How to test my app

Rspec has a nice syntax. However, it's not rails default. But, listen :

I discover that **minitest** has a **Minitest::Spec** module providing a **rspec-like syntax**.

Here the way to go ! Or, maybe, just keep minitest syntax...

See **Annexe** to learn using tests when devise authentication is on the flow...

Natively

You only need to require the module.

```
# any test file
require 'test_helpers'
require 'minitest/spec'
```

That's it !

Choosing between test syntax or spec syntax is a matter of taste. The former tests the validity of assertions; the later require specifications to have expected results.

So :

```
# instead of writing
class MymodelControllerTest < ActionController::TestCase
  test 'successful response to index' do
    get :index
    assert_response :success
  end
end

# you write
describe MymodelController do
  it 'succeed in getting index' do
    get :index
    _.must_respond_with :success
  end
end
```

I think the main interest is in `describe` line. No need to explicitely define a class subclassing the right superclass.

A gem simplify the use of Minitest::Specs : `minitest-spec-rails` . No need to require 'minitest/spec'...

Expectations

Remarque : take care to call `value(obj).must_xxx` rather than `obj.must_xxx` to avoid issues with threaded tests. Equivalent syntaxes are `expect(obj).must_xxx` and `_(obj).must_xxx` .

`value` can also receive a block : `value { .. }.must_xxx` (as `expect` or `_`).

The `expect` syntax is attractive but I'm in trouble with the plain english it gives : « expect something must be ... ». I'd prefer « expect something to be ... » or « expect something does ». Then I may rewrite (not too long) `Minitest::Expectations` module to fill my needs :

```
module Minitest::Expectations # Monkey patching, put it inside test_helper.rb
  infect_an_assertion :assert_equal, :equals
  infect_an_assertion :assert_not_equal, :do_not_equals
  infect_an_assertion :assert_response, :response_is
  infect_an_assertion :assert_kind_of( klass ), :is_kind_of( klass )
end
```

Then spec lines would look like :

```
expect( 1 + 1 ).equals 2
expect( 1 + 4 ).do_not_equals 2
expect.response_is :success
expect( obj ).is_kind_of( ActiveRecord::Base )
```

Perhaps the best way would be to use `_` syntax : « _ something must be ... ».

Defaults

```
value(obj).must_be(operator, expected) # for example, 10.must_be :<, 11
value(obj).must_be_close_to # the equivalent of assert_in_delta
value(obj).must_be_empty # Fails unless obj.empty?
value(obj).must_be_instance_of(klass) # Fails unless obj.class == klass
value(obj).must_be_kind_of(klass) # Fails unless obj is of class klass or
klass is one of its superclasses.
value(obj).must_be_nil
value(obj).must_be_same_as # tests for true object equality
lambda {}.must_be_silent
value(obj).must_be_within_delta
value(obj).must_be_within_epsilon
value(obj).must_equal(other) # Does a ==/eql? comparison between two objects.
value(obj).must_include(other)
value(obj).must_match(regex) # A regular expression match, e.g.
"hello".must_match /w+/
lambda {}.must_output(stdout, [stderr..]) # The block should have certain
output on stdout or stderr. Set stdout to nil just to check stderr.
lambda {}.must_raise(exception)
value(obj).must_respond_to(message)
value(obj).must_throw(sym)

# The above are all positive valueations but the opposite ones are easy to build
# as in most cases you can switch must with wont. For example:
wont_be, wont_be_empty, wont_be_instance_of, wont_be_kind_of
wont_be_nil, wont_be_same_as, wont_equal,
wont_include, wont_match, wont_respond_to
```

Annexe

Configure tests so devise authentication let you operate tests..

- First, complete the monkey patching of `class ActiveSupport::TestCase :`

```
# test/test_helper.rb
# --------------------------------------------------
# for devise
include Devise::Test::IntegrationHelpers
include Warden::Test::Helpers

def log_in( account )
  if integration_test?
    # use warden helper
    login_as( account, :scope => :account )
  else # controller_test, model_test
    # use devise helper
   sign_in( acccount )
  end
end
# --------------------------------------------------
```

- Second, complete your controller tests (or whatever need an authenticated user)

```
# stays_controller_test.rb
class StaysControllerTest < ActionDispatch::IntegrationTest
  include Devise::Test::IntegrationHelpers

  #--------------------------------------------------
  # for devise user
  setup do
    get '/accounts/sign_in'
    sign_in account( :account_01 )
    post account_session_url

    # if you want to test that things are working correctly, uncomment below
    # follow_redirect!
    # assert_response :success
  end
  #--------------------------------------------------

  test "the truth" do
    assert true
  end
end
```

- Third, create an account fixture

```
# test/fixtures/accounts.yml
account_01:
  id: 1
  email: mazu@sfr.fr
  encrypted_password: <%= Account.new.send( :password_digest, 'tttttttt' ) %>
```