Thiết kế chi tiết server-side cho API của FeelEvent

1PAC. INC.

Version 0.1.1
2016-09-08

##

1. MỞ ĐẦU

Đây là bảng thiết kế chi tiết server-end của trang **World School** (FeelEvent), một trang được tách ra từ Feelnote .

1.1. Chú thích về mặt kỹ thuật

- Vì các thông tin về hội thảo (event) cũng được sử dụng ở Feelnote nên tất cả các bảng (table) của cơ sở dữ liệu sẽ được cấu trúc tại Feelnote DB .
- Mặc dù giao diện của Feelnote được thiết kế theo cơ cấu SPA (Single Page Application) , nhưng trang **World School** là một trang web mở được công bố rộng rãi, nên dựa trên quan điểm SEO giao diện của nó sẽ được thiết kế theo phương thức truyền thống MPA(Multi Page Application) .

1.2. Cấu hình hệ thống

- Nginx
- Ruby (phiên bản 2.3.1)
- Ruby on Rails (phiên bản 5.0.0.1)
- MySQL (phiên bản 5.7)
- Redis

1.3. Nguyên tắc code

- Nguyên tắc code cơ bản
  - https://github.com/fortissimo1997/ruby-style-guide/blob/japanese/README.md
  - https://github.com/fortissimo1997/ruby-style-guide/blob/japanese/README.ja.md
- Các nguyên tắc code khác của Ruby (bổ sung cho nguyên tắc cơ bản ở trên)
  - https://github.com/styleguide/ruby
- Nguyên tắc code cơ bản cho Rails
  - https://github.com/bbatsov/rails-style-guide/blob/master/README.md

1.3.1. Nguyên tắc code riêng của 1PAC

Mục định nghĩa

- Không thêm dòng trống vào sau `class`.
- Không thêm dòng trống vào giữa 2 `end` cuối cùng.

Cách dùng unless, if đặt sau

Các nguyên tác tối thiểu dưới đây được đặt ra để giữ cho code vẫn có thể dễ hiểu khi có nhiều xử lý phân nhánh if.(trừ một số ngoại lệ).

- Chỉ được áp dụng if, unless đặt sau khi sử dùng `return`, `raise` để thực hiện ngưng xử lý. Ví dụ:

```
return unless @user.signed_in?
```

- Đối với trường hợp thêm xử lý hoặc đặt giá trị cho biến số và tiếp tục xử lý sau đó, thì dù cho có là 1 dòng đi nữa cũng phải dùng cấu trúc if bình thường. Ví dụ:

```
if @flow.init?
  @user.assign_attributes @flow.get_data(:form)
end
```

1.4. Bổ sung

1.4.1. Thiết kế chi tiết sử dụng cả JSON Schema

- Các thông tin như đường dẫn của API, tham số (Request), cách thức trả lời (response) được quản lý dưới dạng `JSON Schema`.
    - Tên tập tin thiết kế API tương ứng sẽ được ghi tại dòng `JSON Schema` trong mỗi phần.

Tập tin dạng Markdown được lưu tại thư mục `doc/api_design/build/ja/` trong cùng Github.

1.4.2. Tên của các cột của bảng trong database

Nội dung của các tham số trong bản thiết kế được ghi theo loại(type) và giá trị(value) của Rails (ActiveRecord) không phải ghi theo loại và giá trị thực trên cơ sở dữ liệu.

1.4.3. Ví dụ

- Trên cơ sơ dữ liệu
  - *event_items.published* là loại `tinyint` và có giá trị là `1:Published`, `0:unpublished`
- Cách ghi trên thiết kế
  - Khi *event_items.published* có giá trị là `true(published)` thì làm xxx…

2. Cấu trúc thư mục

Dưới đây là thông tin về cấu trúc thư mục(directory) và tập tin (file) của application.
* Chỉ trích dẫn các directory và file chính.

```
.
├── Gemfile      # File quản lý, thiết định các gem cơ bản
├── Gemfile.lock # File quản lý, giữ phiên bản tất cả các gem sử dụng trong app
├── Rakefile     # Thiết định các task có thể chạy từ terminal
├── app/         # Thư mục chính lưu giữ application
│   ├── assets/      # Thư mục chứa các file hình ảnh và style sheets
│   ├── cells/       # Thư mục chứa có xử lý logic của gem Cells
│   ├── controllers/ # Thư mục chứa các controller của Rails
│   ├── helpers/     # Thư mục chứa các module hỗ trợ cho Rails
│   ├── mailers/     # Thư mục chứa các controller cho xử lý email
│   ├── models/      # Thư mục chứa các định nghĩa model của Rails
│   └── views/       # Thư mục chứa các view của Rails
├── config/ # Thư mục chứa các file thiết định của project, application
│   ├── application.rb # File ghi thiết định chung của application cho mọi môi trường
│   ├── boot.rb # File khởi động chính
│   ├── database.yml # File ghi thiết định cơ sở dư liệu(database)
│   ├── environment.rb # File thiết định các môi trường
│   ├── environments/ # Thư mục chứa file ghi thiết định riêng cho mỗi môi  trường
│   ├── initializers/ # Thư mục chứa các file thiết lập cơ bản ban đầu
│   ├── locales/ # Thư mục chứa các file từ điển định nghĩa chung
│   └── routes.rb # File thiết lập routing của Rails
├── config.ru # File thiết định của Racka
├── db/       # Thư mục chứa các thông tin của database
├── doc/      # Thư mục chứa các tài liệu hướng dẫn, thiết kế
├── lib/      # Thư mục chứa các libray của Rails
├── log/      # Thư mục chứa các file log
├── public/   # Thư mục gốc của web server
├── spec/     # Thư mục chứa các file code kiểm tra (test code)
├── tmp/      # Thư mục chứa các file, thông tin lưu dữ tạm thời
└── vendor/   # Thư mục chứa các library bên ngoài (ví dụ gem của rails)
```

3. Tips

Hướng dẫn chung về nhưng `Gem` khác.

3.1. Decorator (Draper)

Gem tạo ra một lớp Presenter(Draper) nằm giữa view và model nhằm giảm thiểu các logic gây khó hiểu không cần thiết trong model và view.

3.1.1. Ví dụ

app/decorators/user_decorator.rb

```
class UserDecorator < Draper::Decorator
  delegate_all

  def full_name
    "#{first_name} #{last_name}"
  end
end
```

app/views/me/sample.html.erb

```
<h1><%= @user.full_name %></h1>
```

Các thông tin khác của Draper có thể xem tại link Github dưới
https://github.com/drapergem/draper

4. Các lưu ý khi lập trình

Dưới đây là 2 lưu ý chính trong quá trình lập trình. * Sử dụng `RSpec` để thực hiện các kiểm tra. * Bắt buộc phải tạo `Pull Request` để có thể kiểm tra chéo code của nhau.

4.1. Sử dụng RSpec để kiểm tra trong lập trình

Web Application này không sử dụng `Minitest` của `Rails` mà sẽ dùng `RSpec` để thực hiện kiểm tra code trong lập trình. * FactoryGirl **Mặc dù Rails có chức năng** `Fixture` **để tạo test data, nhưng application lần này sẽ sử dụng** `FactoryGirl` **.** * **Faker** Một `Gem` chuyên dùng để tạo các dữ liệu ảo trong test.

4.1.1. Các loại TEST

- Test đơn thể độc lập
  - Thực hiện kiểm tra các method của Model, Decorator, Helper
- Test dành cho controller
  - Dùng kiểm tra cả quá trình tạo view, ngăn ngừa lỗi gọi method không được định nghĩa, hoặc gây lỗi trong view.

Sách tham khảo

> Everyday Rails - RSpec###Rails#####
> https://leanpub.com/everydayrailsrspec-jp/read

4.2. Kiểm tra code chéo dựa trên Pull-Request

Thực hiện tạo branch và chia nhỏ các commit theo mức hợp lý. Thực hiện kiểm tra kỹ trên máy tính của mình trước khi đưa lên Github tạo Pull-Request và nhờ người khác kiểm tra chéo.
* Về cơ bản, chỉ được thực hiện `Merge` sau khi **đã được kiểm tra chéo và có comment đồng ý.**

4.2.1. Các kiểm tra tối thiểu trước khi nhờ kiểm tra chéo

Thực hiện tất cả các điều dưới đây trước khi nhờ kiểm tra chéo.

- Kiểm tra xem có code không phù hợp với nội dung của branch không.

- Message trong mỗi commit có đơn giản, dễ hiểu không.
- Đã thực hiện tinh chỉnh, tối giản code chưa (refactoring)
- Method có thiết định public, private phù hợp hay chưa.
- Code có thực sự dê đọc và dễ hiểu không
- Tên của Class/Method/biến số có phù hợp không
- Có trùng lặp với các code đã có sẵn hay không
- Có gây ảnh hưởng tới các code đã có sẵn không
- Có các comment tại vị trí cần thiết hay chưa
- Đã xoá hết các code dùng trong debbug chưa
- Có thể viết code một cách khác nào tốt hơn hay không
- Tên file có hợp lý chưa
- Các file không cần thiết có bị commit chung hay không
- Có chứa lỗi đánh máy không
- Có tuân thủ đúng các qui tắc lập trình không
- Validate trong model có phù hợp chưa
- Có vấn đề gì về mặt bảo mật không
- Code có đi lệch khỏi các qui tắc của framework không
- Có chứa các sử lý mà có thể thay thế bằng library có sẵn không
- Có gây các tiềm ẩn gây nặng xử lý, hoặc nguy hiểm về bảo mật trong tương lai không
- Có cần thực hiện xử lý transaction cho cơ sở dữ liệu không
- Có cần sử dụng bộ nhớ đệm để giảm thiểu xử lý không
- Có chứa các SQL không cần thiết không
- Code kiểm tra có chứa đủ các trường hợp hay chưa
- Chương trình có chạy đúng tại nhưng điều kiện đặc biệt không
- Xử lý chặn lỗi (error handling) có được thực hiện phù hợp, đầy đủ chưa

5. A0101 Lưu event yêu thích (Register favorite event)

5.1. Thông số cơ bản

JSON Schema	api_design/build/unit/ja/A01.md
Controller, Action	api/favorites#create

5.2. Chi tiết

1. Xử lý trước
    - Kiểm tra login
        - Kiểm tra login dựa trên CookieSession. Nếu chưa login thì trả về `403 Response`
2. Xử lý chính
    - Tạo và lưu thông tin event yêu thích(*event_user_items*).
3. Xử lý khi có lỗi
    - Nếu không tìm được thông tin event trong *event_items* đúng theo `event_item_id` thì trả lại `404 Response`.
    - Nếu event đang không được công bố, (*event_items.published*) có giá trị khác **true (published)**, thì trả về `404 Response`
    - Nếu hiện tại không phải là thời gian sự kiện được công bố,
      thời gian nằm ngoài khoản *event_items.published_started_at ~ event_items.published_ended_at*, thì trả về giá trị `404 Response`
    - Nếu như event đã được lưu vào danh sách yêu thích trước đó rồi thì trả về `409 Response`

6. A0102 Xoá event khỏi danh sách yêu thích (Delete favorite event)

6.1. Thông số cơ bản

JSON Schema	api_design/build/unit/ja/A01.md
Controller, Action	api/favorites#destroy

6.2. Chi tiết

1. Xử lý trước
    - Kiểm tra login
        - Kiểm tra login dựa trên CookieSession. Nếu chưa login thì trả về `403 Response`
2. Xử lý chính
    - Xoá event được chỉ định khỏi danh sách yêu thích (*event_user_items*)
3. Xử lý khi có lỗi
    - Nếu không tìm được thông tin event trong *event_items* đúng theo `event_item_id` thì trả lại `404 Response`.
    - Nếu event đang không được công bố, (*event_items.published*) có giá trị khác **true (published)**, thì trả về `404 Response`
    - Nếu hiện tại không phải là thời gian sự kiện được công bố,
      thời gian nằm ngoài khoản *event_items.published_started_at ~ event_items.published_ended_at*, thì trả về giá trị `404 Response`
    - Nếu như không có event đã chỉ định trong danh sách yêu thích thì trả về `404 Response`