

The history of testing framework in Ruby

Kouhei Sutou

ClearCode Inc.

*RubyKaigi 2015
2015-12-12*

Silver sponsor



Silver Sponsors

C.

lished in 2004
cial network,
merged as one
ile tech pioneers,
world's first



ClearCode Inc.

Free software is important in
ClearCode. We develop/support
software with our free
software development
experiences. We feed back our

Goal

目標

You know about
`/\Atest.+unit\z/i`

`/\Atest.+unit\z/i`を知ること

Test

テスト

Which Rubies do include
`/\Atest.+unit\z/i?`

`/\Atest.+unit\z/i`を含んでいるRubyはどれでしょう？

1.4	1.6	1.8
1.9.1	1.9.2	1.9.3
2.0	2.1	2.2
2.3		

Answer

答え

Which Rubies do include
`/\Atest.+unit\z/i?`

`/\Atest.+unit\z/i`を含んでいるRubyはどれでしょう？

1.4	1.6	1.8
1.9.1	1.9.2	1.9.3
2.0	2.1	2.2
2.3		

History

歷史

Characters

登場人物

testsupp	RubyUnit
Lapidary	
Test::Unit	test/unit
test-unit	minitest
RSpec	

Ruby 1.3 times

Ruby 1.3の時代

- ✓ 1998-1999
- ✓ The first testing framework for Ruby was released
最初のRuby用のテストフレームワークがリリース
- ✓ testsupp

testsupp

- ✓ 1999-4-11: [ruby-talk:00634]
ANN: testsupp.rb 0.1
- ✓ Perl's Test like API
PerlのTestモジュールのようなAPI
- ✓ Perl's Test exists since 1998
PerlのTestモジュールは1998からある

testsupp: API

```
require "testsupp"  
include TestSupp  
  
start_tests  
ok "Hello" == "Hello"  
end_tests
```

Ruby 1.4 times

Ruby 1.4の時代

- ✓ 1999-2000
- ✓ The first xUnit testing framework for Ruby was released

最初のRuby用のxUnit系テストフレームワークがリリース

- ✓ RubyUnit

RubyUnit

- ✓ Since 1999-11-20 at least
少なくとも1999年11月20日には存在していた
- ✓ [ruby-list:18594]
"RubyUnit のページは当分更新されないと思うです。 "
- ✓ The first xUnit testing
framework for Ruby
最初のRuby用のxUnit系テストフレームワーク

FYI: XP

参考情報：XP

- ✓ eXtream Programing
- ✓ The first XP book was published at 1999
最初のXPの本は1999年に出版
- ✓ xUnit was born from XP
xUnitはXPが生み出した

RubyUnit: API

```
require "runit/testcase"  
class TestCalc < RUNIT::TestCase  
  def test_add  
    assert_equal(3, 1 + 2)  
  end  
end
```

RubyUnit: Impression

感想

Ruby-ish

Rubyっぽい

RubyUnit: FYI

RubyUnit : 参考情報

- ✓ Test generator exists
テストジェネレーターがある
- ✓ Test exec command exists
テスト実行コマンドがある
- ✓ rubyunit gem exists since 2014-10-12
rubyunitというgemが2014年10月12日から存在する
- ✓ Not related product
関係のないプロダクト

Ruby 1.6 times: 1

Ruby 1.6の時代：1

✓ 2000-2002

✓ The second xUnit testing framework for Ruby was released

2つ目のRuby用のxUnit系テストフレームワークがリリース

✓ Lapidary

Lapidary

- ✓ The first release was
2001-03-20

最初のリリースは2001年3月20日

- ✓ The second xUnit testing
framework for Ruby

2つ目のRuby用のxUnit系テストフレームワーク

- ✓ Maybe
たぶん

Lapidary: API

```
require "Lapidary/TestCase"  
class TC_Adder < Lapidary::TestCase  
  def testAdd  
    assertEquals(3, 1 + 2)  
  end  
end
```

Lapidary: Impression

感想

Not Ruby-ish

Rubyっぽくない

Lapidary: FYI

参考情報

- ✓ Has GUI test runner
GUIのテスト実行機能アリ

Ruby 1.6 times: 2

Ruby 1.6の時代：2

- ✓ Started trying to bundle a testing framework to Ruby

テストフレームワークをRubyにバンドルする
試みを開始

- ✓ RubyUnit + Lapidary = Test::Unit

Test::Unit

- ✓ Since 2002-02:
[ruby-talk:34744]
- ✓ The first `/\Atest.+unit\z/i`
最初の`/\Atest.+unit\z/i`
- ✓ The author is the author of
Lapidary
Lapidaryの作者が作者

Test::Unit: API

```
require "test/unit"  
class TC_Adder < Test::Unit::TestCase  
  def test_add  
    assert_equal(3, 1 + 2)  
  end  
end
```


Test::Unit: Impression

感想

Ruby-ish

Rubyっぽい

Test::Unit: FYI 1

参考情報1

- ✓ API is based on RubyUnit
APIはRubyUnitのよう
- ✓ Has RubyUnit compatible API
RubyUnit互換APIアリ
- ✓ Impl. is based on Lapidary
実装はLapidaryベース
- ✓ Has GUI test runner
GUIのテスト実行機能アリ

Wrap up: Before Ruby 1.8

Ruby 1.8の前までのまとめ

- ✓ testsupp: Perl like API
- ✓ RubyUnit: Ruby-ish xUnit
- ✓ Lapidary: Not Ruby-ish xUnit
- ✓ Test::Unit:
 - ✓ RubyUnit API + Lapidary Impl.

Ruby 1.8 times: 1

Ruby 1.8の時代：1

✓ 2003-2013

✓ Ruby bundled a testing framework

Rubyがテストフレームワークをバンドルした

✓ `Test::Unit` aka `test/unit`

バンドルされたのは`Test::Unit`

`test/unit`とも呼ばれる

FYI: Why test/unit?

参考情報：なぜtest/unitと呼ぶか

```
require "test/unit"
```

Ruby 1.8 times: 2

Ruby 1.8の時代：2

RSpec was born

RSpec誕生

RSpec

- ✓ Since 2005-08
- ✓ Tool for Behavior Driven Development
振る舞い駆動開発用のツール
- ✓ Active development
開発は活発

RSpec 0.1.0: API

```
# 2005-08
require "spec"
class SpecAdd < Spec::Context
  def add
    (1 + 2).should_equal 3
  end
end
```


RSpec: 1.0.0: API

```
# 2007-05  
require "spec"  
describe "Add" do  
  it "should support positive + positive" do  
    (1 + 2).should == 3  
  end  
end
```

RSpec 1.0.0: Impression

感想

Engl-ish

英語っぽい

RSpec: FYI

参考情報

- ✓ Test exec command exists
テスト実行コマンドがある
- ✓ RubyUnit like
RubyUnitみたい
- ✓ More features than Test::Unit
Test::Unitより機能が多い
- ✓ Mock, Not implemented, ...
モックや未実装など

Test::Unit in Ruby 1.8

Ruby 1.8バンドル後のTest::Unit

- ✓ No active development
開発は停滞
- ✓ Maintainer was changed
メンテナー交代
- ✓ The author of minitest
後のminitestの作者

RSpec and Test::Unit

- ✓ RSpec is getting better but Test::Unit isn't changed...

RSpecはよくなっているのにTest::Unitは変わらない。。。。

- ✓ Test::Unit should also be getting better!

Test::Unitももっとよくしたい!

Test::Unit: new maintainer

新しいメンテナー

- ✓ New maintainer said:
新メンテナー曰く
- ✓ Can't maintain Test::Unit because
it's too complex
Test::Unitは複雑すぎてメンテナンスできない

Wrap up: Ruby 1.8 times

Ruby 1.8時代のまとめ

- ✓ Ruby started bundling

`Test::Unit` (`/\Atest.+unit\z/i`)

RubyがTest::Unitのバンドルを始めた

- ✓ But Test::Unit was died...

バンドル後、Test::Unitは死んだ。。。。

- ✓ RSpec was born and active

RSpecが生まれ、活発だった

Ruby 1.9 times

Ruby 1.9の時代

✓ 2009-2014

✓ Ruby dropped `Test::Unit`

Rubyが`Test::Unit`を捨てた

✓ Because `Test::Unit` is too complex
for new maintainer

新メンテナーには`Test::Unit`は複雑すぎるから

Test::Unit → test-unit

- ✓ Released as test-unit gem

test-unit gemとしてリリース

- ✓ Still `/\Atest.+unit\z/i`

名前変更後も`/\Atest.+unit\z/i`

- ✓ `test-unit 1.2.3 == Test::Unit` in Ruby 1.8

- ✓ Maintainer was changed

メンテナー交代

- ✓ Me

私

test-unit

- ✓ Since 2008-03
- ✓ Active development
開発は活発
- ✓ High backward compatibility
高い後方互換性
- ✓ Exception: Inheritance
例外：継承時の挙動

minitest

- ✓ Since 2008-10
- ✓ Active development
開発は活発
- ✓ The author is the new
maintainer of `Test::Unit`
作者は`Test::Unit`の新メンテナー

minitest and Test::Unit

- ✓ Mini, simple, clean and fast
小さくてシンプルでキレイで速い
- ✓ The author said:
作者曰く
 - ✓ 100% Test::Unit compatible
Test::Unitと100%互換
 - ✓ But it's not compat in fact...
実際は互換ではなかった。。

test/unit in Ruby 1.9

- ✓ Provide Test::Unit API

Test::Unit互換APIを提供

- ✓ Wrapper of minitest

minitestのラッパー

- ✓ Ruby developers developed it
not minitest author

minitestの作者ではなくRuby開発者が開発

- ✓ Not Test::Unit but `/\Atest.+unit\z/i`

Test::Unitではないが`/\Atest.+unit\z/i`ではある

Wrap up: Ruby 1.9 times

Ruby 1.9時代のまとめ

✓ Ruby dropped Test::Unit

RubyがTest::Unitを捨てた

✓ Test::Unit → test-unit gem

✓ Ruby bundled minitest

Rubyはminitestをバンドルした

✓ Ruby created test/unit

(test/unit provided Test::Unit API)

RubyはTest::Unit APIを提供するtest/unitを新しく作った

Ruby 2.0 times

Ruby 2.0の時代

✓ 2013-

✓ No highly important things

特筆すべきことはない

✓ test-unit, minitest and RSpec
were actively developed

test-unitもminitestもRSpecも活発に開発が進んでいた

Ruby 2.1 times: 1

Ruby 2.1の時代：1

✓ 2013-

✓ minitest 5 introduced
incompatible changes

minitest 5で非互換の変更が入った

minitest 5

- ✓ `test/unit` maintainer said:
test/unitメンテナー曰く
- ✓ Can't maintain `test/unit` with
`minitest 5`
minitest 5向けのtest/unitはメンテナンスできない
- ✓ `test/unit` for `minitest` was
dropped
minitestのラッパーとしてのtest/unitを捨てた

Is no test/unit problem?

test/unitがないことは問題なのか？

- ✓ Ruby used test/unit for Ruby's tests

RubyはRubyのテストにtest/unitを使っていた

- ✓ No test/unit means existing tests can't be worked

test/unitがなくなると既存のテストが動かない

Solution idea: 1

解決案：1

- ✓ Keep bundling and using minitest 4

minitest 4のバンドルを継続し、使い続ける

- ✓ Users will be confused because the latest Ruby doesn't bundle the latest minitest

最新のRubyに最新のminitestがバンドルされていないとユーザーは混乱しそう

Solution idea: 2

解決案：2

- ✓ Bundle the latest minitest

最新のminitestをバンドルする

- ✓ Ruby uses minitest 4 and test/unit for its tests

Rubyのテストにはminitest 4とtest/unitを使う

Accepted!

Note for idea 2

案2の補足

- ✓ Need to care existing

Test::Unit API users

既存のTest::Unit APIユーザーをケアする必要がある

- ✓ Backward compat is important!

後方互換性は重要！

- ✓ Bundle test-unit gem too for

Test::Unit API

Test::Unit互換APIのためにtest-unit gemもバンドル

Wrap up: Ruby 2.1 times

Ruby 2.1時代のまとめ

- ✓ Ruby developers decided to maintain minitest 4 and test/unit for Ruby's tests

Ruby開発者はRubyのテスト用にminitest 4とtest/unitをメンテナンスすることを決めた

Ruby 2.2 times: 1

Ruby 2.2の時代：1

✓ 2014-

✓ Upgraded bundled minitest to 5

バンドルしているminitestを5に更新

✓ Moved minitest 4 and test/unit to test/

minitest 4とtest/unitはtest/以下に移動

Upgraded minitest to 5

minitest 5に更新

```
- lib/minitest/*.rb
+ test/lib/minitest/*.rb
- lib/test/unit.rb
+ test/lib/test/unit.rb
- lib/test/unit/**/*.rb
+ test/lib/test/unit/**/*.rb
```


Ruby 2.2 times: 2

Ruby 2.2の時代：2

- ✓ Bundled test-unit gem again for users

再びtest-unit gemをバンドル

- ✓ Ruby 1.9 dropped test-unit gem

Ruby 1.9はtest-unit gemを捨てた

Bundled test-unit again

test-unitを再びバンドル

test-unit gem provides:

```
lib/test-unit.rb  
lib/test/unit.rb  
lib/test/unit/**/*.rb
```

test/unit in Ruby 2.2

Ruby 2.2のtest/unit

- ✓ For Ruby developers

Ruby開発者にとっては

- ✓ `test/lib/test/unit`

- ✓ For users

ユーザーにとっては

- ✓ `lib/test/unit = test-unit gem`

FYI: test/lib/test/unit

- ✓ Includes useful features for Ruby's tests

Ruby本体のテストに便利な機能が入っている

- ✓ Leak checker for file descriptor, thread, process...

リークチェッカー：ファイルディスクリプターやスレッドのリークやゾンビプロセスを検出

- ✓ Memory usage profiler

メモリー使用量のプロファイラー

Wrap up: Ruby 2.2 times

Ruby 2.2時代のまとめ

- ✓ Bundled minitest 5
minitest 5をバンドル
- ✓ Bundled test-unit again
再びtest-unitをバンドル
- ✓ Ruby's tests used forked
mintiest 4 and test/unit
Rubyのテストはフォークしたminitest 4とtest/unitを使う

Ruby 2.3 times

Ruby 2.3の時代

✓ 2015-

✓ No highly important things

特筆すべきことはない

✓ test-unit, minitest, test/lib/
test/unit and RSpec were actively
developed

test-unitもminitestもtest/lib/test/unitもRSpecも活
発に開発が進んでいた

Wrap up: History: 1

歴史のまとめ：1

- ✓ Ruby 1.4 and 1.6 didn't bundle any testing framework

Ruby 1.4と1.6はテストフレームワークをバンドルしていなかった

- ✓ Test::Unit was born in Ruby 1.6 times

Ruby 1.6の時代にTest::Unitが生まれた

- ✓ The first `/\Atest.+unit\z/i`

Wrap up: History: 2

歴史のまとめ: 2

✓ Ruby 1.8 bundled Test::Unit

Ruby 1.8はTest::Unitをバンドルした

✓ `/\Atest.+unit\z/i` in 1.8

✓ Ruby 1.9 dropped Test::Unit and bundled minitest

Ruby 1.9はTest::Unitを捨ててminitestをバンドルした

Wrap up: History: 3

歴史のまとめ: 3

- ✓ Ruby 1.9 provided test/unit as a wrapper of minitest

Ruby 1.9はminitestのラッパーとしてtest/unitを提供した

- ✓ test/unit provided Test::Unit compatible API

test/unitはTest::Unitの互換APIを提供した

- ✓ `/\Atest.+unit\z/i` in 1.9

Wrap up: History: 4

歴史のまとめ：4

✓ Ruby 2.0 is same as Ruby 1.9

Ruby 2.0の状況はRuby 1.9と変わらない

✓ `test/unit` is `/\Atest.+unit\z/i` in 2.0

つまり、2.0での`/\Atest.+unit\z/i`は`test/unit`

Wrap up: History: 5

歴史のまとめ：5

- ✓ Ruby 2.1 didn't upgrade minitest to 5

Ruby 2.1はminitestを5に更新しなかった

- ✓ minitest 5 has backward incompatible changes

minitest 5には後方非互換の変更があった

- ✓ They break Ruby's tests

非互換を受け入れるとRubyのテストが動かなくなる

Wrap up: History: 6

歴史のまとめ: 6

- ✓ Ruby 2.2 upgraded minitest

Ruby 2.2はminitestを5に更新した

- ✓ Dropped test/unit as minitest wrapper

Ruby 2.2はminitestのラッパーのtest/unitを捨てた

- ✓ Used forked minitest 4 and test/unit for its tests

Ruby 2.2はRubyのテスト用にminitest 4のフォークとtest/unitを使うことにした

Wrap up: History: 7

歴史のまとめ: 7

- ✓ Ruby 2.2 bundled test-unit gem again

Ruby 2.2はtest-unit gemを再びバンドルした

- ✓ `/\Atest.+unit\z/i` in 2.2

Wrap up: History: 8

歴史のまとめ：8

- ✓ `test/lib/test/unit` was evolved for Ruby's tests
`test/lib/test/unit`はRubyのテスト用に進化した
- ✓ `/\Atest.+unit\z/i` in 2.2 too

Wrap up: History: 9

歴史のまとめ：9

- ✓ No highly important things in Ruby 2.3

Ruby 2.3には特筆すべきことはない

What's `/\Atest.+unit\z/i`

`/\Atest.+unit\z/i`はなにか

Test::Unit compatible API

Test::Unit互換API

Test::Unit compat API

Test::Unit互换API

```
class XXXTest < Test::Unit::TestCase
  def test_xxx
    assert_equal(xxx, yyy)
  end
end
```

Test::Unit API spirit

Test::Unit APIの魂

Ruby-ish

Rubyらしさ

Ruby-ish in Test::Unit

Test::UnitでのRubyらしさ

- ✓ Write test as normal Ruby script

普通のRubyスクリプトのようにテストを書ける

- ✓ It's important in minitest too
minitestもこれを大事にしている

By example: 1
実例：1

Grouping tests

テストのグループ化



class

Group: 1

グループ化 : 1

```
class TestAdd < Test::Unit::TestCase
  def test_positive_positive
    assert_equal(3, 1 + 2)
  end

  def test_positive_negative
    assert_equal(-2, 1 + -3)
  end
end
```

Group: 2

グループ化 : 2

```
class TestSub < Test::Unit::TestCase
  def test_positive_positive
    assert_equal(-2, 1 - 3)
  end

  def test_positive_negative
    assert_equal(4, 1 - -3)
  end
end
```

Group: 3

グループ化 : 3

```
class TestCalc < Test::Unit::TestCase
  class TestAdd < self # TestAdd < TestCalc
    # def test_positive_positive
    # def test_positive_negative
  end
  class TestSub < self # TestAdd < TestCalc
    # def test_positive_positive
    # def test_positive_negative
  end
end
```

Wrap up: Group: 1

グループ化のまとめ: 1

- ✓ Test case has tests
テストケースは複数のテストを持つ
- ✓ Test case is class
テストケースをクラスとして実装
- ✓ Test is method
テストをメソッドとして実装
- ✓ Class has methods
クラスは複数のメソッドを持つ

Wrap up: Group: 2

グループ化のまとめ: 2

- ✓ Sub test case is sub class
サブテストケースはサブクラスとして実装
- ✓ Sub test case is-a parent test case
TestAdd is a test case of TestCalc
サブテストケースは親テストケースとis-aの関係

FYI: Group

参考情報：グループ化

- ✓ `Test::Unit` and `minitest` don't support sub test case is a sub class

`Test::Unit`と`minitest`はサブクラスでサブテストケースを作ることをサポートしていない

- ✓ `test-unit` only supports it
test-unitだけがサポートしている

- ✓ It's the only one incompatible
唯一の非互換

FYI: Group DSL

参考情報：グループ化DSL

```
class TestCalc < Test::Unit::TestCase
  sub_test_case "+" do
    test "positive + negative" {}
  end
  sub_test_case "-" do
    test "positive - negative" {}
  end
end
```

test-unit feature

By example: 2
实例: 2

Share tests

テストの共有



module

Share: 1

共有: 1

```
module KVSTests
  def test_read_write
    @kvs[:key] = :value
    assert_equal(:value, @kvs[:key])
  end
end
```

Share: 2

共有: 2

```
class TestKVSHash < Test::Unit::TestCase
  include KVSTests
  def setup; @kvs = {}; end
end
class TestKVSOpenStruct < Test::Unit::TestCase
  include KVSTests
  def setup; @kvs = OpenStruct.new; end
end
```

Wrap up: Share

共有のまとめ

- ✓ Tests can be shared by module
モジュールでテストを共有できる
- ✓ Module is used for sharing method implementations in Ruby
Rubyではモジュールはメソッドの実装を共有するために使われている

By example: 3

実例：3

Assertion

アサーション



Write as-is

そのまま書く

Assert: Equal

アサート : 等価

```
def test_add
  assert_equal(3,
               1 + 2)
               # ↑ ↓ as-is
  # RSpec: expect(1 + 2).to eq(3)
end
```

Assert: Predicate

アサート：述語

```
def test_exist_readme
  assert(File.exist?("README"))
        # ↑ as-is ↓ not as-is
        # RSpec: expect(File).to exist("README")
end
```

Message from minitest

```
1) Failure:  
TestFile#test_exist_readme [test-file-minitest.rb:5]:  
Failed assertion, no message given.
```

Less information

Message from RSpec

```
1) File README should exist
   Failure/Error: expect(File).to exist("README")
     expected File to exist
   # ./spec-file.rb:5:in `block (2 levels) in <top (required)>'
```

Colorized and snippet

Message from test-unit

```
Failure: <false> is not true.  
test_exist_readme(TestFile)  
test-file-test-unit.rb:5:in `test_exist_readme'  
  2:  
  3: class TestFile < Test::Unit::TestCase  
  4:   def test_exist_readme  
=> 5:   assert(File.exist?("README"))  
  6:   end  
  7: end
```

Colorized and snippet

power-assert

```
def test_exist_readme
  assert do # ↓ as-is
    File.exist?("README")
  end
end
```

Built-in in test-unit

Message: power-assert

Failure:

```
File.exist?("README")
|   |
|   false
File (NOTE: ← Receiver information)
test_exist_readme(TestFile)
/usr/lib/ruby/vendor_ruby/power_assert.rb:36:in `start'
test-file-power-assert.rb:5:in `test_exist_readme'
2:
3: class TestFile < Test::Unit::TestCase
4:   def test_exist_readme
=> 5:   assert do
6:     File.exist?("README")
7:   end
8: end
```

FYI: test-unit and power-assert

- ✓ Don't recommend "all power-assert"
すべてpower-assertは非推奨
- ✓ Recommend "power-assert only for predicate"
述語だけpower-assertを推奨

Assert: Equal

```
# @group.users should return  
# sorted names of user  
assert_equal(user_names.sort,  
             @group.users)  
             # ↑ Align-able  
  
assert do  
  @group.users == user_names.sort  
end
```

Align

縦に並べる

- ✓ Indicate compare targets visually
視覚的に比較対象ということを示す
- ✓ Clarify differences visually
視覚的に違いを明確にする

Experiment: 1

実験: 1

Find a difference!

違うところを探せ!

Hello | Heilo

Experiment: 2

実験: 2

Find a difference!
違うところを探せ!

Hello
Heilo

Align in Ruby

Rubyで縦に並べる

```
# Arguments are natural  
method(arg1,  
        arg2)  
  
# Operands aren't natural  
operand1 ==  
operand2
```

引数だと自然だけどオペランドだと不自然

FYI: test-unit and power-assert

- ✓ Don't recommend "all power-assert"
すべてpower-assertは非推奨
- ✓ Recommend "power-assert only for predicate"
述語だけpower-assertを推奨

FYI: Fixture

参考情報：フィクスチャー

Ensure clean test environment

キレイなテスト環境を用意する仕組み

Fixture: e.g.

フィクスチャー：例

```
def setup;      p :s;  end
def teardown;  p :t;  end
def test_1;    p :t1; end
def test_2;    p :t2; end
# :s -> :t1 -> :t ->
# :s -> :t2 -> :t
```


Fixture: Open file

フィクスチャー：ファイルを開く

```
def setup
  @file = File.open("x")
end
def teardown
  @file.close
end
```

Fixture: Open file

フィクスチャー：ファイルを開く

Not Ruby-ish

Rubyっぽくない

Fixture: Open File

フィクスチャー：ファイルを開く

```
def setup # Ruby-ish version
  File.open("x") do |file|
    @file = file
    yield
  end # Close file automatically
end
```

New feature in test-unit

Wrap up: Test::Unit API

まとめ: Test::Unit API

- ✓ Ruby-ish is important in Test::Unit API

Test::Unit APIではRubyらしいことは重要

Wrap up: Test::Unit API

まとめ: Test::Unit API

- ✓ (Sub) test case by (sub) class
 (サブ) テストケースには (サブ) クラスを使う
- ✓ Share tests by module
 テストの共有にはモジュールを使う
- ✓ as-is in assert
 アサートではそのままの使い方で書く

Wrap up: Test::Unit API

まとめ: Test::Unit API

- ✓ More Ruby-ish
さらにRubyっぽく
- ✓ Support block based fixture
ブロックを使ったフィクスチャーもサポート

Goal

目標

You know about
`/\Atest.+unit\z/i`

`/\Atest.+unit\z/i`を知ること