Лекция 6
Minitest
CI
Contributing

План

- Minitest
- CI
- Contributing

Подробнее о Minitest

Чего мы хотим от тестового фреймворка?

- Способ писать тесты
- Способ запускать тесты
- Способ получать подробные результаты тестирования

Иерархия тестов

- Шаг теста (test step) – «кнопка нажимается»
- Тест (test case) – «объект создается, кнопка нажимается, объект удаляется»
- Набор тестов (test suite) – «Все на странице работает нормально», «Все методы класса работают нормально»

Два подхода к тестированию

Assert-Style VS Spec-Style

```ruby
require 'test_helper'

class MyprojectTest < Minitest::Test
  def test_that_it_has_a_version_number
    refute_nil ::Myproject::VERSION
  end

  def test_it_does_something_useful
    assert false
  end
end
```

```ruby
require 'test_helper'

describe "My Project" do
  it "has a version number" do
    value(::Myproject::VERSION).wont_be_nil
  end

  it "does something useful" do
    value(4 + 4).must_equal 8
  end
end
```

Assert-Style VS Spec-Style

Assert:

- Чистый Ruby
- Читаемость
- Не очень строго

Spec:

- Читаемость
- DSL
- Более строгая парадигма

Value

```
      _(1 + 1).must_equal 2
 value(1 + 1).must_equal 2
expect(1 + 1).must_equal 2
```

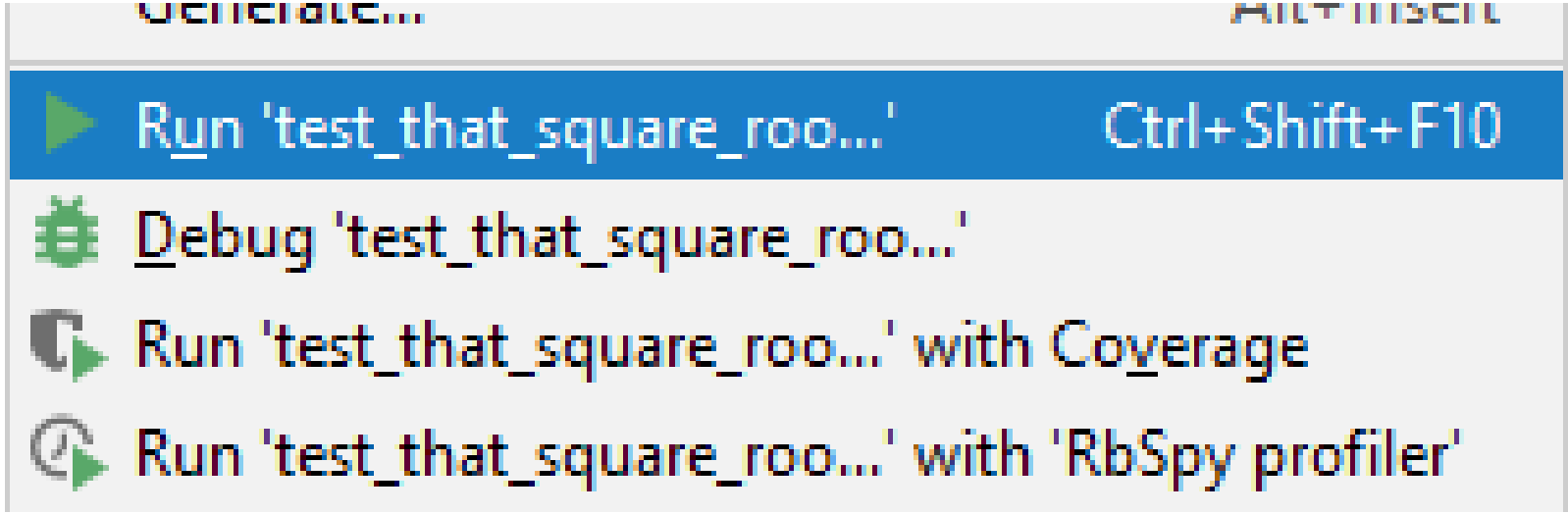
Пишем тест на sqrt

```ruby
module SquareRoot
  class Error < StandardError; end

  def self.square_root(value)

  end
end
```

```ruby
class SquareRootTest < Minitest::Test

  def test_that_square_root_of_9_is_3
    result = square_root(9)
    assert_equal 3, result
  end

end
```

Как запускать?

- `$ ruby test/square_root_test.rb`



Опции запуска

```
-h, --help                      #Display this help.
-s, --seed SEED                 #Sets random seed. Also via env. Eg: SEED=n rake
-v, --verbose                   #Verbose. Show progress processing files.
-n, --name PATTERN              #Filter run on /regexp/ or string.
-e, --exclude PATTERN           #Exclude /regexp/ or string from run.
```

Assertions

```
assert(reverse('abc') == 'cba')
assert(reverse('abc') == 'cba', "reverse('abc') did not return 'cba'")
assert(list.empty?, 'The list is not empty as expected.')
assert_equal('cba', reverse('abc'))
```

Тихие asserts

assert_mock
assert_raises
assert_silent

Floating Point

```
value_a = 2
value_b = 1.9999999999999999
puts value_a == value_b
```

```
assert_in_delta 3.1415, Math::PI, 0.0001
assert_in_delta 3.1415, Math::PI, 0.00001
```

```
Minitest::Assertion:
Expected |3.1415 - 3.141592653589793|
(9.265358979293481e-05) to be <= 1.0e-05.
```

Сравниваем объекты

```
assert_same(arr, arr.sort!)
```

Проверяем nil

```
assert_nil(find_todos_list('Groceries'))
```

Проверяем коллекции

```
list = []
assert_empty(list)

list = %w(abc def xyz)
assert_includes(list, 'xyz')
```

Регулярные выражения

```
assert_match(/not found/, error_message)
```

Подготовка окружения

```
class DatabaseTest < Minitest::Test
  def setup
    @myapp = MyApp.new
  end
 ...
  def teardown
    @myapp.cleanup
  end
end
```

Проверка исключений

```
def test_with_negative_number
  assert_raises(Math::DomainError) { square_root(-3) }
end
```

Проверяем вывод

```ruby
def test_has_no_output
  assert_silent { update_database }
end
```

```ruby
def test_stdout_and_stderr #nil, String, RegExp
  assert_output('', /No records found/) do
    print_all_records
  end
end
```

Другой способ проверить вывод

```
def test_stdout_and_stderr
  out, err = capture_io do
    print_all_records
  end

  assert_equal('', out)
  assert_match(/No records found/, err)
end
```

capture_subprocess_io

Тестируем классы

```
assert_instance_of SomeClass, object

assert_kind_of SomeClass, object

assert_respond_to object, :empty?
```

Refutes

```
a = [...]
assert a.object_id != method(a).object_id, 'method(a) returns original Array'


refute(a.object_id == method(a).object_id,
                    'method(a) returns copy of original Array')


 refute_equal a.object_id, method(a).object_id


 refute_same a, method(a)
```

Нет refute

```
assert_output
assert_raises
assert_send
assert_silent
assert_throws
```

Benchmarks

Асимтотика!

```ruby
class MyprojectBenchmark < Minitest::Benchmark
  def bench_my_linear
    assert_performance_linear do |n|
      n.times do
        n*n
      end
    end
  end

  def bench_my_constant
    assert_performance_constant do |n|
      n*n
    end
  end
end
```

Настраиваем эксперимент

```
def self.bench_range
  [1, 100, 10000]
end
```

Spec-style

```
describe "My Project" do
  if ENV["BENCH"] then
    bench_performance_linear "my_linear" do |n|
      n.times do
        n*n
      end
    end
  end
end
```

Кастомные Asserts

```ruby
class CustomAssertionTest < Minitest::Test
  def assert_uppercase(str, msg = nil)
    msg = message(msg) { "Expected #{mu_pp(str)} to be uppercase" }
    assert(str == str.upcase, msg)
  end

  def test_custom_assertion
    assert_uppercase 'HAHAHA'
  end
end
```

Reporters

```
Minitest::Reporters::DefaultReporter
Minitest::Reporters::SpecReporter
Minitest::Reporters::ProgressReporter
Minitest::Reporters::RubyMateReporter
Minitest::Reporters::RubyMineReporter
Minitest::Reporters::JUnitReporter
Minitest::Reporters::MeanTimeReporter
Minitest::Reporters::HtmlReporter
```

```
require 'minitest/reporters'
Minitest::Reporters.use! [Minitest::Reporters::SpecReporter.new]
```

Continuous Integration

Зачем?

- Хотим пушить часто
- Не хотим часто проводить отладку
- Хотим увеличить прозрачность
- Хотим автоматизировать скучные процессы
- Не хотим ждать

Что нужно сделать?

- Иметь один репозиторий
- Автоматизировать билд
- Билд должен тестироваться автоматически
- Каждый коммит должен собираться
- Билды должны собираться быстро
- Тестировать в специально подготовленном окружении
- Всем должны быть доступны артефакты
- Всем должны быть доступны результаты билда
- Автоматизировать деплой

Как устроен процесс?

- Затянули код к себе
- Написали – запушили
- CI проверяет состояние репозитория
- CI выполняет сборку и тестирует
- CI готовит итоговую сборку (возможно, опять тестирует)
- CI помечает сборку
- CI информирует команду
- Если надо – чиним
- На шаг 1

Применительно к Gem

1. Прогнать тесты, если приехал новый коммит
2. Прогнать performance
3. Отправить готовый гем на rubygems

git tags

```
git tag v1.4-lw

git tag -a v1.4 -m "my version 1.4“

git push origin --tags

git tag -d v1.4-lw
```

Настраиваем Travis CI

```
---
sudo: false
language: ruby
cache: bundler
rvm:
  - 2.6.3
before_install: gem install bundler -v 2.0.2
```

Rakefile

```ruby
require "bundler/gem_tasks"
require "rake/testtask"

Rake::TestTask.new(:test) do |t|
  t.libs << "test"
  t.libs << "lib"
  t.test_files = FileList["test/**/*_test.rb"]
end

task :default => :test
```

Нужны вспомогательные гемы

```
spec.add_development_dependency "bundler", "~> 2.0"
spec.add_development_dependency "rake", "~> 10.0"
spec.add_development_dependency "minitest", "~> 5.0"
spec.add_development_dependency "rubocop"
```

Конфигурируем деплой

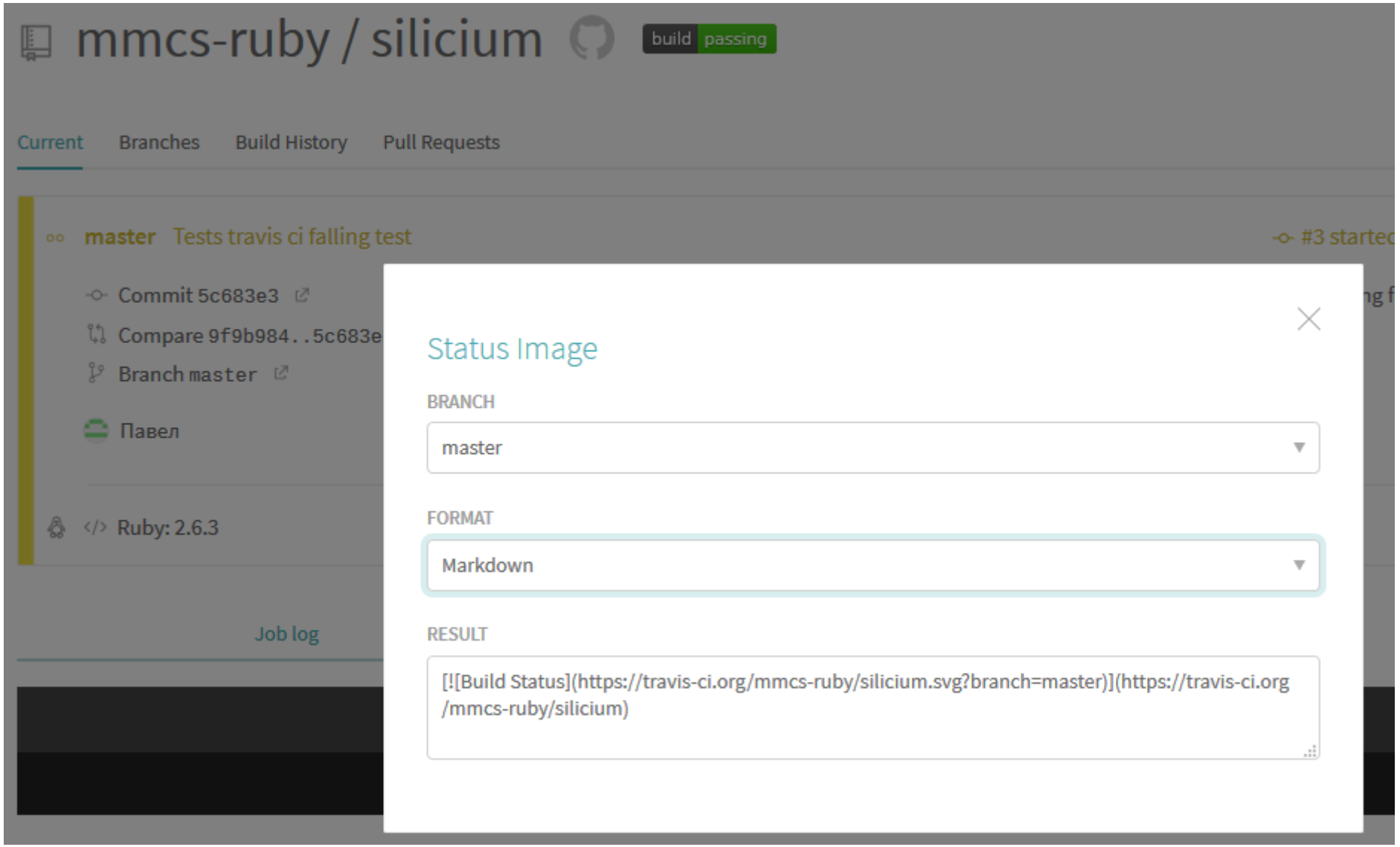
```
---
sudo: false
language: ruby
cache: bundler
rvm:
  - 2.6.3
before_install: gem install bundler -v 2.0.2

before_deploy: bundle exec rake test

deploy:
  provider: rubygems
  api_key: API_KEY
  on:
    tags: true
```

Проверяем

```ruby
def test_it_will_fall
  assert false
end
```

```
230   1) Failure:
231 SiliciumTest#test_it_will_fall [/home/travis/build/mmcs-ruby/silicium/test/silicium_test.rb:9]:
232 Expected false to be truthy.
233
234 2 runs, 2 assertions, 1 failures, 0 errors, 0 skips
235 rake aborted!
236 Command failed with status (1): [ruby -I"lib:test:lib" -I"/home/travis/build/mmcs-ruby/silicium/ve
    /ruby/2.6.0/gems/rake-10.5.0/lib/rake/rake_test_loader.rb" "test/silicium_test.rb" ]
237 /home/travis/build/mmcs-ruby/silicium/vendor/bundle/ruby/2.6.0/bin/ruby_executable_hooks:24:in `ev
238 /home/travis/build/mmcs-ruby/silicium/vendor/bundle/ruby/2.6.0/bin/ruby_executable_hooks:24:in `<m
239 Tasks: TOP => default => test
240 (See full trace by running task with --trace)
241 The command "bundle exec rake" exited with 1.
242 store build cache
245
246
247 Done. Your build exited with 1.
```

Бейджик !


mmcs-ruby / silicium
build passing
Current
Branches
Build History
Pull Requests
master Tests travis ci falling test
#3 started
Commit 5c683e3
Compare 9f9b984..5c683e
Branch master
Павел
Ruby: 2.6.3
Job log
Status Image
BRANCH
master
FORMAT
Markdown
RESULT
[![Build Status](https://travis-ci.org/mmcs-ruby/silicium.svg?branch=master)](https://travis-ci.org/mmcs-ruby/silicium)

ReadMe

Silicium

- Теория вероятностей
- Геометрия
- Аналитическое задание функций
- Графы
- Численное интегрирование
- Методы оптимизации
- Построение графиков
- Операции с матрицами
- Разреженные матрицы

Теория вероятностей

- Комбинаторика
- Поиск мат. ожидания и дисперсии для таблично заданной функции
- Методы для бросков кубиков, выбора карты, других игровых механик

Геометрия

- Расстояние от точки до прямой
- Массивы точек – поиск оболочки, поиск пары ближайших
- Площадь фигур
- Пересечение фигур

Аналитическое задание функций

- Минимум – полиномы
- Тригонометрия, логарифм, экспонента, остальные
- Символьное дифференцирование

Графы

- Описание
- Редактирование
- Поиск пути
- Ориентированные графы
- Связность

Построение графиков

- С использование сторонних гемов
- Диаграммы
- Точки на графике
- Оси, подписи

Методы оптимизации

- Монте-Карло
- Градиентный спуск
- Генетический алгоритм

Операции с матрицами

- C-extensions (BLAS, LAPACK)
- Нативная реализация
- Базовые операции
- Решение СЛАУ

Разреженные матрицы

- Реализация на триплетах
- Более сложные реализации
- Добавление элемента
- Умножение

Как начать?

- Оставить логин в Moodle
- Получить приглашение в организацию
- Сделать форк
- Начать писать тесты и код
- Сделать pull request в основной репозиторий

Как сделать ценный вклад?

- Issue
- MRE
- Итерации
- Squashing commits
- CoC

Что оценивается?

- Реализация
- TDD
- Code Conventions
- Соблюдение сроков

Ссылки

- https://launchschool.com/blog/assert-yourself-an-introduction-to-minitest
- https://www.thoughtworks.com/continuous-integration