DanBooks

By Daniel kinyo

Overview

The application will be used to retrieve the book listing from the New york Times public api

(https://api.nytimes.com/svc/books/v3/lists/names.json?api-key=APIkey)

where books are listed with the relevant points they can be purchased from (i.e Amazom.com) along with other information including being featured in the current best seller list etc.

After retrieving this books the users can then;

- add these books to their library ,
- follow the relevant website of choice to buy the book i.e www.amazon.com.
- recommend the book to anyone using their email.
- Also the users can add reviews for these books.

Language of choice is Ruby on rails no added dependencies for stack used

Github repository https://github.com/kinyodan/danbooks.git

Assumptions Made

1. No Authentication or user login/logout added, but a user_token cookie has been used for session designation and each cookie user_ token is taken to be a single user.
2. Users can add as many reviews as they can with no restriction, so one user can add multiple reviews for the same book.
3. A users_library items are unique so no duplication is allowed user cannot add same book to library
4. Recommendations are single email only so no mailing list and also no duplication allowed
5. No mailer actions added for recommendations only saving is done forwarding of emails negated but can be easily implemented with relevant mailers.

Specifications

Ruby version used ruby 3.0.4p208 (2022-04-12 revision 3fa771dded) [x86_64-linux]

RAILS Version used Rails 7.0.5

Bundler version 2.2.33

Customs gems added outside the rails new app scaffold auto generated are

```
gem 'figaro'
gem "rspec-rails"
gem 'factory_bot_rails'
gem 'faker'
gem 'reek'
gem 'rouge-rails'
gem 'rubocop', require: false
gem 'standard', group: %i[development test]
```

Bootstrap and Jquery included from CDN for front end view styling

Code Specifications

Code of highly commented so ease of understanding my thought flow.

Most of the code is to be found in the "**controllers/concerns**" folder.This is to make code easily reusable and also isolate large code sets out of the models or controllers and when needed can simply be called from outside.

All the concerns have been included inside applicationController making them globally available and reusable across all child controllers

Very little large code sets are to be found in controllers so the controllers can be thin,most of the work is done in the relevant concerns file which are designated with the word "**manager**", i.e **ReviewsManager** inside the file "**reviews_manager**" module concern handles all code related to reviews no matter where the code is implemented be it inside the reviews controller or not.

Testing

Testing tools;

- Rspec
- Rails 7 out of the box inbuilt testing

I decided to retain the already generated tests present in test folder , that are generated as part of rails 7 scaffold generators and used this test to cover controller tests as they covered all the relevant testing parameters and also more unit tests , which effectively gave the application near 100 % test coverage so , kindly run both sets of tests.

Rspec: using the "rspec" command

Rails 7 out of box tests: run using the "rails test" command

NB: incase of bundling errors delete lockfile , even though lockfile has been ignored when pushing to github, but just as a point of note incase of problems caused by differences in bundler versions , rails versions or even Ruby distributions one may be using.