[Phần 1. Các bước từ khi hình thành app đến khi thực thi 3](#_Toc429131465)

[**Bài 1.Hình thành Rails App** 3](#_Toc429131466)

[**Bài 2.Khởi động Rails App bằng cách sử dụng WEBrick** 7](#_Toc429131467)

[**Bài 3.Quản lý Gem bằng cách sử dụng Bundler** 7](#_Toc429131468)

[**Bài 4.Mô hình MVC trong Rails** 7](#_Toc429131469)

[**Bài 5.Hiển thị nội dung với đường dẫn tĩnh** 8](#_Toc429131470)

[**Bài 6.Hình thành controller, action và thiết lập router** 8](#_Toc429131471)

[**Bài 7.Hình thành View, cung cấp Data** 15](#_Toc429131472)

[**Bài 8.Hình thành Model và sử dụng Database** 16](#_Toc429131473)

[Phần 2: Hình thành mô hình App với Scaffold 30](#_Toc429131474)

[**Bài 1.File được hình thành theo Scaffold** 30](#_Toc429131475)

[**Bài 2.Thực thi App đã được hình thành bằng Scaffold** 30](#_Toc429131476)

[Phần 3: Controller và Action 30](#_Toc429131477)

[**Bài 1.Controller và các quy ước đặt tên** 30](#_Toc429131478)

[**Bài 2.Hình thành và gọi Action** 30](#_Toc429131479)

[**Bài 3.Gọi Template từ Action** 30](#_Toc429131480)

[**Bài 4.Xuất Text từ Action** 30](#_Toc429131481)

[**Bài 5.Xuất các dạng JSON/XML từ Action** 30](#_Toc429131482)

[**Bài 6.Thiết lập Redirect(chuyển tiếp)** 30](#_Toc429131483)

[**Bài 7.Lấy parameter** 30](#_Toc429131484)

[Phần 4: Thiết lập Routing 30](#_Toc429131485)

[**Bài 1.Phương pháp thiết lập, xác nhận và vị trí ghi của routing** 30](#_Toc429131486)

[**Bài 2.Thiết lập routing đến root(/)** 30](#_Toc429131487)

[**Bài 3.Routing đã được sử dụng trước kia** 30](#_Toc429131488)

[**Bài 4.Thiết lập routing sử dụng match method** 30](#_Toc429131489)

[**Bài 5.Thiết lập routing của resource base** 30](#_Toc429131490)

[**Bài 6.Thêm action vào routing** 30](#_Toc429131491)

[**Bài 7.Thiết lập action sử dụng chỉ routing đến action** 30](#_Toc429131492)

[Phần 5: Model và Database 30](#_Toc429131493)

[**Bài 1.File cài đặt Database khi dùng SQLite** 30](#_Toc429131494)

[**Bài 2.File cài đặt Database khi dùng MySQL** 31](#_Toc429131495)

[**Bài 3.Hình thành Database** 31](#_Toc429131496)

[**Bài 4.Hình thành Model và dạng dữ liệu chỉ định khả năng** 31](#_Toc429131497)

[**Bài 5.Chức năng Migration là gì?** 32](#_Toc429131498)

[**Bài 6.Thành lập bảng sử dụng Migration** 32](#_Toc429131499)

[**Bài 7.Thay đổi bảng, thêm migration mới** 32](#_Toc429131500)

[**Bài 8.Rollback của Migration** 32](#_Toc429131501)

[**Bài 9.Thiết lập điều kiện NOT NULL** 32](#_Toc429131502)

[**Bài 10.Nhập dữ liệu đầu vào với rake db:seed** 32](#_Toc429131503)

[Phần 6: Template(View) và Helper Method 32](#_Toc429131504)

[**Bài 1. Hình thành Template và sử dụng từ Action** 32](#_Toc429131505)

[**Bài 2. Sử dụng Layout** 32](#_Toc429131506)

[**Bài 3. Thay đổi Layout sử dụng với mỗi action và controller** 32](#_Toc429131507)

[**Bài 4. Template bộ phận** 32](#_Toc429131508)

[**Bài 5. Sử dụng code Ruby trong Template** 32](#_Toc429131509)

[**Bài 6. Chuyển giá trị đến Template sử dụng biến instance** 32](#_Toc429131510)

[**Bài 7. Escape của văn tự đặc thù HTML** 32](#_Toc429131511)

[**Bài 8. Hình thành link sử dụng method link\_to** 32](#_Toc429131512)

[**Bài 9. Không gắn link cho page hiện tại và page giống nó** 32](#_Toc429131513)

[**Bài 10. Hình thành link mail sử dụng phương thức mail\_to** 32](#_Toc429131514)

[**Bài 11. Hình thành image tag sử dụng image\_tag method** 32](#_Toc429131515)

[**Bài 12. Hình thành Favicon sử dụng favicon\_link\_tag method** 32](#_Toc429131516)

**Ruby on Rails入門**

Dịch chuỗi bài 「Ruby on Rails　入門」tại page:

+ <http://www.rubylife.jp/rails/> của tác giải Tatsuo Ikura

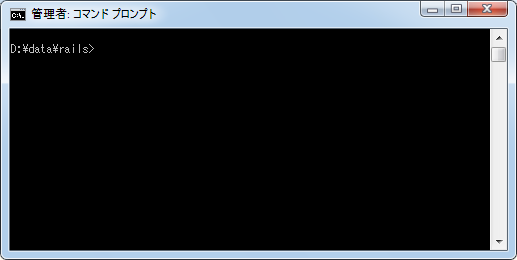
Do tất cả các bài viết đều được thực thi trên windown nên mình sẽ giữ nguyên như những gì tác giả viết.

# **Phần 1. Các bước từ khi hình thành app đến khi thực thi**

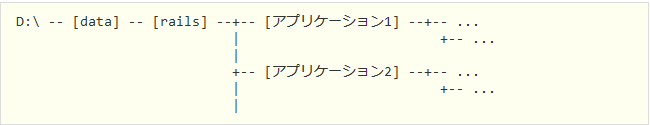
## **Bài 1.Hình thành Rails App**

* 1. Sử dụng Directory cho các Rails App

Gói gọn tất cả công việc lại trong 1 folder chính sẽ giúp ta dễ dàng thực hiện việc tìm kiếm , cũng như thuyết minh cho những ví dụ sau này. Khi nói đến thư mục chính thì chúng ta sẽ chấp nhận rằng tôi đang nói đến thư mục này.Hãy đặt nó tại nơi bạn cảm thấy yêu thích và dễ dàng tìm đến.



Ở đây tôi chọn ổ D/data/rails sẽ là nơi chứa các rails app do mình tạo ra theo cấu trúc:



Vì vậy từ giờ khi tôi nói tới thư mục chính thì bạn sẽ hiểu đó chính là thư mục D/data/rails của tôi nhé

* 1. Hình thành Rails App

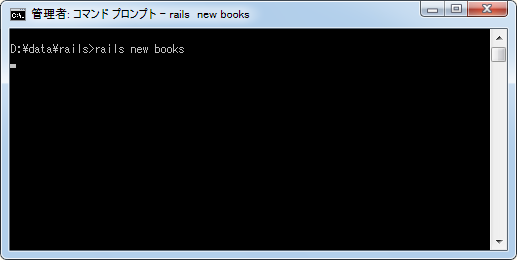
Để xây dựng 1 rails app thì tại thư mục chính , bạn hãy bật command prompt và thực thi theo những gì được ghi sau đây:

C:\Users\chu\Desktop\create_rails_app_1.PNG

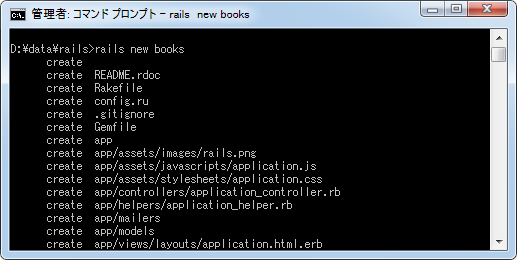
VD:

Taọ 1 project có tên là book

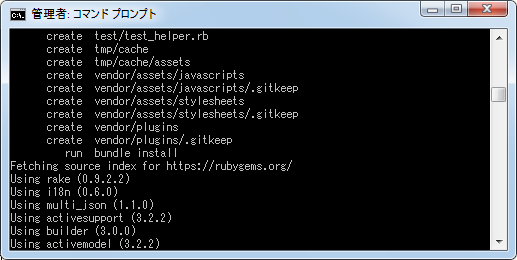
C:\Users\chu\Desktop\vd1.PNG



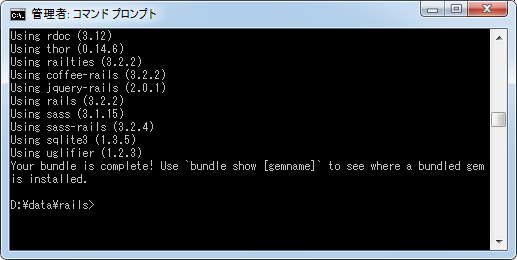
Tiếp theo hãy chú ý, có rất nhiều file được sinh ra



Khi các file đã được tạo ra thì lệnh “bundle install” sẽ tự động được thực thi(bundle install là gì sẽ được giải thích ở phần sau)



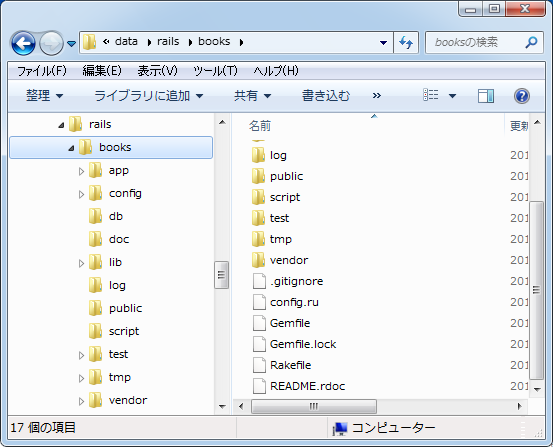
Và đến đây là kết thúc quá trình tạo ra 1 rails app rồi



* 1. Cấu trúc file

Trong thư mục chính, bạn hãy để ý có phải vừa có 1 thư mục mới cùng tên với rails app vừa tạo được sinh ra. Đó chính là thư mục chứa mã nguồn mà bạn cần làm việc với nó để có thể tạo ra các sản phẩm từ rails.

Bây giờ bạn hãy vào D:/data/rails/book/ và đây chính là những gì chúng ta thu được khi tạo ra project rails



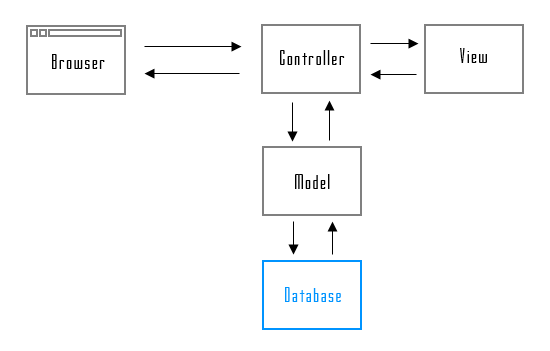
## **Bài 2.Khởi động Rails App bằng cách sử dụng WEBrick**

## **Bài 3.Quản lý Gem bằng cách sử dụng Bundler**

## **Bài 4.Mô hình MVC trong Rails**

Khi phát triển Rails App thì các thuật ngữ như Model/ Controller/ View thường xuyên được sử dụng. Vậy chúng là gì ?

Model / Controller / View là những thuật ngữ được sử dụng trong cấu trúc mô hình MVC. Trong đó Model là phần điều khiển, giải quyết các vấn đề data, View là tạo ra kết quả để người dùng có thể thấy được và Controller là phần xử lý yêu cầu từ người dùng, bằng cách tiến hành hợp tác với View và Model.



Sơ đồ trên chính là cấu trúc cơ bản của mô hình MVC.Chính vì thế mà khi Rails App được sử dụng mô hình này, thì những request(yêu cầu) từ phía người dùng sẽ có thể dễ dàng xác nhận.

1. Routing và Controller
2. Model
3. View
4. Trả về kết quả

## **Bài 5.Hiển thị nội dung với đường dẫn tĩnh**

(Phần 2,3,4,5 sẽ được hoàn thành sau)

## **Bài 6.Hình thành controller, action và thiết lập router**

Trong Rails, action sẽ được hình thành bên trong các controller được gọi khi có request(yêu cầu) từ phía người dùng.Như vậy để xác định được các tương tác của action và controller thì ta sẽ thử tạo 1 project đơn giản để kiểm tra việc này. Ngoài ra thì ta cũng sẽ nói về cách ghi lại routing trong Rails.

1. Tạo Controller

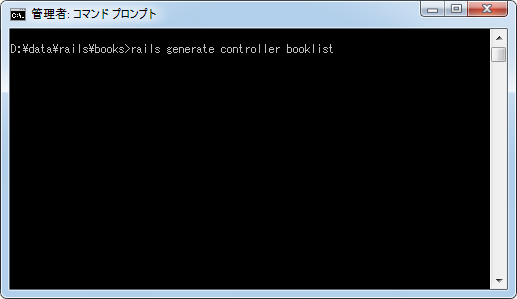
Như những gì đã ghi chép trong bài 4, Controller và Action được gọi ra từ phía người dùng, Controller sẽ sử dụng model để tập hợp những dữ liệu cần thiết và dùng view để gửi trả kết quả lại về phía người dùng.

Trước hết, ta hãy khởi tạo controller bằng cách di chuyển vào bên trong directory của rails app và thực thi lệnh sau:

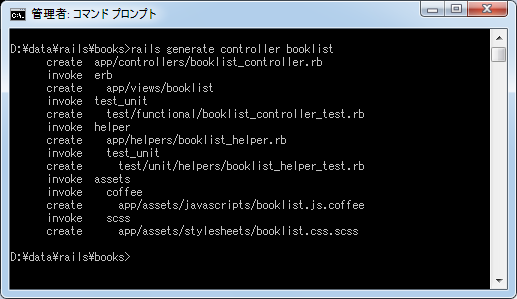
C:\Users\chu\Desktop\controller_create.PNG

VD:Bây giờ ta thử tạo 1 controller có tên booklist

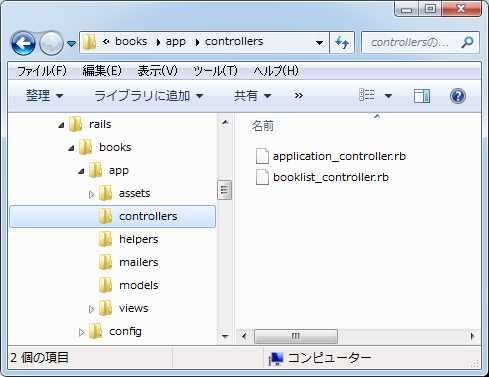
C:\Users\chu\Desktop\controller_create_vd.PNG



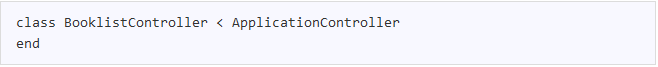
Tiếp theo sẽ có 1 số file tự động được sinh ra



Trong folder controller sẽ có file “app/controller/book\_controller.rb” được sinh ra



Mở file này ra bằng Text Editor, và đây là nội dung được ghi trong file

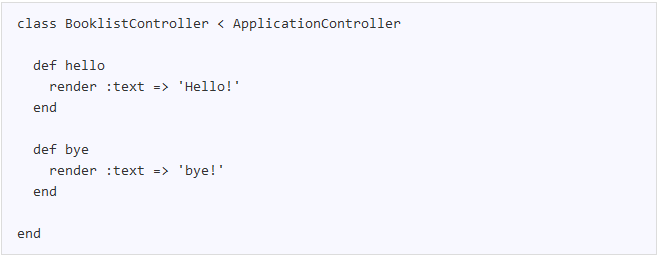


Ta thấy khi controller được hình thành, thì nó kế thừa từ lớp ApplicationController, tên nó được định nghĩa “tên\_controller + Controller”, và tên\_controller được viết hoa các chữ cái đầu theo đúng quy ước đặt tên trong Rails.

1. Tạo Action

Bởi vì controller đã được khởi tạo nên bây giờ ta sẽ đi tạo thành các action. Trong Rails thì các action được định nghĩa bên trong controller. Action được định nghĩa chính là các method bên trong lớp controller. Ví dụ , action “hello” ở ví dụ dưới đây được tạo thành với tư cách chính là phương thức “hello” của lớp BooklistController.

Tương tự như action hello, ta cũng tạo ra action bye



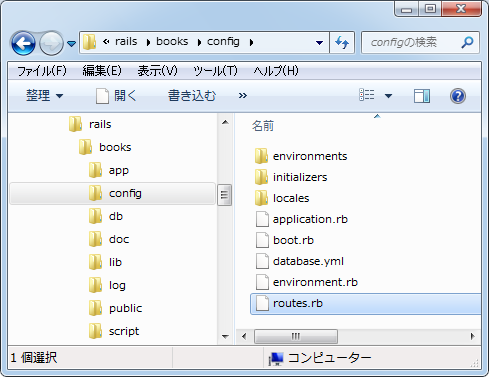
Hãy định nghĩa các phương thức public làm phương thức biểu thị cho action (Trong Ruby, nếu ko có chỉ định rõ ràng thì phương thức luôn được hiểu là public)

Bây giờ để kiển tra action vừa tạo, khi action được gọi , thay vì sử dụng view, ta sẽ gửi trực tiếp text khai báo trong action đến phía người dùng.

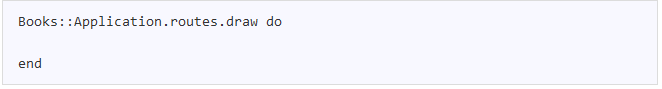
\*Chú ý: Trong method có ghi: 「render :text => ‘Hello!’」.Đây là phần text sẽ được gửi phản hồi về phía người dùng. Do lần này test controller và action là chính nên mình sẽ ko nói kĩ về phần render này.

1. Thiết lập Routing

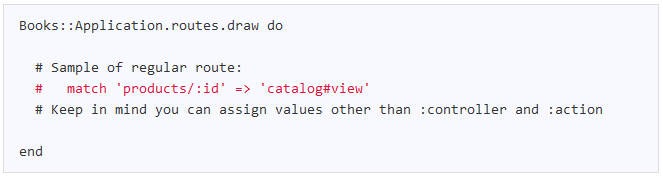
Cuối cùng, ta sẽ đi thiết lập routing. Raoutting là gì ? Nó là hành động định nghĩa việc action nào sẽ thực thi khi có lời gọi nào đó từ phía người dùng.Thiết lập routing ở đây chính là tiến hành thiết lập các nội dung trong file routes.rb(nằm trong project\_directory/config/routes.rb) – file này được tự động sinh ra khi khởi tạo rails app.



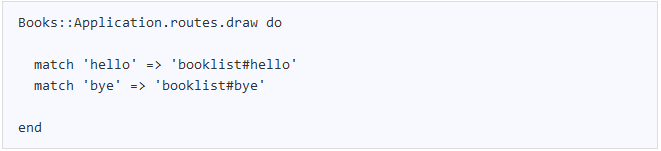
Sử dụng Text Editor mở file này ra. Bởi vì phần đầu đều được comment bằng dấu “#” nên để có thể định nghĩa bất cứ điều gì thì ta phải xóa hết phần comment đó đi



Trong phần comment đó, đa phần là những mẫu sample về việc thiết lập routing.Mặc dù có nhiều phương pháp để thiết lập routing nhưng lần này ta sẽ thử sử dụng phương pháp định nghĩa từng lời gọi action khi người dùng nhập trực tiếp URL path



Khi sử dụng hình thức thiết lập routing như thế này thì nếu người dùng mở url “http://localhost:3000/hello” từ brower thì action “hello” trong controller “booklist” sẽ được gọi. Và ta phải định nghĩa việc đó trong routes.rb(Tương tự với action bye)



Như vậy việc thiết lập routing đã kết thúc.

P/s: Phiên bản Rails của tác giả đã cũ (Rails3.x) nên khi chuyển sang 4.x thì lệnh trên đã phát sinh lỗi, và sửa lại như sau:

|  |
| --- |
| Rails.application.routes.draw do  match 'hello' => 'booklist#hello' , :via => [:get, :post]  match 'bye' => 'booklist#bye' , :via => [:get, :post]  end |

1. Xác nhận tương tác

Sau khi khởi động app Rails thì ta sẽ đi xác nhận lại tương lại trong app.Vào bên trong thư mục rails app và thực thi lệnh

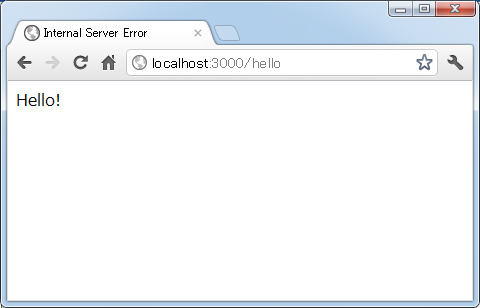
>rails server



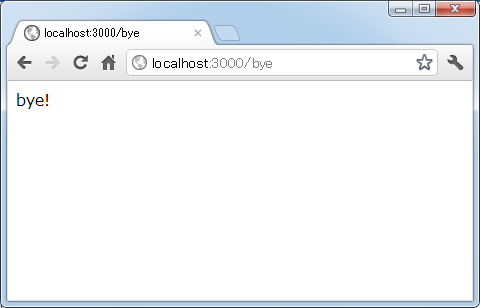
Tiếp theo hãy truy cập vào đường dẫn sau từ bower

C:\Users\chu\Desktop\url.PNG

Theo như những gì đã định nghĩa trong routes, thì action hello trong controller booklist sẽ được gọi và hiển thị trên màn hình



Tương tự với action bye



Như vật theo như cách trên thì sau khi khởi động Rails App,ta có thể gửi phản hồi kết quả thực thi action lại tới người dùng khi nhận được request từ phía người dùng thông qua brower.

## **Bài 7.Hình thành View, cung cấp Data**

**(chưa xong)**

Ở bài trước, ta đã gửi phản hồi kết quả trực tiếp đến người dùng bằng các action được gọi từ phía người dùng. Tuy nhiên trong thực tế, Rails lại không làm như thế, mà Rails sử dụng các kết quả được xử lý tùy theo action và phản hồi nó lại phía người dùng bằng cách sử dụng View. Ở bài này, ta sẽ tìm hiểu vai trò của View, đồng thời tạo thử 1 project đơn giản.

1. Vai trò của View

Đầu tiên ta sẽ đi xác nhận vai trò của View.Đối với các ứng dụng, khi nhận được request từ phía người dùng thì sẽ phản hồi lại những kết quả nào đó.

(Chưa dịch xong)

1. Xây dựng Template
2. Gọi Template từ action
3. Xác nhận tương tác
4. Truyền tham số giữa controller và view

(Bài 7 chưa dịch)

## **Bài 8.Hình thành Model và sử dụng Database**

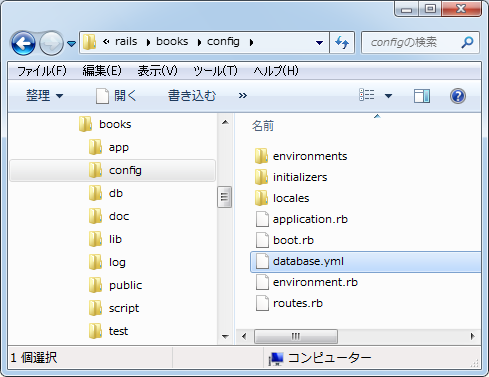
Controller, Action và cả View đều đã có những ví dụ đơn giản để bạn có thể hiểu về nó rồi. Cuối cùng ta sẽ thử tạo 1 project tiến hành trao đổi dữ liệu được chứa trong database hình thành từ model

1. File thiết lập Database

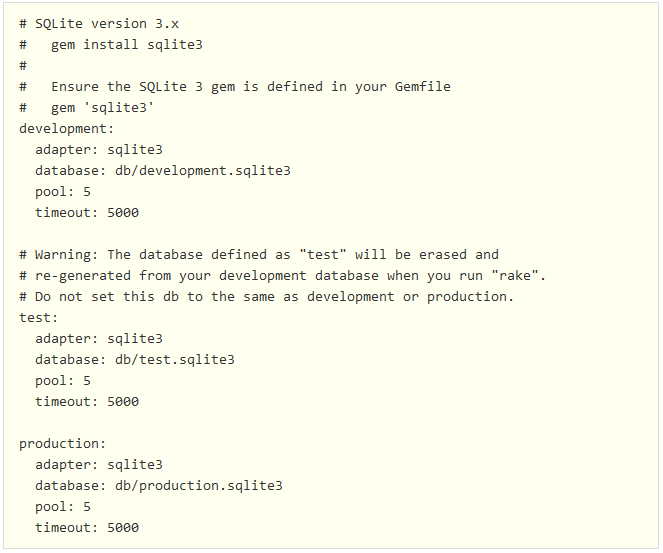
Khi bạn tạo Rails App theo câu lệnh

>rails new app\_name

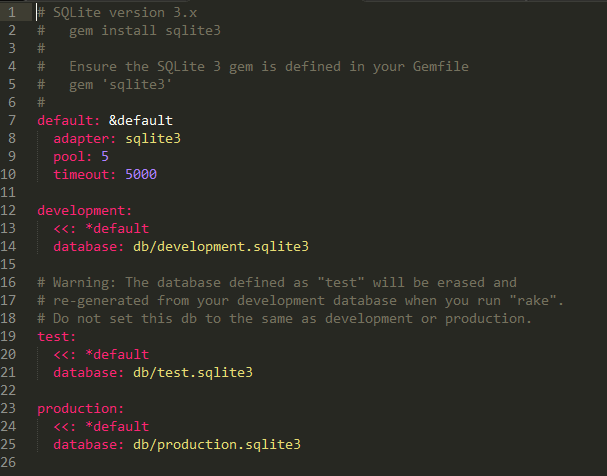
Nếu không có chỉ thị rõ ràng rằng sẽ sử dụng mô hình cơ sở dữ liệu nào thì Rails sẽ mặc định bạn sẽ sử dụng SQLite3. Và đồng thời file config/database.yml dùng để thiết lập khi sử dụng Database cũng được sinh ra.



Bạn có thể mở file database.yml và sẽ thấy nội dung như sau



P/s: Đây là nội dung file database.yml với Rails4.x



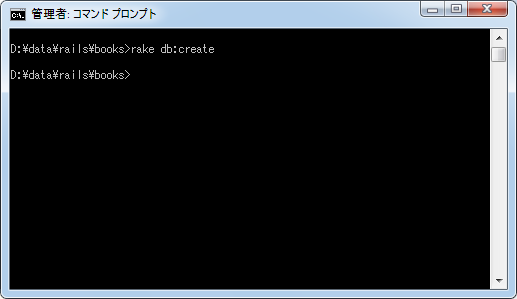
Trong file thiết lập được phân chia làm 3 loại mode đó là development , test và production, được dùng tương ứng với nhà phát triển, tester và người sử dụng. Trong trường hợp nếu bạn sử dụng hệ CSDL MySQL thì trong file này sẽ định nghĩa thêm username và password.

Ở đây mặc dù đã giản lược về phương pháp thiết lập cụ thể nhưng trong mode development bạn có thể thấy file database của SQLite tương ứng với tên “db/development.sqlite3” được thành lập

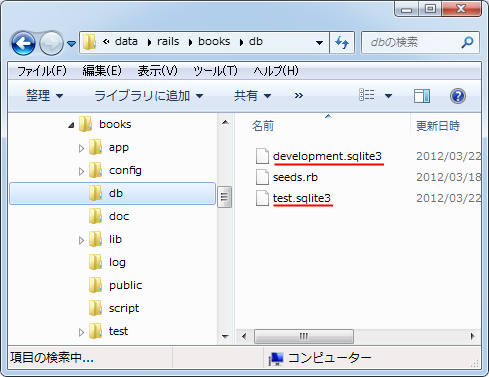
1. Hình thành Database

Bây giờ việc đầu tiên là ta sẽ bắt tay vào tạo Database.Database được sử dụng có 2 loại là SQLite và MySQL. Bây giờ ta sẽ đi vào bên trong thư mục Rails App và khởi động command prompt, sau đó thực thi lệnh sau đây để tạo database.

C:\Users\chu\Desktop\create_db.PNG

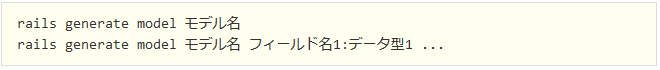


Vào directory có tên db trong thư mục rails app, tại đây thì database giành cho nhà phát triển và tester đã được hình thành



1. Hình thành Model

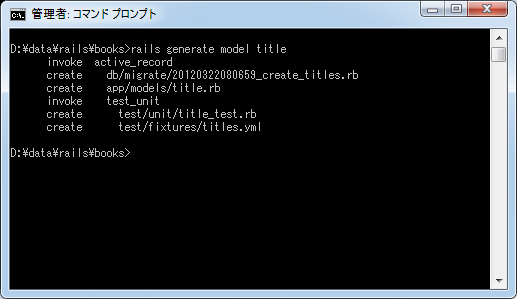
Ta bắt tay vào xây dựng Model.Di chuyển đến thưc mục rails app và thực thi lệnh sau



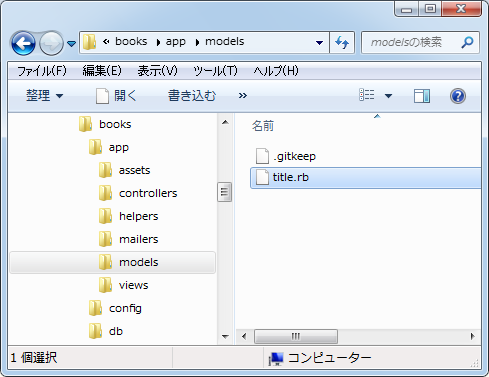
Giải thích: Với cách ghi đầu tiên thì khi hình thành Model thì file dùng để tạo thành, thay đổi table khi migration được gọi cũng tự động được sinh ra. Còn với cách khai báo thứ 2 thì các trường trong table , kiểu dữ liệu của các trường sẽ được chỉ định. Trong ví dụ này ta sử dụng cách đầu tiên khai báo.

Ta thành lập 1 model có tên là “title”

C:\Users\chu\Desktop\table_title.PNG



Sau khi thực thi lệnh trên thì có 1 số file đã được sinh ra nhưng ở đây ta chú ý tới file được ghi lại liên quan tới Model vừa được tạo ra là “app/model/title.rb”



Bây giờ ta sẽ dùng Text Editor để mở file title.rb vừa tạo ra

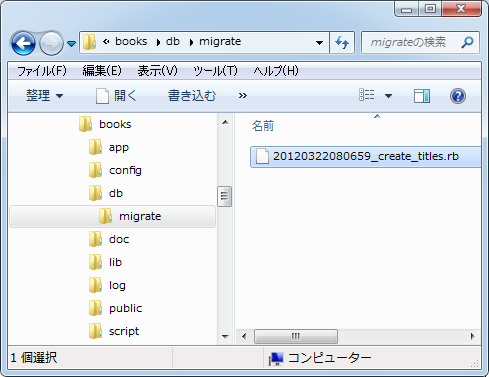


Model được hình thành theo cách trên thì đều kế thừa từ lớp 「ActiveRecord::Base」 và tên lớp được định nghĩa theo dạng 「tên\_model」với chữ cái đầu được viết hoa.

Lớp này được sử dụng để ghi lại việc quản lý liên quan đến thao tác, thông tin của database. Tuy nhiên do lớp ActiveRecord::Base là lớp cơ bản đã được định nghĩa hết rồi, nên nó có thể quản lý 1 cách dễ dàng mà ko cần khai báo gì cả.

1. Hình thành Table sử dụng Migration

Khi Model được khởi tạo thì đồng thời một file có tên “db/migrate/ 20120322080659\_create\_titles.rb” cũng được khởi tạo.Đây chính file được gọi là được gọi tới migration.

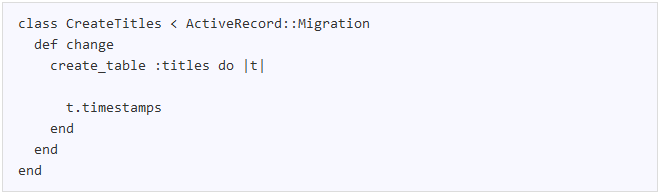


Sau đây mình sẽ giải thích 1 cách đơn giản về migration.Database thì đã được thành lập nhưng việc nạp dữ liệu vào thì cần phải thành lập các bảng(table). Khả năng xây dựng các bảng bằng khả năng của Database và không phụ thuộc vào Rails là đương nhiên.Thế nhưng trong Rails đã và đang chia sẻ chức năng được gọi là “Migration” được dùng để tạo và thay đổi các bảng.

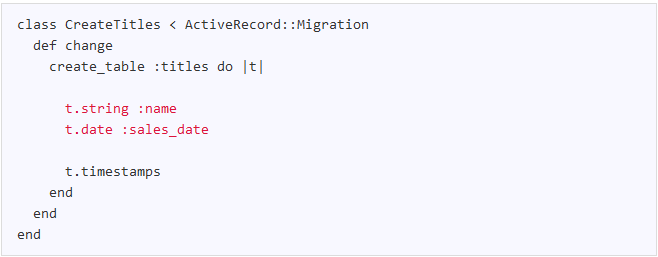
Vậy thì Migration là gì ? Mặc dù tạo bảng nhưng không tạo trực tiếp mà dựa vào những file script đã được tạo(file script được dùng để tạo table), và thực thi các file script đó để tạo ra bảng. Nhiều người thắc mắc tại sao lại phải tốn thời gian như vậy thì có 2 nguyên nhân sau:

1. Có thể dễ dàng xử lý một cách tương tự(giống nhau)tại các môi trường khác nhau bằng cách ghi lại những xử lý đã tiến hành liên quan tới table
2. Có thể hình thành bảng dựa trên file script mà ko cần quan tâm đến các mô hình cơ sở dữ liệu khác nhau. Đây là lợi điểm.

Bây giờ ta quay trở lại với file “20120322080659\_create\_titles.rb”, ta sẽ mở nó bằng Text Editor

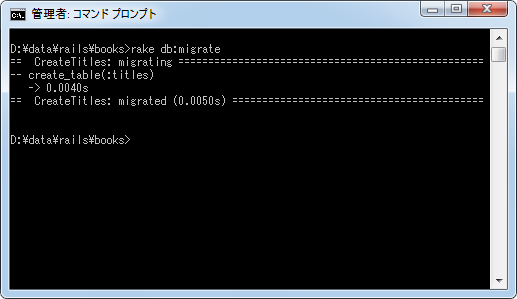


Bây giờ ta sẽ thêm 2 trường vào db bằng cách chỉnh sửa file này như sau:



Bây giờ ta sẽ sử dụng migration. Di chuyển command prompt đến thực mục rails app và thực thi lệnh

C:\Users\chu\Desktop\run_migrate.PNG

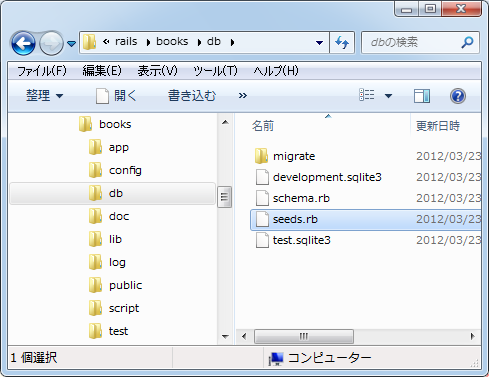


MigrantionScript đã được tạo ra đã khởi tạo table nên khi thực thi script nó sẽ tạo table bên trong database.

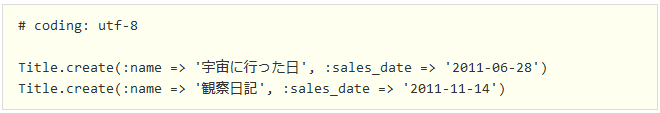
1. Hình thành dữ liệu Test

Tạo Model , tạo table quan trọng bên phía database đã hoàn thành.Bây giờ để tạo 1 sample đơn giản thì ta sẽ đi đăng kí 1 số dữ liệu vào bảng vừa tạo.

Trong Rails, cơ cấu đăng ký dữ liệu vào bảng đã được chuẩn bị sẵn. Ta vào trong thư mục db và sẽ thấy file “seed.rb” đã được hình thành.

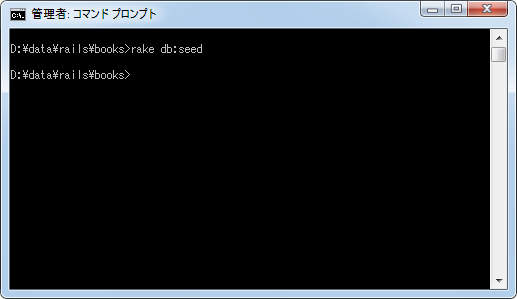


Sử dụng TextEditor mở file này lên và thêm vào đó như sau:



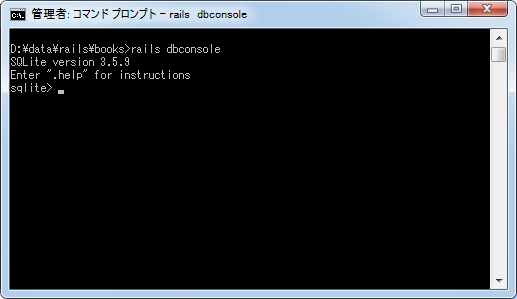
Ta thêm data vào bảng bằng cách sử dụng method create của Model Title. Sau khi chỉnh sửa xong ta nhớ phải lưu lại bằng mã UTF-8 để khi lưu vào database dữ liệu sẽ ko bị lỗi font(với tiếng Nhật or Việt). Tiếp theo ta sẽ di chuyển ra thư mục chính của rails app và thực thi đoạn script sau đây:

C:\Users\chu\Desktop\save_data.PNG

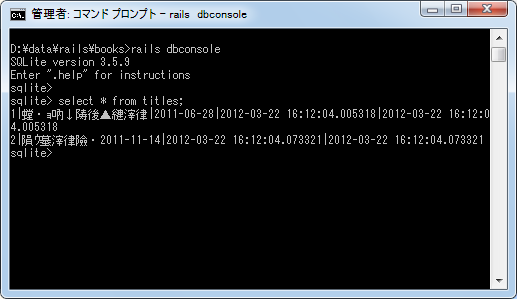


Như vậy đã có 2 dữ liệu được đăng ký vào bảng “title”. Tuy nhiên để cho chắc chắn, ta nên xác nhận lại.Theo thiết lập hiện tại của database, ta khởi động commandline tool của SQLite bằng lệnh

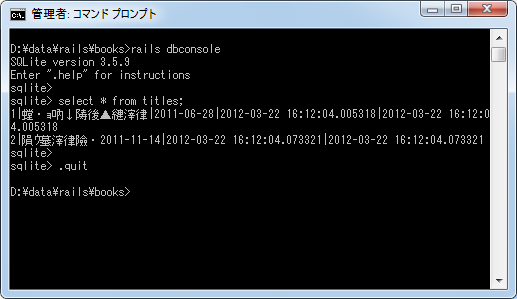
C:\Users\chu\Desktop\dbconsole.PNG



Lấy toàn bộ dữ liệu trong bảng “title”



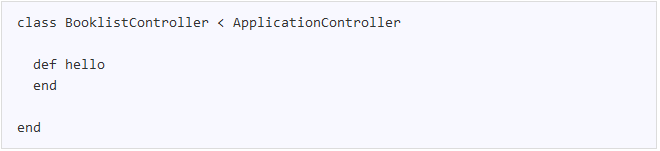
Command prompt của Win ko hỗ trợ UTF-8 nên sẽ lỗi khi hiển thị tiếng Nhât.Bây giờ để thoát khỏi database tool của sqlite ta dùng lệnh “.quit”



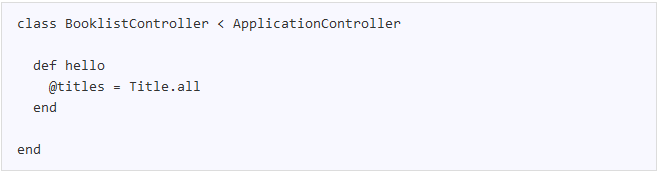
1. Sử dụng Model để lấy dữ liệu từ Database

Bây giờ ta sẽ sử dụng Model để lấy dữ liệu có trong bảng “title”. Bây giờ ta sẽ sửa action method một chút, mở file “app/controller/

booklist\_controller.rb”



Sử dụng action hello đã được tạo từ các bài học trước. Và sửa như sau:



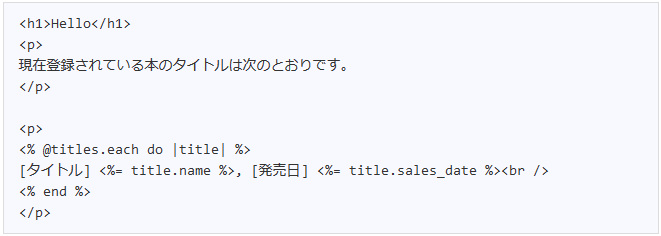
Trong Model Title thì method all sẽ trả về tất cả các dữ liệu trong bảng theo kiểu mảng của object của lớp Model.

Bây giờ ta sẽ chỉnh sửa View bằng cách sửa file “app/views/booklist/

hello.html.erb”.



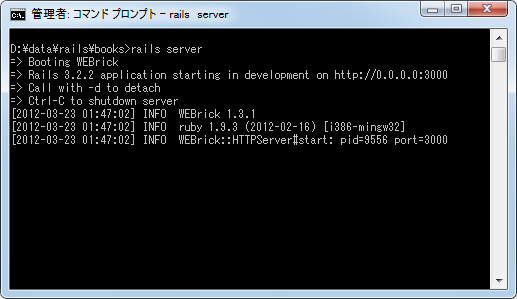
Sửa thành



Màn hình sẽ hiển thị title và ngày bán lấy được theo thứ tự từ mảng đối tượng của lớp Model được truyền từ Action method.

1. Xác nhận tương tác

Bây giờ ta sẽ khởi động Rails App để xác nhận kết quả.



Vào brower và truy cập vào url

C:\Users\chu\Desktop\url.PNG

Và đây là kết quả:



Đến đây ta đã biết cách sử dụng Model để lấy dữ liệu từ database, và hiển thị trên brower kết quả lấy được. Như vậy là ta đã hiểu qua về cách sử dụng Controller, View, Model trong Rails thông qua ví dụ đơn giản trên.

==================================================================

# **Phần 2: Hình thành mô hình App với Scaffold**

Trong Rails app thì controller , view và model thì khi hình thành từng thành phần trong đó, nó sẽ tiến hành thiết lập các routing quan trọng. Trong rails thì có 1 chức năng được gọi là Scaffold có thể hình thành 1 cách sơ lược và tiến hành tổng hợp các tác nghiệp như đã nói ở trên. Trong phần này, ta sẽ đi nói rõ hơn về phương pháp hình thành rails app bằng cách sử dụng chức năng Scaffold

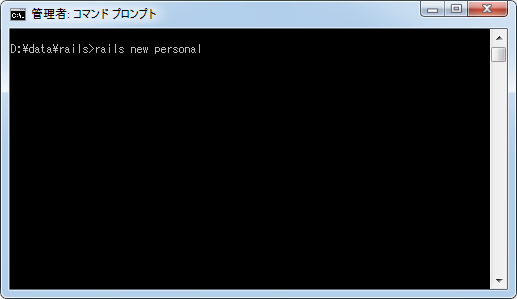
## **Bài 1.File được hình thành theo Scaffold**

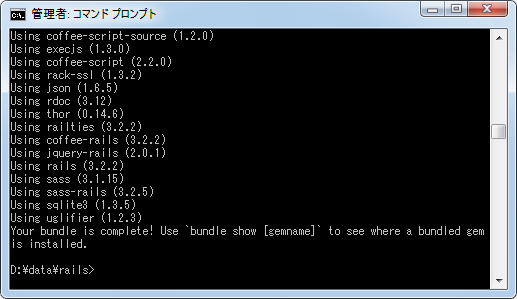
Scaffold là 1 chức năng của Rails. Chỉ cần thực thi 1 lệnh command thì nó sẽ tự động hình thành các Controller, View , Model quan trọng của Rails App giúp cho mình. Như vậy là mình chẳng cần làm gì nếu sử dụng scaffold. Tuy nhiên trong bài học này thì mình muốn biết những file đã được hình thành như thế nào.

1. Thực thi command để sử dụng chức năng Scaffold

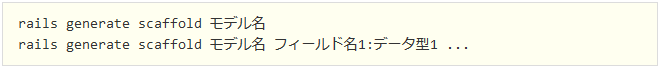
Đầu tiên, vẫn như thường lệ, ta cần tạo 1 Rails App.

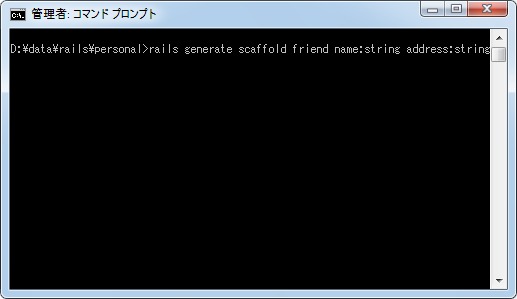
C:\Users\chu\Desktop\new_rail_app.PNG



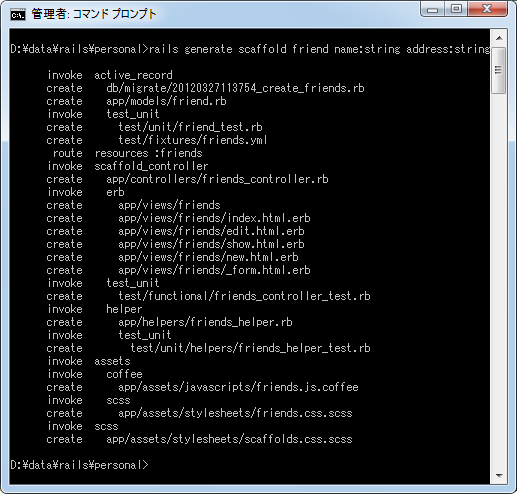


Đến đây là rails app đã được tạo xong. Tiếp theo ta sẽ thực thi lệnh command để hình thành Controller, Model,.. thông qua việc sử dụng chức năng Scaffold. Làm theo những gì ghi sau đây:





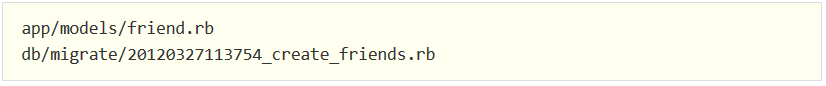
Đợi các file được tự động sinh ra



Và bây giờ ta sẽ đi xác nhận những file đã được sinh ra.

1. Model và Migration Script

Tiếp theo là 2 file được ghi chép liên quan tới model và migration



Ta sẽ thử mở file có liên quan đến model bằng text editor

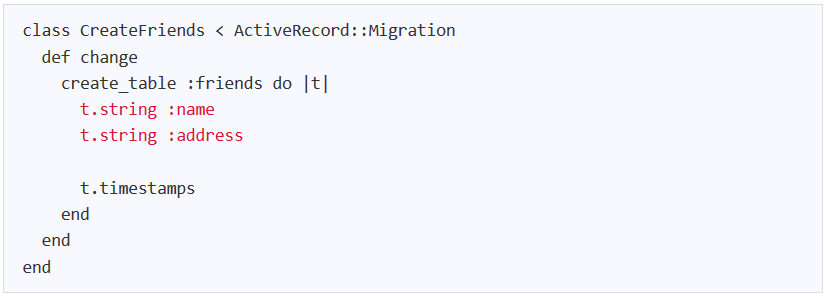


Mặc dù không có định nghĩa liên quan đến lớp nhưng model trong trường hợp này để có thể tổ chức quản lý một cách đơn giản mà không cần ghi gì cả thì nó được định nghĩa…

(クラスに関する定義しかありませんが、モデルの場合は継承している「ActiveRecord::Base」クラスに基本的なメソッドは定義されているため、特に何も記述しなくても簡単な処理を行なうことができます。)

Đối với migration script thì khi đã thực hiện command …

マイグレーションスクリプトにはコマンドを実行した時に引数に指定したカラムに関する記述が含まれています。



1. Controller và Action

File tiếp theo chúng ta để ý tới là file được viết liên quan tới Controller.

C:\Users\chu\Desktop\scaffold_ctl_1.PNG

Mở file bằng text editor

|  |
| --- |
| class FriendsController < ApplicationController  before\_action:set\_friend,only: [:show, :edit, :update, :destroy]  # GET /friends  # GET /friends.json  def index  # Khai bao 1 bien friends chua tat ca data ve friend  @friends = Friend.all  end  # GET /friends/1  # GET /friends/1.json  def show  end  # GET /friends/new  def new  @friend = Friend.new  end  # GET /friends/1/edit  def edit  end  # POST /friends  # POST /friends.json  def create  @friend = Friend.new(friend\_params)  respond\_to do |format|  if @friend.save  format.html { redirect\_to @friend, notice: 'Friend was successfully created.' }  format.json { render :show, status: :created, location: @friend }  else  format.html { render :new }  format.json { render json: @friend.errors, status: :unprocessable\_entity }  end  end  end  # PATCH/PUT /friends/1  # PATCH/PUT /friends/1.json  def update  respond\_to do |format|  if @friend.update(friend\_params)  format.html { redirect\_to @friend, notice: 'Friend was successfully updated.' }  format.json { render :show, status: :ok, location: @friend }  else  format.html { render :edit }  format.json { render json: @friend.errors, status: :unprocessable\_entity }  end  end  end  # DELETE /friends/1  # DELETE /friends/1.json  def destroy  @friend.destroy  respond\_to do |format|  format.html { redirect\_to friends\_url, notice: 'Friend was successfully destroyed.' }  format.json { head :no\_content }  end  end  private  # Use callbacks to share common setup or constraints between actions.  def set\_friend  @friend = Friend.find(params[:id])  end  # Never trust parameters from the scary internet, only allow the white list through.  def friend\_params  params.require(:friend).permit(:name, :address)  end  end |

Trong controller thì các metod action là “index” , “show” , “new” , “edit” , “create” , “update” , “destroy”　đã được tự động định nghĩa.Như vậy ta có thể tiến hành lấy và thêm dữ liệu bằng cách gọi các action này.

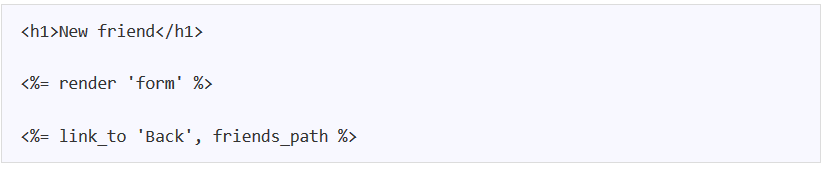
1. View

Tiếp theo là file được ghi liên quan tới view



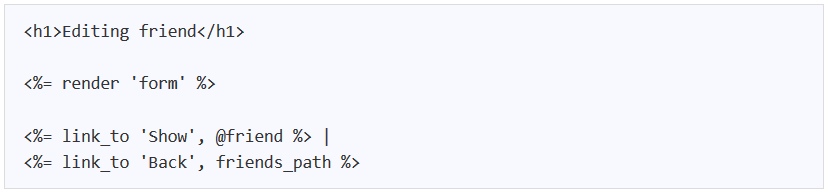
File cuối cùng “\_form.html.erb” là thứ được đọc từ view khác nên ngoài nó ra thì view là các file view tương ứng với các action.

Giờ ta sẽ mở file “new.html.erb” bằng TextEditor



render ‘form’ chính là đang đọc file “\_form.html.erb”

Tiếp theo ta sẽ mở file “edit.html.erb” bằng TextEditor



Trong file này cũng có mã render ‘form’ thực hiện việc đọc nội dung file “\_form.html.erb”

File “\_form.html.erb” là thứ hiển thị form dùng để thay đổi dữ liệu đã tồn tại hay là đăng ký dữ liệu mới. Bởi vì form là phần chung nên được tạo ra như là 1 file riêng biệt để các file như là “new.html.erb” hay “edit.html.erb” gọi tới nó.

Phía trên là các file chủ yếu được sinh ra khi thực thi command bằng việc sử dụng chức năng scaffold. Controller và action , view , model được tạo ra, không chỉ có vậy mà những chức năng đọc và thêm mới cơ bản của dữ liệu cũng được hình thành.

## **Bài 2.Thực thi App đã được hình thành bằng Scaffold**

**(chưa)**

1. Thực thi Migration Script
2. Khởi động Application

(chưa viết xong)

==================================================================

# **Phần 3: Controller và Action**

## **Bài 1.Controller và các quy ước đặt tên**

**(chưa hoàn thành)**

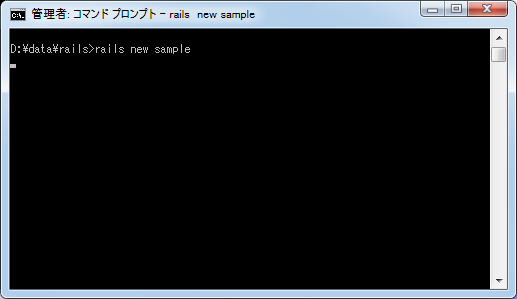
Khi tạo Rails App thì ta cần chú ý tới cách tạo 1 controller. Bài này mình sẽ đi sâu vào việc giải thích về cách tạo ra 1 controller cũng như đặt tên cho 1 controller sao cho chuẩn.

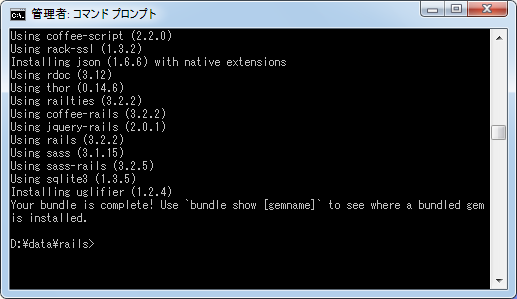
1. Cách tạo Controller

Đầu tiên là phương pháp khởi tạo controller.Ta có thể tạo ra controller bằng cách thủ công, tuy nhiên bằng cách thực thi command thì những file quan trong cũng sẽ tự động được sinh ra sẽ giúp bản giảm thời gian và tránh nhầm lẫn.

C:\Users\chu\Desktop\create_controller.PNG

Ta thử tạo 1 sample app có tên “sample”





(chưa viết xong)

1. Quy tắc đặt tên cho Controller
2. Các thêm vào thêm

## **Bài 2.Hình thành và gọi Action**

**(Chưa hoàn thành)**

## **Bài 3.Gọi Template từ Action**

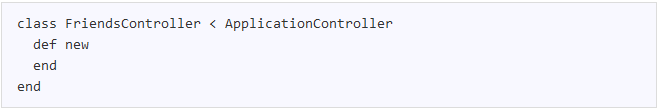
Khi action được gọi từ phía người dùng, thì action đó sẽ tiến hành lấy dữ liệu từ database thông qua model và sau đó sẽ phản hồi lại người dùng một kết quả nào đó. Khi đó, action sẽ sử dụng view(template) để hiển thị kết quả phản hồi cho người dùng. Ở bài này, ta sẽ tìm hiểu về cách action gọi đến template.

1. Gọi Template

Trong Rails, khi action được gọi,việc phản hồi 1 kết quả nào đó trở lại phía người dùng là rất quan trọng.Trong action method, bạn có thể trực tiếp xuất ra kết quả và gửi nó về phía người dùng, nhưng thực tế thì Rails sử dụng các template tương ứng với view để xuất ra các kết quả mà action muốn gửi về phía người dùng.

Nếu không có chỉ định đăc biệt gì thì action sẽ gọi đến template tương ứng mà nó sử dụng, có tên đặt theo dạng “actionname.html.erb”. Do đó, bên trong action method, trong trường hợp không tiến hành xuất ra trực tiếp hay redirect(chuyển hướng) thì ở cuối của method,nó sẽ tự động gọi đến template tương ứng với nó.

Đơn giản là ta có thể thấy thông qua ví dụ sau đây. Bên trong action method không có gì hay có nghĩa là nó ko tiến hành xử lý bất kì điều gì nhưng, cuối cùng thì nó vẫn được tự động gọi đến template có tên “actionname.html.erb”.



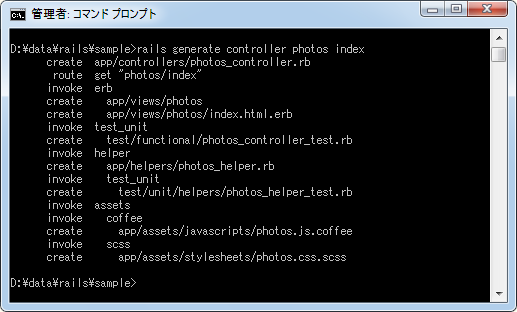
Folder đặt template được gọi tự động từ action đó là “app/views/controllername”. Ví dụ: controller “friends” có action “new” thì action này sẽ sử dụng template là “app/views/friends/new.html.erb”. Trong trường hợp nếu không sử dụng cách tạo tự theo lệnh

>rails generate controller new\_controller action1 action2

Thì nếu tạo bằng thủ công, bạn hãy tạo template trong đúng folder app/views/controller\_name\_folder/actionname.html.erb

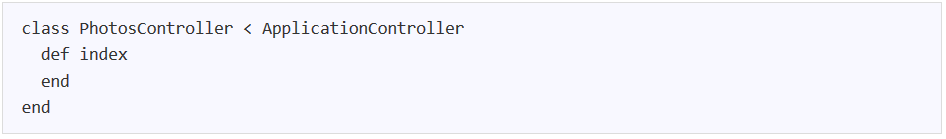
Bây giờ ta sẽ tiến hành thử nghiệm thực tế.Trong app “sample” ta tạo thêm 1 controller mới có tên “photos” và action index của nó.

C:\Users\Chu Quang Vien\Desktop\create_photos_controller.PNG



Mở class controller vừa được tạo ra(app/controllers/

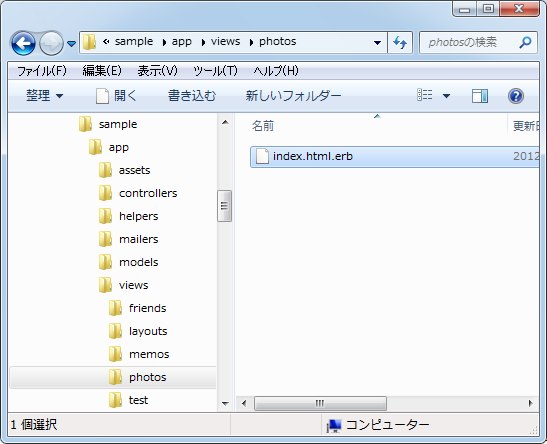
photos\_controller.rb)



Bên trong action method “index” ta không định nghĩa gì cả.

Tiếp theo, ta sẽ tạo template được gọi từ action “index”. Trong trường hợp này thì template đó chính là “app/views/photos/

Index.html.erb”. Thế nhưng khi khởi tạo controller thì action đồng thời được tạo ra và default template tương ứng cũng tự động được sinh ra.



Mở file index.html.erb đã tự động được sinh ra và xác nhận nội dung trong nó có như những gì được ghi dưới đây hay không.

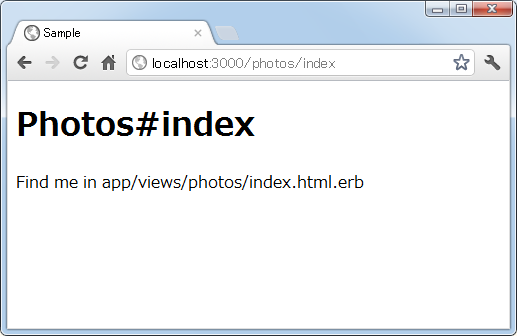


Khi action được gọi, nếu giữa chừng không có gì khác được xuất ra(đưa ra) thì template có tên giống với tên của action được gọi sẽ được gọi ra, tùy vào template mà kết quả tạo ra được phản hồi tới người dùng. Trong trường hợp này, khi action index được gọi thì dựa vào template tương ứng là index.html.erb mà kết quả sinh ra được phản hồi lại về phía người dùng.

Nào, bây giờ ta hãy thực thi bằng app “sample”



Truy câp bằng brower theo địa chỉ “http://localhost:3000/photos/index” và đây là kết quả thu được



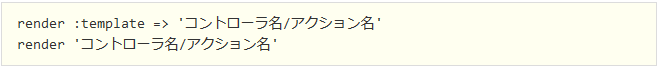
1. Gọi Template được sử dụng bởi action khác

Nếu bên trong action có tiến hành điều kiện phân kì(???) thì với kết quả như thế thì không chỉ default template mà template của những action khác cũng có thể gọi được. Trong trường hợp như thế này thì nếu sử dụng những câu khai báo dưới đây thì ta có thể chỉ định template gọi đến một cách minh bạch.

Đầu tiên, các action trong cùng 1 controller có thể gọi được đến template tương ứng của nhau



Bây giờ để gọi template tương ứng với action chứa trong 1 controller khác thì ta thực hiện như những gì được ghi dưới đây



Trong từng trường hợp trên, hãy chú ý một điểm đó là nó chỉ hình thành kết quả gửi phản hồi về phía người dùng bằng cách gọi đến template của các action đó chứ ko phải là gọi tới action đã được chỉ định.

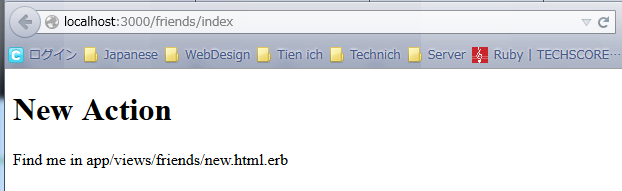
VD: Từ action index bên trong controller friends, ta sẽ gọi template của action new trong cùng controller như sau:



Trong trường hợp này, thực tế template được gọi đến là “app/views/

friends/new.html.erb” chứ ko phải default template là “app/views/

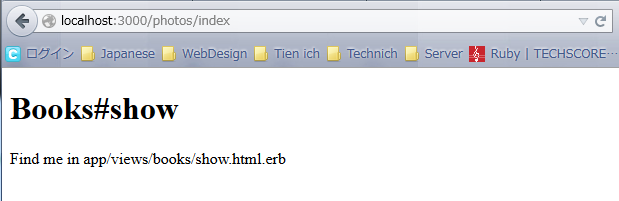
friends/index.html.erb”



Tiếp theo, ta sẽ gọi template tương ứng với action show bên trong controller books từ action index của controller photos như sau



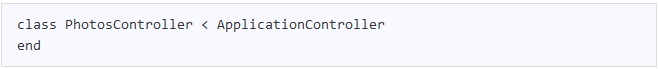
Kết quả thu được:



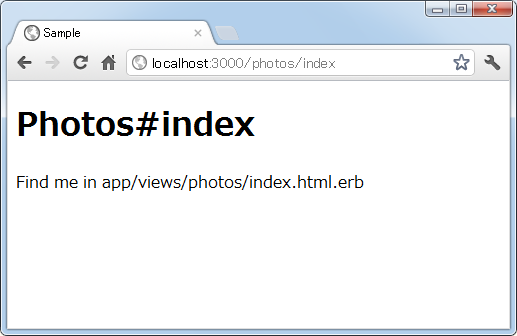
1. Trong trường hợp chỉ có template mà không có action method

Khi action được gọi thì mặc dù action method không được định nghĩa nhưng template có tên gắn với action đó tồn tại thì template vẫn được sử dụng và xuất ra kết quả vè phía người dùng.

Bây giờ ta sẽ thử nghiệm thực tế, trong controller photos ta sẽ xóa đi action index



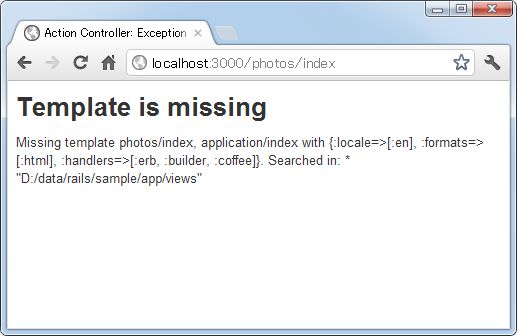
Măc dù action index không có nhưng template tương ứng với nó là “app/views/photos/index.html.erb” vẫn tồn tại nên khi ta truy cập vào đường dẫn “http://localhost:3000/photos/index” thông qua brower thì vẫn thu được kết quả như sau



1. Trong trường hợp không tồn tại template

Trong Action nếu không đưa ra kết quả trực tiếp thì template gắn với action sẽ tự động được gọi đến.Tuy nhiên nếu không trường hợp template này không tồn tại thì sẽ phát sinh lỗi. Bây giờ ta sẽ sử dụng controller có sẵn đó là lớp controller photos(“app/controller

s/photos\_controller.rb”), và ta sẽ xóa đi file template gắn với action index đó là (“app/views/photos/index.html.erb”), và sau đó ta sẽ truy cập vào đường dẫn <http://localhost:3000/photos/index> từ brower.



Và đây là lỗi phát sinh “Missing Template”.

## **Bài 4.Xuất Text từ Action**

Chuyển kết quả về phía người dùng sử dụng template của view được hình thành một cách bình thường, tuy nhiên với những test đơn giản thì việc chuyển kết quả trực tiếp từ action cũng rất phổ biến. Thì bài hôm nay mình sẽ đi giải thích về phương pháp xuất ra Text về phía người dùng sử dụng method render.

1. Xuất Text

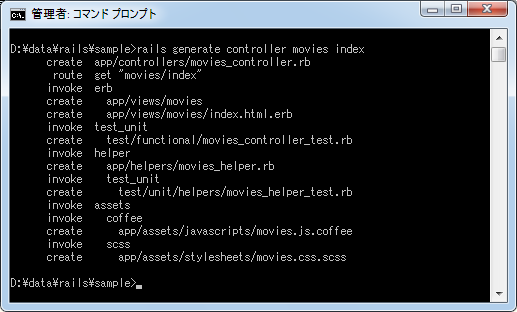
Với việc tiến hành đưa ra trực tiếp theo request ta sử dụng phương thức render. Phương thức render là gì ? Render method được sử dụng rất nhiều với việc muốn chuyển lại 1 cái gì đó. Trong trường hợp muốn xuất ra Text thì ra sẽ sử dụng mã sau:

C:\Users\chu\Desktop\render_text.PNG

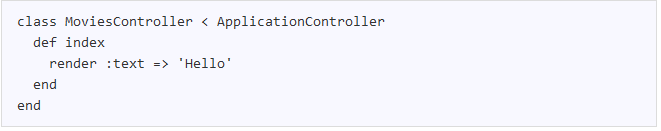
Khi khai báo trên được thực thi thì nó sẽ chuyển đoạn văn bảnテキストđã chỉ định về phía người dùng.

Bây giờ ta sẽ tiến hành thử nghiệm thực tế bằng cách tiến hành tạo 1 controller movies trong rails app “sample”

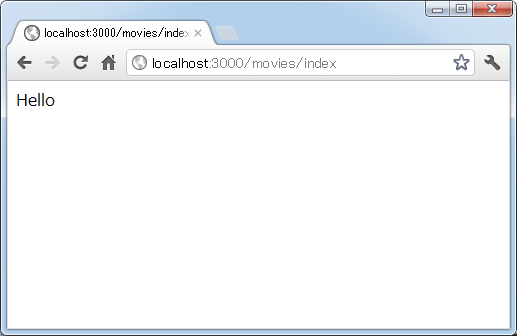
C:\Users\chu\Desktop\movies_controller.PNG



Tiếp theo ta sẽ sửa file nguồn movies\_controller.rb như sau



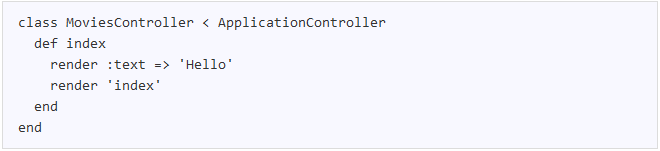
Bây giờ truy cập vào đường dẫn “http://localhost:3000/movies/index”



Như vậy ta đã xuất được text trực tiếp từ action ra màn hình mà ko cần thông qua template

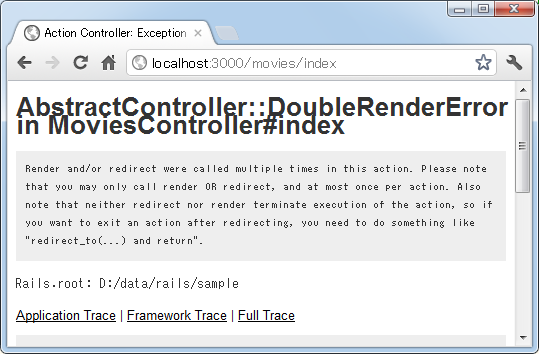
1. DoubleRenderError

Nếu bây giờ mình muốn vừa sử dụng template, vừa xuất kết quả trực tiếp về phía người dùng thì sao nhỉ? Có phải sẽ là như thế này



Mình sẽ thử xem có ra kết quả không nhỉ. Truy cập vào đường dẫn

“http://localhost:3000/movies/index” và đây là kết quả



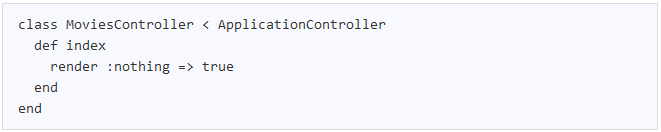
Trên thực tế thì Rails ko cho phép gửi trả 2 lần kết quả về phía người dùng như vậy. Nên chỉ có thể sử dụng duy nhất 1 lần render, nếu sử dụng render :text => ‘Hello’ thì sẽ ko thể gọi đến template được nữa

1. Không xuất ra gì cả

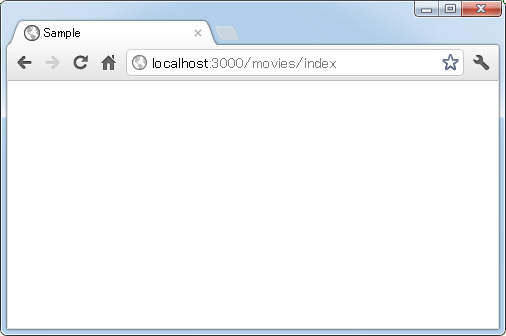
Việc gửi phản hồi lại về phía người dùng khi có request từ người dùng là rất quan trọng, nhưng ta có thể phản hồi lại bằng cách không có hiển thị gì với lệnh “render :nothing”

C:\Users\chu\Desktop\render_nothing.PNG

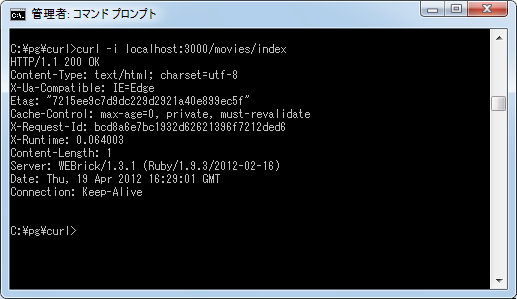
Trong action index ta sửa mã như sau



Kết quả



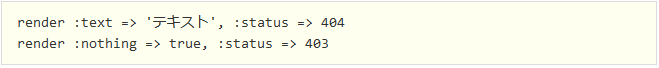
Ngoài ra thay vì phải test trực tiếp trên màn hình thì ta có thể sử dụng command prompt để kiểm tra bằng lệnh curl như sau



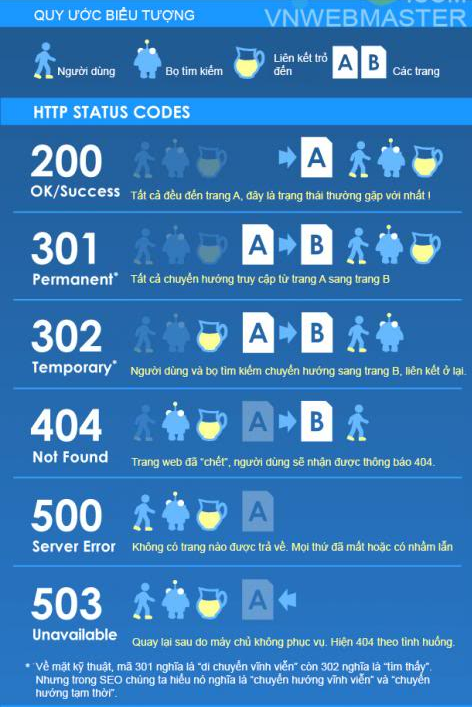
Ta sẽ nhận phản hồi status code là 200.

1. Status Option

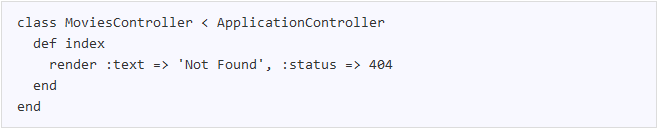
Khi sử dụng Status Option thì bạn có thể gửi lại status code tùy ý



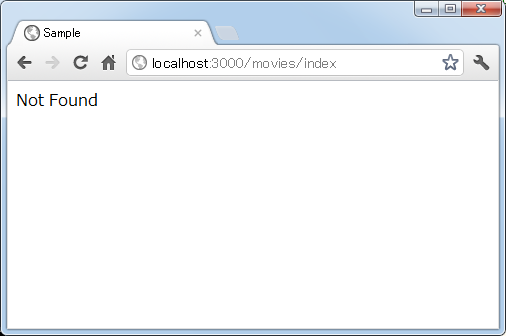
Mà Status code là gì nhỉ ? Nó chính là mã trạng thái HTTP được phản hồi lại khi người dùng gửi request đến webserver.



Sửa mã nguồn của action index



Kết quả



1. Layout Option

Đã hiểu, dịch sau

## **Bài 5.Xuất các dạng JSON/XML từ Action**

1. Xuất ra dạng JSON
2. Xuất ra dạng XML
3. Lựa chọn hình thức sử dụng respond\_to method

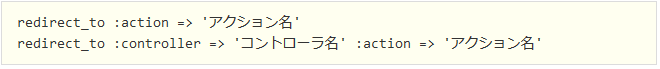
## **Bài 6.Thiết lập Redirect(chuyển tiếp)**

Redirect là gì? Redirect là việc yêu cầu request gọi tới 1 url khác khi nhận được tín hiệu gửi từ brower của người dùng.Ban đầu thì từ action được gọi, nó ko chỉ gọi đến action khác, mà hãy chú ý tới điểm nó còn thay thế request từ người dùng thành url mới.Và bài hôm nay sẽ nói rõ về cách thức redirect hoạt động

1. Xử lý khi redirect

Khi muốn thực thi redirect thì ta sử dụng method redirect\_to. Trước khi redirect ta có thể chỉ định url bên ngoài app hoặc là gọi tới 1 action khác trong app.

Nếu muốn redirect tới 1 actuon khác thì ta làm theo những gì ghi dưới đây



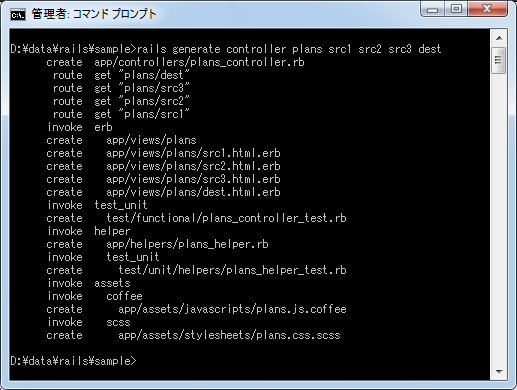
Nếu muốn redirect tới action bên trong cùng controller thì ta chỉ cần chỉ định tên của action muốn redirect tới. còn nếu muốn redirect tới action của controller khác thì tao phải khai báo thêm cả controller chứa action đó theo cấu trúc bên trên.

Ngoài ra ta có thể chỉ định url của site khác hoặc path



Thử ngiệm: Ta tạo thêm 1 controller plans với các action là src1 , src2 ,src3,dest trong app sample

C:\Users\chu\Desktop\vf.PNG



Chỉnh sửa nội dung của file plans\_controller.rb như sau



(Phần sau về nhà viết tiếp)

1. Thay đổi code status
2. DoubleRenderError khi sử dụng Redirect

**Phụ luc**: Sự khác biệt giữa render và redirect\_to

<http://trwtnb.blogspot.jp/2009/11/rubyrailsrenderredirectto.html>

<http://ruby-rails.hatenadiary.com/entry/20140808/1407427457>

## **Bài 7.Lấy parameter**

Thông thường khi sử dụng 1 app thì không chỉ đơn giản là gửi request, ví dụ khi đăng ký dữ liệu mới thì trên brower ta phải nhập vào thông tin, đồng thời ta phải gửi đi cả request và parameter. Như vậy ở bài này ta sẽ đi tìm hiểu phương pháp lấy và sử dụng parameter đã được gửi đến.

1. Lấy Parametter

Việc gửi paramatter như thế nào cùng với request từ người dùng sẽ được trình bày ở phần 4(Thiết lập routing) nhưng trong trường hợp sử dụng GET của phương thức HTTP thì Rails sẽ gửi request có bao gồm parameter với URL được biểu diễn như những gì được viết tiếp theo đây.

Ví dụ , trong controller “travels” có action “show” được gọi đến thì khi đó request có bao gồm giá trị của parameter có tên là “id”, sẽ được truy cập với URL có dạng như sau

C:\Users\chu\Desktop\url_with_parametter.PNG

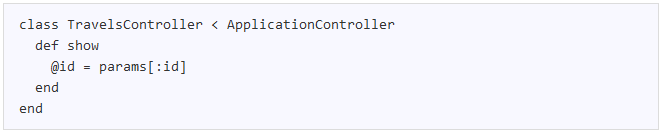
Routing sẽ được thiết lập như sau

C:\Users\chu\Desktop\routing_parametter.PNG

Khi ta khai báo như vậy thì ta có thể lấy được giá trị của parameter id là “10” chứa trong request.Tiếp theo, bên trong action method ta có thể lấy giá chị chỉ định của parameter như sau

C:\Users\chu\Desktop\khaibaopara.PNG

Bây giờ ta sẽ tiến hành lấy và sử dụng giá trị của parameter



(Về nhà viết tiếp)

1. Parametter trong các hình thức querry

Trong Rails có lẽ ko được sử dụng nhiều nhưng việc chuyển parameter thông qua hình thức query có thể thực hiện được. Hình thức query chính là URL được viết theo dang sau

C:\Users\chu\Desktop\url_query.PNG

Sau URL sẽ là dấu “?” và tiếp theo đó là “tên\_parametter=giá\_trị”

Trong trường hợp có nhiều parameter thì ta sẽ dùng ký tự “&” để ngăn cách chúng.Tiếp theo ta sẽ thiết lập routing bằng cách chỉnh sửa trong file config/routes.rb



(Ve nha viet tiep)

1. Lấy Parametter khi sử dụng truyền tin bằng POST

Phần này chỉ đúng có Rails3, sang Rails4 sẽ có lỗi

<http://blog.codebook-10000.com/entry/20140419/1397903226>

1. Chuyển hash bằng parametter
2. Chuyển mảng bằng parameter

Phần d,e sử dụng lại ví dụ của phần c nên cũng ko thực hiện được

==================================================================

# **Phần 4: Thiết lập Routing**

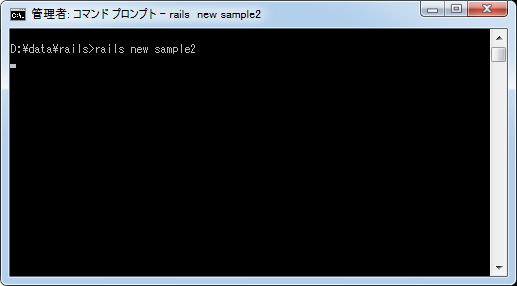
**Phần 4.1: Routing**

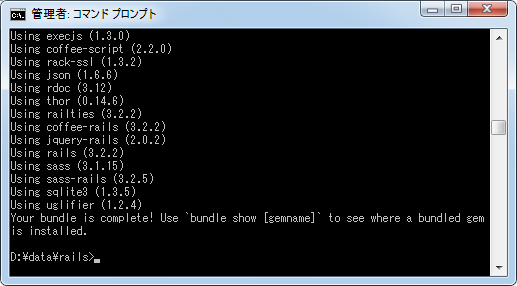
## **Bài 1.Phương pháp thiết lập, xác nhận và vị trí ghi của routing**

Trong bài này mình sẽ thuyết minh về vị trí ghi lại thiết lập liên quan đến routing được sử dụng trong Rails.Ngoài ra mình cũng sẽ nói về thứ tự xác định routing đã được thiết lập.

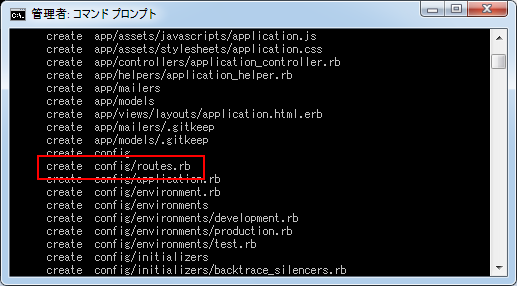
1. File ghi routing

File ghi lại thiết lập routing là ghi vào file tự động được sinh ra “config/routes.rb” khi Rails App sinh ra. Ta thử kiểm chứng bằng cách tạo ra 1 app rails thực tế có tên “sample2”

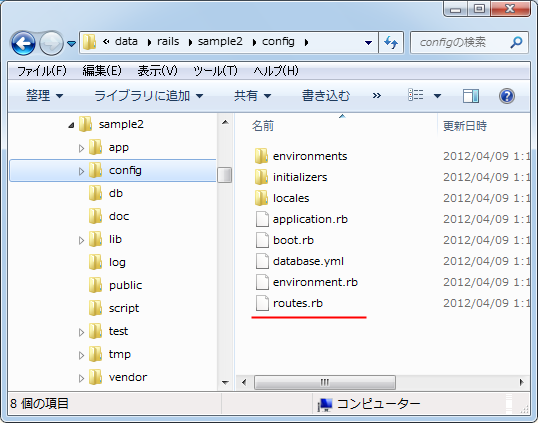




Khi Rails App được sinh ra thì có rất nhiều file cũng tự động được sinh ra, trong số đó thì có file “config/routes.rb” ghi lại những thiết lập liên quan đến routing cũng được hình thành.



Ta có thể xác nhận vị trí thực tế của file này



“routes.rb” là file script được viết bằng ngôn ngữ Ruby. Để xác nhận nội dung bên trong ta có thể mở nó bằng TextEditor.

(Phần nội dung file)

Các dòng bắt đầu bằng ký tự # là các dòng comment. Các mẫu sampke của cách viết routing được viết dưới dạng comment. Giờ ta hãy xóa hết toàn bộ nội dung comment.

(Bổ sung hình ảnh sau và dịch nốt phần còn lại)

1. Phương pháp xác nhận routing đã được thiết lập

Thiết lập liên quan tới routing được viết trong file “routes.rb” có thể xác nhận bất cứ lúc nào bằng cách thực thi command dưới đây

C:\Users\chu\Desktop\routest.PNG

(Phần còn lại sẽ dịch sau)

Như vậy bằng cách thực thi câu lệnh command đơn giản đó thì ta có thể xác định thiết lập liên quan tới routing đang được sử dụng

## **Bài 2.Thiết lập routing đến root(/)**

Phần này nói về việc làm sao để thiết lập root cho 1 rails app. Root ở đây chính là nơi mà rails ap sẽ redirect đến khi app được khởi động.(Phần này sẽ được dịch lại sau)

## **Bài 3.Routing đã được sử dụng trước kia**

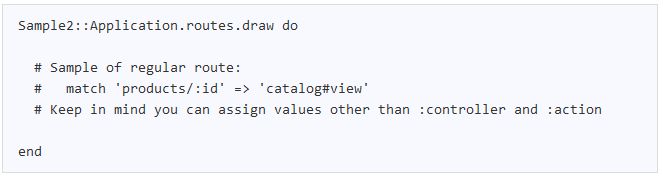
Về nhà dịch phần này

## **Bài 4.Thiết lập routing sử dụng match method**

Bài này sẽ thuyết mình vè phương pháp thiết lập routing ko phải là resource base tiếp tục của trang trước(???-> tối nghĩa vãi). Tại đây ta sẽ nói về cách thành lập routing kết nối 1-1 cho action được gọi từ URL của request, sử dụng phương thức match.

1. Hình thức viết cơ bản đã được sử dụng của match method

Routing đã sử dụng phương thức match sẽ kết nối action gọi từ path url của request từ người dùng.Trong file “config/routes.rb” có rất nhiều comment ví dụ về cách thiết lập nhưng lần này ta chỉ sử dụng những gì được ghi dưới đây



Tiếp theo là cách khai báo method match

C:\Users\chu\Desktop\match.PNG

“パス” ở đây chính là phần path của url trong request “http://

localhost:3000/パス”.Ví dụ khi thực thi truy cập và url có dạng là

“http://localhost:3000/disp” thì ta sẽ gọi đến action show trong controller book, như vậy ta sẽ khai báo routing như sau

C:\Users\chu\Desktop\match_vd.PNG

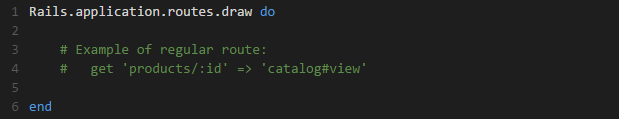
Với url bao gồm cả path như vậy thì ta có thể thiết lập action gọi ra.Tương tự như ví dụ trên trong trường hợp ta muôn truy cập vào url có dạng “http://localhost:3000/books/show” thì ta chỉ việc thiết lập routing như sau

C:\Users\chu\Desktop\vd2.PNG

(Phần tiếp theo về nhà dịch)



P/s: Trong Rails 4 thì phần này đã bị bỏ và thay thế bằng



Và mình test thử với match thì báo lỗi và nó đưa ra cảnh báo là ko nên sử dụng match trong phương thức HTTP, nếu có cần có thêm

“:via => [:get , :post]” để khai báo về phương thức sử dụng, giữ cả 2 nếu muốn sử dụng cả 2 phương thức get và post, nếu ko có thể sử dụng tùy ý từng phương thức.match có thể dùng cho mọi loại phương thức, còn get thì chỉ dùng trong http, mặc dù mục đích sử dụng là như nhau. Như vậy

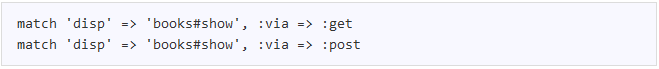
match ‘welcome/index’ => ‘welcome#index’ , :via => [:get]

get ‘welcome/index’ => ‘welcome#index’

2 cách viết trên là tương đương.Tiếp tục bài giảng nào.

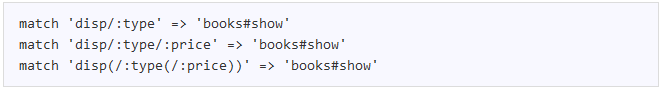
1. Chỉ định phương thức HTTP

Về dịch





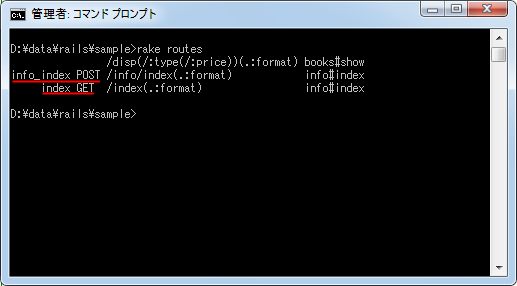
1. Chỉ định parameter

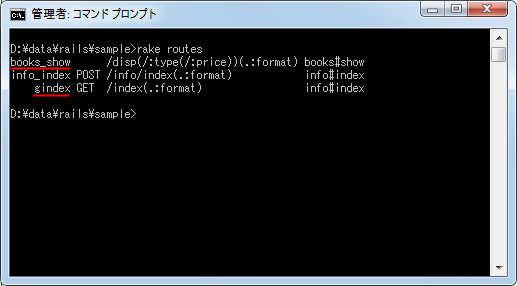


C:\Users\chu\Desktop\match_parametter_2.PNG

1. Xác nhận tương tác
2. Gắn tên vào routing bằng as option

As sẽ được sử dụng để chỉ định tên cho routing, và được sử dụng trong phần “xây dựng link sử dụng phương thức link\_to” trong phần 6(Về dịch)





==================================================================

**Phần 4.2: Routing RESTful base**

## **Bài 5.Thiết lập routing của resource base**

Dựa theo cách suy nghĩ gọi là REST mà các ứng dụng được tạo ra thì ta gọi đó là ứng dụng RESTful. Với version hiện tại của Rails,để tạo ra các ứng dụng RESTfull thì có rất nhiều chức năng đã được chuẩn bị sẵn, nếu không vì lý do gì đặc biệt thì ta nên dùng những chức năng này tiến hành tạo ra các ứng dụng ngắn gọn.Bài hôm nay mình sẽ nói về phương pháp thiết lập routing của resource base trong trường hợp tạo ra ứng dụng Rails RESTful.

1. Thiết kế RESTful

Đầu tiên ta sẽ nói 1 cách đơn giản về REST. REST là gì ?REST là 1 phương pháp thiết kế ứng dụng, nó tiến hành tổ chức các thao tác sử dụng “GET,POST,DELETE,PUT” của phương thức HTTP đối với việc biểu diễn URL bằng resource của đối tượng tương tác.Nếu truy cập vào 1 URL có sẵn bằng cách sử dụng GET thì ta có thể lấy được kết quả nhưng, không thể tiến hành những việc như là tạo mới nếu truy cập vào cùng 1 URL bằng PUT

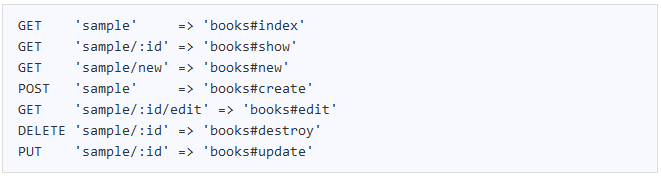
Trong Rails,resource của đối tượng dĩ nhiên là các table của database rồi.URL biểu diễn resource này,vì dụ như là “http://local

host:3000/sample”. Nếu dựa theo cách nghĩ của REST thì đối với 1 URL,nó sẽ cố gắng tiến hành xây dựng dữ liệu nếu truy cập bằng PUT, lấy dữ liệu nếu truy cập bằng GET của phương thức HTTP.

Trong Rails, action được gọi từ request của người dùng nên mỗi tổ hơp URL và phương thức HTTP đã được quyết định là tốt nếu xây dựng routing cho action được gọi.Với lập luận như vậy ta hãy xem ví dụ tiếp theo đây



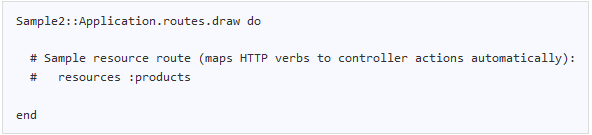
Thế nhưng, thực tế thì trong trường hợp truy cập bằng GET, những trường hợp như chỉ lấy dữ liệu chỉ định, hay lấy dữ liệu trong bảng cũng có.Ngoài ra, khi đăng ký dữ liệu mới hay thay đổi những dữ liệu có sẵn thì có thể ta có thể gọi ra màn hình đăng ký, rât tiện lợi. Như vậy ta sẽ định nghĩa 7 loại routing như sau:



Trong 1 app RESTfull thì bạn có thể tạo routing cho 7 action với tên giống nhau nhưng khác controller với resource của đối tượng.Nếu nhìn URL thì có thể dễ dàng hiểu nó tiến hành thao tác như thế nào với cái gì.

1. Thiết lập routing sử dụng phương thức resource

Tất nhiên ta có thể dễ dàng tạo ra routing thủ công dựa vào các tổ hợp RESTful nhưng trong Rails, để tạo ra routing bằng các resource base thì các tổ hợp tiện dụng đã được chuẩn bị sẵn.Bây giờ ta hãy vào file “config/routes.rb” và xem những ghi chép ở phần đầu file



Ta chú ý đến hình thức khai báo

C:\Users\chu\Desktop\resource_kt.PNG

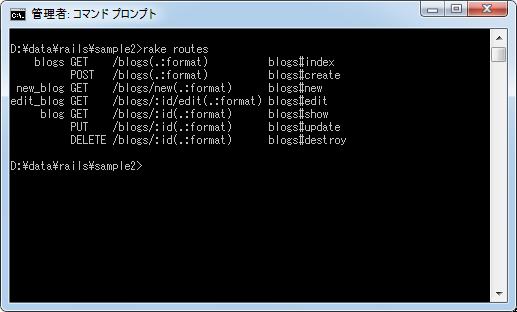
Với phương thức resource cùng với tên resource của đối tượng được chỉ định thì lệnh routing ở trên sẽ tự động tạo ra 7 routing mà ta đã ghi ở phần a.

Tên của resource được chỉ định ở đây chính là url của request hoặc là tên của controller chứa action được gọi.

Nào, bây gườ để xác định resource method khi thành lập sẽ thực hiện thiết lập routing như thế nào thì ra sẽ sửa 1 chút trong file routes.rb như sau

(Bổ sung hình ảnh sau)

Ta thực thi “rake routes” (lệnh này sẽ hiện lên các tương thích giữa URL và method trong controller)



Ý nghĩa



Vậy là ta đã có thể xác định việc thành lập 7 loại routing. Tên resource được chỉ định là tên controller hoặc là url của request được sử dụng.

1. Xác nhận tương tác

(Về nhà dịch, chỉ là test lại các khai báo resource có thực thi ko)

## **Bài 6.Thêm action vào routing**

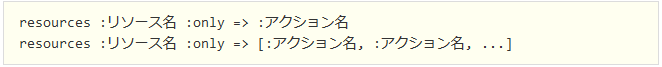
1. Thêm action
2. Cách viêt khác khi thêm action
3. Xác nhận tương tác

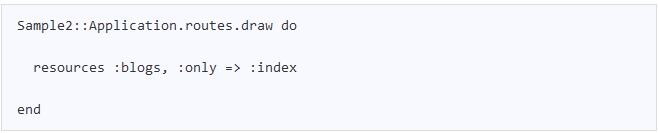
(Về nhà dịch sau)

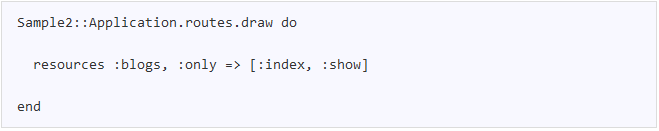
## **Bài 7.Thiết lập action sử dụng chỉ routing đến action**

1. Chỉ định action quan trong

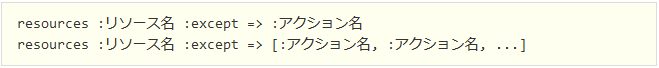
Đầu tiên là phương pháp chỉ định những action quan trọng







1. Chỉ định action không quan trọng



(Về dịch – chưa đọc phần này)

==================================================================

# **Phần 5: Model và Database**

**Phần 5.1: Cơ bản về Model**

## **Bài 1.File cài đặt Database khi dùng SQLite**

Khi hình thành Rails App thì nó cũng tự động hình thành file “database.yml” ghi lại những thiết lập liên quan tới database.Trong bài hôm nay ta sẽ đi làm rõ cách nhìn của file thiết lập database này trong trường hợp sử dụng SQLite.

1. File “config/database.yml”
2. Mode thực thi(chế độ thực thi)
3. Giải thích về file thiết lập

(Về nhà viết)

## **Bài 2.File cài đặt Database khi dùng MySQL**

<Phần này đã được viết rõ trong phần bài học về MySQL>

Để sử dụng được mysql thì mình dùng thử lệnh

>mysql –u root –p

Enter passwork:

Hế, làm gì có pass mà nhập.. Tìm cách reset lại pass, thì tham khảo tại đây

<https://gist.github.com/kenzo0107/94509f453f2d49fc2500>

>sudo su – (đi đến thư mục root trong máy ảo)

>Password: vagrant

>service mysqld start

>mysql\_secure\_installation

Nó sẽ hỏi có muốn reset ko? Nhập pass hiện tại(do lúc đầu mình ko cài pass nên mình enter luôn). Và nó yêu cầu nhập pass mới.Sau khi tạo xong pass mới thì mình test thử mysql

>mysql –u root –p

Enter pass: <Nhập pass vừa đổi vào>

mysql>

Vậy là từ giờ ta có thể thực thi lệnh sql tại đây. Muốn thoát thì Ctr + C

Tất cả những phần kia có thể tham khảo tại

<http://www.dbonline.jp/mysql/>

## **Bài 3.Hình thành Database**

## **Bài 4.Hình thành Model và dạng dữ liệu chỉ định khả năng**

**Phần 5.2: Migration**

## **Bài 5.Chức năng Migration là gì?**

## **Bài 6.Thành lập bảng sử dụng Migration**

## **Bài 7.Thay đổi bảng, thêm migration mới**

## **Bài 8.Rollback của Migration**

## **Bài 9.Thiết lập điều kiện NOT NULL**

## **Bài 10.Nhập dữ liệu đầu vào với rake db:seed**

**(Về dịch)**

==================================================================

# **Phần 6: Template(View) và Helper Method**

**Phần 6.1: Cơ bản về Template**

## **Bài 1. Hình thành Template và sử dụng từ Action**

## **Bài 2. Sử dụng Layout**

## **Bài 3. Thay đổi Layout sử dụng với mỗi action và controller**

## **Bài 4. Template bộ phận**

## **Bài 5. Sử dụng code Ruby trong Template**

## **Bài 6. Chuyển giá trị đến Template sử dụng biến instance**

**Phần 6.2: Helper Method**

## **Bài 7. Escape của văn tự đặc thù HTML**

## **Bài 8. Hình thành link sử dụng method link\_to**

## **Bài 9. Không gắn link cho page hiện tại và page giống nó**

## **Bài 10. Hình thành link mail sử dụng phương thức mail\_to**

## **Bài 11. Hình thành image tag sử dụng image\_tag method**

## **Bài 12. Hình thành Favicon sử dụng favicon\_link\_tag method**