Swift Testing の
パラメトライズテスト

宇佐見公輔

2024-11-14

株式会社ゆめみ

自己紹介

- 宇佐見公輔
  - 株式会社ゆめみ　iOS テックリード

← ゆめみ大技林
技術書典 17 で頒布中

月刊 I/O →
最近いくつか寄稿

Swift Testing とは

Swift Testing とは

Swift 用の単体テストフレームワーク

- Swift 6、Xcode 16 に統合された
- 従来の XCTest に比べて、Swift の機能をより活用
- Swift Testing と XCTest は同じプロジェクトで混在可
  - そのため、少しずつ移行できる

Swift Testing の構成要素

- テスト関数
  - `@Test` 属性
- 期待値の確認
  - `#expect` マクロ
- テストスイート
  - `@Suite` 属性
- トレイト
  - `TestTrait` / `SuiteTrait`

リファレンス

リファレンス

[Swift Testing | Apple Developer Documentation]

- ドキュメントは結構ある
- ただ、マクロのリファレンスなどはやや読みづらい
- そこで解説本を執筆中
  - 技術書典 17 に間に合わせたかったけど・・・

実際のリファレンスの例

```
@freestanding(expression)
macro expect(
    _ condition: Bool,
    _ comment: @autoclosure () -> Comment? = nil,
    sourceLocation: SourceLocation = #_sourceLocation
)
```

実際の使い方

```
#expect(library.count == 1)
```

テスト関数

テスト関数

```
import XCTest
class MySimpleLibrarytTests: XCTestCase {
    func testAddBookSuccessfully() {
        // ...
    }
}
```

```
import Testing
struct MySimpleLibrarytTests {
    @Test func addBookSuccessfully() {
        // ...
    }
}
```

期待値の確認

期待値の確認

```swift
// XCTest
func testAddBookSuccessfully() {
    library.addBook(book)
    XCTAssertEqual(library.count, 1)
    XCTAssertNotNil(library.findBook(byISBN: "1234"))
}

// Swift Testing
@Test func addBookSuccessfully() {
    library.addBook(book)
    #expect(library.count == 1)
    #expect(library.findBook(byISBN: "1234") != nil)
}
```

Xcode 上の表示

#expect マクロはテスト失敗時の結果をきれいに表示する

```
@Test func addBookSuccessfully() {
8       let library = MySimpleLibrary()
9       let book = MySimpleBook(isbn: "1234567890123")
10      library.addBook(book)
11
12      #expect(library.count == 1)
13      #expect(library.findBook(byISBN: "1234567890123") == book)   Expectation failed: (library.findBook(byISB...
        Results
          library.findBook(byISBN: "1234567890123") : MySimpleBook(isbn: "0000000000000")
          Optional<MySimpleBook>
            some : MySimpleBook(isbn: "0000000000000")
            MySimpleBook
          book : MySimpleBook(isbn: "1234567890123")
          MySimpleBook
            isbn : "1234567890123"
            String
14  }
```

期待値の確認

コメント

`#expect` にコメントを書くと、テスト失敗時に表示してくれる

```swift
// 空の本棚に 1 冊追加したので、本の数は 1 冊になるはず
#expect(library.count == 1)
```

```
◆ Test addBookSuccessfully() recorded an issue at
MySimpleLibrarytTests.swift:13:5: Expectation failed: (library.count → 0) == 1
↳ // 空の本棚に1冊追加したので、本の数は1冊になるはず
```

テストスイート

テストスイート

```
struct MySimpleLibrarytTests {
    @Test func addBookSuccessfully() {
        // ...
    }
}
```

- テスト関数を含む型が、自動的にテストスイートになる
- 後述するトレイトを使う場合は `@Suite` 属性を指定する

Swift の機能の活用

```
final class MySimpleLibrarytTests {
    init() async throws {
        // ...
    }

    deinit {
        // ...
    }
}
```

- 専用の `setUp` の代わりに、通常の `init` が使える
- `actor` や `@MainActor` なども使える

トレイト

トレイト

テスト関数やテストスイートの振る舞いを指定する

```
@Suite(.timeLimit(.minutes(1)))
struct MySimpleLibrarytTests {
    @Test func addBookSuccessfully() {
        // ...
    }
}
```

- テストスイートのタイムアウト時間を指定する例

用意されているトレイト

- `.enabled` / `.disabled`
- `.timeLimit`
- `.serialized`
- `.tags`
- `.bug`
- `.isRecursive`

Swift Testing の機能

Swift Testing の機能

Swift Testing ならではの機能

- エラーのテスト
- パラメトライズテスト
- テストの並列実行

今回は、パラメトライズテストについて紹介

パラメトライズテスト

テスト関数に引数を渡す

入力や期待値をテスト関数の外側から渡せる

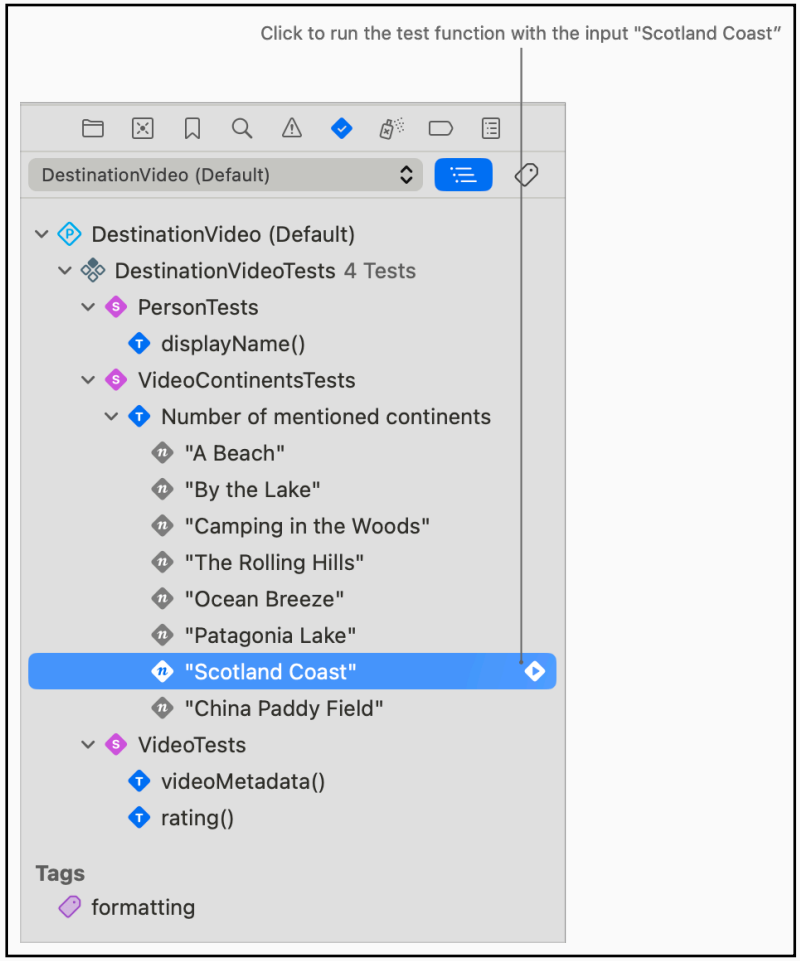
```
@Test(arguments:[(
    input: ...,
    expected: ...
)])
func addBook(input: InputData, expected: ExpectedData) {
    library.addBook(input.book)
    #expect(library.count == expected.count)
}
```

複数のデータでテストする

同じテスト関数を入力を変えて何度も実行できる

```
@Test(arguments:[
    (input: ..., expected: ...),
    (input: ..., expected: ...),
    (input: ..., expected: ...)
])
func addBook(input: InputData, expected: ExpectedData) {
    library.addBook(input.book)
    #expect(library.count == expected.count)
}
```

Xcode 上の表示



Xcode のテストナビゲータで、パラメータごとに分かれて表示される

- それぞれのテスト結果がわかる
- 特定パラメータだけ実行できる

まとめ

まとめ

- Swift Testing の構成要素
  - テスト関数、期待値の確認、テストスイート、トレイト
- Swift Testing ならではの機能
  - エラーのテスト、パラメトライズテスト、並列実行