

# RSpec & TDD

## Tutorial

<https://ihower.tw>

2016/8

# Who Am I?

- 張文鈞 a.k.a. ihower
  - <https://ihower.tw>
  - Build web application since 2002



# Ruby on Rails 實戰聖經

使用 *Rails 4.2* 及 *Ruby 2.1*

電子書和簡體版本準備中。如果您有任何意見、鼓勵或勘誤，歡迎來信給我，謝謝。

我是 ihower，本書介紹 Ruby on Rails 這套開放原始碼的網站開發框架。如果您對這本書有任何意見或勘誤指教，歡迎來信和我聯絡。

[關於本書](#)

[回首頁](#)

## Part 1: 入門實作

1. Ruby on Rails 簡介
2. 安裝 Rails 開發環境
3. Rails 起步走
4. Ruby 程式語言入門
5. 手工打造 CRUD 應用程式
6. RESTful 應用程式
7. Active Record 基本操作與關聯設計
8. RESTful 綜合應用

## Part 2: 深度剖析

1. 環境設定與 Bundler
2. 路由 (Routing)
3. Action Controller: 控制 HTTP 流程
4. Active Record: 資料表操作
5. Active Record: 資料庫遷移 (Migration)
6. Active Record: 資料表關聯
7. Active Record: 資料驗證及回呼
8. Active Record: 追蹤功能
9. Action View: 樣板設計
10. Action View: Helpers 方法
11. Ajax 應用程式
12. Assets Pipeline

The screenshot shows the top navigation bar of the alphacamp.co website. It includes the logo 'ALPHAcamp' with an orange 'A', followed by links for 'ABOUT US', 'BOOTCAMP', 'STUDENTS', 'SEMINARS', 'APPLY', and 'COUNTRY'. The 'APPLY' button is highlighted with an orange background.

# 網站開發工程師實戰營

**FULL TIME | 每週五天**

十週 Ruby on Rails 課程實戰訓練，帶你學會 Rails、HTML、CSS 和 JavaScript，

打通前端、後端開發任督二脈，成為全球炙手可熱的網路全端工程師（Full Stack Developer），或親手實現創業的 idea！



瞭解更多

## 十週內打造全方位的 **FULL STACK** 開發能力

### 你將會學到

- ✓ Ruby 程式語言
- ✓ 测試驅動開發
- ✓ Rails 網站開發框架
- ✓ Web API 設計
- ✓ 前端網頁設計
- ✓ 網站效能調校
- ✓ 版本控制開發流程
- ✓ 網站安全

# Agenda

- 什麼是測試？寫測試有什麼好處？
- RSpec 語法介紹
- TDD Kata 練習

# 什麼是測試？



Search



Home Profile Messages Who To Follow



ihower ▾



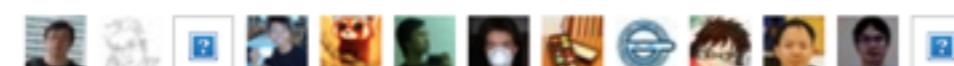
@ihower

Wen-Tien Chang

為什麼測試喜歡搞這麼多名目：單元測試、驗收測試、需求測試、煙霧測試、回歸測試、整合測試、系統測試、功能測試、確認(**validation**)測試、白箱測試、黑箱測試、灰箱測試、錯誤處理測試、壓力測試、效能測試、安全性測試、相容性測試、使用性測試、完整性測試、結構測試、安裝測試、

27 Sep via web

Retweeted by [sandzhang](#) and 15 others



Sorry, I'm not QA guy :/

# 測試 Overview

單元測試  
Unit Test

整合測試  
Integration Test

驗收測試  
Acceptance Test

## Verification

確認程式的執行結果有沒有對?  
Are we writing the code right?



## Validation

檢查這軟體有滿足用戶需求嗎?  
Are we writing the right software?

## 單元測試 Unit Test

測試個別的類別和函式結果正確

## 整合測試 Integration Test

測試多個類別之間的互動正確

## 驗收測試 Acceptance Test

從用戶觀點測試整個軟體

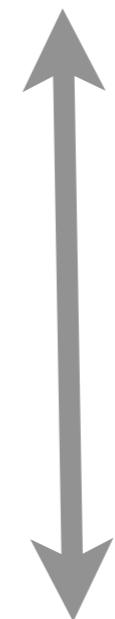


單元測試  
Unit Test

整合測試  
Integration Test

驗收測試  
Acceptance Test

白箱測試



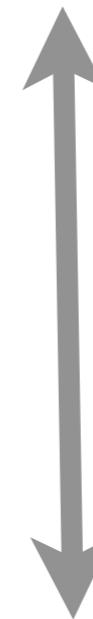
黑箱測試

單元測試  
Unit Test

整合測試  
Integration Test

驗收測試  
Acceptance Test

Developer 開發者



QA 測試人員

# Verification

確認程式的執行結果有沒有對?

Are we writing the code right?

# 我們天天都在檢查程 式的結果有沒有對

- 程式寫完後，手動執行看看結果對不對、打開瀏覽器看看結果對不對？(如果你是web developer 的話)
- 聽說有人只檢查 syntax 語法沒錯就 commit 程式的？

# 手動測試很沒效率

特別是複雜或 dependency 較多的程式，你可能會偷懶沒測試所有的執行路徑

# 有三種路徑，你得執行(手動測試)三次才能完整檢查

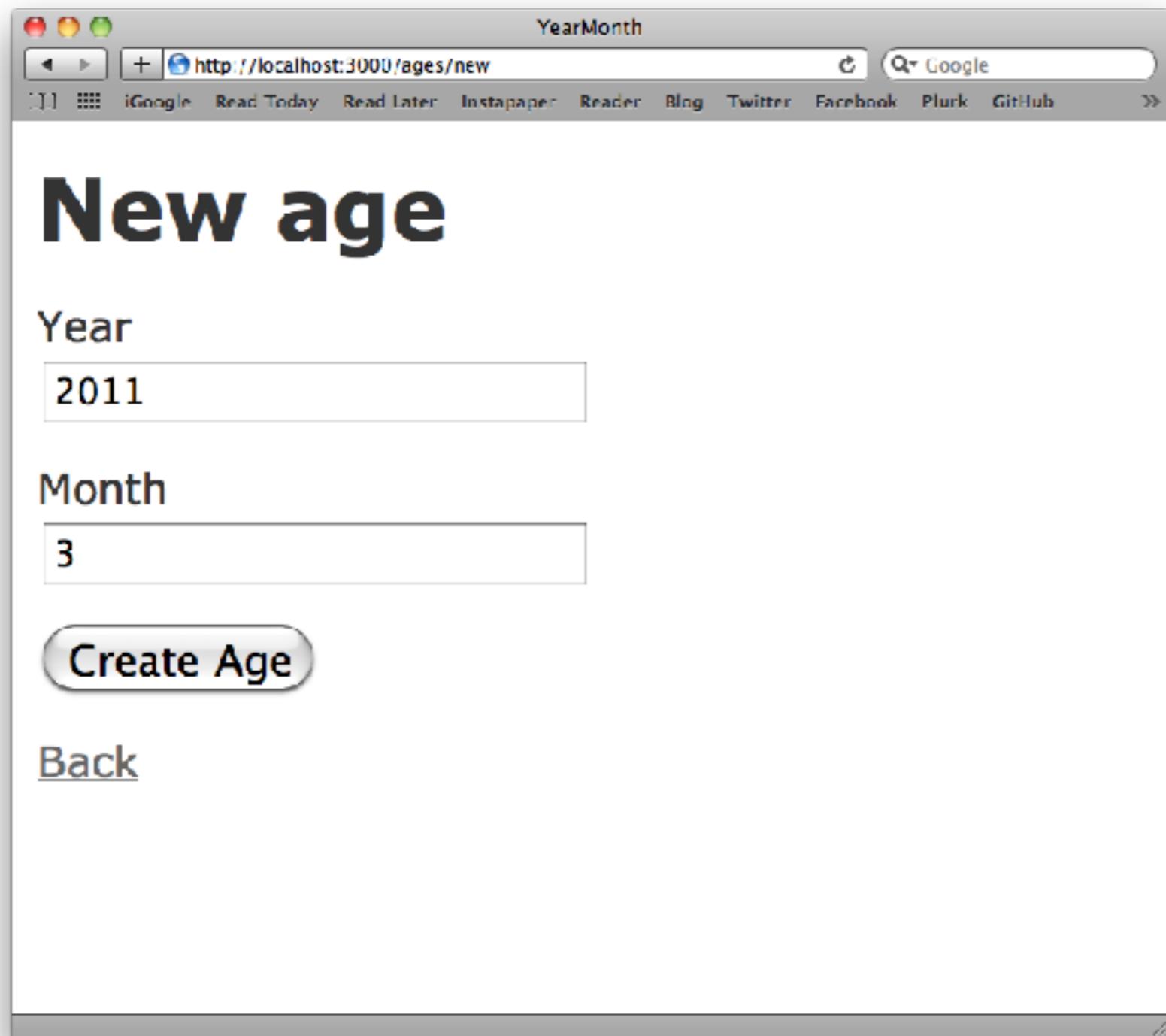
```
def correct_year_month(year, month)

    if month > 12
        if month % 12 == 0
            year += (month - 12) / 12
            month = 12
        else
            year += month / 12
            month = month % 12
        end
    end

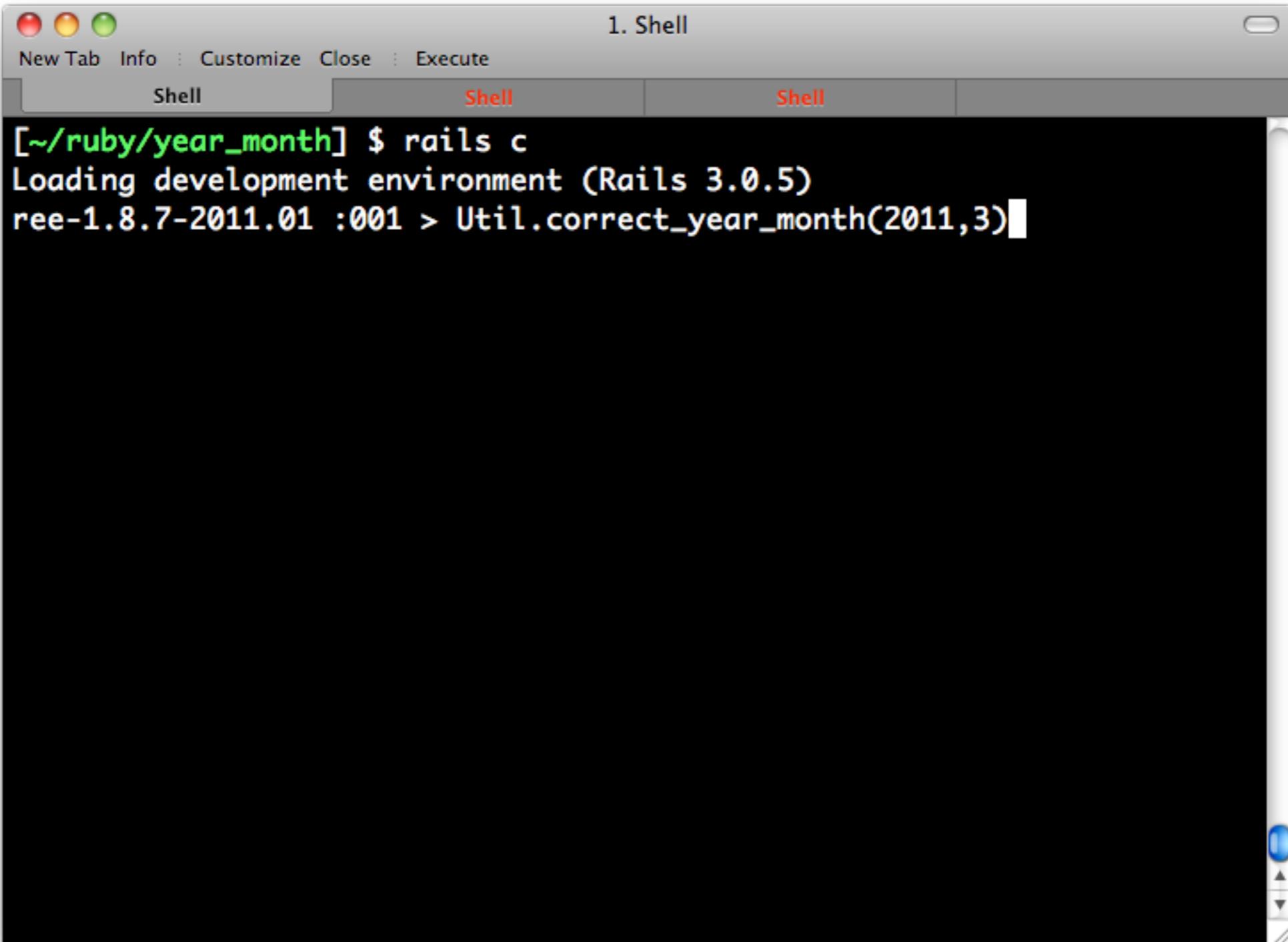
    return [year, month]

end
```

# 用介面手動測試？



# 用 Console 介面手動測?



The screenshot shows a Mac OS X terminal window with three tabs labeled "Shell". The active tab displays the following text:

```
[~/ruby/year_month] $ rails c
Loading development environment (Rails 3.0.5)
ree-1.8.7-2011.01 :001 > Util.correct_year_month(2011,3)
```

# 自動化測試

讓程式去測試程式

# I. Instant Feedback

寫好測試後，很快就可以知道程式有沒有寫對

```
it "should return correct year and month" do
    expect(correct_year_month(2011, 3)).to eq [2011, 3]
    expect(correct_year_month(2011, 14)).to eq [2012, 2]
    expect(correct_year_month(2011, 24)).to eq [2012, 12]
end
```

寫測試的時間

小於

debug 除錯的時間

## 2. 回歸測試及重構

- 隨著程式越寫越多，新加的程式或修改，會不會搞爛現有的功能？
- 重構的時候，會不會造成破壞？
- 如果有之前寫好的測試程式，那們就可以幫助我們檢查。

# 3. 幫助你設計 API

- 寫測試的最佳時機點：先寫測試再實作
- TDD (Test-driven development)
  - Red-Green-Refactor 流程
- 從呼叫者的角度去看待程式，關注介面，協助設計出好用的 API。

# 4. 一種程式文件

- 不知道程式的 API 怎麼呼叫使用?
- 查測試程式怎麼寫的，就知道了。

# 小結

- 你的程式不 Trivial -> 寫測試節省開發時間
- 你的程式不是用過即丟 -> 回歸測試和重構
- TDD -> 設計出可測試，更好的 API 介面
- API 怎麼用 -> 測試也是一種文件

怎麼寫測試?

# xUnit Framework

# xUnit Framework

每個程式語言都有這樣的框架

- test case 依序執行各個 test：
  - Setup 建立初始狀態
  - Exercise 執行要測的程式
  - Verify (assertion) 驗證狀態如預期
  - Teardown 結束清理資料

# Ruby 的 Test::Unit

```
class OrderTest < Test::Unit::TestCase  
  
  def setup  
    @order = Order.new  
  end  
  
  def test_order_status_when_initialized  
    assert_equal @order.status, "New"  
  end  
  
  def test_order_amount_when_initialized  
    assert_equal @order.amount, 0  
  end  
  
end
```

Method 的命名  
是個麻煩

# RSpec 寫法

```
describe Order do
  before do
    @order = Order.new
  end

  context "when initialized" do
    it "should have default status is New" do
      expect(@order.status).to eq("New")
    end

    it "should have default amount is 0" do
      expect(@order.amount).to eq(0)
    end
  end
end
```



# RSpec 第一印象

- 語法 Syntax 不同
  - 程式更好讀
  - 更像一種 spec 文件

# RSpec

- RSpec 是一種 Ruby 的測試 DSL  
(Domain-specific language)
  - Semantic Code : 比 Test::Unit 更好讀，寫的人更容易描述測試目的
  - Self-documenting : 可執行的規格文件
- 非常多的 Ruby on Rails 專案採用 RSpec 作為測試框架

# RSpec (cont.)

- 不是全新的測試方法論
- 是一種改良版的 xUnit Framework
- 演進自 TDD，改稱 BDD (Behavior-driven development)
  - 先寫測試，後寫實作

# Learn RSpec

- syntax
- syntax
- syntax
- more syntax

describe 和 context  
幫助你組織分類

# 要測的東西是什麼？

```
→ describe Order do  
  # ...  
end
```

# 或是

```
describe "A Order" do  
  # ...  
end
```



通常是一個類別

# 可以 Nested 加入想要測試的方法是哪個

```
describe Order do  
  → describe "#amount" do  
    # ...  
  end  
end
```

通常開頭用 # 表示 instance method  
dot 開頭表示 class method

# 可以再 nested 加入不同情境

```
describe Order do
    describe "#amount" do
        → context "when user is vip" do
            # ...
            end
        → context "when user is not vip" do
            # ...
            end
        end
    end
end
```

每個 it 就是一小段測試

Assertions 又叫作 Expectations

# 加入 it

```
describe Order do

    describe "#amount" do
        context "when user is vip" do
            → it "should discount five percent if total > 1000" do
                #
            end

            → it "should discount ten percent if total > 10000" do
                #
            end
        end

        context "when user is not vip" do
            → it "should discount three percent if total > 10000" do
                #
            end
        end
    end

end
```

`expect(...).to` 或 `to_not`

所有物件都有這個方法來定義你的期望

```
describe Order do

  describe "#amount" do
    context "when user is vip" do

      it "should discount five percent if total >= 1000" do
        user = User.new( :is_vip => true )
        order = Order.new( :user => user, :total => 2000 )
        → expect(order.amount).to eq(1900)
      end

      it "should discount ten percent if total >= 10000" { ... }
    end
    context "when user is vip" { ... }
  end

end
```

# 輸出結果(red)

```
1. Shell
New Tab Info : Customize Close : Execute
[~/ruby] $ rspec order_spec.rb -fs

Order
#amount
  when user is vip
    should discount five percent if total >= 1000 (FAILED - 1)
    should discount ten percent if total >= 10000 (FAILED - 2)
  when user is not vip
    should discount three percent if total > 10000 (FAILED - 3)

Failures:

1) Order#amount when user is vip should discount five percent if total >= 1000
Failure/Error: order = Order.new( :total => 2000 )
ArgumentError:
  wrong number of arguments (1 for 0)
# ./order_spec.rb:15:in `initialize'
# ./order_spec.rb:15:in `new'
# ./order_spec.rb:15

2) Order#amount when user is vip should discount ten percent if total >= 10000
Failure/Error: order = Order.new( :total => 10000 )
ArgumentError:
  wrong number of arguments (1 for 0)
# ./order_spec.rb:20:in `initialize'
# ./order_spec.rb:20:in `new'
# ./order_spec.rb:20
```

(狀態顯示為在寫 Order 主程式)

# 輸出結果(green)



```
[~/ruby] rspec order_spec.rb -fs

Order
  #amount
    when user is vip
      should discount five percent if total >= 1000
      should discount ten percent if total >= 10000
    when user is not vip
      should discount three percent if total > 10000

Finished in 0.00102 seconds
3 examples, 0 failures
[~/ruby] $
```

漂亮的程式文件

# before 和 after

- 如同 xUnit 框架的 setup 和 teardown
- before(:each) 每段 it 之前執行
- before(:all) 整段 describe 執行一次
- after(:each)
- afte(:all)
- 搭配 describe 和 context 放在適當的位置

```
describe Order do

  describe "#amount" do
    context "when user is vip" do

      → before(:each) do
        @user = User.new( :is_vip => true )
        @order = Order.new( :user => @user )
      end

      it "should discount five percent if total >= 1000" do
        @order.total = 2000
        @order.amount.should == 1900
      end

      it "should discount ten percent if total >= 10000" do
        @order.total = 10000
        @order.amount.should == 9000
      end

    end
    context "when user is vip" { ... }
  end

end
```

# skip

略過不跑，可以先列出來打算要寫的測試

```
RSpec.describe "an example" do  
  it "is skipped" # 沒有block  
  skip "is skipped" do  
    end  
  
 xit "is skipped using xit" do  
    end  
end
```

New Tab Info : Customize Close : Execute

1. Shell

[~/ruby] \$ rspec order\_spec.rb -fs

```
Order
#amount
  when user is vip
    should discount five percent if total >= 1000
    should discount ten percent if total >= 10000
  when user is not vip
    should discount three percent if total > 10000
#paid?
  should be false if status is new (PENDING: Not Yet Implemented)
  should be true if status is paid or shipping (PENDING: No reason given)

Pending:
Order#paid? should be false if status is new
# Not Yet Implemented
# ./order_spec.rb:39
Order#paid? should be true if status is paid or shipping
# No reason given
# ./order_spec.rb:41

Finished in 0.00137 seconds
5 examples, 0 failures, 2 pending
[~/ruby] $
```

# pending

把失敗的測試暫時改成 pending

```
RSpec.describe "an example" do
  pending("something else getting finished") do
    expect(1).to eq(2)
  end
end
```

New Tab Info : Customize Close : Execute

1. Shell

[~/ruby] \$ rspec order\_spec.rb -fs

```
Order
#amount
  when user is vip
    should discount five percent if total >= 1000
    should discount ten percent if total >= 10000
  when user is not vip
    should discount three percent if total > 10000
#paid?
  should be false if status is new (PENDING: Not Yet Implemented)
  should be true if status is paid or shipping (PENDING: No reason given)

Pending:
Order#paid? should be false if status is new
# Not Yet Implemented
# ./order_spec.rb:39
Order#paid? should be true if status is paid or shipping
# No reason given
# ./order_spec.rb:41

Finished in 0.00137 seconds
5 examples, 0 failures, 2 pending
[~/ruby] $
```

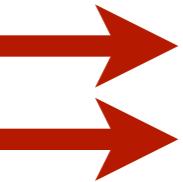
```
let(:name) { exp }
```

- 有使用到才會運算(lazy)，並且在同一個 example 測試中多次呼叫會 Memoized 快取起來。
- 相較於 before(:each) 可增加執行速度
- 因此就不用 instance variable 放 before 裡了，增加程式可讀性

```
describe Order do
  describe "#amount" do
    context "when user is vip" do
      let(:user) { User.new( :is_vip => true ) }
      let(:order) { Order.new( :user => user ) }

      it "should discount five percent if total >= 1000" do
        order.total = 2000
        order.amount.should == 1900
      end

      it "should discount ten percent if total >= 10000" do
        order.total = 10000
        order.amount.should == 9000
      end
    end
    context "when user is vip" { ... }
  end
end
```



# 一些別名方法

哪個念起來順就用哪個

```
describe Order do # describe 和 context 其實是 alias  
  # it, specify, example 其實都一樣  
  it { ... }  
  specify { ... }  
  example { ... }  
  
end
```

# 一些慣例

- 一個 rb 檔案配一個同名的 \_spec.rb 檔案
  - 容易找
  - guard 等工具容易設定
  - editor 有支援快速鍵
- describe “#title” 是 instance method
- describe “.title” 是 class method

# 輸出格式

- rspec filename.rb 預設不產生文件
- rspec filename.rb -fd 輸出 specdoc 文件
- rspec filename.rb -fh 輸出 html 文件

# HTML format

鼓勵寫出高 spec coverage 程式，因為可以當做一種文件

The screenshot shows a web browser window displaying an RSpec test report at `file:///Users/ihower/ruby/report.html`. The title bar says "RSpec results". The top navigation bar includes links for iGoogle, Read Today, Read Later, Instapaper, Reader, Blog, Twitter, Facebook, Plurk, GitHub, Tumblr, Yahoo!, WebTV, Redmine(O), Redmine(T), 翻譯(G), and 字典(Y). The main content area has a red header with "RSpec Code Examples" and "11 examples, 1 failure, 1 pending" followed by "Finished in 0.023062 seconds". It lists various test cases under categories like Order, status, #amount, when user is vip, when user is not vip, #paid?, #receiver\_name, #ship!, and with paid. The "with paid" section contains a detailed error message from the test runner.

```
11 examples, 1 failure, 1 pending
Finished in 0.023062 seconds

Passed Failed Pending
Order
  should be valid
status
  should == "New"
#amount
  when user is vip
    should discount five percent if total >= 1000
    should discount ten percent if total >= 10000
  when user is not vip
    should discount three percent if total > 10000
#paid?
  should be false if status is new
  should be true if status is paid or shipping
#receiver_name
  should be user name
#ship!
with paid
  should call gateway API
  (Mock "ezcat").deliver(#[RSpec::Mocks::Mock:0x809e3830 @name="user"])
    expected: 1 time
    received: 0 times
./order_spec.rb:84

82   context "with paid" do
83     it "should call gateway API" do
84       expect {
85         order = Order.new(status: "New")
86         order.discount_code = "VIP10"
87         order.save!
88       }.to change { order.paid? }.from(false).to(true)
89     end
90   end
91
92   context "when user is not vip" do
93     it "should discount three percent if total > 10000" do
94       order = Order.new(status: "Not VIP", amount: 15000)
95       order.discount_code = "VIP10"
96       order.save!
97       expect(order.discount_code).to eq("VIP10")
98       expect(order.discount).to eq(1500)
99     end
100   end
101
102   context "when user is not paid" do
103     it "should be false if status is new" do
104       order = Order.new(status: "New")
105       expect(order.paid?).to be_falsey
106     end
107   end
108
109   context "when user is paid" do
110     it "should be true if status is paid or shipping" do
111       order = Order.new(status: "Paid")
112       expect(order.paid?).to be_truthy
113     end
114   end
115
116   context "when user is not paid" do
117     it "should be false if status is new" do
118       order = Order.new(status: "New")
119       expect(order.paid?).to be_falsey
120     end
121   end
122
123   context "when user is not paid" do
124     it "should be false if status is new" do
125       order = Order.new(status: "New")
126       expect(order.paid?).to be_falsey
127     end
128   end
129
130   context "when user is not paid" do
131     it "should be false if status is new" do
132       order = Order.new(status: "New")
133       expect(order.paid?).to be_falsey
134     end
135   end
136
137   context "when user is not paid" do
138     it "should be false if status is new" do
139       order = Order.new(status: "New")
140       expect(order.paid?).to be_falsey
141     end
142   end
143
144   context "when user is not paid" do
145     it "should be false if status is new" do
146       order = Order.new(status: "New")
147       expect(order.paid?).to be_falsey
148     end
149   end
150
151   context "when user is not paid" do
152     it "should be false if status is new" do
153       order = Order.new(status: "New")
154       expect(order.paid?).to be_falsey
155     end
156   end
157
158   context "when user is not paid" do
159     it "should be false if status is new" do
160       order = Order.new(status: "New")
161       expect(order.paid?).to be_falsey
162     end
163   end
164
165   context "when user is not paid" do
166     it "should be false if status is new" do
167       order = Order.new(status: "New")
168       expect(order.paid?).to be_falsey
169     end
170   end
171
172   context "when user is not paid" do
173     it "should be false if status is new" do
174       order = Order.new(status: "New")
175       expect(order.paid?).to be_falsey
176     end
177   end
178
179   context "when user is not paid" do
180     it "should be false if status is new" do
181       order = Order.new(status: "New")
182       expect(order.paid?).to be_falsey
183     end
184   end
185
186   context "when user is not paid" do
187     it "should be false if status is new" do
188       order = Order.new(status: "New")
189       expect(order.paid?).to be_falsey
190     end
191   end
192
193   context "when user is not paid" do
194     it "should be false if status is new" do
195       order = Order.new(status: "New")
196       expect(order.paid?).to be_falsey
197     end
198   end
199
200   context "when user is not paid" do
201     it "should be false if status is new" do
202       order = Order.new(status: "New")
203       expect(order.paid?).to be_falsey
204     end
205   end
206
207   context "when user is not paid" do
208     it "should be false if status is new" do
209       order = Order.new(status: "New")
210       expect(order.paid?).to be_falsey
211     end
212   end
213
214   context "when user is not paid" do
215     it "should be false if status is new" do
216       order = Order.new(status: "New")
217       expect(order.paid?).to be_falsey
218     end
219   end
220
221   context "when user is not paid" do
222     it "should be false if status is new" do
223       order = Order.new(status: "New")
224       expect(order.paid?).to be_falsey
225     end
226   end
227
228   context "when user is not paid" do
229     it "should be false if status is new" do
230       order = Order.new(status: "New")
231       expect(order.paid?).to be_falsey
232     end
233   end
234
235   context "when user is not paid" do
236     it "should be false if status is new" do
237       order = Order.new(status: "New")
238       expect(order.paid?).to be_falsey
239     end
240   end
241
242   context "when user is not paid" do
243     it "should be false if status is new" do
244       order = Order.new(status: "New")
245       expect(order.paid?).to be_falsey
246     end
247   end
248
249   context "when user is not paid" do
250     it "should be false if status is new" do
251       order = Order.new(status: "New")
252       expect(order.paid?).to be_falsey
253     end
254   end
255
256   context "when user is not paid" do
257     it "should be false if status is new" do
258       order = Order.new(status: "New")
259       expect(order.paid?).to be_falsey
260     end
261   end
262
263   context "when user is not paid" do
264     it "should be false if status is new" do
265       order = Order.new(status: "New")
266       expect(order.paid?).to be_falsey
267     end
268   end
269
270   context "when user is not paid" do
271     it "should be false if status is new" do
272       order = Order.new(status: "New")
273       expect(order.paid?).to be_falsey
274     end
275   end
276
277   context "when user is not paid" do
278     it "should be false if status is new" do
279       order = Order.new(status: "New")
280       expect(order.paid?).to be_falsey
281     end
282   end
283
284   context "when user is not paid" do
285     it "should be false if status is new" do
286       order = Order.new(status: "New")
287       expect(order.paid?).to be_falsey
288     end
289   end
290
291   context "when user is not paid" do
292     it "should be false if status is new" do
293       order = Order.new(status: "New")
294       expect(order.paid?).to be_falsey
295     end
296   end
297
298   context "when user is not paid" do
299     it "should be false if status is new" do
300       order = Order.new(status: "New")
301       expect(order.paid?).to be_falsey
302     end
303   end
304
305   context "when user is not paid" do
306     it "should be false if status is new" do
307       order = Order.new(status: "New")
308       expect(order.paid?).to be_falsey
309     end
310   end
311
312   context "when user is not paid" do
313     it "should be false if status is new" do
314       order = Order.new(status: "New")
315       expect(order.paid?).to be_falsey
316     end
317   end
318
319   context "when user is not paid" do
320     it "should be false if status is new" do
321       order = Order.new(status: "New")
322       expect(order.paid?).to be_falsey
323     end
324   end
325
326   context "when user is not paid" do
327     it "should be false if status is new" do
328       order = Order.new(status: "New")
329       expect(order.paid?).to be_falsey
330     end
331   end
332
333   context "when user is not paid" do
334     it "should be false if status is new" do
335       order = Order.new(status: "New")
336       expect(order.paid?).to be_falsey
337     end
338   end
339
340   context "when user is not paid" do
341     it "should be false if status is new" do
342       order = Order.new(status: "New")
343       expect(order.paid?).to be_falsey
344     end
345   end
346
347   context "when user is not paid" do
348     it "should be false if status is new" do
349       order = Order.new(status: "New")
350       expect(order.paid?).to be_falsey
351     end
352   end
353
354   context "when user is not paid" do
355     it "should be false if status is new" do
356       order = Order.new(status: "New")
357       expect(order.paid?).to be_falsey
358     end
359   end
360
361   context "when user is not paid" do
362     it "should be false if status is new" do
363       order = Order.new(status: "New")
364       expect(order.paid?).to be_falsey
365     end
366   end
367
368   context "when user is not paid" do
369     it "should be false if status is new" do
370       order = Order.new(status: "New")
371       expect(order.paid?).to be_falsey
372     end
373   end
374
375   context "when user is not paid" do
376     it "should be false if status is new" do
377       order = Order.new(status: "New")
378       expect(order.paid?).to be_falsey
379     end
380   end
381
382   context "when user is not paid" do
383     it "should be false if status is new" do
384       order = Order.new(status: "New")
385       expect(order.paid?).to be_falsey
386     end
387   end
388
389   context "when user is not paid" do
390     it "should be false if status is new" do
391       order = Order.new(status: "New")
392       expect(order.paid?).to be_falsey
393     end
394   end
395
396   context "when user is not paid" do
397     it "should be false if status is new" do
398       order = Order.new(status: "New")
399       expect(order.paid?).to be_falsey
400     end
401   end
402
403   context "when user is not paid" do
404     it "should be false if status is new" do
405       order = Order.new(status: "New")
406       expect(order.paid?).to be_falsey
407     end
408   end
409
410   context "when user is not paid" do
411     it "should be false if status is new" do
412       order = Order.new(status: "New")
413       expect(order.paid?).to be_falsey
414     end
415   end
416
417   context "when user is not paid" do
418     it "should be false if status is new" do
419       order = Order.new(status: "New")
420       expect(order.paid?).to be_falsey
421     end
422   end
423
424   context "when user is not paid" do
425     it "should be false if status is new" do
426       order = Order.new(status: "New")
427       expect(order.paid?).to be_falsey
428     end
429   end
430
431   context "when user is not paid" do
432     it "should be false if status is new" do
433       order = Order.new(status: "New")
434       expect(order.paid?).to be_falsey
435     end
436   end
437
438   context "when user is not paid" do
439     it "should be false if status is new" do
440       order = Order.new(status: "New")
441       expect(order.paid?).to be_falsey
442     end
443   end
444
445   context "when user is not paid" do
446     it "should be false if status is new" do
447       order = Order.new(status: "New")
448       expect(order.paid?).to be_falsey
449     end
450   end
451
452   context "when user is not paid" do
453     it "should be false if status is new" do
454       order = Order.new(status: "New")
455       expect(order.paid?).to be_falsey
456     end
457   end
458
459   context "when user is not paid" do
460     it "should be false if status is new" do
461       order = Order.new(status: "New")
462       expect(order.paid?).to be_falsey
463     end
464   end
465
466   context "when user is not paid" do
467     it "should be false if status is new" do
468       order = Order.new(status: "New")
469       expect(order.paid?).to be_falsey
470     end
471   end
472
473   context "when user is not paid" do
474     it "should be false if status is new" do
475       order = Order.new(status: "New")
476       expect(order.paid?).to be_falsey
477     end
478   end
479
480   context "when user is not paid" do
481     it "should be false if status is new" do
482       order = Order.new(status: "New")
483       expect(order.paid?).to be_falsey
484     end
485   end
486
487   context "when user is not paid" do
488     it "should be false if status is new" do
489       order = Order.new(status: "New")
490       expect(order.paid?).to be_falsey
491     end
492   end
493
494   context "when user is not paid" do
495     it "should be false if status is new" do
496       order = Order.new(status: "New")
497       expect(order.paid?).to be_falsey
498     end
499   end
500
501   context "when user is not paid" do
502     it "should be false if status is new" do
503       order = Order.new(status: "New")
504       expect(order.paid?).to be_falsey
505     end
506   end
507
508   context "when user is not paid" do
509     it "should be false if status is new" do
510       order = Order.new(status: "New")
511       expect(order.paid?).to be_falsey
512     end
513   end
514
515   context "when user is not paid" do
516     it "should be false if status is new" do
517       order = Order.new(status: "New")
518       expect(order.paid?).to be_falsey
519     end
520   end
521
522   context "when user is not paid" do
523     it "should be false if status is new" do
524       order = Order.new(status: "New")
525       expect(order.paid?).to be_falsey
526     end
527   end
528
529   context "when user is not paid" do
530     it "should be false if status is new" do
531       order = Order.new(status: "New")
532       expect(order.paid?).to be_falsey
533     end
534   end
535
536   context "when user is not paid" do
537     it "should be false if status is new" do
538       order = Order.new(status: "New")
539       expect(order.paid?).to be_falsey
540     end
541   end
542
543   context "when user is not paid" do
544     it "should be false if status is new" do
545       order = Order.new(status: "New")
546       expect(order.paid?).to be_falsey
547     end
548   end
549
550   context "when user is not paid" do
551     it "should be false if status is new" do
552       order = Order.new(status: "New")
553       expect(order.paid?).to be_falsey
554     end
555   end
556
557   context "when user is not paid" do
558     it "should be false if status is new" do
559       order = Order.new(status: "New")
560       expect(order.paid?).to be_falsey
561     end
562   end
563
564   context "when user is not paid" do
565     it "should be false if status is new" do
566       order = Order.new(status: "New")
567       expect(order.paid?).to be_falsey
568     end
569   end
570
571   context "when user is not paid" do
572     it "should be false if status is new" do
573       order = Order.new(status: "New")
574       expect(order.paid?).to be_falsey
575     end
576   end
577
578   context "when user is not paid" do
579     it "should be false if status is new" do
580       order = Order.new(status: "New")
581       expect(order.paid?).to be_falsey
582     end
583   end
584
585   context "when user is not paid" do
586     it "should be false if status is new" do
587       order = Order.new(status: "New")
588       expect(order.paid?).to be_falsey
589     end
590   end
591
592   context "when user is not paid" do
593     it "should be false if status is new" do
594       order = Order.new(status: "New")
595       expect(order.paid?).to be_falsey
596     end
597   end
598
599   context "when user is not paid" do
600     it "should be false if status is new" do
601       order = Order.new(status: "New")
602       expect(order.paid?).to be_falsey
603     end
604   end
605
606   context "when user is not paid" do
607     it "should be false if status is new" do
608       order = Order.new(status: "New")
609       expect(order.paid?).to be_falsey
610     end
611   end
612
613   context "when user is not paid" do
614     it "should be false if status is new" do
615       order = Order.new(status: "New")
616       expect(order.paid?).to be_falsey
617     end
618   end
619
620   context "when user is not paid" do
621     it "should be false if status is new" do
622       order = Order.new(status: "New")
623       expect(order.paid?).to be_falsey
624     end
625   end
626
627   context "when user is not paid" do
628     it "should be false if status is new" do
629       order = Order.new(status: "New")
630       expect(order.paid?).to be_falsey
631     end
632   end
633
634   context "when user is not paid" do
635     it "should be false if status is new" do
636       order = Order.new(status: "New")
637       expect(order.paid?).to be_falsey
638     end
639   end
640
641   context "when user is not paid" do
642     it "should be false if status is new" do
643       order = Order.new(status: "New")
644       expect(order.paid?).to be_falsey
645     end
646   end
647
648   context "when user is not paid" do
649     it "should be false if status is new" do
650       order = Order.new(status: "New")
651       expect(order.paid?).to be_falsey
652     end
653   end
654
655   context "when user is not paid" do
656     it "should be false if status is new" do
657       order = Order.new(status: "New")
658       expect(order.paid?).to be_falsey
659     end
660   end
661
662   context "when user is not paid" do
663     it "should be false if status is new" do
664       order = Order.new(status: "New")
665       expect(order.paid?).to be_falsey
666     end
667   end
668
669   context "when user is not paid" do
670     it "should be false if status is new" do
671       order = Order.new(status: "New")
672       expect(order.paid?).to be_falsey
673     end
674   end
675
676   context "when user is not paid" do
677     it "should be false if status is new" do
678       order = Order.new(status: "New")
679       expect(order.paid?).to be_falsey
680     end
681   end
682
683   context "when user is not paid" do
684     it "should be false if status is new" do
685       order = Order.new(status: "New")
686       expect(order.paid?).to be_falsey
687     end
688   end
689
690   context "when user is not paid" do
691     it "should be false if status is new" do
692       order = Order.new(status: "New")
693       expect(order.paid?).to be_falsey
694     end
695   end
696
697   context "when user is not paid" do
698     it "should be false if status is new" do
699       order = Order.new(status: "New")
700       expect(order.paid?).to be_falsey
701     end
702   end
703
704   context "when user is not paid" do
705     it "should be false if status is new" do
706       order = Order.new(status: "New")
707       expect(order.paid?).to be_falsey
708     end
709   end
710
711   context "when user is not paid" do
712     it "should be false if status is new" do
713       order = Order.new(status: "New")
714       expect(order.paid?).to be_falsey
715     end
716   end
717
718   context "when user is not paid" do
719     it "should be false if status is new" do
720       order = Order.new(status: "New")
721       expect(order.paid?).to be_falsey
722     end
723   end
724
725   context "when user is not paid" do
726     it "should be false if status is new" do
727       order = Order.new(status: "New")
728       expect(order.paid?).to be_falsey
729     end
730   end
731
732   context "when user is not paid" do
733     it "should be false if status is new" do
734       order = Order.new(status: "New")
735       expect(order.paid?).to be_falsey
736     end
737   end
738
739   context "when user is not paid" do
740     it "should be false if status is new" do
741       order = Order.new(status: "New")
742       expect(order.paid?).to be_falsey
743     end
744   end
745
746   context "when user is not paid" do
747     it "should be false if status is new" do
748       order = Order.new(status: "New")
749       expect(order.paid?).to be_falsey
750     end
751   end
752
753   context "when user is not paid" do
754     it "should be false if status is new" do
755       order = Order.new(status: "New")
756       expect(order.paid?).to be_falsey
757     end
758   end
759
760   context "when user is not paid" do
761     it "should be false if status is new" do
762       order = Order.new(status: "New")
763       expect(order.paid?).to be_falsey
764     end
765   end
766
767   context "when user is not paid" do
768     it "should be false if status is new" do
769       order = Order.new(status: "New")
770       expect(order.paid?).to be_falsey
771     end
772   end
773
774   context "when user is not paid" do
775     it "should be false if status is new" do
776       order = Order.new(status: "New")
777       expect(order.paid?).to be_falsey
778     end
779   end
780
781   context "when user is not paid" do
782     it "should be false if status is new" do
783       order = Order.new(status: "New")
784       expect(order.paid?).to be_falsey
785     end
786   end
787
788   context "when user is not paid" do
789     it "should be false if status is new" do
790       order = Order.new(status: "New")
791       expect(order.paid?).to be_falsey
792     end
793   end
794
795   context "when user is not paid" do
796     it "should be false if status is new" do
797       order = Order.new(status: "New")
798       expect(order.paid?).to be_falsey
799     end
800   end
801
802   context "when user is not paid" do
803     it "should be false if status is new" do
804       order = Order.new(status: "New")
805       expect(order.paid?).to be_falsey
806     end
807   end
808
809   context "when user is not paid" do
810     it "should be false if status is new" do
811       order = Order.new(status: "New")
812       expect(order.paid?).to be_falsey
813     end
814   end
815
816   context "when user is not paid" do
817     it "should be false if status is new" do
818       order = Order.new(status: "New")
819       expect(order.paid?).to be_falsey
820     end
821   end
822
823   context "when user is not paid" do
824     it "should be false if status is new" do
825       order = Order.new(status: "New")
826       expect(order.paid?).to be_falsey
827     end
828   end
829
830   context "when user is not paid" do
831     it "should be false if status is new" do
832       order = Order.new(status: "New")
833       expect(order.paid?).to be_falsey
834     end
835   end
836
837   context "when user is not paid" do
838     it "should be false if status is new" do
839       order = Order.new(status: "New")
840       expect(order.paid?).to be_falsey
841     end
842   end
843
844   context "when user is not paid" do
845     it "should be false if status is new" do
846       order = Order.new(status: "New")
847       expect(order.paid?).to be_falsey
848     end
849   end
850
851   context "when user is not paid" do
852
```

# Cheat Sheet

- <http://www.rubypigeon.com/posts/rspec-core-cheat-sheet/>
- <http://www.rubypigeon.com/posts/rspec-expectations-cheat-sheet/>

# Code Kata

- 一種練習方法，透過小型程式題目進行鍛鍊，就像學功夫套拳，重複練功一樣
- 除了自己練，也可以 pair-programming
- TDD 需要練習才能抓到訣竅，與其說它是一種測試方式，不如說它更像一種設計手法
- 抓到 TDD 測試精神和訣竅，比學再多漂亮的語法糖更重要

<https://blog.alphacamp.co/2015/03/02/tdd-kata/>

# 準備 Kata 環境

<https://github.com/ihower/ruby-kata>

- Gemfile 設定 rspec, guard
- 安裝 growl notification (optional)
- bundle

# Continuous Testing

- 使用 guard-rspec
- 程式一修改完存檔，自動跑對應的測試
  - 節省時間，立即回饋

TDD 誤竅一：

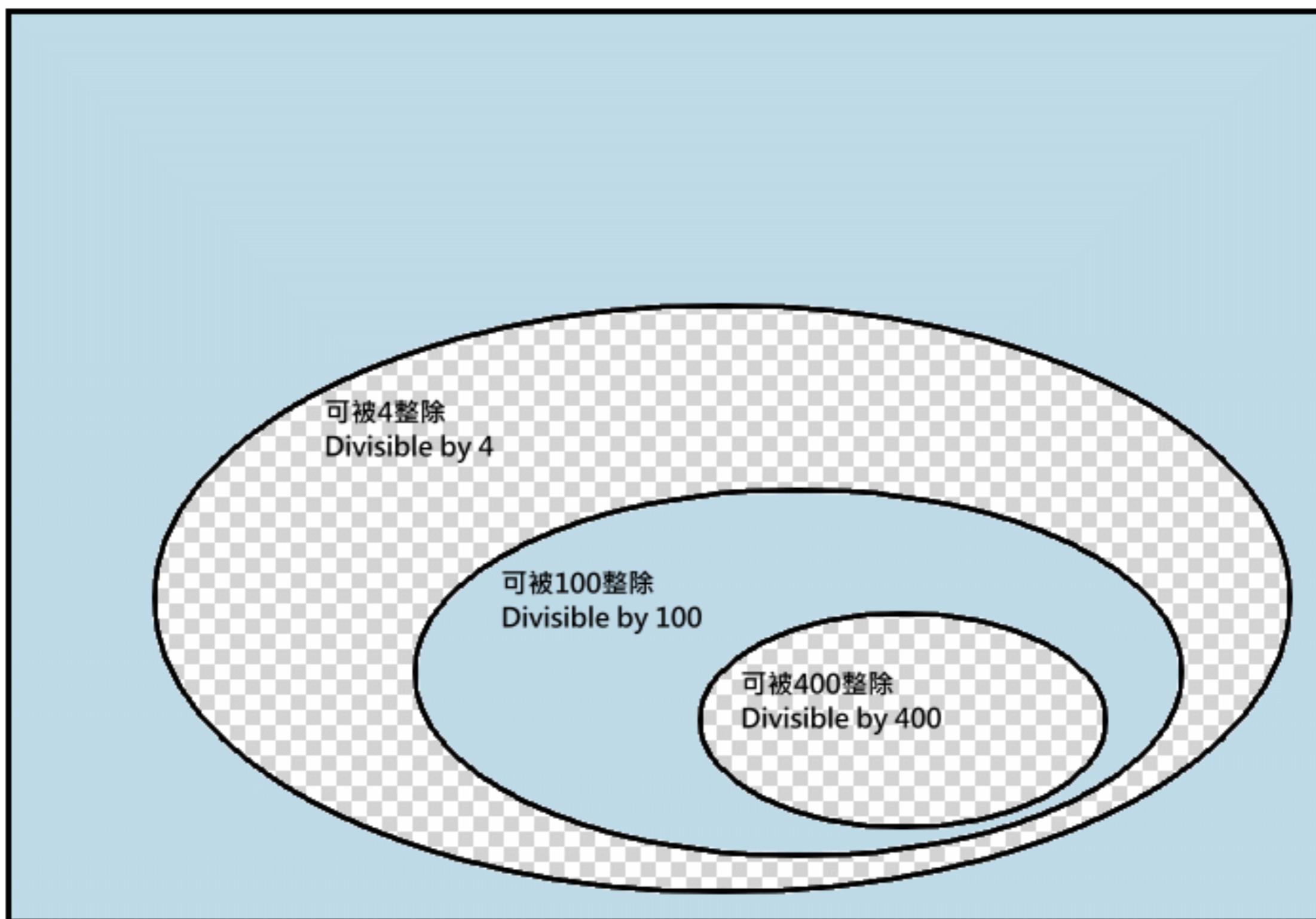
# Red-Green-Refactor development cycle

# FizzBuzz

- 逢三整除，輸出 Fizz
- 逢五整除，輸出 Buzz
- 逢三五整除，輸出 FizzBuzz

# Leap Years

- 判斷閏年
  - 西元年份除以400可整除，為閏年。
  - 西元年份除以4可整除但除以100不可整除，為閏年。
  - 其他則是平年



平年      阖年  
Common year      Leap year

# Roman Numerals

- 轉換羅馬數字

- 1 -> I
- 4 -> IV
- 5 -> V
- 9 -> IX
- 10 -> X
- 20 -> XX

# What have we learned?

- 一個 it 裡面只有一種測試目的，最好就只有一個 expectation
- 要先從測試 failed 失敗案例開始
  - 確保每個測試都有效益，不會發生砍掉實作卻沒有造成任何測試失敗
- 一開始的實作不一定要先直攻一般 case。  
可以一步一步在 cycle 中進行思考
- 測試程式碼的可讀性比 DRY 更重要

As the tests get more specific, the code gets more generic.

Programmers make specific cases work by writing code that makes the general case work.

# Three Laws of TDD

- 一定是先寫一個不通過的單元測試，才開始實作功能
- 每次只新加一個單元測試，只需要剛剛好不通過即可，不要一次加多個測試情境
- 每次實作功能時，只需要剛剛好通過測試即可，不多也不少

TDD 訣竅二：

# 讓測試引導 API 設計

# Bowling Game

<http://www.sportcalculators.com/bowling-score-calculator>

- 計算保齡球分數
  - X (strike) 累加後兩球分數
  - / (spare) 累加後一球分數

BowlingGame

```
# roll
# roll_many(1,2,3,4)
# or roll_many("1234512345")
# finished?
# score
```

# What have we learned?

- 透過寫測試，定義出哪些是類別公開的  
    介面(API)
- 測試碼方便呼叫的API，就是好API
- 不需要公開的，定義成 private 方法，  
    讓實作有更好的物件導向封裝。
- 不需要針對 private 方法直接寫單元測  
    試，而是透過 public 方法間接測試

# Reference:

- The RSpec Book
- The Rails 3 Way
- Foundation Rails 2
- xUnit Test Patterns
- everyday Rails Testing with RSpec
- <http://pure-rspec-rubynation.herokuapp.com/>
- <http://jamesmead.org/talks/2007-07-09-introduction-to-mock-objects-in-ruby-at-lrug/>
- <http://martinfowler.com/articles/mocksArentStubs.html>
- <http://blog.rubybestpractices.com/posts/gregory/034-issue-5-testing-antipatterns.html>
- <http://blog.carbonfive.com/2011/02/11/better-mocking-in-ruby/>

**Thanks.**