Minitest Introduction

Victor H. Goff III

Table of Contents

Minitest Introduction as presented for Nepal Ruby and Rails Users Group on 22 SEP 20

Presented by: Victor H. Goff III
Email: keeperotphones@gmail.com

Testing in General

- Everyone tests
- Sometimes we "Spike"
  - We can record the spike and use that for example usage documentation
  - We can spike and move the spike information to a more formal format

    IRB as a test platform

    ```
    1 >> 1 + 2
    2 => 3
    3 >> 'bananas'.capitalize
    4 => "Bananas"
    ```

    This works well when generating documentation because we can see when the example usage breaks, if we evaluate our documentation against the version of Ruby that is installed.

- We don’t really require a formal testing framework
  - It is fundamentally something we can do ourselves

Why do we need a testing framework?

Self made assertion

```
def assert(expression, expectation)
  expression == expectation
end

def refute(expression, expectation)
  !assert(expression, expectation)
end
```

1. *What things might we need more than this?*

   We need to know how to capture output, we need to know how to let excpeptions happen while reporting them, and perhaps an option to fail on first failure, or not to fail at all. We would also need to know how to reasonably report pass/fail.

2. *Do we really want to maintain a test suite on top of our application?*

   Probably not. Let the folks that are interested in providing this service provide the service.

3. *Is it important to know how to write your own test framework?*

   I think so. It helps you to appreciate what they do for us, while also making us aware of the decisions that were made. This is similar to "why don't we make our own web framework?" Interesting problem, but we have better things to do than the lower level things, usually.

4. *Are we going to write a test framework in this talk?*

   No. Well, I did not plan on it. At least not more than what is shown in "Self made assertion"

Using our own assertions

Using our self made assertions

```
1   puts assert(1 + 2, 3)
2   puts refute(1 + 2, 2)
```

This can be good enough, indeed, even very good for recording our learnings, as we explore different things, light weight, and we can always add the features that we want.

Why Minitest?

- Minitest is a popular testing framework for Ruby
  - It is PORO in that there is nothing that is really "not Ruby" about it.
  - It has both Spec and Unit style syntax
- It is available with Ruby (bundled)
  - It is also available to be installed as a Gem
    - It should be included in your `Gemfile`, probably under the *development* and *test* sections
  - Install the latest available version
    - `gem install minitest`

Test Unit Style Assertions

Test Style

Table 1. Unit Assertions

assert	assert_empty	assert_equal
assert_in_delta	assert_in_epsilon	assert_includes
assert_instance_of	assert_kind_of	assert_match
assert_nil	assert_operator	assert_output
assert_path_exists	assert_predicate	assert_raises
assert_respond_to	assert_same	assert_send
assert_silent	assert_throws	refute
refute_empty	refute_equal	refute_in_delta
refute_in_epsilon	refute_includes	refute_instance_of
refute_kind_of	refute_match	refute_nil
refute_operator	refute_path_exists	refute_predicate
refute_respond_to	refute_same	

Spec Style Assertions

Spec Style

Table 2. Spec Assertions

must_be	must_be_close_to	must_be_empty
must_be_instance_of	must_be_kind_of	must_be_nil
must_be_same_as	must_be_silent	must_be_within_delta
must_be_within_epsilon	must_equal	must_include
must_match	must_output	must_raise
must_respond_to	must_throw	path_must_exist
path_wont_exist	wont_be	wont_be_close_to
wont_be_empty	wont_be_instance_of	wont_be_kind_of
wont_be_nil	wont_be_same_as	wont_be_within_delta
wont_be_within_epsilon	wont_equal	wont_include
wont_match	wont_respond_to	

As before, this listing is generated directly from my (active) version of Minitest.

How do we run a test?

We can run the tests by using Ruby.

Terminal

```
1 $ ruby test/test_hello.rb   ①
2 Run options: --seed 38327
3
4 # Running:
5
6 .                           ②
7
8 Finished in 0.000679s, 1472.4614 runs/s, 1472.4614 assertions/s.
9
10 1 runs, 1 assertions, 0 failures, 0 errors, 0 skips
```

① The file name is a convention for pattern matching

② The report here is compact, showing that the test passed

The Behavior We Wish We Had

spec_hello.rb

```
require 'minitest/autorun'
require 'hello.rb'

describe 'hello method' do                    ①
  it 'must greet the world' do                ②
    expect(hello).must_equal 'Hello, World!'
  end

  it 'must give a string that is not the same as any other string' do
    expect(hello).wont_be_same_as 'Hello, World!' ③
  end

  it 'must be silent, not output for any reason, including failure' do
    _ { hello }.must_be_silent                ④
  end

  it 'must not return an empty string' do
    _(hello).wont_be :empty?                   ⑤
  end
end
```

① `describe` creates a class for us.

② The idea that we are specifying.

③ Equality is not the same as identity.

Calling assertions on objects directly is deprecated.

④ We can test for output via standard error or standard out.

⑤ Testing that the string is not empty. Test should be self explanatory.

Tests as a Unit Test

hello_test.rb

```ruby
require 'minitest/autorun'
require 'hello.rb'

class TestHello < Minitest::Test
  def test_hello_method
    assert_equal 'Hello, World!', hello
  end

  def test_it_gives_a_new_string
    refute_same 'Hello, World!', hello
  end

  def test_is_silent_not_giving_output_or_error
    assert_silent { hello }
  end

  def test_is_not_empty
    refute_empty hello
  end
end
```

Sample Code

The following code makes the tests pass.

hello.rb

```
def hello
  'Hello, World!'
end
```

Test Report

Test Report

```
Run options: --seed 63063

# Running:

....

Finished in 0.000843s, 4747.0247 runs/s, 5933.7808 assertions/s.

4 runs, 5 assertions, 0 failures, 0 errors, 0 skips
```

Questions about the Syntax?

Questions about the Syntax?

1. *Can the styles be mixed?*

   Yes, the styles can be mixed. Great question!

Questions about the Tests?