

テスト・インテグレーション・フレームワーク に入門してみた Swift編

2015/05/23

第72回関東Cocoa勉強会


藤本 尚邦

Self-Introduction

- Hisakuni Fujimoto / 藤本尚邦
 - @fhisa, <https://github.com/hisa>
- Freelance Programmer (野良プログラマ)
- RubyCocoa.framework creator
 - It had been installed in Mac OS X from 10.5 to 10.9.
- iOS developer of half year experience

Agenda

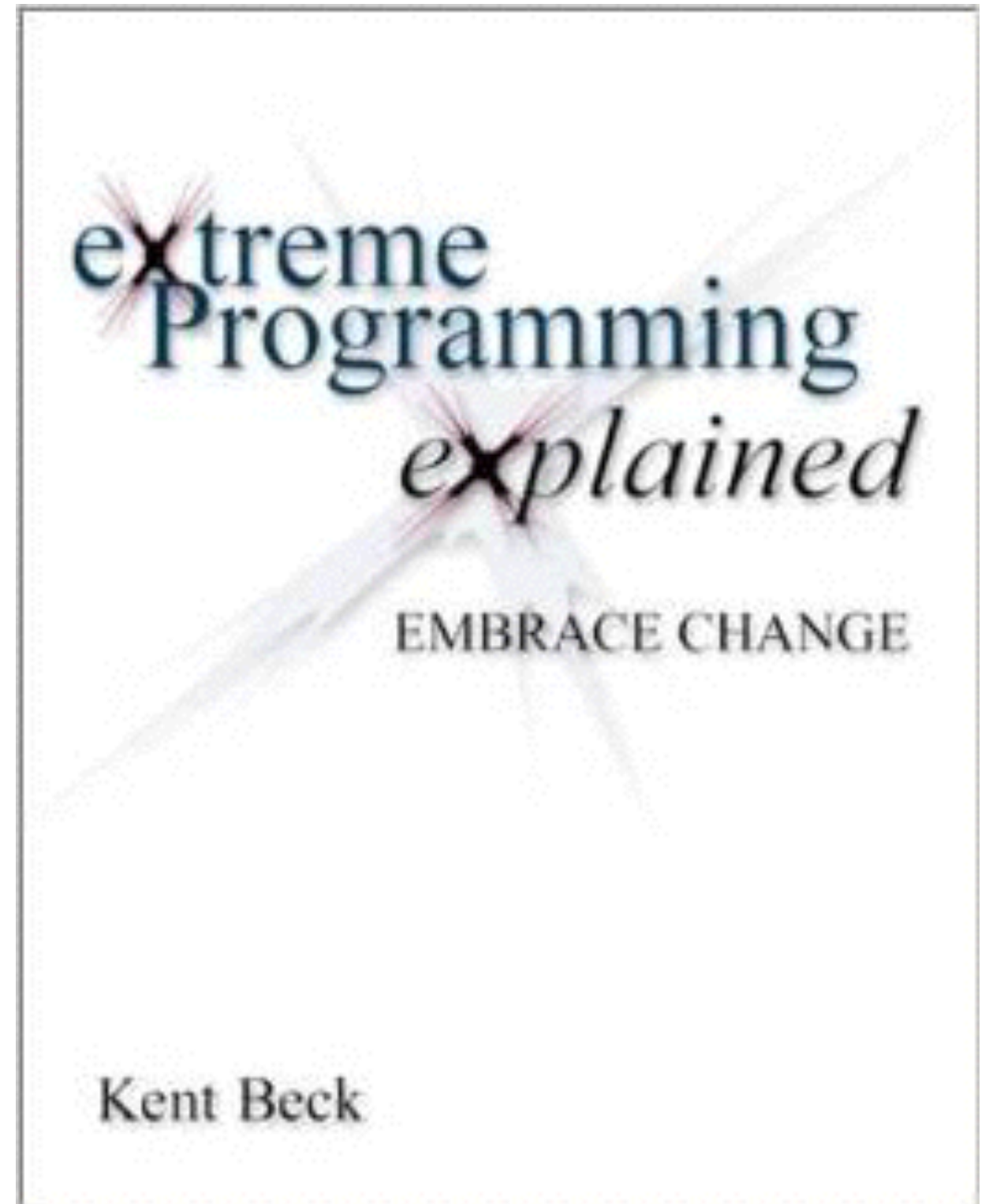
1. テスティングフレームワークの歴史
2. XCTest
3. RSpec
4. Quick
5. まとめ

A white cat with grey patches is sitting on a gravel surface. The cat is looking towards the right. The background is a blurred outdoor setting.

テストイングフレーム ワークの歴史

テストファースト

XPエクストリーム・プログラミングのプラクティスの内の1つとして右の本(1999年)で紹介されていた



テスト駆動開発(TDD)

- ・ まず単体テスト(ユニットテスト)を書く
- ・ テストを実行すると失敗する
- ・ コード本体を書く
- ・ (繰り返す)
- ・ テストの実行に成功する

xUnit

- ・ プログラムの単体テスト(ユニットテスト)を行うためのテスト
テイングフレームワークの総称
- ・ 単体テストを自動化する
- ・ JUnit (Java)
- ・ Test::Unit (Ruby)
- ・ XCTest (Objective-C, Swift)
- ・ その他たくさん

振舞駆動開発(BDD)

- ・ TDDの発展形
- ・ 「振る舞い」や「制約条件」すなわち「要求仕様」に近い形で、自然言語を併記しながらテストコードを記述
- ・ RSpec (Ruby) = Quick の元ネタ
- ・ Quick (Swift, Objective-C, Foundation.frameworkに依存している)
- ・ Sleipnir (Swiftのみで実装されている)
- ・ その他いろいろ

A small, fluffy kitten with white, grey, and orange patches is sitting on a gravel surface. The kitten has large, pointed ears and yellow-green eyes. The text "XC Test" is overlaid in the center of the image.

XC Test

NSDateクラスの単体テスト(一部)

```
import Foundation
import XCTest
```

```
class NSDateTests: XCTestCase {
```

```
    var 今: NSDate!
```

```
    var 1分後: NSDate!
```

```
    override func setUp() {
```

```
        super.setUp()
```

```
        今 = NSDate()
```

```
        1分後 = NSDate(timeInterval: 60.0, sinceDate: 今)
```

```
    }
```

```
    func test_等値演算子() {
```

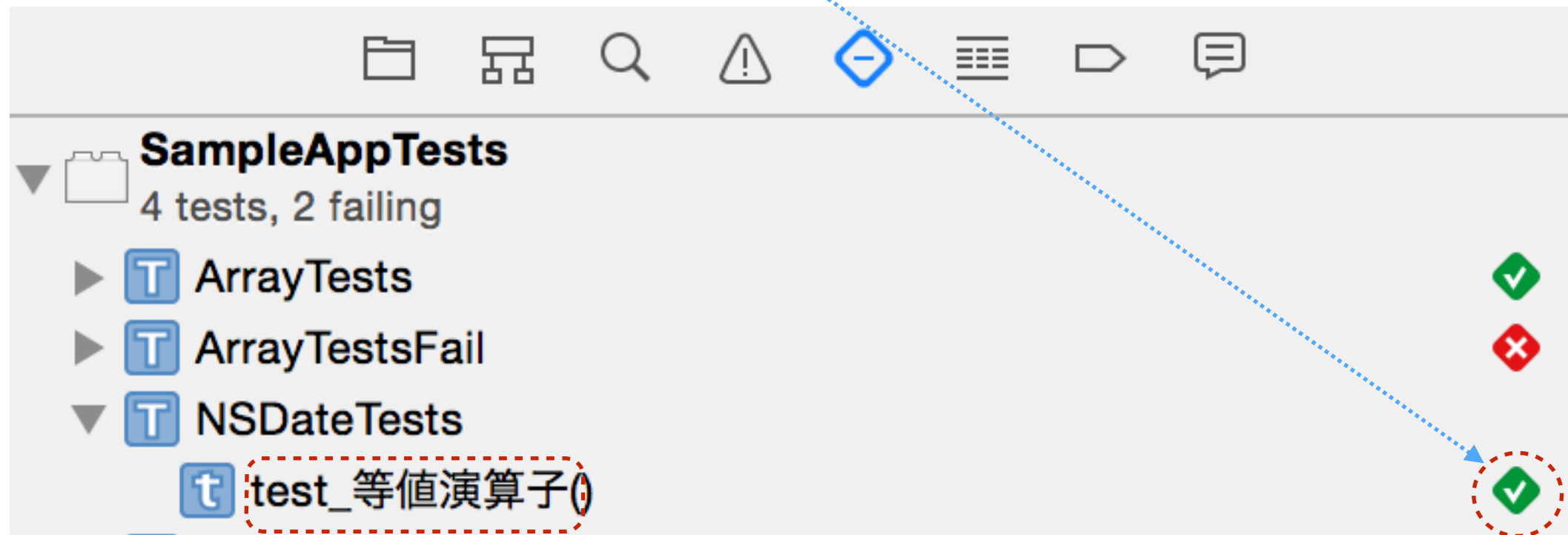
```
        XCTAssertTrue(今 == 今)
```

```
        XCTAssertFalse(今 == 1分後)
```

```
    }
```

```
}
```

テストを実行するとTest Navigatorに
緑のマークが表示される



テスト実行でコンソールに出力された内容

```
Test Suite 'NSDateTests' started at 2015-05-23 00:21:44 +0000
Test Case '-[SampleAppTests.NSDateTests test_等値演算子]' started.
Test Case '-[SampleAppTests.NSDateTests test_等値演算子]' passed (0.000 seconds).
Test Suite 'NSDateTests' passed at 2015-05-23 00:21:44 +0000.
    Executed 1 test, with 0 failures (0 unexpected) in 0.000 (0.001) seconds
```


あえて失敗するテストを実行してみる…


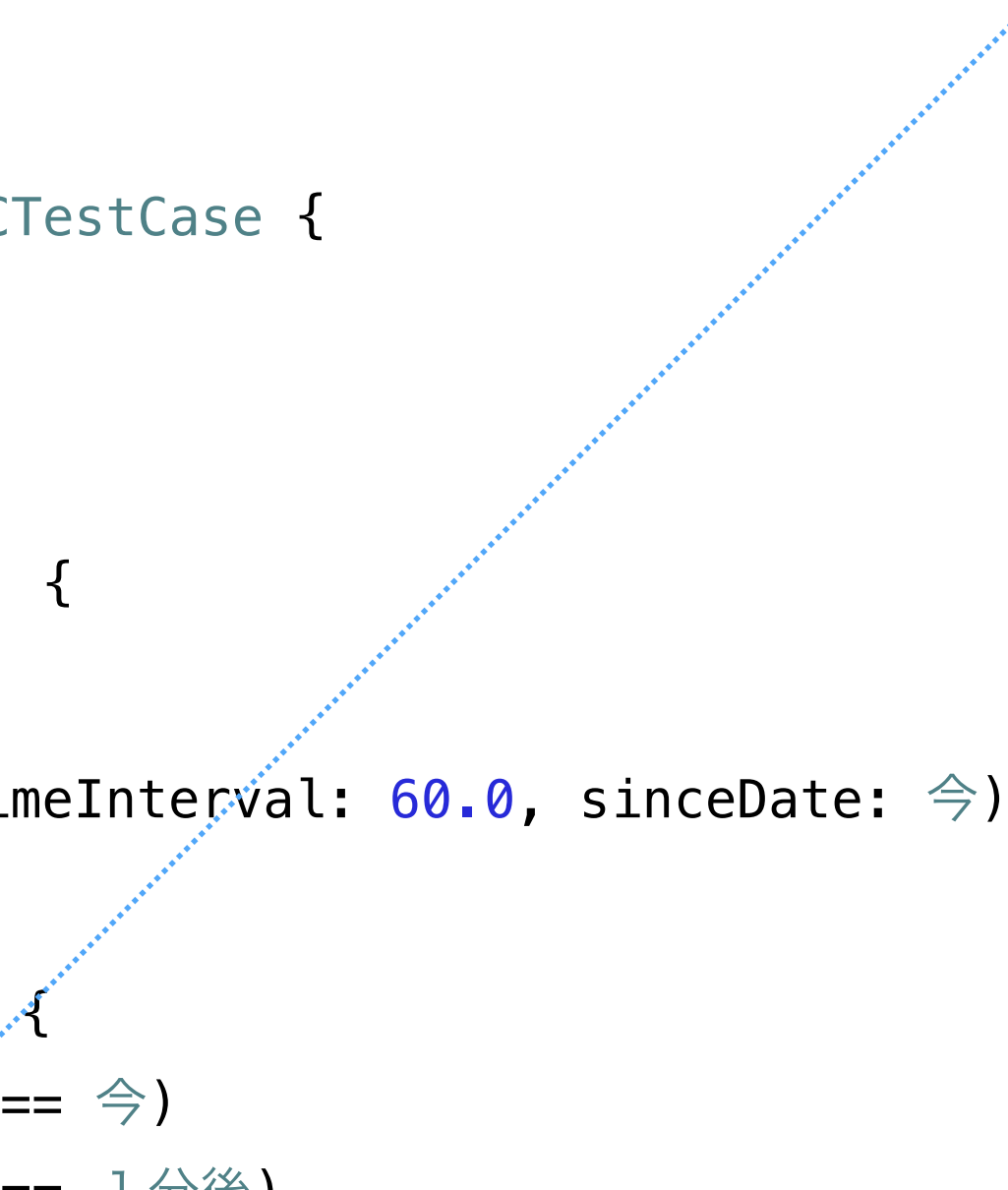
```
import Foundation
import XCTest

class NSDateTestsFail: XCTestCase {

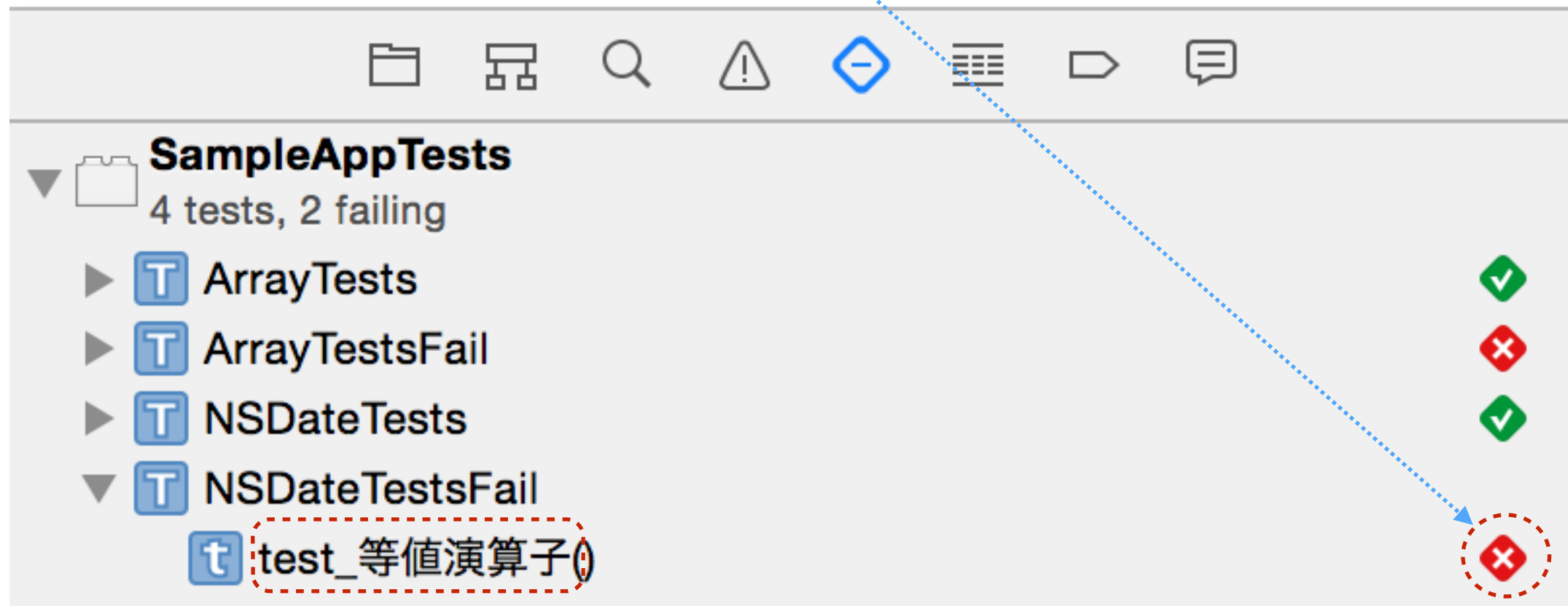
    var 今: NSDate!
    var 1分後: NSDate!

    override func setUp() {
        super.setUp()
        今 = NSDate()
        1分後 = NSDate(timeInterval: 60.0, sinceDate: 今)
    }

    func test_等値演算子() {
        XCTAssertTrue(今 == 今)
        XCTAssertTrue(今 == 1分後)
    }
}
```



Test Navigator の失敗したテストに 赤いマークが表示される



コンソール出力

```
Test Suite 'NSDateTestsFail' started at 2015-05-23 00:21:44 +0000
Test Case '-[SampleAppTests.NSDateTestsFail test_等値演算子]' started.
../SampleAppTests_fail.swift:17: error: -[SampleAppTests.NSDateTestsFail test_等値演
算子] : XCTAssertTrue failed -
Test Case '-[SampleAppTests.NSDateTestsFail test_等値演算子]' failed (0.001
seconds).
Test Suite 'NSDateTestsFail' failed at 2015-05-23 00:21:44 +0000.
    Executed 1 test, with 1 failure (0 unexpected) in 0.001 (0.001) seconds
```

A small, fluffy kitten with white, grey, and orange patches is sitting on a gravel surface. The kitten has large, upright ears and is looking towards the right. The background is a blurred, light-colored surface.

RSpec

RSpec

Behaviour Driven

Development for **Ruby**.

Making TDD Productive and Fun.

Timeクラスの仕様定義(一部)

```
describe Time do

  describe "==" do

    context "今と1分後について" do

      before do
        @now = Time.now
        @after_one_minute = @now + 60.0
      end

      it "今と今は等しい" do
        expect(@now == @now).to eq true
      end

      it "今と1分後は等しくない" do
        expect(@now == @after_one_minute).to eq false
      end
    end
  end
end
```

テストの結果、仕様を満たしていた

```
$ rspec sample_spec.rb
```

```
..
```

```
Finished in 0.00103 seconds (files took 0.09761 seconds to load)  
2 examples, 0 failures
```

並行世界でのTimeクラス仕様

```
describe Time do
  describe "==" do
    context "今と1分後について" do
      before do
        @now = Time.now
        @after_one_minute = @now + 60.0
      end

      it "今と今は等しい" do
        expect(@now == @now).to eq true
      end

      it "今と1分後は等しい" do
        expect(@now == @after_one_minute).to eq true
      end
    end
  end
end
```

The diagram illustrates the relationship between the title and the test cases. Two blue dotted arrows originate from the title '並行世界でのTimeクラス仕様'. One arrow points to the test case 'it "今と今は等しい"', and the other points to the test case 'it "今と1分後は等しい"'. In the second test case, the description '今と1分後は等しい' and the expected result 'true' are circled with red dotted lines.

こちらの世界では仕様を満たしていなかった

```
$ rspec sample_spec.rb  
.F
```

Failures:

1) Time == 今と1分後について 今と1分後は等しい

Failure/Error: expect(@now == @after_one_minute).to eq true

expected: true
got: false

(compared using ==)
./hoge.rb:18:in `block (4 levels) in <top (required)>'

Finished in 0.00162 seconds (files took 0.15771 seconds to load)
2 examples, 1 failure

Failed examples:

rspec ./hoge.rb:17 # Time == 今と1分後について 今と1分後は等しい

ここらへんが describe や context や it に
渡した文字列に対応している

A small, fluffy kitten with white, grey, and orange patches is sitting on a gravel surface. The kitten has large, pointed ears and yellow-green eyes. The word "Quick" is overlaid in the center of the image.

Quick

Quick

Quick is a behavior-driven development framework for **Swift** and Objective-C. Inspired by RSpec, Specta, and Ginkgo.

Quickの特徴

- ・ Quick=テストングフレームワーク
- ・ Nimble=matcher
- ・ 上記2つで構成されているが
- ・ Nimbleは必須ではない模様
- ・ XCTestで個々のテストを書くこともできた

Quickのインストール

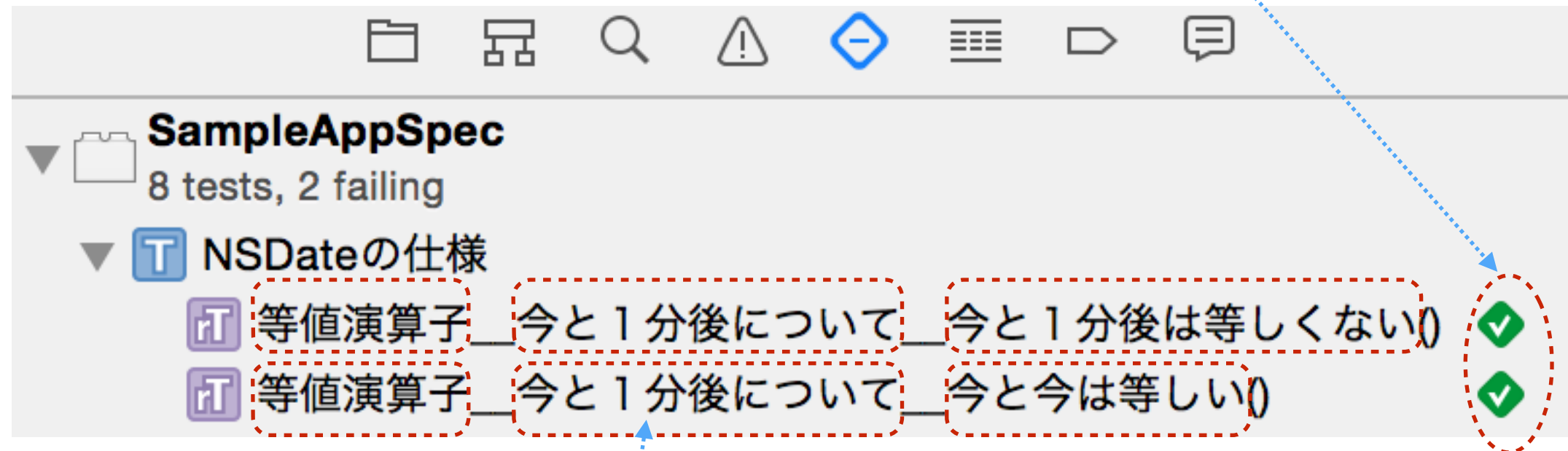
1. 3種類のインストール方法が推奨されているが...
2. Carthogeで入れる → 挫折 (Carthogeがまともに動かない)
3. CocoaPodsで入れる → 挫折(リンカーの設定とか修正が必要)
4. `git submodule add` で入れる → これに落ち着いた

```
import Foundation
import Quick
import Nimble
```

NSDateクラスの仕様定義(一部)

```
class NSDateの仕様: QuickSpec {
    override func spec() {
        describe("等値演算子") {
            context("今と1分後と1分前について") {
                var 今: NSDate!
                var 1分後: NSDate!
                beforeEach {
                    今 = NSDate()
                    1分後 = NSDate(timeInterval: 60.0, sinceDate: 今)
                }
                it("今と今は等しい") {
                    expect(今 == 今).to(beTrue())
                }
                it("今と1分後は等しくない") {
                    expect(今 == 1分後).to(beFalse())
                }
            }
        }
    }
}
```

テストの結果、仕様を満たしていた (Test Navigatorの表示)



ここらへんが describe や context や it に
渡した文字列に対応している

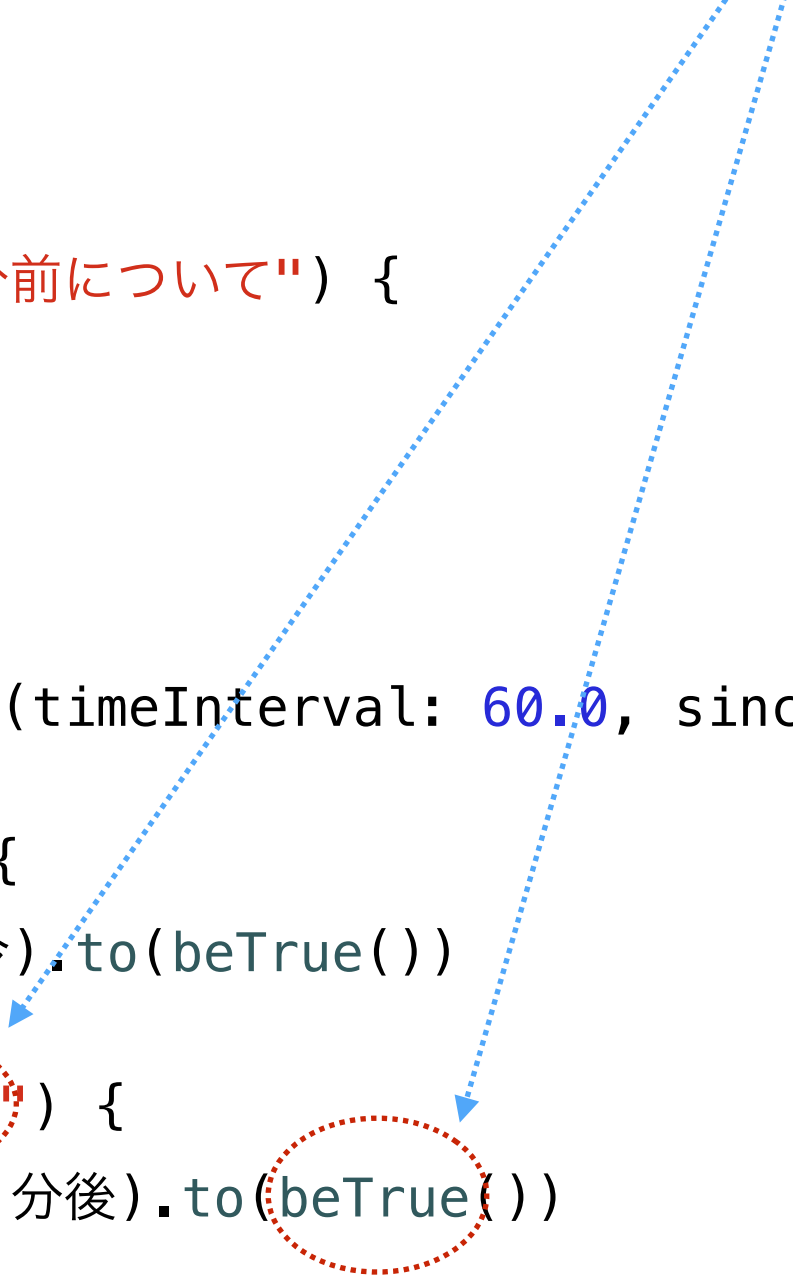
テストの結果、仕様を満たしていた (コンソール出力)

```
Test Suite 'NSDateの仕様' started at 2015-05-22 23:23:15 +0000
Test Case '-[SampleAppSpec.NSDateの仕様 等値演算子__今と1分後について__今と今は等しい]'
started.
Test Case '-[SampleAppSpec.NSDateの仕様 等値演算子__今と1分後について__今と今は等しい]' passed
(0.002 seconds).
Test Case '-[SampleAppSpec.NSDateの仕様 等値演算子__今と1分後について__今と1分後は等しくない]'
started.
Test Case '-[SampleAppSpec.NSDateの仕様 等値演算子__今と1分後について__今と1分後は等しくない]'
passed (0.000 seconds).
Test Suite 'NSDateの仕様' passed at 2015-05-22 23:23:15 +0000.
    Executed 2 tests, with 0 failures (0 unexpected) in 0.002 (0.017) seconds
```

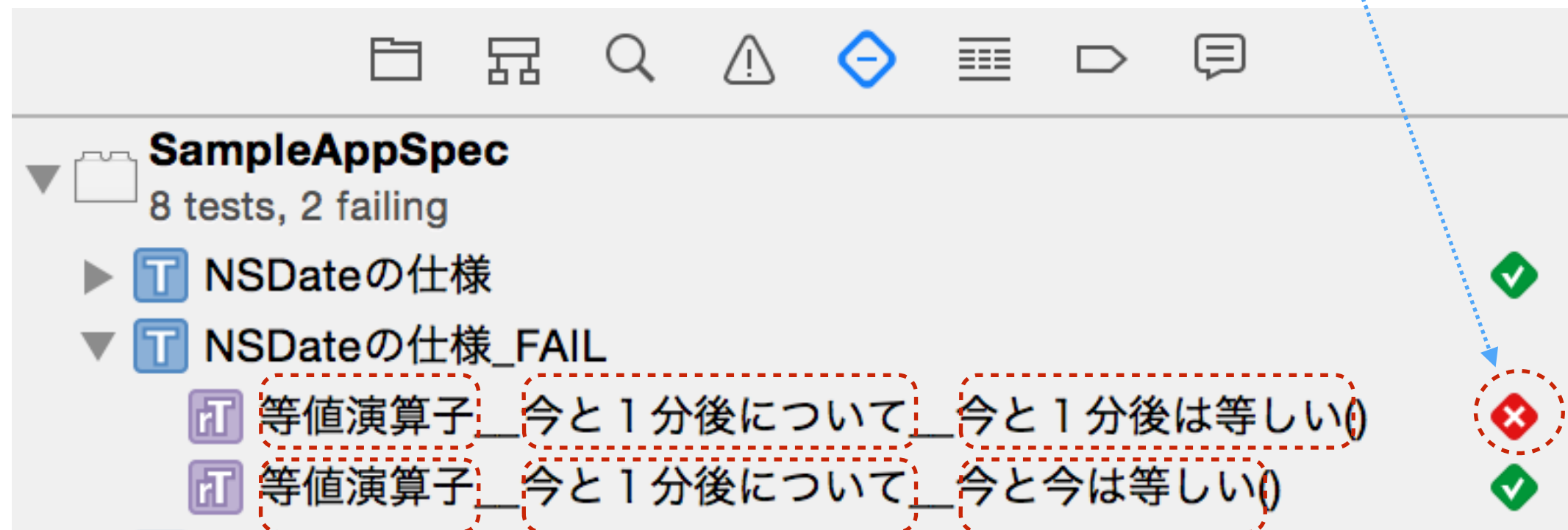
```
import Foundation
import Quick
import Nimble
```

並行世界でのNSDateクラス仕様

```
class NSDateの仕様: QuickSpec {
    override func spec() {
        describe("等値演算子") {
            context("今と1分後と1分前について") {
                var 今: NSDate!
                var 1分後: NSDate!
                beforeEach {
                    今 = NSDate()
                    1分後 = NSDate(timeInterval: 60.0, sinceDate: 今)
                }
                it("今と今は等しい") {
                    expect(今 == 今).to(beTrue())
                }
                it("今と1分後は等しい") {
                    expect(今 == 1分後).to(beTrue())
                }
            }
        }
    }
}
```



こちらの世界では仕様を満たしていない
(Test Navigatorの表示)



こちらの世界では仕様を満たしていない (コンソール出力)

Test Suite 'NSDateの仕様_FAIL' started at 2015-05-22 23:40:31 +0000

Test Case '-[SampleAppSpec.NSDateの仕様_FAIL 等値演算子__今と1分後について__今と今は等しい]' started.

Test Case '-[SampleAppSpec.NSDateの仕様_FAIL 等値演算子__今と1分後について__今と今は等しい]' passed (0.000 seconds).

Test Case '-[SampleAppSpec.NSDateの仕様_FAIL 等値演算子__今と1分後について__今と1分後は等しい]' started.

/Users/fhisa/Desktop/CocoaStudy-20150523-Fujimoto/SampleApp/SampleAppSpec/SampleAppSpec_fail.swift:24: error: -[SampleAppSpec.NSDateの仕様_FAIL 等値演算子__今と1分後について__今と1分後は等しい] : failed - expected to be true, got <false>

Test Case '-[SampleAppSpec.NSDateの仕様_FAIL 等値演算子__今と1分後について__今と1分後は等しい]' failed (0.001 seconds).

Test Suite 'NSDateの仕様_FAIL' failed at 2015-05-22 23:40:31 +0000.

Executed 2 tests, with 1 failure (0 unexpected) in 0.002 (0.003) seconds

まとめ

1. XCTestは、インストール不要で手軽
2. Quickは、インストールに一苦労した
3. XCTestは、テストの書き方がシンプルで簡単
4. Quickは、実行可能な仕様書として使えそうなところが良い
5. Quickで良い仕様(テスト)書くためには、文章構成力が問われる
6. Quickは、Nimble の expect 記法を覚えるまでが大変そう

Thank You!



2015/05/23
第72回関東Cocoa勉強会
藤本 尚邦