

Lezione 8: Test automatici

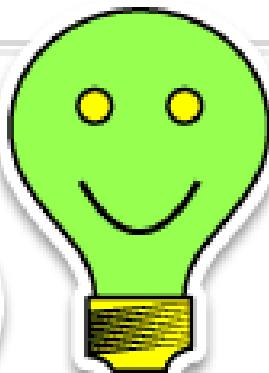
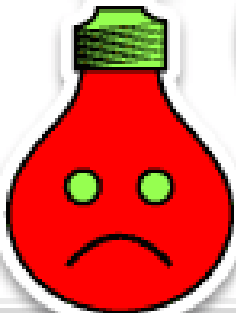
Matteo Vaccari

<http://matteo.vaccari.name/>
matteo.vaccari@uninsubria.it



(cc) Matteo Vaccari. Published in Italy.
Attribution – Non commercial – Share alike 2.5

**Debugging
sucks.**



Testing rocks.

Benvenuti nel XXI secolo

Testare è assolutamente essenziale

Debuggare fa tanto ventesimo secolo

Scrivere test è divertente

Clean code that works.

Is out of reach of even the best programmers, some of the time,
and out of reach of most programmers (like me) most of the time

— Kent Beck

Come li vogliamo i test?

- ▶ Automatici
- ▶ Ripetibili
- ▶ Isolati
- ▶ Self-checking

Tipi di test in Rails

- ▶ "unit" test: modelli
- ▶ "functional" test: controller (e indirettamente i modelli)
- ▶ "integration" test: più di un modello
- ▶ performance test

Unit test

my_test.rb

```
require 'test/unit'
```

```
class MyTest < Test::Unit::TestCase
  def test_sum
    assert_equal 4, 2+2
  end
end
```

```
$ ruby my_test.rb
```

```
Started
```

```
.
```

```
Finished in 0.000366 seconds.
```

```
1 tests, 1 assertions, 0 failures, 0 errors
```

```
$
```

Un database per i test

- ▶ motti_development
- ▶ motti_test
- ▶ motti_production

Copia la struttura del db di produzione nel db di test:

```
$ rake db:test:prepare
```

invoca db:test:prepare ed esegue tutti i test:

```
$ rake test
```

Che cosa testare?

Fare test **positivi** e **negativi**

```
def test_find_by_author
  Quote.delete_all
  quote = Quote.create(:body => 'bla', :author => 'Pippo')

  # Test positivo: lo troviamo Pippo?
  assert_equal quote, Quote.find_by_author('Pippo')

  # Test negativo: che succede se cerchiamo un autore che non c'è?
  assert_equal nil, Quote.find_by_text('foobar')
end
```

Che cosa testare?

Test dei modelli:

- ▶ tutte le validazioni
- ▶ tutte le associazioni
- ▶ tutti i metodi aggiunti al modello

Test dei controller:

- ▶ almeno un test positivo per ogni azione
- ▶ almeno un test negativo per ogni azione

Testare le validazioni

app/models/product.rb

```
class Product
  validates_presence_of :title, :description, :image_url
end
```

```
def test_invalid_with_empty_attributes
  product = Product.new
  assert !product.valid?
  assert product.errors.invalid?(:title)
  assert product.errors.invalid?(:description)
  assert product.errors.invalid?(:image_url)
end
```

Testare le validazioni

```
class Product
  def validate
    if price.nil? || price < 0.01
      errors.add(:price, "should be at least 0.01")
    end
  end
end

def test_positive_price
  product = Product.new(:title => "My Book Title",
    :description => "yyy", :image_url => "zzz.jpg")

  product.price = -1
  assert !product.valid?
  assert_equal "should be at least 0.01", product.errors.on(:price)

  product.price = 0
  assert !product.valid?
  assert_equal "should be at least 0.01", product.errors.on(:price)

  product.price = 1
  assert product.valid?
end
```

Fixtures: sample data

ruby_book:

id: 1

title: Programming Ruby

description: Dummy description

price: 1234

image_url: ruby.png

rails_book:

id: 2

title: Agile Web Development with Rails

description: Dummy description

price: 2345

image_url: rails.png

Assertion methods

```
assert(User.find_by_name("dave"), "user 'dave' is missing")
```

```
assert_equal(3, Product.count)
```

```
assert_not_equal(0, User.count, "no users in database")
```

```
assert_nil(User.find_by_name("willard"))
```

```
assert_not_nil(User.find_by_name("henry"))
```

```
assert_raise(ActiveRecord::RecordNotFound) do
```

```
  Product.find(bad_id)
```

```
end
```

Testing of controllers

```
require 'test_helper'
```

```
class HelloControllerTest < ActionController::TestCase
```

```
  # Replace this with your real tests.
```

```
  test "the truth" do
```

```
    assert true
```

```
  end
```

```
end
```

```
# quotes_controller.rb
def index
  list
  render :action => 'list'
end

# quotes_controller_test.rb
def test_index
  get :index
  assert_response :success
  assert_template 'list'
end
```

```
def create
  @quote = Quote.new(params[:quote])
  if @quote.save
    flash[:notice] = 'Quote was successfully created.'
    redirect_to :action => 'list'
  else
    render :action => 'new'
  end
end

def test_create_ok
  num_quotes = Quote.count

  post :create, :quote => {:body => 'bla bla', :author => 'pippo'}

  assert_response :redirect
  assert_redirected_to :action => 'list'
  assert_equal 'Quote was successfully created.', flash[:notice]

  assert_equal num_quotes + 1, Quote.count
end
```

```
def create
  @quote = Quote.new(params[:quote])
  if @quote.save
    flash[:notice] = 'quote was successfully created.'
    redirect_to :action => 'list'
  else
    render :action => 'new'
  end
end
```

```
def test_failed_create
  num_quotes = Quote.count

  post :create, :quote => {}

  assert_response :success

  assert_invalid assigns(:quote)

  assert_equal num_quotes, Quote.count
end
```

Testare le view

```
assert_select "title", "Pragprog Books Online Store"
assert_select "title", /Online/

assert_select "div#cart"
assert_select "div#cart table tr", :count => 3
assert_select "div#cart table tr.total-line td:last-of-type", "$57.70"

assert_select "div#cart" do
  assert_select "table" do
    assert_select "tr", :count => 3
    assert_select "tr.total-line td:last-of-type", "$57.70"
  end
end
```

Strategie per iniziare a testare

- ▶ test di integrazione
- ▶ test-driven development

Test di integrazione

```
def assert_xml(contents)
  REXML::Document.new(contents)
end
```

```
def test_index
  get '/quotes'
  assert_response :success
  assert_xml @response.body
end
```

Test-Driven Development

