Rails

Model-View-Controller (MVC) Framework

- Persistence $\triangleq$ db
- Logic $\triangleq$ your code and rack appserver
- Presentation $\triangleq$ nginx
- Client $\triangleq$ browser

Controller

- Stored in `controllers/` directory
- Get data from models
  - Stored in `models/` directory
  - Gets data from db
  - Subclasses of `ApplicationRecord`
- Passes data to view
  - Stored in `views/` directory
  - Subclasses of `ActionView`
- Subclasses of `ApplicationController`

Config

- Routes (stored in `routes.rb`) map incoming URLs to controller actions and extract away optional params
- `:value` in defined route maps the value to the requested value in the path
  - Available in `params`

Philosophy

- Convention over configuration
- Is naming follows certain conventions, no need for config files
- DRY
- Relies on introspection and metaprogramming, blocks, modules, and symbols

Commands

- `rails new` makes a new app
- `rails g[enerate]` scaffolds a new object
  - Can specify columns and attributes with `name:type` params
- `rails s[erver]` runs the app
- Can use `--pretend` or `-p` flag to dry run and show changes

SQL

- Rails generates SQL statements at runtime
- CRUD $\triangleq$ {`INSERT INTO`, `SELECT`, `UPDATE`, `DELETE`}
- Database table name derived from model's name (lowercase)
- Database table column names are getters and setters for model attributes
  - Getters and setters do not modify instance variables

Model

-
- Created in memory

```ruby
# Our "complete" Movie class:
class Movie < ApplicationRecord ; end
# ApplicationRecord is a ActiveRecord::Base instance.

movie = Movie.new
movie.title = 'The Help'
movie.rating = 'PG-13'

# OR
movie = Movie.new(:title => 'The Help', :rating => 'PG-13')
```

Figure 1: Screenshot_2023-09-14_at_3.55.00_PM.png

- Must call `save` or `save!` to persist the data to the db
- `create` does both `new` and `save`
- All record have an `id` method if it has been saved
- `DataMapper` associates separate mapper with each model
  - Alternative to `ApplicationRecord`

Routing

- Stored in `config/routes.rb`
- eg: `get 'pastries/:flavor'=> 'pastries#eat', as: 'eat_dessert'`
  - Says call method eat on `PastriesController`
  - Populates `params[:flavor]` from URL
  - Creates an action `eat_dessert_path` that returns the path
-
  - Can auto generate these routes by adding `resources :movies` in `routes.rb`
- Can create an HTML form for a path by using `form_for`
- Can show routes and actions from bash with `rails routes` or `rake routes`

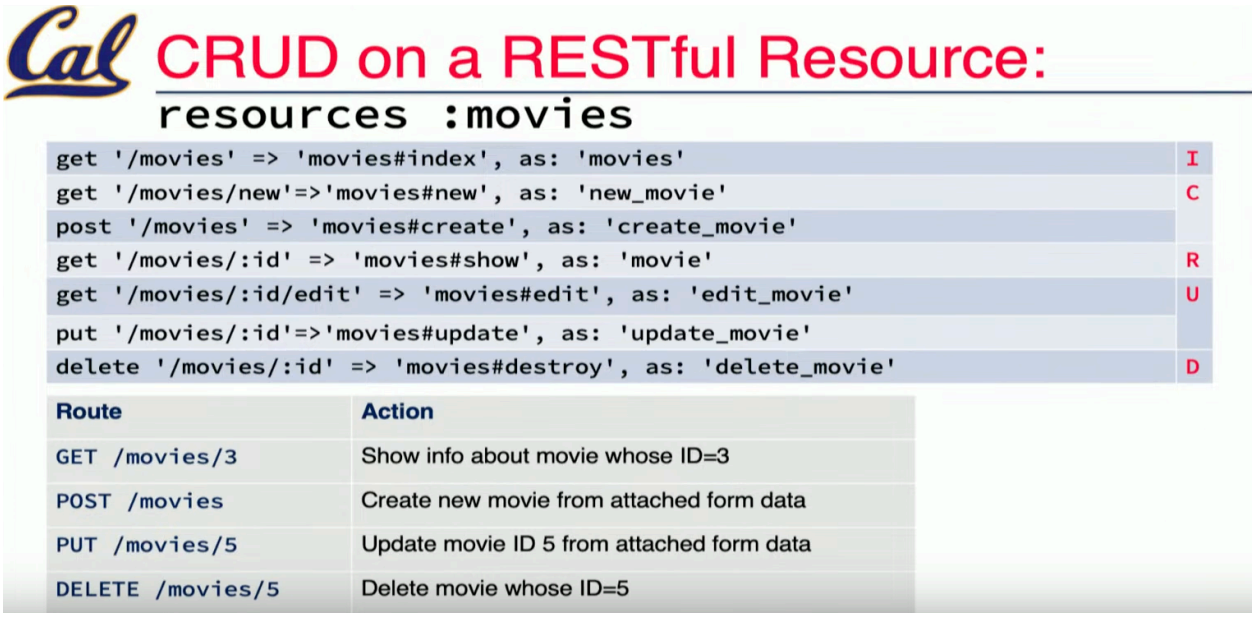

Figure 2: Screenshot_2023-09-14_at_5.12.55_PM.png