

# TEST-DRIVEN DEVELOPMENT WRITE CONFIDENCE CODE 🍑

# 內容

- » 什麼是 TDD?
- » 為什麼要 TDD? 為什麼不做 TDD?
- » 如何進行 TDD?
  - » 紅燈(Red)、綠燈(Green)及重構(Refactor)

# TDD

## TEST-DRIVEN DEVELOPMENT

- » 為什麼寫測試？
  - » 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
  - » 有重構 (refactor) 的可能性



# TDD

## TEST-DRIVEN DEVELOPMENT

- » 為什麼不寫測試？
  - » 沒時間
  - » 不知道怎麼寫
  - » 測試太慢了
  - » 測試很容易爛掉

# TDD

## TEST-DRIVEN DEVELOPMENT

- » TDD = 寫程式來測試你的程式
- » 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？

# TDD

## 實際動手作

» 銀行

» 存錢功能

» 可以存錢

» 不能存小於或等於 0 的金額

# TDD

## 實際動手作

» 銀行

» 領錢功能

» 可以領錢

» 不能領小於或等於 0 的金額

» 不能領超過本身餘額

# TDD

## 實際動手作

» 使用 RSpec :

```
» gem install rspec
```



# TDD

## 實際動手作

# 檔案名稱：bank\_spec.rb

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

# TDD

## 實際動手作

- » 執行 `rspec ./bank_spec.rb`
- » 一定會出錯！(紅燈)
- » 為什麼會出錯？
- » 沒出錯就一定是你還沒睡醒

# TDD

## 實際動手作

問：怎麼解決剛剛發生的錯誤訊息？

# TDD

## 實際動手作

# 檔案名稱：bank\_spec.rb

```
RSpec.describe Bank do
  describe "存錢功能" do
    it "可以存錢" do
      bank = Bank.new(10)
      bank.deposit 20
      expect(bank.balance).to be 30
    end
  end
end
```

# 假設有一個 Bank，一開始有 10 元  
# 存入 20 塊  
# 現在應該有 30 塊

# TDD

## 實際動手作

» 這時候才開始實作真正功能

```
class Bank
  attr_reader :balance

  def initialize(amount)
    @balance = amount
  end

  def deposit(amount)
    @balance += amount
  end
end
```

# TDD

## 實際動手作

- » 執行 `rspec ./bank_spec.rb`
- » 這時候應該就會正確了！〔綠燈〕
- » 測試有過不保證就是功能正確
- » 然後繼續寫下一個測試，繼續紅燈！

# 想想看...

» 問題：既然 TDD 是寫程式來測試程式，那 TDD 的程式要試來測試？

# TDD

## TEST-DRIVEN DEVELOPMENT

- » 所以，為什麼寫測試？
  - » 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
  - » 有重構 (refactor) 的可能性



# 聯絡 高見龍 (EDDIE)

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