XCTest から Swift Testing へ

宇佐見公輔

2024-09-25

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
- iOSDC Japan 2024
  - パンフレット記事 / ポスターセッション

Swift Testing とは

Swift Testing とは

- Swift 用の単体テストフレームワーク
  - 従来の XCTest に比べて、Swift の機能をより活用
- Swift 公式の GitHub リポジトリで公開されている
  - https://github.com/swiftlang/swift-testing
- Xcode 16 に統合された
  - Xcode の UI と連携する

XCTest と Swift Testing

- XCTest
  - 従来からの単体テストフレームワーク
- Objective-C 時代から存在、Swift にも適合
  - https://github.com/swiftlang/swift-corelibs-xctest
- XCTest と Swift Testing は同じプロジェクトで混在可能
  - そのため、少しずつ移行していくことが可能

Swift Testing の基本

テスト関数定義：XCTest

```
import XCTest

class FoodTruckTests: XCTestCase {
    func testEngineWorks() {
        // ...
    }
}
```

- XCTestCase のサブクラス内で定義する必要
- メソッド名を test 始まりで命名する必要

テスト関数定義：Swift Testing

```
import Testing

struct FoodTruckTests {
    @Test
    func engineWorks() {
        // ...
    }
}
```

- テスト関数はどこで定義してもよい
- `@Test` 属性をつければメソッド名は自由

テストの実装

```swift
// XCTest
func testEngineWorks() throws {
    XCTAssertNotNil(engine.parts.first)
    XCTAssertGreaterThan(engine.batteryLevel, 0)
    XCTAssertTrue(engine.isRunning)
}

// Swift Testing
@Test func engineWorks() throws {
    try #require(engine.parts.first != nil)
    #expect(engine.batteryLevel > 0)
    #expect(engine.isRunning)
}
```

Optional 値の Unwrap

```swift
// XCTest
func testEngineWorks() throws {
    let part = try XCTUnwrap(engine.parts.first)
}

// Swift Testing
@Test func engineWorks() throws {
    let part = try #require(engine.parts.first)
}
```

テスト実行時の setup/teardown

```
// XCTest
class FoodTruckTests: XCTestCase {
    override func setUp() async throws {
        // ...
    }
}

// Swift Testing
final class FoodTruckTests {
    init() async throws {
        // ...
    }
}
```

変換ツール

- XCTest のコードを Swift Testing に変換するツール
  - https://github.com/giginet/swift-testing-revolutionary

Swift Testing の機能

パラメータを変えてテスト

```
enum Food {
  case burger, iceCream, burrito, noodleBowl, kebab
}

@Test(arguments: [
    Food.burger, .iceCream, .burrito, .noodleBowl, .kebab
])
func foodAvailable(_ food: Food) async throws {
    let foodTruck = FoodTruck(selling: food)
    #expect(await foodTruck.cook(food))
}
```

複数のパラメータ

```
@Test(arguments: zip(Food.allCases, 1 ... 100))
func makeLargeOrder(of food: Food, count: Int) async
throws {
  let foodTruck = FoodTruck(selling: food)
  #expect(await foodTruck.cook(food, quantity: count))
}
```

テスト実行制御

```
@Test(.enabled(if: FoodTruck.sells(.arepas)))
func arepasAreTasty() {
    // ...
}

@Test(.timeLimit(.seconds(30))
func serveLargeOrder() {
    // ...
}
```

アノテーション

```
@Test func grillWorks() async {
    withKnownIssue("Grill is out of fuel") {
        try FoodTruck.shared.grill.start()
    }
}
```

非同期処理のテスト

非同期処理のテスト：XCTest

```
func testTruckEvents() async {
    let soldFood = expectation(description: "…")
    FoodTruck.shared.eventHandler = { event in
        soldFood.fulfill()
    }
    await Customer().buy(.soup)
    await fulfillment(of: [soldFood])
}
```

expectation を使う。fulfill() が呼ばれたら成功。

非同期処理のテスト：Swift Testing

```
@Test func truckEvents() async {
    await confirmation("…") { soldFood in
        FoodTruck.shared.eventHandler = { event in
            soldFood()
        }
        await Customer().buy(.soup)
    }
}
```

confirmation を使う。confirmed が呼ばれたら成功。

注意点

- XCTest の `expectation` では `await fulfillment()` で完了待ちをしていた。
- Swift Testing の `confirmation` は完了待ちをしない。ブロックを抜けるまでに `confirmed` メソッドが呼ばれないと失敗扱い。

この点では、不便になっているようにも見えるが・・・

解決策(1)

Swift Concurrency の `withCheckedContinuation` を使う。

```
@Test func truckEvents() async {
  await confirmation("…") { soldFood in
    await withCheckedContinuation { continuation in
      FoodTruck.shared.eventHandler = { event in
        soldFood()
        continuation.resume()
      }
      await Customer().buy(.soup)
    }
  }
}
```

解決策(2)

pointfreeco/swift-concurrency-extras の megaYield を使う。

```
@Test func truckEvents() async {
  await confirmation("…") { soldFood in
    FoodTruck.shared.eventHandler = { event in
      soldFood()
    }
    await Customer().buy(.soup)
    await Task.megaYield()
  }
}
```

まとめ

XCTest から Swift Testing へ

- XCTest から段階的に移行できる
- 機械的に移行できる部分も多い
- パラメータテストが便利
- 非同期処理のテストは少し変わっているので注意