

# 開發網站應用程式 使用 Ruby on Rails

Test-Driven Development(TDD)

# My Bentobox

Application:

Designed by:

## Storage

Backend.  
How the application stores data.

資料庫

## Logic

Backend.  
How the application works.

Ruby

Rails

## Style and structure

Frontend.  
How the application looks.

HTML

CSS

JavaScript

## Infrastructure

Backend.  
How the application runs.

伺服器

Git

# Test-Driven Development (TDD)

測試驅動開發

什麼是測試？

# 關於測試

- 用滑鼠點一點也是一種測試。
- 測試其實是一段程式碼。
  - 寫測試 = 寫一段**程式**，確認另一段程式可正常運作。

# 測試功能

# 計算分數

```
def calc_score(x, y)  
    return 2 * x * y  
end
```

# 測試它是否正常運作

```
if calc_score(3, 5) == 30  
    puts "success!"  
else  
    puts "error"  
end
```

所以，為什麼要寫測試？



**When i try to fix a bug**



確保程式可以依照預期  
行為運作



請工讀生用滑鼠點一點不行嗎？

為什麼不寫測試？

# 為什麼不寫測試

- 「光寫主程式都沒時間了， 哪裡還有時間寫測試」
- 「跑測試太慢了」
- 「測試很脆弱耶， 不小心改一下就爛掉了」
- 「不知道怎麼寫」



重點

# 什麼是 TDD

# Test-Driven Development (TDD)

測試驅動開發

# TDD 是一種**開發方法**

重點是 development

先寫測試, 再寫程式



# 先寫測試, 再寫程式

# 測試它是否正常運作

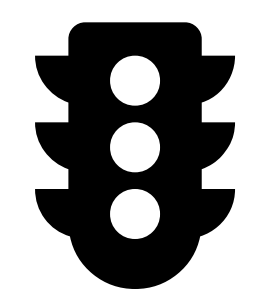
```
if calc_score(3, 5) == 30
  puts "success!"
else
  puts "error"
end
```

執行後會?

明明 `calc_score` 還沒寫，  
是要怎麼測？

測試**不存在**的功能,  
**假設**它可正常運作

在公堂之上  
大膽假設一下

 紅燈、綠燈

測試**失敗**（紅燈），

再想辦法**修正**這個問題（綠燈）

可以用更有效率、  
更有系統的方式來寫測試

# 測試專用套件



# minitest 與 RSpec

為什麼選用 RSpec

安裝 RSpec

# 安裝 RSpec

```
$ gem install rspec
```

```
Fetching rspec-mocks-3.8.0.gem
```

```
Fetching rspec-expectations-3.8.3.gem
```

```
Fetching rspec-support-3.8.0.gem
```

```
Fetching rspec-3.8.0.gem
```

```
...略...
```

```
Done installing documentation for diff-lcs, rspec-support,  
rspec-mocks, rspec-expectations, rspec-core, rspec after 3  
seconds
```

```
6 gems installed
```



動手寫測試！

# ATM 功能

規格

- **存錢**功能

- 可以存錢
- 不可以存 0 元或是小於 0 元的金額（越存錢越少！）

- **領錢**功能

- 可以領錢
- 不能領 0 元或是小於 0 元的金額（越領錢越多！）
- 不能領超過本身餘額

# 先寫測試

 檔案名稱 atm\_spec.rb

```
RSpec.describe ATM do
  describe "存錢功能" do
    it "可以存錢" do
      # 測試內容待會寫...
    end
  end
end
```



# 執行測試, 發生錯誤

```
$ rspec atm_spec.rb
```

```
An error occurred while loading ./atm_spec.rb.
```

```
Failure/Error:
```

```
...略...
```

```
NameError:
```

```
uninitialized constant ATM
```

```
# ./atm_spec.rb:1:in `'
```

```
No examples found.
```

```
Finished in 0.00021 seconds (files took 0.09145 seconds to load)
```

```
0 examples, 0 failures, 1 error occurred outside of examples
```



你什麼時候有寫了 **ATM** 類別的錯覺了！

該怎麼解決這個錯誤訊息？

# 繼續寫測試

# 先寫測試

📄 檔案名稱 atm\_spec.rb

```
RSpec.describe ATM do
  describe "存錢功能" do
    it "可以存錢" do
      atm = ATM.new(10)      # 有一個 ATM 帳戶，一開始有 10 元
      atm.deposit 20         # 存入 20 元
      expect(atm.balance).to be 30    # 現在應該有 30 元
    end
  end
end
```

等等...

你的 **ATM** 什麼時候有 **deposit** 功能了？

所以，執行一定會發生錯誤！



然後想辦法通過這個測試

測試**失敗**（紅燈），

再想辦法**修正**這個問題（綠燈）

小步前進

## 【 冷<sup>カ</sup>知<sup>チ</sup>識<sup>シ</sup> 】

- 即然 TDD 是**寫程式來測試程式**， 那這些 TDD 的程式要誰來測試？

所以，為什麼要寫測試？

# 為什麼要寫測試

- 測試本身就是規格(Spec)
- 寫出更有信心的程式碼
- 可以做出比較好的設計
- 將來有重構(refactor)的可能性

# 常見問題

- 「我要怎麼測試還不存在的程式碼？」
- 「我怎麼知道哪些東西應該要測？」