


BUILD RUBY ON RAILS APPLICATION PROTOTYPING LESSION 1



開始之前...

- » 儘量發問
- » Line 討論群組
- » 補充資料
 - » 「為你自己學 Ruby on Rails」
 - » <https://railsbook.tw/>

開始之前...

- » 大部份的同學沒辦法只在上課用聽的或用看的就學得會
- » 請盡量跟著敲打範例
- » 回家一定要練習！
- » 卡關一定要發問！

內容

- » 簡介
- » 環境安裝
- » 你的第一個 Rails 應用程式 (Blog)
 - » 使用 Scaffold
 - » 資料表關連及驗證

簡介

» 什麼是 Ruby?

» 什麼是 Ruby on Rails?

RUBY

- » Ruby 是一種泛用型的程式語言(不限定開發網站)
- » 腳本式程式語言(`scripting language`)
- » 發明人：松本行弘(まつもとゆきひろ) (Matz)
- » 首次公開發表： 1995 年(其實是 1993 年就出生了)
 - » 它已經 25 歲了！

RUBY ON RAILS

- » 簡稱 "Rails"，已少使用 "RoR"
- » 是用 Ruby 這個程式語言所開發出來的網站開發框架
- » 發明人：David Heinemeier Hansson (DHH)
- » 首次公開發表：2004 年

RUBY ON RAILS

» 常見縮寫：

» Convention over Configuration (CoC)

» Don't Repeat Yourself (DRY)

» Create, Read, Update, Delete (CRUD)

» Model, View, Controller (MVC)

ENVIRONMENT & INSTALLATION

» Ruby

» `ruby -v` 檢視目前 Ruby 版本

» `ver 2.0` + (建議 2.5 以上)

» RVM (<https://rvm.io/>)

» Ruby Version Manager

» 可安裝多個不同版本的 Ruby，而且不需要 `root` 權限

環境安裝

» Gem

» 什麼是 Gem?

» 安裝 Gem

» 安裝最新版本 `gem install rails`

» 安裝指定版本 `gem install rails -v 4.2.10`

環境安裝

» Bundler

» 為什麼使用 bundler?

» 緊緊相依的心如何 say goodbye

環境安裝

» Rails

» 目前最新穩定版本 5.2.0

» `gem install rails`

» 安裝最新測試版本

» `gem install rails --pre`

環境安裝

» 開發工具

» Sublime Text (<https://www.sublimetext.com/>)

» Atom (<https://atom.io/>)

» Vim, Emacs

» RubyMine (<https://www.jetbrains.com/ruby/>)

你的第一個 **RAILS** 應用程式 (**BLOG**)

» 設計

- » 你想要開發一個可以發文的 **Blog** 系統
- » 可以新增使用者(`user`)
- » 每個作者可以新增文章、修改或刪除自己寫的文章(`post`)

你的第一個 **RAILS** 應用程式 (**BLOG**)

建立 **RAILS** 專案

» 建立專案：

```
$ rails new blog
```

» 安裝套件：

```
$ bundle install
```

你的第一個 RAILS 應用程式 (BLOG)

使用者 (User)

欄位名稱	在資料庫裡的資料型態
id	數字(integer, 自動跳號)
name	字串(string)
email	字串(string)

你的第一個 RAILS 應用程式 (BLOG)

SCAFFOLDING

第一步 (完整指令) :

```
$ bin/rails generate scaffold User name:string email:string
```

少打一點 :

```
$ rails g scaffold User name email
```

你的第一個 RAILS 應用程式 (BLOG)

MIGRATION

- » 除了原本的 `name` 跟 `email` 外，還有加了 `timestamps`
- » 自動跳號的 `id` 也自動加上去了

```
class CreateUsers < ActiveRecord::Migration[5.2]
  def change
    create_table :users do |t|
      t.string :name
      t.string :email

      t.timestamps
    end
  end
end
```

你的第一個 RAILS 應用程式 (BLOG)

SCAFFOLDING

第二步：

```
$ bin/rails db:migrate
```

(`bin/rails db:rollback` 指令可以退回上一個動作)

這個指令會把 `migration` 檔案裡的定義轉換成真正的資料表

注意，在 `rails 5.0` 之後可用 `rails db:migrate` 指令做跟 `rake` 指令一樣的事。

你的第一個 RAILS 應用程式 (BLOG)

SCAFFOLDING

第三步：

```
$ bin/rails server
```

如果是在 c9.io 上：

```
$ bin/rails server -b $IP -p $PORT
```

這會在你的電腦上啟動一個伺服器

網址：`http://localhost:3000/`

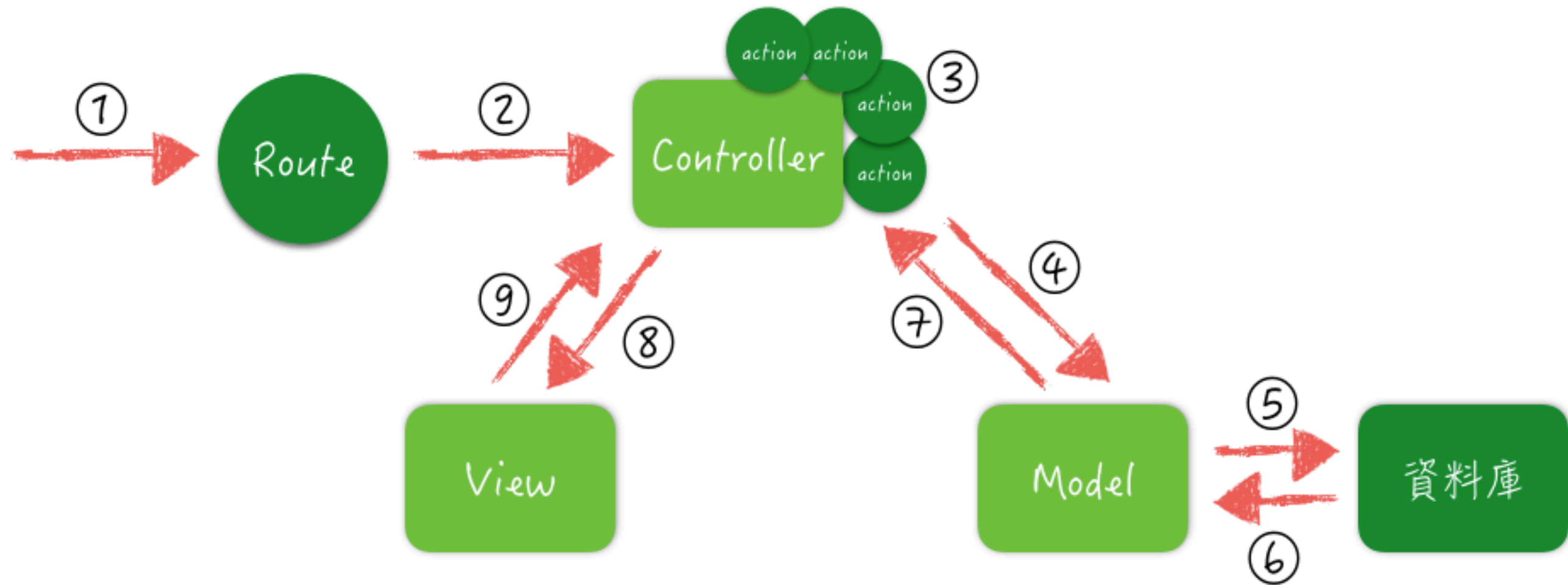
少打一點：

```
$ rails s
```

MVC (模型、視圖、控制器)

- » Model (`app/models/user.rb`)
- » View (`app/views/users/index.html.erb`)
- » Controller (`app/controllers/users_controller.rb`)
- » Router (`config/routes.rb`)

MVC (模型、視圖、控制器)



RAILS 專案的結構

» app/*

» lib/*

» logs/*

» config/*

» db/*

» Gemfile

CONTROLLER

USER CONTROLLER

```
# -----  
# app/controllers/user_controller.rb  
# -----  
class UsersController < ApplicationController  
  def index; end  
  def show; end  
  def new; end  
  def edit; end  
  def create; end  
  def update; end  
  def destroy; end  
end
```


CONTROLLER

USER CONTROLLER

```
# -----  
# app/controllers/user_controller.rb  
# -----  
class UsersController < ApplicationController  
  def index  
    @users = User.all  
  end  
end
```

MODEL

USER MODEL

```
# -----  
# file: app/models/user.rb  
# -----  
class User < ActiveRecord::Base  
end
```

VIEW

USER INDEX PAGE

```
# -----  
# app/views/users/index.html.erb  
# -----
```

...略...

```
<tbody>  
  <% @users.each do |user| %>  
    <tr>  
      <td><%= user.name %></td>  
      <td><%= user.email %></td>  
      <td><%= link_to 'Show', user %></td>  
      <td><%= link_to 'Edit', edit_user_path(user) %></td>  
      <td><%= link_to 'Destroy', user, method: :delete, data: { confirm: 'Are you sure?' } %></td>  
    </tr>  
  <% end %>  
</tbody>
```

...略...

RESTFUL 網址設計⁵

» REST = REpresentational State Transfer

» GET

» POST (create new record)

» PUT & PATCH (update record)

» DELETE (delete record)

⁵ link: <http://zh.wikipedia.org/wiki/REST>

路由 ROUTES

網址路設定

```
# -----
```

```
# config/routes.rb
```

```
# -----
```

```
Rails.application.routes.draw do
```

```
  resources :users
```

```
end
```

路由 ROUTES

```
$ rails routes
```

Prefix	Verb	URI Pattern	Controller#Action
users	GET	/users(:format)	users#index
	POST	/users(:format)	users#create
new_user	GET	/users/new(:format)	users#new
edit_user	GET	/users/:id/edit(:format)	users#edit
user	GET	/users/:id(:format)	users#show
	PATCH	/users/:id(:format)	users#update
	PUT	/users/:id(:format)	users#update
	DELETE	/users/:id(:format)	users#destroy

你的第一個 RAILS 應用程式 (BLOG)

文章 (Post)

欄位名稱	在資料庫裡的資料型態
id	數字(integer, 自動跳號)
title	字串(string)
content	文字(text)
user_id	數字(integer, 對應用 User 的 id 欄位)

你的第一個 RAILS 應用程式 (BLOG)

SCAFFOLDING

第一步：

```
$ rails g scaffold Post title content:text  
user:references
```

第二步：

```
$ rails db:migrate
```


你的第一個 RAILS 應用程式 (BLOG)

MIGRATION

```
class CreatePosts < ActiveRecord::Migration[5.2]
  def change
    create_table :posts do |t|
      t.string :title
      t.text :content
      t.references :user, foreign_key: true

      t.timestamps
    end
  end
end
```

資料表關連 (RELATIONSHIP)

» 一位使用者 (User) 可以寫很多篇文章 (Post)

```
# -----  
# app/models/user.rb  
# -----
```

```
class User < ApplicationRecord  
  has_many :posts  
end
```

資料表關連 (RELATIONSHIP)

» 每篇文章 (Post) 都屬於某位使用者 (User)

```
# -----  
# file: app/models/post.rb  
# -----  
class Post < ApplicationRecord  
  belongs_to :user  
end
```

資料表關連 (RELATIONSHIP)

VIEW

```
<div class="field">
  <%= form.label :user_id %>
  <%= form.text_field :user_id %>
</div>
```

改成

```
<div class="field">
  <%= form.label :user_id %>
  <%= form.collection_select :user_id, User.all, :id, :name %>
</div>
```

資料表關連 (RELATIONSHIP)

» 使用 `rails console` 指令進入控制台模式：

```
>> me = User.first           # 取得第一個會員
>> me.posts                  # 查詢該會員所有的文章
>> the_post = Post.first     # 取得第一篇文章
>> the_post.user             # 查詢該文章的作者是誰
```

驗證 內容長度

```
# -----  
# app/models/post.rb  
# -----
```

```
class Post < ApplicationRecord  
  validates :content, length: { maximum: 20 }  
end
```

New Post

1 error prohibited this post from being saved:

- Content is too long (maximum is 100 characters)

Content

12121212121212121212	
12121212121212121212	

User

122

Create Post

[Back](#)

想想看，為什麼 **RAILS** 這麼厲害？

» Model

```
# -----  
# app/models/user.rb  
# -----
```

```
class User < ApplicationRecord  
  # 明明什麼都沒寫  
end
```


RAILS 常用快速鍵

- » rails generate 📌 rails g
- » rails destroy 📌 rails d
- » rails server 📌 rails s
- » rails console 📌 rails c
- » rails dbconsole 📌 rails db
- » bundle install 📌 bundle
- » rake test 📌 rake

學到了什麼？

- » Rails 專案結構
- » MVC
- » Scaffolding
- » Restful 路由
- » 資料表關連
- » 資料驗證

課後練習

1. 試著使用 Rails 的 Scaffold 建立以下功能：
 - a. 每間書店可以賣很多書。
 - b. 每本書應該都有一位作者。
 - c. 完成後請把作業上傳至 GitHub。
 - d. 加分題：每本書可能不只一位作者。
 - e. 加分題：試著套用 Bootstrap 讓畫面變好看一點。

聯絡資訊

- » Blog: <https://kaochenlong.com>
- » Facebook: <https://www.facebook.com/eddiekao>
- » Twitter: <https://twitter.com/eddiekao>
- » Email: eddie@5xruby.tw
- » Mobile: +886-928-617-687