby
class CreateCharacters < ActiveRecord::Migration[7.0]
  def change
    create_table :characters do |t|
      t.string :name
      t.string :actor
      t.string :email

      t.timestamps
    end
  end
end

```

Let’s try Another

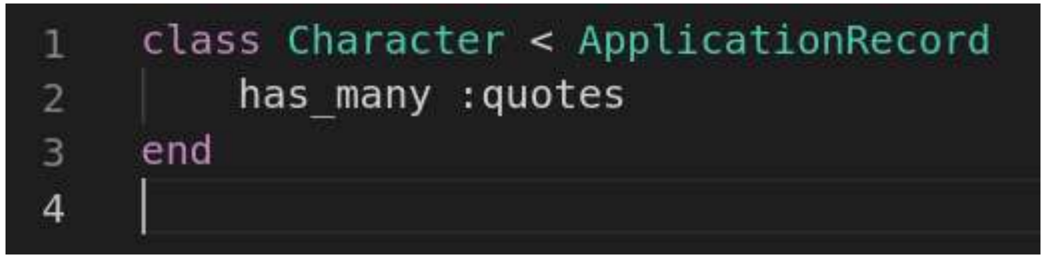
- rails g model Quote
  - We’ll update the migration file here as well, to ensure the appropriate columns are set up.
  - Since we want each quote to belong to a character, we’ll set up a table reference!



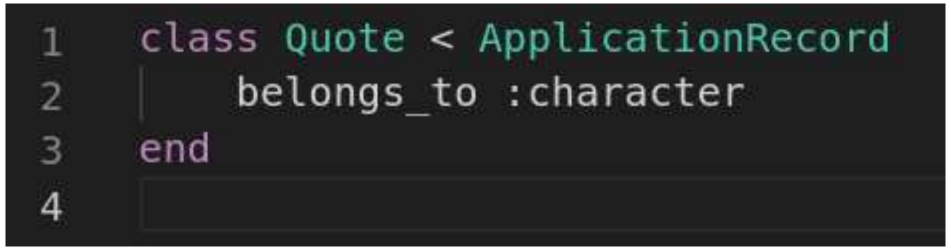
```
class CreateQuotes < ActiveRecord::Migration[7.0]
  def change
    create_table :quotes do |t|
      t.string :quote

      t.references :character, foreign_key: true, index: true

      t.timestamps
    end
  end
end

```

Relationships

- Now, for Rails and ActiveRecord to understand any relationships between our tables, it is paramount that we describe these in our models
- In this case, each character may have many quotes, so we'll update the models like so…



```ruby
class Character < ApplicationRecord
    has_many :quotes
end

```



```ruby
class Quote < ApplicationRecord
    belongs_to :character
end

```

Rails Migration Command

- In order to generate the tables we’ve described in our migration files, we can run the following Rails command:
  - rails db:migrate
- Awesome! The table is present, but not yet populated… our next goal will be adding our test data

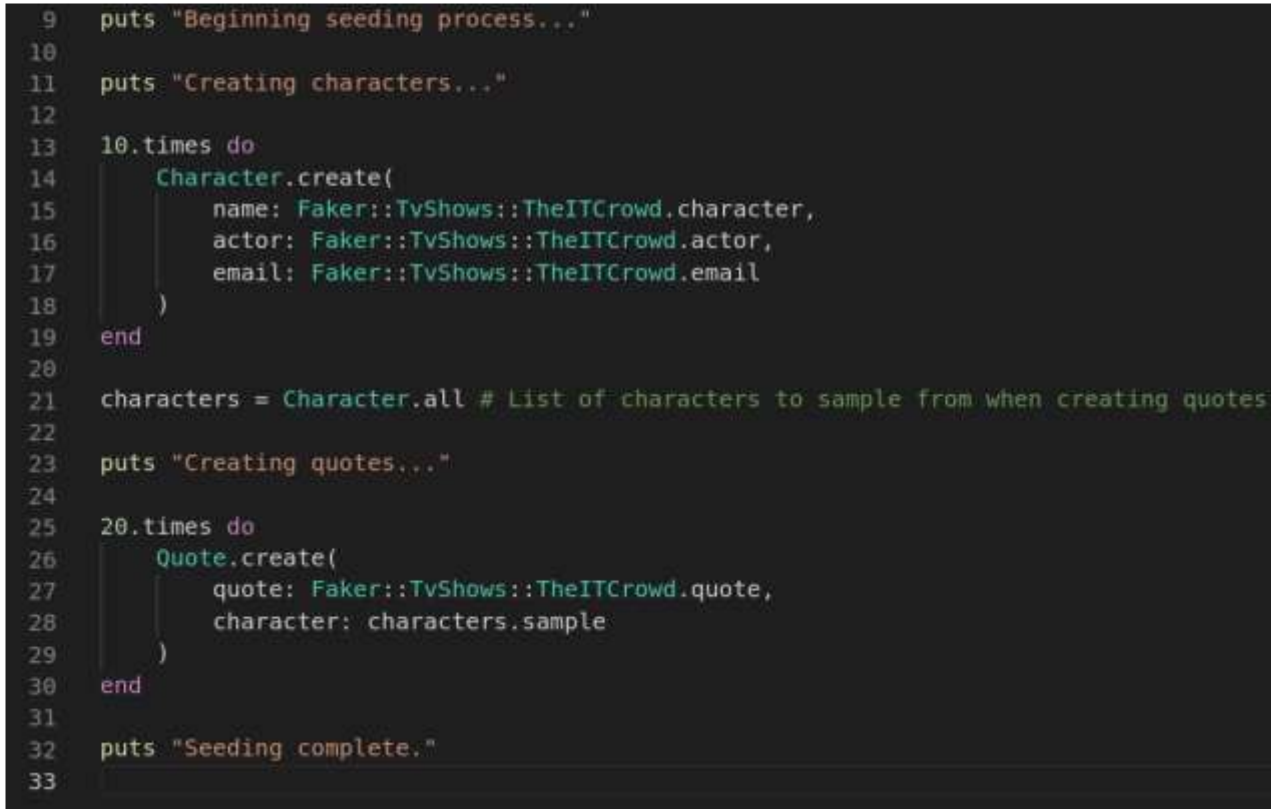
Faker

- For populating and testing outputs during development of an application, a library to quickly insert realistically-sized and formatted information saves a lot of time and effort
- The most widely-used and ported library for this is Faker!
- Earlier in this presentation, we asked you to add **gem “faker”** into your Gemfile and run the **bundle** command to have it installed—this will allow us to use this gem in populating our database

Seeding the Database

- Let’s fill that database with information from the Faker gem!
- Access db/seeds.rb and insert the following…

- Note how we can use .times to create high quantities of test characters and quotes very quickly!
- Run **rails db:seed** to have this file execute and add to your database



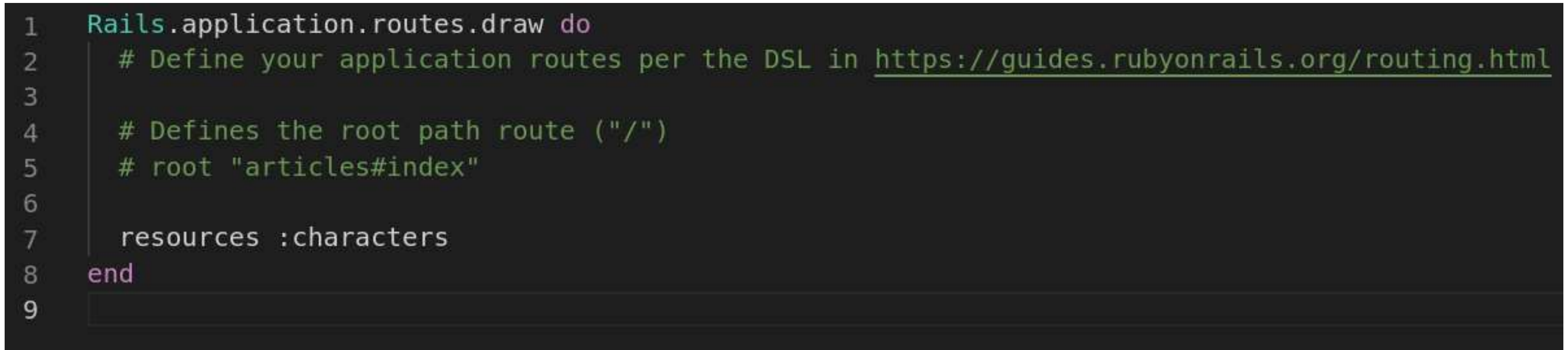
```ruby
puts "Beginning seeding process..."

puts "Creating characters..."

10.times do
    Character.create(
        name: Faker::TvShows::TheITCrowd.character,
        actor: Faker::TvShows::TheITCrowd.actor,
        email: Faker::TvShows::TheITCrowd.email
    )
end

characters = Character.all # List of characters to sample from when creating quotes

puts "Creating quotes..."

20.times do
    Quote.create(
        quote: Faker::TvShows::TheITCrowd.quote,
        character: characters.sample
    )
end

puts "Seeding complete."

```

Check that your Database is Populated

- Let’s try out the Rails console again, and see if we can find some data using ActiveRecord… (pay close attention to the output for each of these tests!)
  - rails c
  - Character.all
  - Character.find 2
  - Quote.all
  - example_quote = Quote.find 2
  - example_quote.character
  - exit
- Note how powerful ActiveRecord is, in helping us find the character attached to our quote! Because we described the relationship, Rails can help us very easily locate connected data like this

Routes

- On your website, users will often expect there to be pages at specific paths… for example, perhaps there is a welcome page on **/home**, and more information at the **/about** path
- The way we control what happens at a specific end-point or path is via the applications routes, these are adjusted via **config/routes.rb**
- Try the following in your file…
  - **resources :characters**



```ruby
Rails.application.routes.draw do
  # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html

  # Defines the root path route ("/")
  # root "articles#index"

  resources :characters
end

```

More on Routes

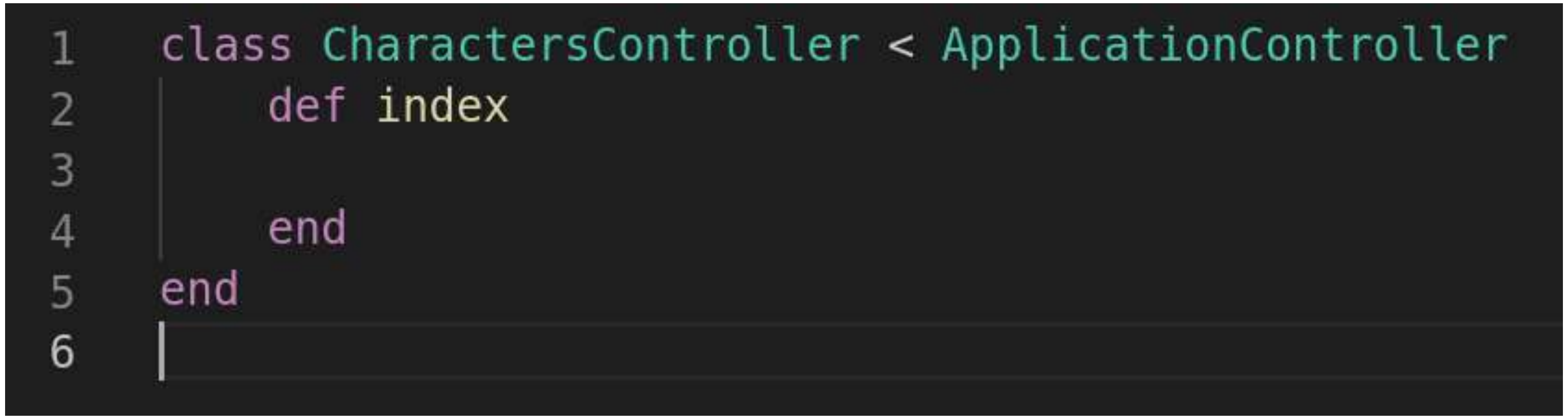
- Running **rails routes** in the latest version(s) of Ruby on Rails should output a full listing of configured routes, including our brand new characters paths
- Note that using “resources” like we did in our file will generate the standard CRUD screens for the resource (index, create, edit, display, and delete)
- If you try running **rails s** in the terminal and visiting **localhost:3000/characters**, however, you’ll be met with an error for now—why is that!?
- Next, we need a controller for Characters, to let the app know how to handle each of these routes

Index Route and Controller Method

- When using Rails' built-in "resources" routes, it is important to match your controller method names to the route names
- Traditionally, the page that lists all of the resource is referred to as the "index," let's start with that
- Firstly, generate the controller for your resource (in this case, Character)
    - **rails g controller characters**
- This generates a new file in **app/controllers**

Adding to your Controller

- We’ll need a method representing the index page to resolve the error you may have received earlier, let’s add that now:



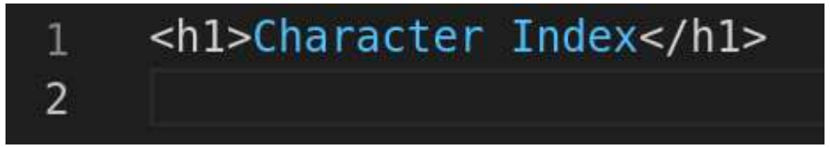
```ruby
class CharactersController < ApplicationController
    def index

    end
end

```

The M and the C… what’s Missing?

- We now have the M[odel] and the C[ontroller] present in the application for listing our characters… there’s one more letter in MVC though!
- That’s right, we’ll need a view—that is—a way we can format our data in HTML and send it to an end-user’s browser
- Recall that when a request from a user hits a Rails app, the routes are checked… assuming a route is matched to a controller, that controller gets to decide how that request is handled
- This means in our controller methods, we have to write any logic related to our output

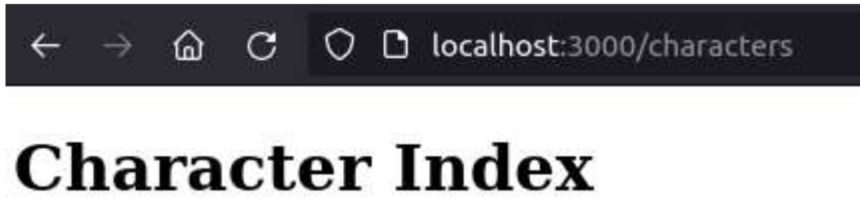
Let’s add the View

- In the **app/views** folder, by this point, there should be a **characters** directory (**app/views/characters**)
- If we want to do things the default and recommended “Rails way” we can simply add a file in here matching our controller method name, and it can work right out-of-the-box
- Create a file, **app/views/characters/index.html.erb**, and populate it like so for a test:



```
<h1>Character Index</h1>

```

- Run **rails s** and visit, in your web browser, **localhost:3000/characters**
- You should be met with the heading we added!

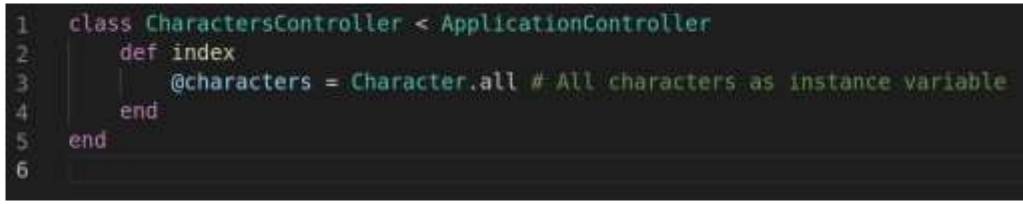


```
localhost:3000/characters
```

Character Index

Views and Controllers

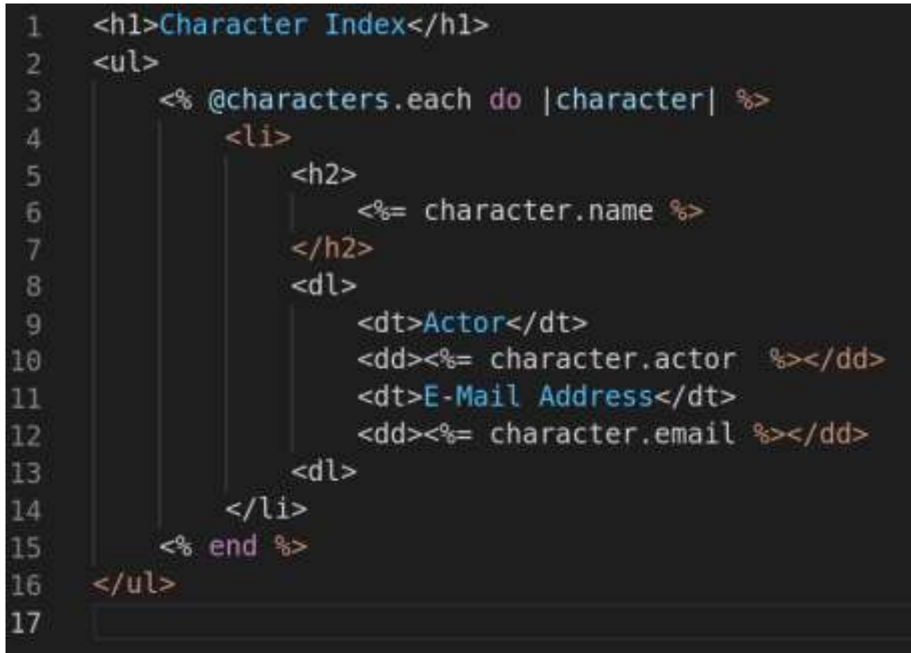
- We’re finally seeing our view! Now to use our controller to pass our view some data from our model… the point of an index screen is to list all of our data, so we’ll want to print out details about our characters here
- Rails is smart, and if we follow convention, will do a lot behind the scenes so we don’t have to worry about it
- Try the following in your controller file (**app/controllers/characters.rb**)
- Notice the use of the @ symbol at the beginning of line 3, this means it is an instance variable—in the controller Rails knows to pass this to your view automatically



```
1  class CharactersController < ApplicationController
2      def index
3          @characters = Character.all # All characters as instance variable
4      end
5  end
6
```

Access Controller Instance Variable in View

- Now that we're grabbing the characters in the controller as an instance variable, we can use this information in the connected view!
- Update your **app/views/characters/index.html.erb** file accordingly
- Since "@characters" is an array of characters, we can loop through and output each
- Note the syntax used in ERB:
  - We use a **<% %>** block to execute Ruby code
  - We use a **<%= %>** block to *puts*, or output, a value



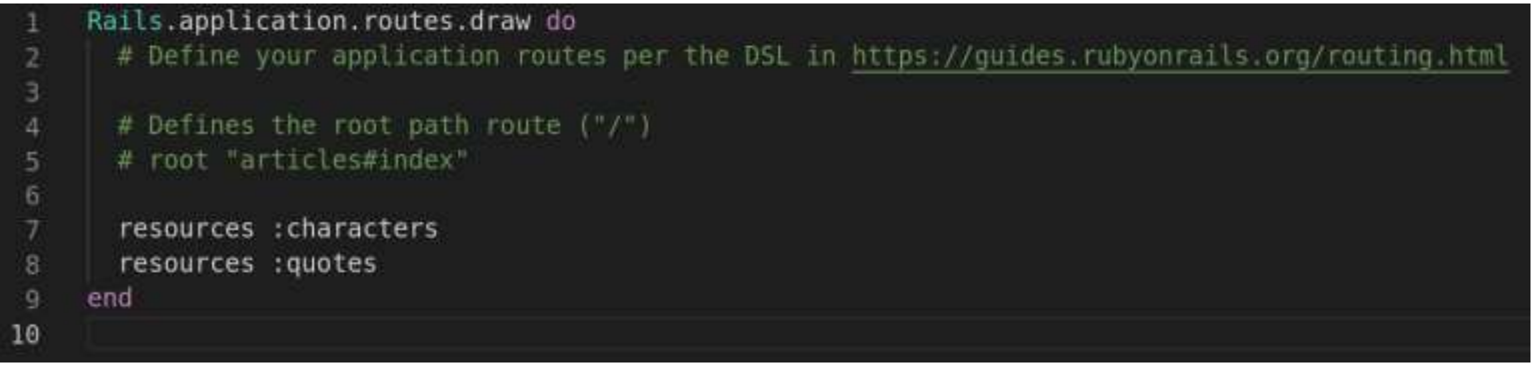
```
<h1>Character Index</h1>
<ul>
    <% @characters.each do |character| %>
        <li>
            <h2>
                <%= character.name %>
            </h2>
            <dl>
                <dt>Actor</dt>
                <dd><%= character.actor  %></dd>
                <dt>E-Mail Address</dt>
                <dd><%= character.email %></dd>
            <dl>
        </li>
    <% end %>
</ul>
```

Quotes Challenge

- See if you can do the same for Quotes…
- You’ll need:
  - An updated routes file
  - A new quotes controller
  - A new quotes index view

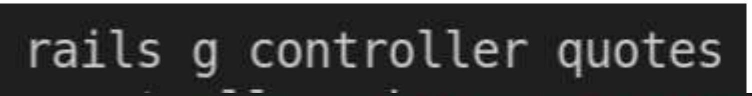
Quotes How-To

- Add quotes to config/routes.rb

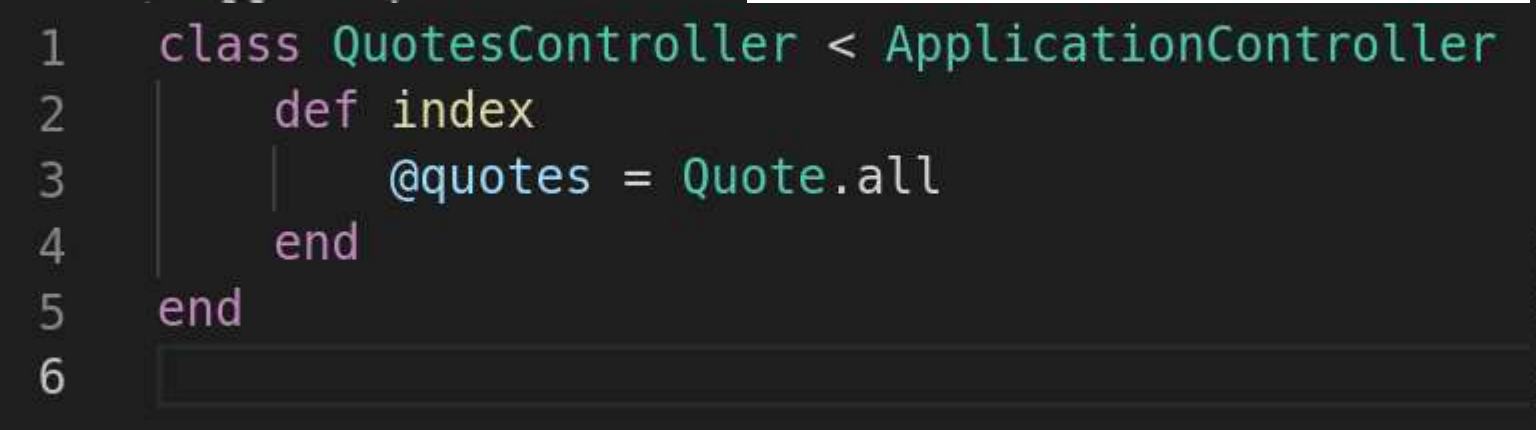


```ruby
Rails.application.routes.draw do
  # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.html

  # Defines the root path route ("/")
  # root "articles#index"

  resources :characters
  resources :quotes
end

```

- Quotes controller index method

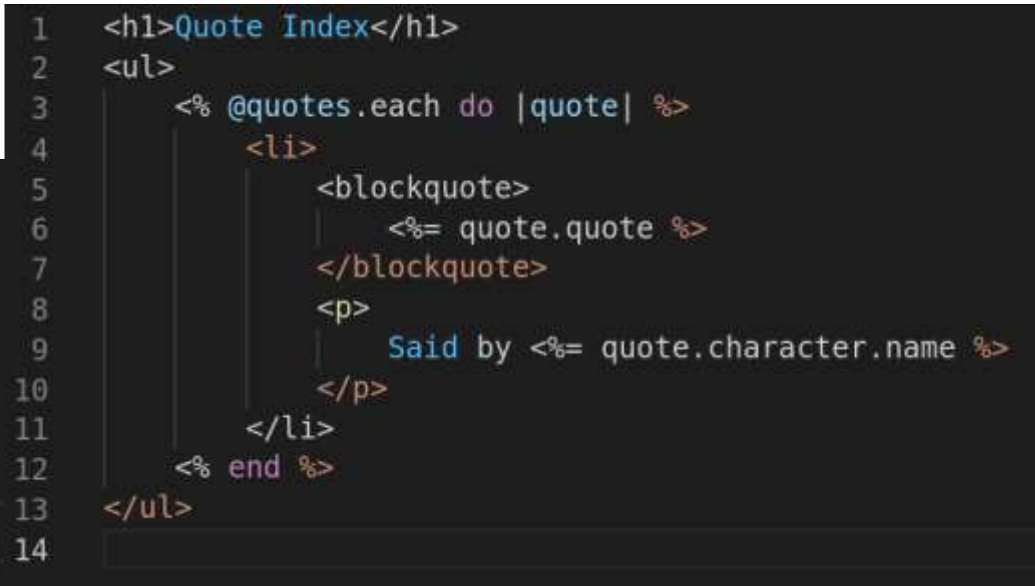


```
rails g controller quotes
```



```ruby
class QuotesController < ApplicationController
    def index
        @quotes = Quote.all
    end
end

```

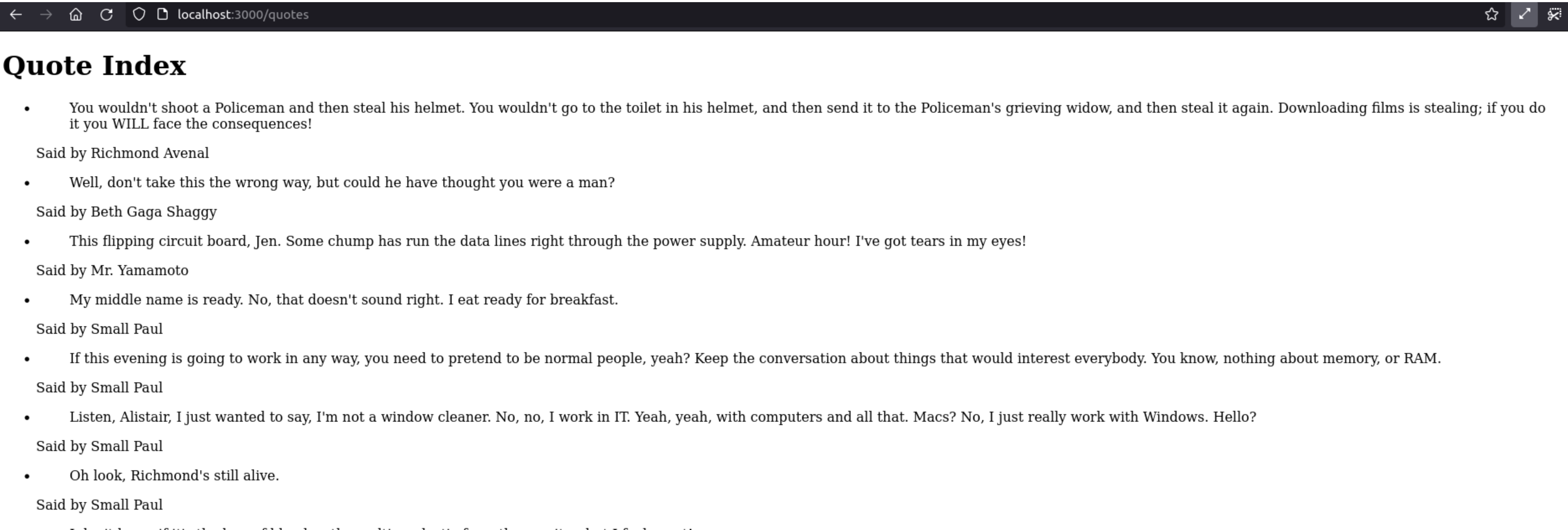
- Quotes index view
  - Notice how ActiveRecord knows what you mean by **quote.character.name**, how it uses the defined relationship in your tables to find the name of the associated character!



```erb
<h1>Quote Index</h1>
<ul>
    <% @quotes.each do |quote| %>
        <li>
            <blockquote>
                <%= quote.quote %>
            </blockquote>
            <p>
                Said by <%= quote.character.name %>
            </p>
        </li>
    <% end %>
</ul>

```

Quotes Index Page

- Once completed, ensure you run rails s and test the quotes index screen in your web browser:



Quote Index

- You wouldn't shoot a Policeman and then steal his helmet. You wouldn't go to the toilet in his helmet, and then send it to the Policeman's grieving widow, and then steal it again. Downloading films is stealing; if you do it you WILL face the consequences!

  Said by Richmond Avenal
- Well, don't take this the wrong way, but could he have thought you were a man?

  Said by Beth Gaga Shaggy
- This flipping circuit board, Jen. Some chump has run the data lines right through the power supply. Amateur hour! I've got tears in my eyes!

  Said by Mr. Yamamoto
- My middle name is ready. No, that doesn't sound right. I eat ready for breakfast.

  Said by Small Paul
- If this evening is going to work in any way, you need to pretend to be normal people, yeah? Keep the conversation about things that would interest everybody. You know, nothing about memory, or RAM.

  Said by Small Paul
- Listen, Alistair, I just wanted to say, I'm not a window cleaner. No, no, I work in IT. Yeah, yeah, with computers and all that. Macs? No, I just really work with Windows. Hello?

  Said by Small Paul
- Oh look, Richmond's still alive.

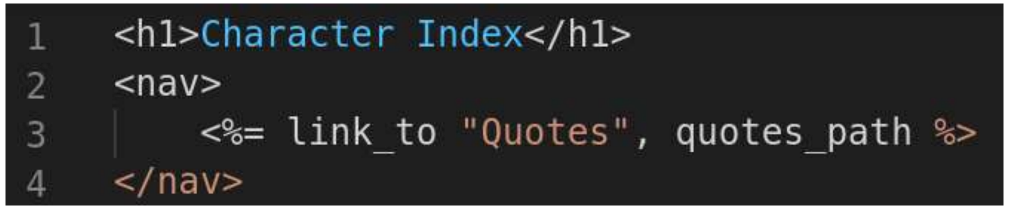
  Said by Small Paul

Links in Rails

- In HTML we know we can write an anchor tag (**<a>**) with an HREF property pointing to a web page, and if someone clicks this link their browser will bring them to that resource
- Ruby on Rails provides a helper function to assist us in more quickly and reliably enter links into our apps: link_to
  - This is composed of the function name, **link_to** followed by…
    - a string representing the text you’d like to have display in the anchor
    - a string representing the URL that the anchor should point to
- Let’s have our characters index and quotes index link to each other to test this out!

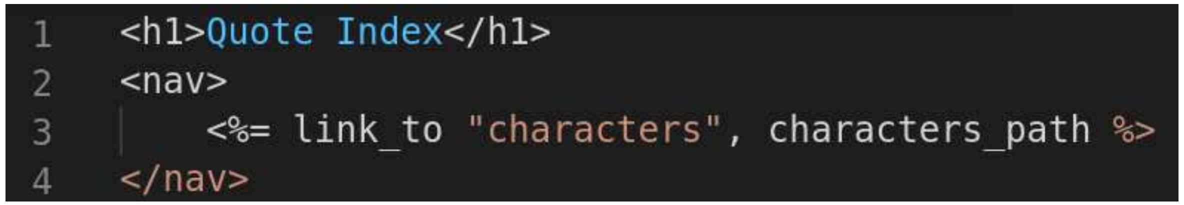
Add link_to to your Views

- Ruby on Rails resources generated these names for us, representing our various URLs:
  - characters_path
  - quotes_path
- Update your app/views/characters/index.html.erb file…



```
<h1>Character Index</h1>
<nav>
    <%= link_to "Quotes", quotes_path %>
</nav>
```

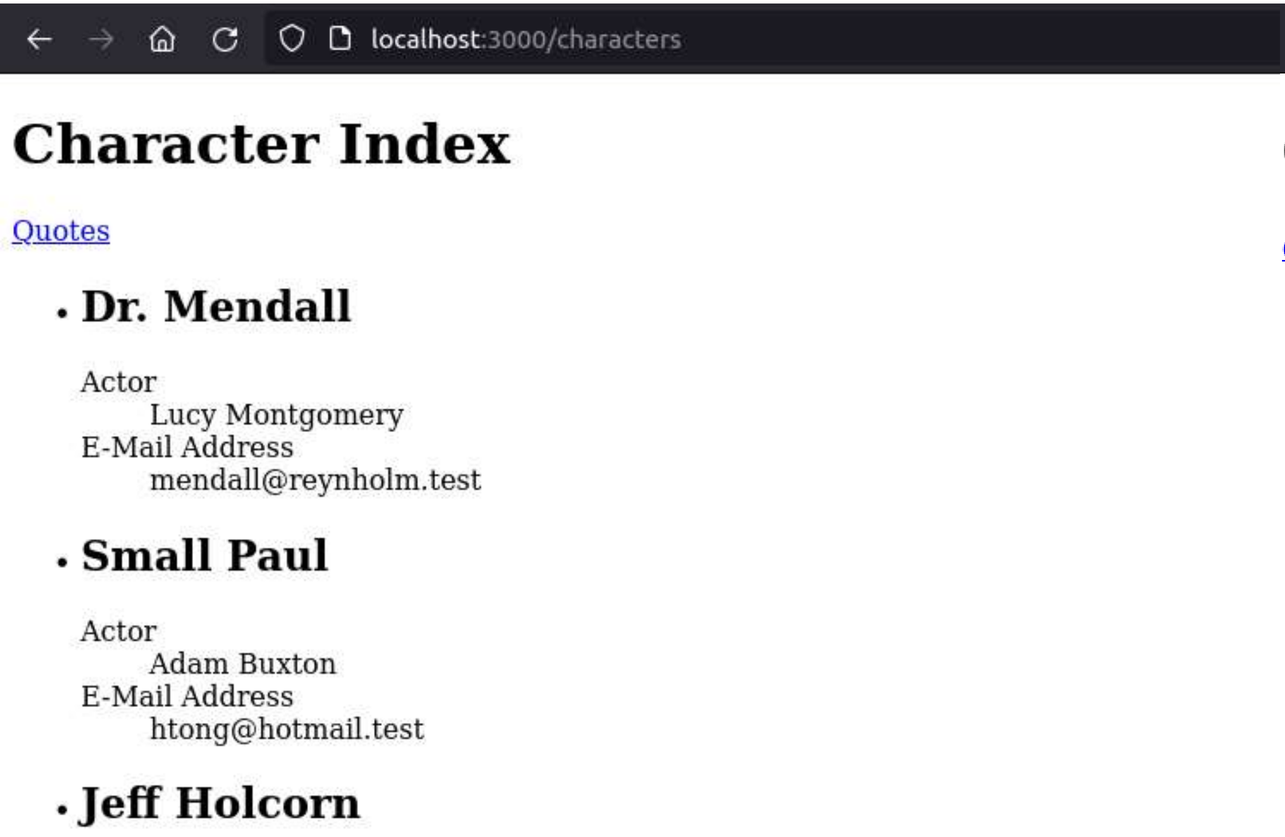
- Update your app/views/quotes/index.html.erb file…



```
<h1>Quote Index</h1>
<nav>
    <%= link_to "characters", characters_path %>
</nav>
```

Try out your Links in the Web Browser

- Ensure that the links on both screens do work!


localhost:3000/characters

Character Index

Quotes

- **Dr. Mendall**

  Actor

  Lucy Montgomery

  E-Mail Address

  mendall@reynholm.test

- **Small Paul**

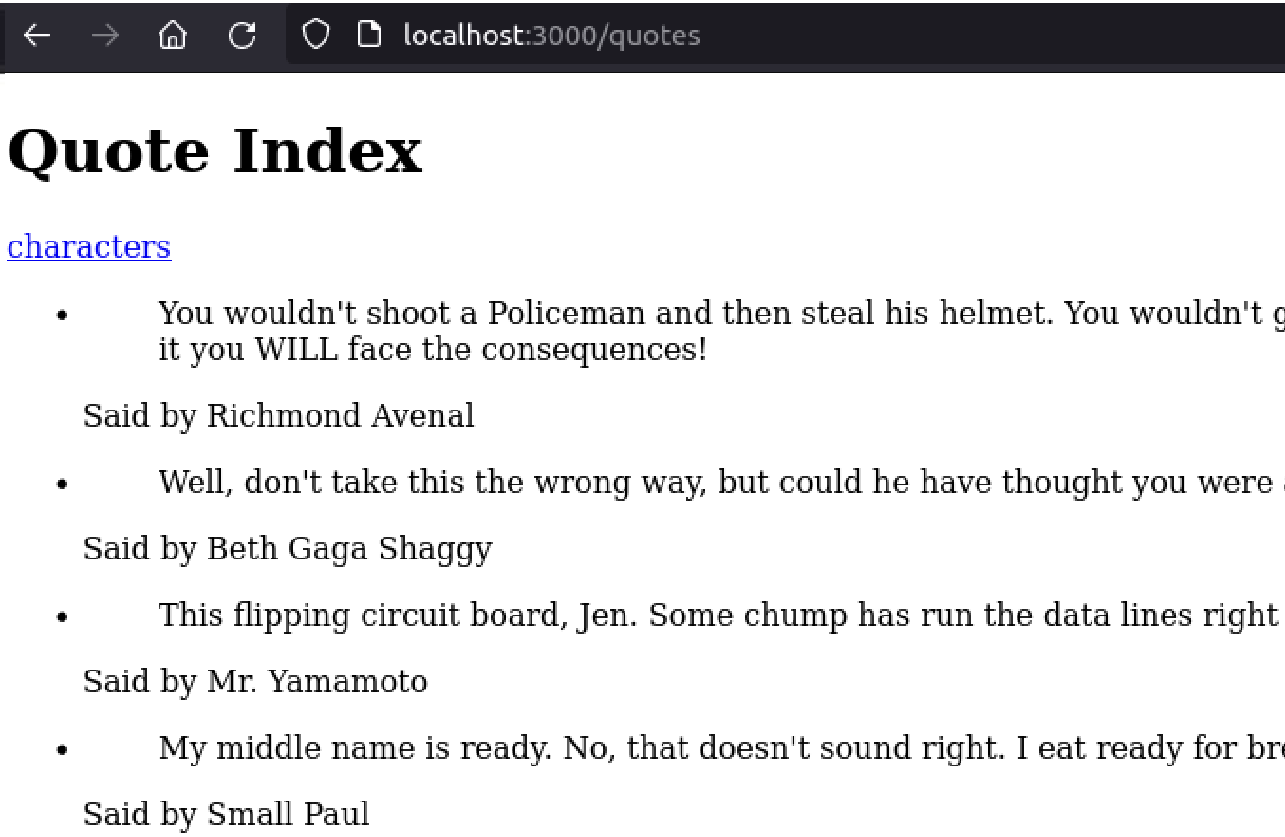
  Actor

  Adam Buxton

  E-Mail Address

  htong@hotmail.test

- **Jeff Holcorn**


localhost:3000/quotes

Quote Index

characters

- You wouldn't shoot a Policeman and then steal his helmet. You wouldn't g it you WILL face the consequences!

Said by Richmond Avenal

- Well, don't take this the wrong way, but could he have thought you were

Said by Beth Gaga Shaggy

- This flipping circuit board, Jen. Some chump has run the data lines right

Said by Mr. Yamamoto

- My middle name is ready. No, that doesn't sound right. I eat ready for br

Said by Small Paul

Exciting! We have 2 Index Pages Working!

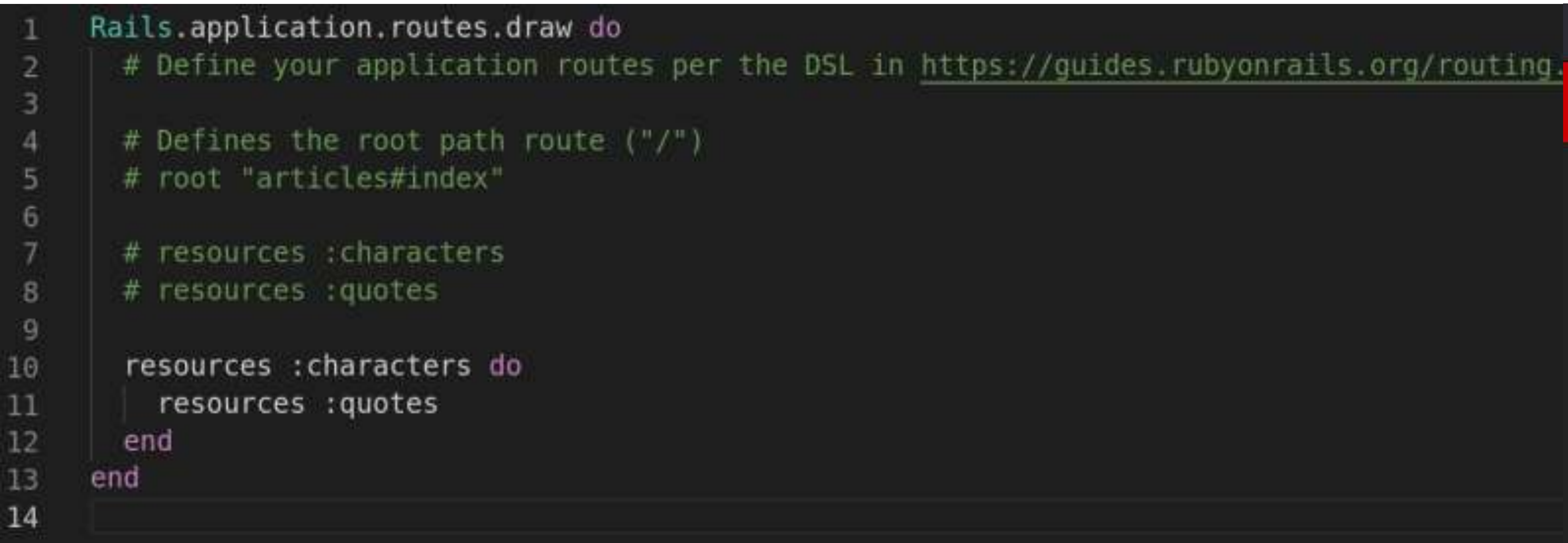
- Pat yourself on the back, you've got a piece of each the model, view, and controller all working together to show not one, but two resources and their related value!
- Note how quickly it is to develop with Ruby on Rails, it advertises itself as one of the most developer-friendly web app development frameworks, and potentially the fastest framework to use in development!
  - Note that other languages, interpreters, or compiled code may run faster than Ruby, and other frameworks may be able to scale better or run more efficiently… there are always pros and cons!
    - Do you need to develop quickly? Pick Ruby on Rails!

Nested Resources

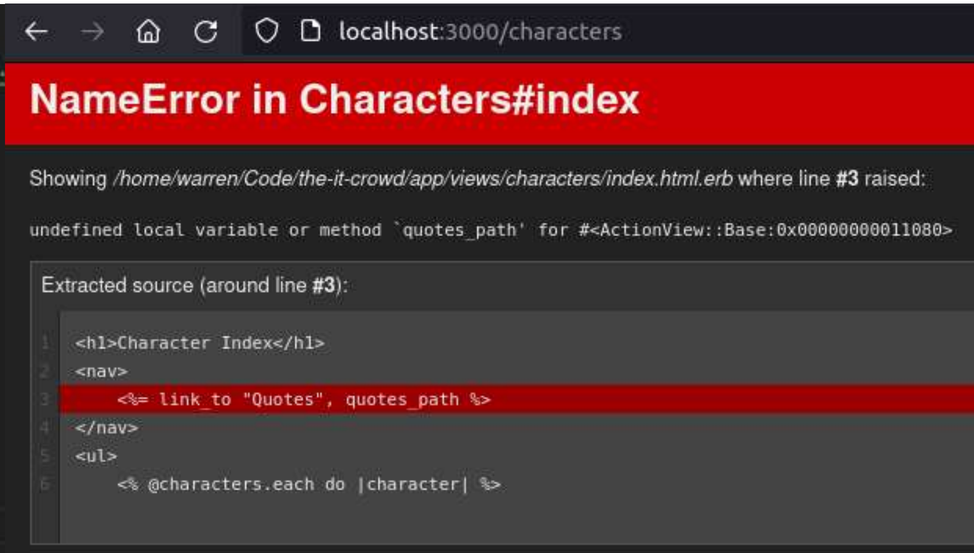
- Ruby on Rails also offers us the option to nest resources
- This is very useful in cases, like what we have here, where we have relationships between tables that end-users may want to see represented in a web page
  - For this particular application, remember we have characters, and each character may have multiple quotes associated with them
  - An end-user may want to view a character and *all* of their associated quotes
  - What would this look like in Rails?
    - There may be a few options, but let’s see a nested resource in action so that you have a nice example of this in-use

Update Routes Again

- Comment out the old “resources” routes, and lets try nesting one
- Characters can have many quotes, in this case, so we’ll nest quotes inside of our characters resources like so…
  - Note that this will break our characters link_to
  - Our original quotes paths have been lost!



```ruby
Rails.application.routes.draw do
  # Define your application routes per the DSL in https://guides.rubyonrails.org/routing.

  # Defines the root path route ("/")
  # root "articles#index"

  # resources :characters
  # resources :quotes

  resources :characters do
    resources :quotes
  end
end
```



localhost:3000/characters

NameError in Characters#index

Showing */home/warren/Code/the-it-crowd/app/views/characters/index.html.erb* where line **#3** raised:

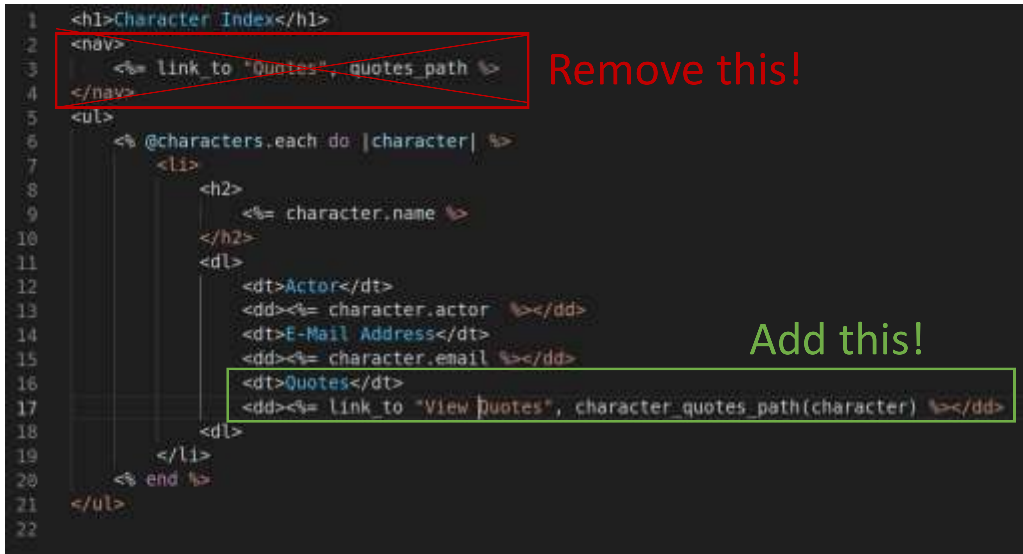
```
undefined local variable or method `quotes_path' for #<ActionView::Base:0x0000000000011080>
```

Extracted source (around line **#3**):

```
<h1>Character Index</h1>
<nav>
    <%= link_to "Quotes", quotes_path %>
</nav>
<ul>
    <% @characters.each do |character| %>
```

Updating the Characters Index

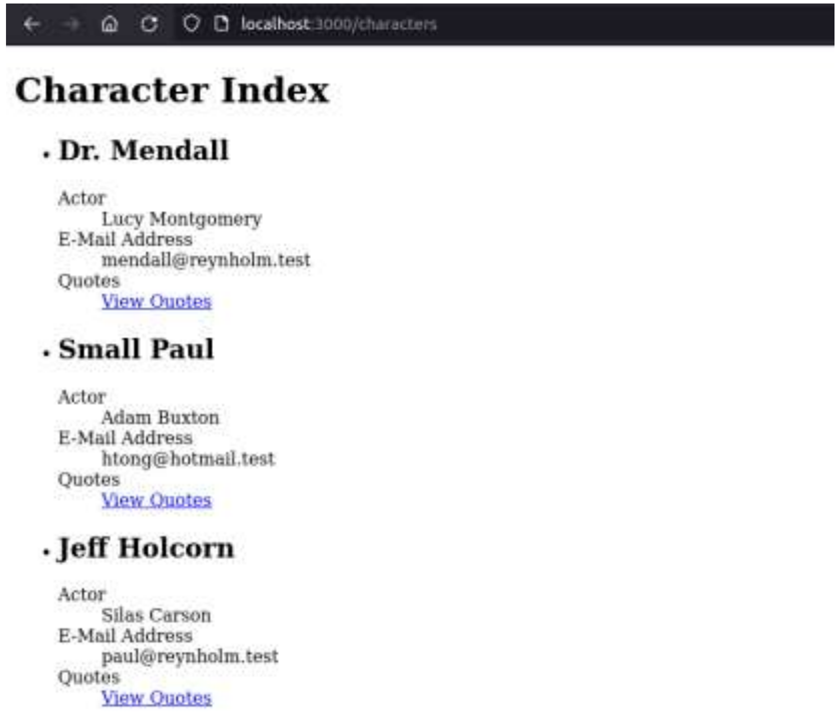
- Because of the new relationship, the generated routes will be a bit different for quotes, as they’re now nested in character entries
- Instead of **quotes_path**, Rails generated for us **character_quotes_path()**, requiring us to pass a character (or their corresponding ID integer) into the parentheses to let it know which character the quotes will relate to
- Remove the navigation we had placed in the app/views/characters/index.html.erb file, we’ll move the link down into the character loop
  - In here we’ll be able to easily pick out individual characters for each link
  - Now each link can send the user to a page that details quotes only associated with that *particular* character!



```
<h1>Character Index</h1>
<nav>
    <%= link_to "Quotes", quotes_path %>
</nav>
<ul>
    <% @characters.each do |character| %>
        <li>
            <h2>
                <%= character.name %>
            </h2>
            <dl>
                <dt>Actor</dt>
                <dd><%= character.actor  %></dd>
                <dt>E-Mail Address</dt>
                <dd><%= character.email %></dd>
                <dt>Quotes</dt>
                <dd><%= link_to "View Quotes", character_quotes_path(character) %></dd>
            <dl>
        </li>
    <% end %>
</ul>
```

Test the Character Screen

- Now have a look at the characters index…
- Note we have links as the end-user to view the quotes for any one character now!
- Try clicking one of these links, there are a couple of things we should yet have a look at…


localhost:3000/characters

Character Index

- **Dr. Mendall**

  Actor
  : Lucy Montgomery
  E-Mail Address
  : mendall@reynholm.test
  Quotes
  : View Quotes

- **Small Paul**

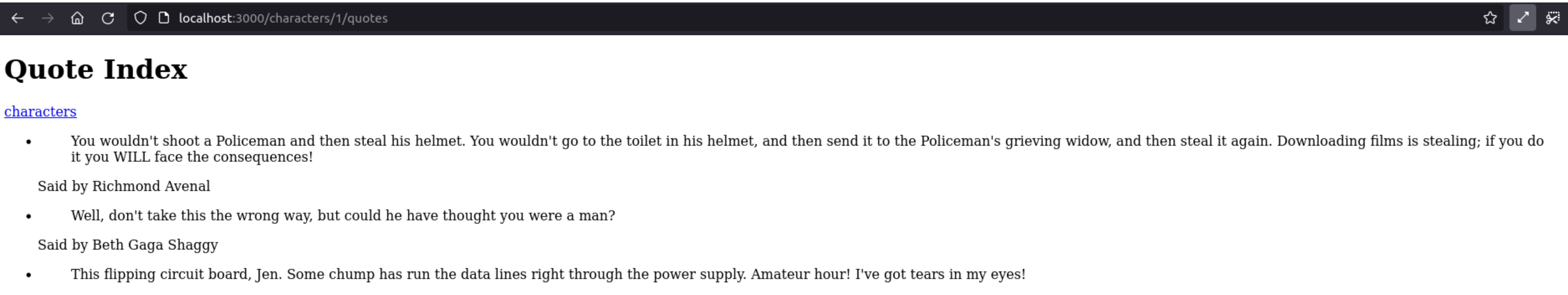
  Actor
  : Adam Buxton
  E-Mail Address
  : htong@hotmail.test
  Quotes
  : View Quotes

- **Jeff Holcorn**

  Actor
  : Silas Carson
  E-Mail Address
  : paul@reynholm.test
  Quotes
  : View Quotes

The Quotes Index View

- Having a look, we’ll notice that in the address bar the path shows the character ID that our quotes should be associated with—however, the controller is still pulling ALL the quotes in our database!
  - Bad news: we want to see only quotes for the character we selected, but we have all of them…
  - Good news: the URL includes the ID for the character we want to target, so we’ll be able to use that to filter the results!



localhost:3000/characters/1/quotes

Quote Index

characters

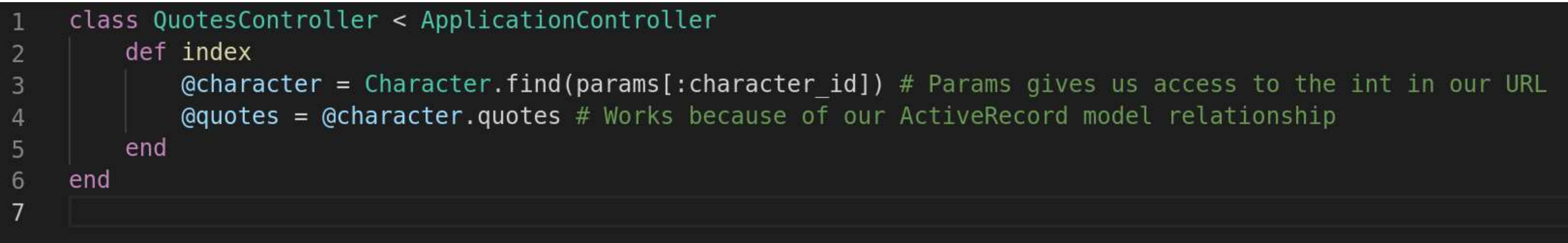
- You wouldn't shoot a Policeman and then steal his helmet. You wouldn't go to the toilet in his helmet, and then send it to the Policeman's grieving widow, and then steal it again. Downloading films is stealing; if you do it you WILL face the consequences!

  Said by Richmond Avenal
- Well, don't take this the wrong way, but could he have thought you were a man?

  Said by Beth Gaga Shaggy
- This flipping circuit board, Jen. Some chump has run the data lines right through the power supply. Amateur hour! I've got tears in my eyes!

Updates for Quotes Controller

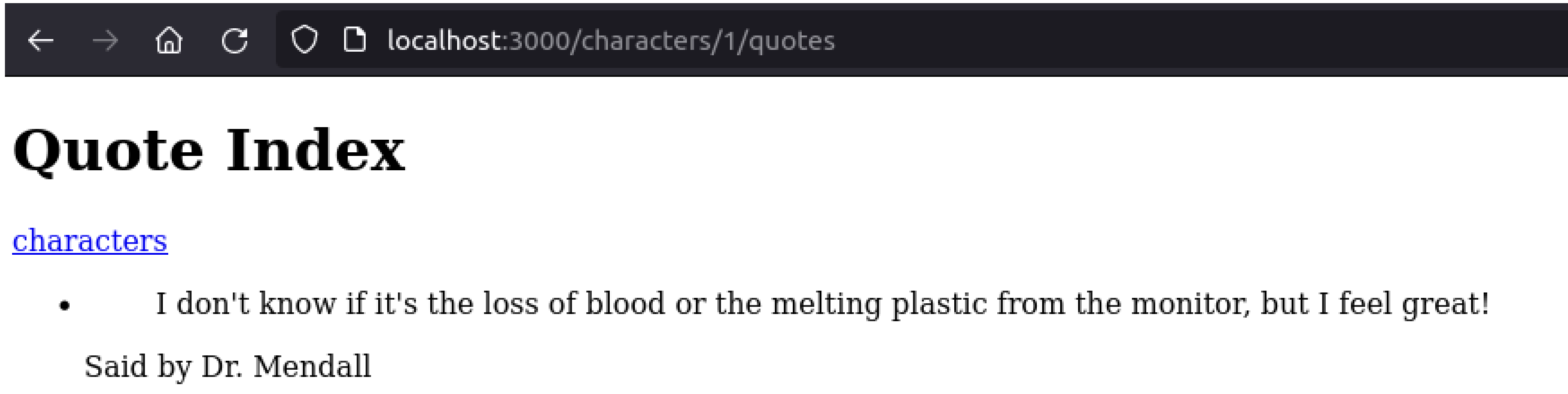
- Remove the contents of index, and let’s start fresh, there are two new things to make note of in our new adjustment…
  - There is a params variable we can access to grab values from our URL, we can use this to find the ID for the character we clicked on in our Characters index!
  - Because of the relationship we described in our models, that Character has many quotes, we are able to call upon the quotes of the character via a simple .quotes, see below how this works out in practice!



```ruby
class QuotesController < ApplicationController
    def index
        @character = Character.find(params[:character_id]) # Params gives us access to the int in our URL
        @quotes = @character.quotes # Works because of our ActiveRecord model relationship
    end
end

```

Try in the Browser Again…

- Now we're seeing only quotes related to the character we clicked on! This is exactly what we were hoping for.
- It is a bit unnecessary and redundant for us to see the "Said by…" message under each quote now, so lets move the character name to our main heading for the view instead



localhost:3000/characters/1/quotes

Quote Index

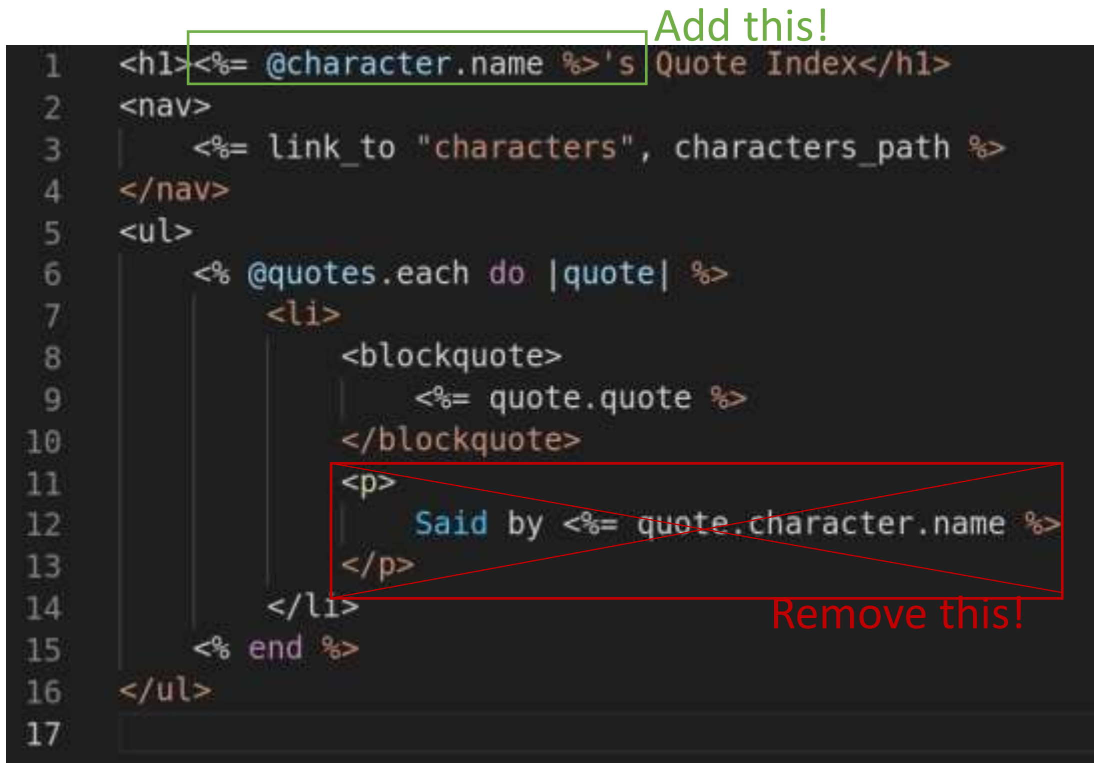
characters

- I don't know if it's the loss of blood or the melting plastic from the monitor, but I feel great!

  Said by Dr. Mendall

Minor Adjustment to the Quotes View

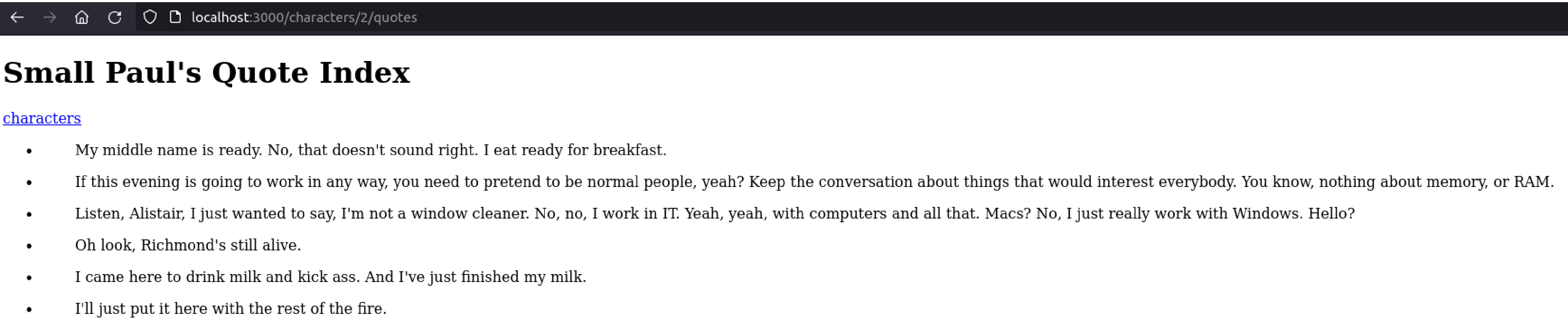
- Let’s simply print the current character’s name at the top of the page, once, so it isn’t repeated everywhere
- Don’t forget to remove the “Said by…” phrase, we won’t be needing that anymore!



```erb
<h1><%= @character.name %>'s Quote Index</h1>
<nav>
    <%= link_to "characters", characters_path %>
</nav>
<ul>
    <% @quotes.each do |quote| %>
        <li>
            <blockquote>
                <%= quote.quote %>
            </blockquote>
            <p>
                Said by <%= quote.character.name %>
            </p>
        </li>
    <% end %>
</ul>
```

Make Sure it Works!

- And there we have it… only quotes associated with the character, and we don’t have the name repeated everywhere—success!



localhost:3000/characters/2/quotes

Small Paul's Quote Index

characters

- My middle name is ready. No, that doesn't sound right. I eat ready for breakfast.
- If this evening is going to work in any way, you need to pretend to be normal people, yeah? Keep the conversation about things that would interest everybody. You know, nothing about memory, or RAM.
- Listen, Alistair, I just wanted to say, I'm not a window cleaner. No, no, I work in IT. Yeah, yeah, with computers and all that. Macs? No, I just really work with Windows. Hello?
- Oh look, Richmond's still alive.
- I came here to drink milk and kick ass. And I've just finished my milk.
- I'll just put it here with the rest of the fire.

Did you know…

- Instead of creating view files, and running Model and Controller generation separately, all of these steps can be done in one Rails command?
- If you know you'll require basic CRUD and all the controller methods and associated views for each, look into the Ruby on Rails scaffold command
    - This is an extremely powerful and time-saving command
    - Try creating a rails app and using this command to create a new resource
        - Note that it even adds a resource reference to your routes!