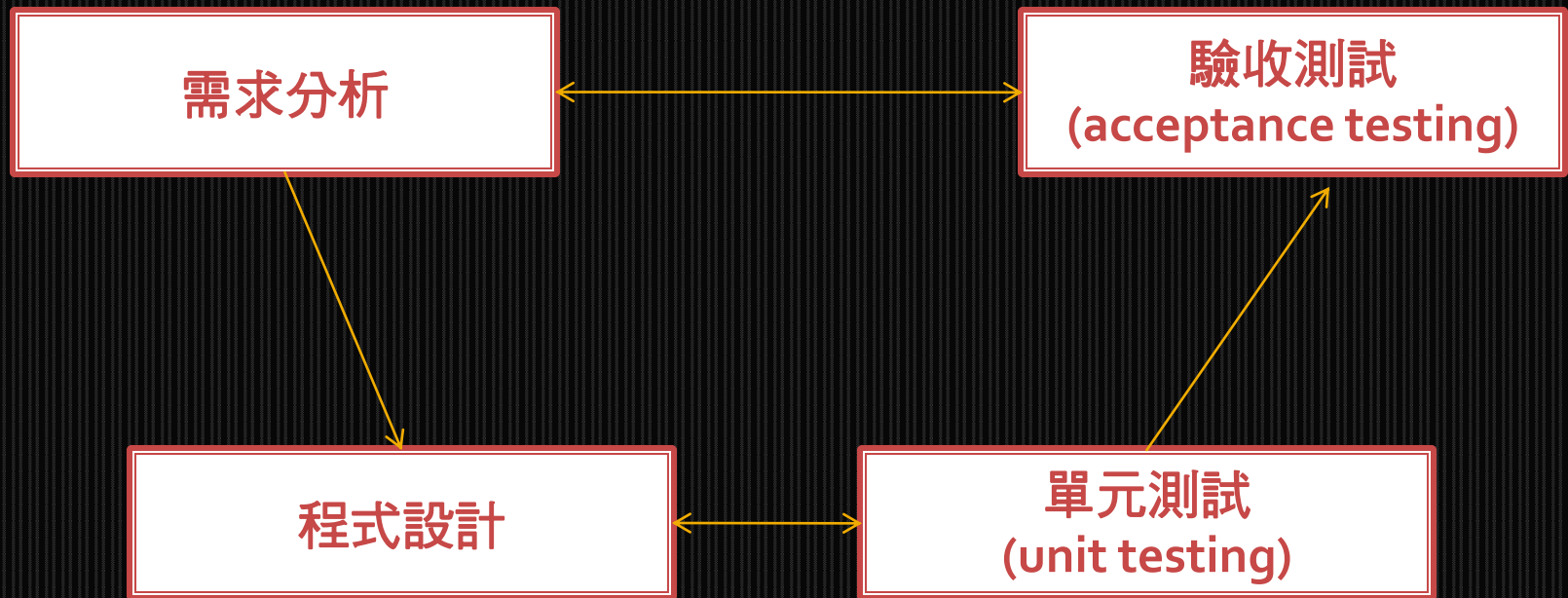


ihowe, handlino

Ruby Tue Day @opcafe 2007/7/17

Rails Testing

測試程序(an agile way)



需求分析

- 使用 User Story，既可以明白需求，又不需要知道太多細節 (太早捕捉細節會白費力氣)
- 最好由 customer team 來撰寫
- 如果 story 太大，請拆小以放進 iteration。
- 搭配 Acceptance tests 一起服用

Example (a job website)

- A user can post her resume to the website.
 - A user can search for jobs.
 - A company can post new job openings.
 - A user can limit who can see her resume
- story size? It's better to have more stories than to have stories that are too large.

Acceptance Testing

- 由QA或客戶撰寫
- 使用某種 scripting 機制和 GUI 軟體。
- 每次 release 前一定要被執行

Test-Driven Development(TDD)

- Unit testing 針對軟體中的最小單位來做驗證。
- 所有產生的code都是為了讓失敗的 unit testing 通過而撰寫。
- 編寫 unit case 和 code之間快速反覆一起演進，而unit case會更早一點。

TDD (cont.)

- 有了 test suite，將非常有利於 refactoring。每次小轉化，都可以藉由執行 unit testing 以確保沒有造成任何破壞。
- 讓我們從呼叫者的角度去看待 code，關注介面，協助設計出便於呼叫 (conveniently callable) 的軟體。
- 可測試的程式碼，協助了設計上的除耦 decouple 化。
- 可當作文件或 demo。

黑白箱

WHITE-TESTING

- Unit testing
- 驗證系統個別機制
- 高度 localized，由寫code的programmer自己寫。
- 目的是提高身為一個programmer的生產效率。

BLACK-TESTING

- Acceptance testing
- 驗證客戶需求有被滿足，軟體能夠正常運作。
- 由不瞭解系統內部架構的人所撰寫: QA或客戶
- 並不關心 programmer 生產力，而是給品管跟客戶開心用。

xUnit Framework

- test case 依序執行各個 test :
 - Setup() 建立初始狀態
 - Exercise
 - Verify (assertion) 驗證狀態如預期
 - Teardown()

Ruby Test:Unit

```
require 'test/unit'
class TC_MyTest <Test::Unit::TestCase
  # def setup
  # end
  # def teardown
  # end

  def test_fail
    assert(false, 'Assertion was false.')
  end
end
end
```

Rails Testing

- Unit testing (for Model) 白
- Functional testing (for Controller) 灰?
- Integration testing (Acceptance testing) 黑
 - Rails 內建模擬，速度快，可讓 developer check-in 前可以執行。
- GUI 的 Acceptance testing 工具
watir or selenium 使用瀏覽器較準，慢，只能讓QA team 跑。

Rails: Unit testing

- 建立測試資料庫
rake db:test:prepare
- 清空資料庫
rake db:test:purge
- 從dev資料庫複製schema
rake db:test:clone_structure

Rails: Unit testing (cont.)

- Step 1: setup data
 - Easy way:
eg. `User = Person.new(:name => 'ihower')`
 - Powerful Way: fixture data
- Step 2: execute something
- Step 3: assertion
 - `assert(boolean,msg)`
 - `assert_equal(expected, actual, msg)`
 - ...

Fixture data

- fixtures :users
- 讀取 /test/fixtures/users.yml

```
ihower:  
  id: 1  
  name: ihower  
  password: <%= User.sha('password') %>  
  email: user1@yhq.com.tw  
h1b:  
  id: 2  
  name: h1b  
  password: <%= User.sha('password') %>
```

Fixture data (cont.)

- 用法:
 - @ihower = users(:ihower)
 - @h1b = users(:h1b)
- 好處:
 - transactional fixtures !! Fast !!
 - reuse in other testing.
共通的測試初始狀態

Example code:

```
require File.dirname(__FILE__) + '/../test_helper'

class GroupTest < Test::Unit::TestCase
  fixtures :user

  def setup
    @creator = user(:ihower)
    @member = user(:h1b)
  end

  def test_group_simple
    g1 = Group.create( :name => 'blah', :user_id => @creator.id )
    assert_equal( @creator.id , g1.user_id )

    g1.join(@member)
    assert_equal( 2 , g1.group_memberships.count )
  end

end
```

Some tips

- 每當接獲bug report，請先撰寫一個 unit testing 來揭發。
- 測試你最擔心出錯的部分，不要等待完美的測試，如果試圖寫太多測試，反而會因為工作量太大而什麼也寫不成。
- 考慮可能出錯的邊界條件。
- 不要因為測試無法捕捉所有bug，就不撰寫測試，因為測試的確可以抓到大多數bug。

Rails: functional testing

- 也是使用 `Ruby Test::Unit` 測試 Controller
- 可以測什麼?
 - response 是否 success, redirect, error?
 - 檢測變數, eg:
 - assert assigns("items")
 - 或 session, cookies, flash 等
 - 檢測 Template 是否正確
 - 檢測 Content

Step1: setup data

- Use fixtures data.
- 可在 `/test/test_helper.rb` 建立常用函式
eg. `login_as(user)`

Step2: send HTTP request

- `get(action,parameters=nil,session=nil,flash = nil)`
- `get :index, :id=> 2`
- `get :show, {'id' => "12"}, {'user_id' => 5}`
- `post :edit, :user => { :user => 'dave', :age='24' }`
- `xhr(:get, :add_to_cart, :id => 11)`
- 要使用 Cookies?
`@request.cookies["user"] = CGI::Cookie.new('name' => 'user' , 'value' =>123456789)`

Step 3: assertion

- `assert_response : success`
- `assert_redirected_to :action => "index"`
- `assert_template "index"`
- `assert_select` 測試 View 有特定內容
 - `assert_select "title", "happy designer"`
 - 支援 Nesting select assertions

Example code:

```
class GroupControllerTest < Test::Unit::TestCase
  ...
  def test_private_group

    get :show, :id => @private_group.id
    assert_redirected_to :action => 'index'

    login_as( @member )

    assert @request.cookies["user"]
    puts @request.cookies["user"].to_yaml

    get :show, :id => @private_group.id
    assert_response :success
    assert_template "show"
  end
end
```

Rails: Integration testing

- Rails 內建的 Acceptance testing
- 可跨 controller 操作，模擬 continuous session
- 支援多個 user session 模擬使用者交錯執行。
- 建立測試檔
`ruby script/generate integration_test user_stories`

Example Code:

```
class UserStoriesTest < ActionController::IntegrationTest
  fixtures :user

  def test_login
    get "/lobby/index"
    assert_response :success

    get "/account/login"
    assert_response :success

    post_via_redirect "/account/login",
                     :user => { :email => 'user1@yhq.com.tw',
                               :plain_password => 'password' }
    assert_response :success
  end
end
```

Acceptance testing

- Usage:
 1. 實際操作瀏覽器錄下動作
 2. 撰寫 scripting 指令
- Selenium (using firefox)
<http://www.openqa.org/selenium/>
- Watir (using ie)
<http://wtr.rubyforge.org/>

Rails Mock objects

- Rails mock is not mock, it's stub!!
- eg.
/test/mocks/ test/some_model.rb
複寫部分 method
- 真正的 Mock object:
<http://mocha.rubyforge.org/>
- 最大的差異是，若 mock object 沒按照預期的 behavior，會有error。
<http://www.martinfowler.com/articles/mocksArentStubs.html>

Example Code:

```
require 'lib/email_gateway'  
  
  def send  
    true  
  end  
  
end
```

rcov

- code coverage 是個量測“測試程式”功效的工具，他會分析有多少比例的 source code 被測試程式測到。
- 安裝 `gem install rcov`
- 執行
`rake test:units:rcov`
`rake test:functional:rcov`

補充: autotest

- 當有新修改，自動執行對應的test!!
- <http://nubyonrails.com/articles/autotest-rails>

Reference Books:

- Agile Web Development with Rails, 2nd
- 重構, Martin Fowler
- 敏捷軟體開發, Robert Cecil Martin
- Test-Driven Development by example, Kent Beck