

# RSpec 心得分享

# About me

- sdlong
- 半路出家的 Rails Developer
- 喜歡分享學程式開發的學習心得

# 問卷調查

- 完全沒有 Rails 開發經驗的請舉手
- 有開發過 Rails 並寫過測試(無論何種) 的請舉手
- RSpec 寫得融會貫通出神入化的請舉手  
(請到外面休息，等下一場的 MarsZ 大大演講謝謝)

# 進入主題

RSpec 心得分享

這場 talk 的內容

# 沒辦法讓你在聽完後

- 馬上變成寫測試大師
- 我要打十個！ 管他是  
unit test,  
integration test,  
front-end test,  
API test,  
3rd party services test  
通通都是一塊蛋糕
- 找到另一半

因時間限制，所以 ...

# 期望幫助你

- 了解 RSpec 的基本運作
- 架構測試撰寫的思維
- 使用讓測試寫更順的好用工具
- 進入實作情境 => Live Demo: 從 0 開始寫出基本的 RSpec 測試



# 流程

- 10 分鐘 - 前言與 RSpec 基本概念
- 20 分鐘 - Live Demo 基本測試撰寫
- 15 分鐘 - Live Demo 翻修程式碼
- 10 分鐘 - 架構測試撰寫的概念與方法論
- 5 分鐘 - Q & A

什麼是 RSpec ?

請看官網介紹，謝謝

這邊著重在實戰經驗分享

# 程式的運作原理

Input => 演算法 => Output

# 測試的運作原理

將測試參數與邏輯丟進去

=> (程式運作) =>

判斷出來的結果是否符合預期


就這麼簡單

# 如果真的想看 RSpec 介紹

請參考 iHower 實戰聖經裡的 自動化測試 章節

為什麼要寫測試？



A man with a beard, wearing a black cap and a yellow robe, is shown in profile, looking upwards. He is holding a string of dark prayer beads. The background is a blurred interior with colorful decorations and a yellow wall with red text.

施主，這個問題你應該要問你自己

# 寫測試的原因

- 接手既有專案，藉由寫測試了解架構
- 專案已經到不得不大改，但是牽一髮動全身的狀態
- 時間太多不知道怎麼打發

# 讓維護與開發更順利

寫測試的目的

撰寫 RSpec 的心法

# MVO 原則

- Minimum Valid Object
- 先建一個 valid (合法, 可行) 的物件, 讓測試 pass
- 做一點小改變
- 讓測試預期變成 no pass

# 以 Rails model 舉例

- 我要撰寫 `class User < ActiveRecord::Base` 的測試
- 裡面有很多 `validate`  
( e.g. : `validates :email, presence: true` )
- 先寫一個可以成功建立，並存入 DB 的物件，然後判斷此物件是過的
- 修改物件裡的參數內容，讓它存入 DB 失敗 => 判斷此物件 fail, 錯誤原因: XXX

到目前為止，只需要先  
了解 MVO 概念就好

接下來我們直接進入  
Live Demo

[Source Code](#)



在演講結束後會把投影片與  
Live Demo 的 source code 分享出來

所以不用急著筆記

專心在 Live Demo 就好

# Live Demo 流程

- 基本測試撰寫
  - 安裝 Rspec
  - 撰寫 Model 測試
  - 使用工具幫助撰寫測試更順暢
  - 撰寫 Controller 測試
- Refactor 測試程式碼

# 基本測試撰寫

Live Demo

# 回顧 Demo 內容

- 先記住 MVO 概念就好
- 以 MVO 概念撰寫 Model 測試要點
  1. 先建立該 model 的 instance
  2. `expect(instance.method).to`（預期結果）
  3. 修改 instance / method 的參數，預期會得出什麼不同結果（像是錯誤訊息）

# 回顧 Demo 內容

- 許多 Rails 內建功能的測試直接用 Gem 處理掉就好  
( `validate`, `before action`, `association ... etc` )

# 回顧 Demo 內容

- Controller 的測試主要在 request / response 上
  1. 確保 status ( 200 或 redirect\_to xxx )
  2. 確保 format ( html / json ... etc )
  3. 業務邏輯的部份可以抽出來變成 service object / form object 另外撰寫它們的測試

# 翻修程式碼

Live Demo



# 翻修程式碼回顧

- 使用 Faker 來自動生成亂數假資料
- 使用 FactoryGirl 來預建常用物件
  1. 像是各種身份的角色、各種狀態的訂單
  2. 物件設定裡面還可以再建物件
  3. 好處是在 `_spec.rb` 裡一樣的變數 (例如 `let(:user)`) 可以因應不同情境，設定成 `admin` / `normal` / 其他身份

# 翻修程式碼回顧

- 善用 `before` 與 `block` 來做出一樣邏輯，但是程式更簡潔
- 使用 `shared_examples_for` 來簡化重複的程式碼

# Gem list

- `gem 'rspec-rails' # 這不用解釋吧 XDD`
- `gem "fuubar" # 用進度百分比顯示 rspec 狀態`
- `gem 'guard-rspec' # 存檔時自動跑該檔案對應的 rspec`
- `gem "terminal-notifier-guard" # 將測試訊息顯示在通知中心上`
- `gem 'shoulda-matchers' # 簡化 model / controller 測試`
- `gem 'faker' # 建測試假資料 (隨機人名、字串、Email ... etc )`
- `gem "factory_girl_rails", "~> 4.0" # 建測試用物件`
- `gem 'database_clean' # 執行測試時自動清空 test DB 的 gem`

以上就是 RSpec 的  
基本概念

即使如此  
實際撰寫測試  
還是會面臨一個問題

環境裝完以後  
腦袋還是一片空白

# 不知道該如何動手？

會有這問題的請舉手

以下是小弟對於  
撰寫測試的概念與方法論



# 概念

- 測試絕對寫不完，別去想『怎麼寫完』
- 專注在『怎麼開始』就好
- 註解多寫不是壞事，直接在 `_spec.rb` 裡寫下對於該測試的想法、測試邏輯、TODO、FIXME
- 先在腦袋裡架構好整個世界觀與運作流程，再開始動手

用講得很簡單  
要怎麼做？

我從某本書上學到一招  
很實用的方法

# 方法論

- 拿一張紙，想著你要完成某件事，寫下以下要點
  1. 想像希望出來的成果是什麼？
  2. 需要做 (Do) 什麼事？
  3. 完成 (Done) 的定義
  4. 過程 (Doing) 的樣貌

這就好像







……他在……

做什麼……！



Image  
Training——

……

30



Image……  
Training——

一流的  
外科醫生，

在腦海中有一幅  
真實的人體影像，  
可以藉此自己做開  
刀的練習——

要是集中力跟對人  
體的瞭解不夠，是  
做不到的。



他現在，

正在做  
巴提斯塔手術。



不只寫測試，很多事物  
都適用這套方法



希望以上內容  
可以實際幫助到大家

# 參考資料 / 學習來源

- Everyday Rails Test with RSpec ([連結](#))
- Tealeaf 第三階課程 ([連結](#))
- RSpec Design Patterns by Adam Cuppy ([連結](#))
- Better Rspec ([連結](#))

Q & A