Testing with RxTest

Mostafa Amer

RxTest

- Unit testing extensions for RxSwift.

- Should make unit testing your operators easy as unit testing RxSwift built-in operators

RxTest Basics

- TestScheduler
- Recorded<Event<T>>
- TestableObservable / TestableObserver
- Subscription

RxTestExt

- TestScheduler Extension
- Testable Observer assertions
- and hopefully more to come …
- https://github.com/RxSwiftCommunity/RxTestExt

Demo

```
protocol ZenViewModelType {
    var isLoading: Driver<Bool> { get }
    var load: AnyObserver<Void> { get }
}

override func viewDidLoad() {
    super.viewDidLoad()
    viewModel.isLoading.drive(spinner.rx.isAnimating).disposed(by: bag)
    load.rx.tap.bind(to: viewModel.load).disposed(by: bag)
}
```

Record Events

```
func testRequestCorrectURL() {
    _ = scheduler.record(source: sut.zen())      // 1
    scheduler.start()                            // 2
    XCTAssertEqual(mockSession.requestedURL, "https://api.github.com/zen")
}
```

1. Record events from given observable type into a TestableObserver
2. Start scheduler

Playback Events

```
func testCallAPIWhenRequested() {
    scheduler.bind([next(10, ())], to: sut.load) // 1
    scheduler.start()                            // 2
    XCTAssertEqual(mockAPI.zenCalled, 1)
}
```

1. Deliver given events to the observer
2. Start scheduler

Testing Drivers

```
func testLoadingStateFollowsRequest() {
    mockAPI.zenEvents = [next(10, "Hello, world!"), completed(10)]
    scheduler.bind([next(5, ())], to: sut.load)
    SharingScheduler.mock(scheduler: scheduler) {                    // 1
        let loading = scheduler.record(source: sut.isLoading)
        scheduler.start()
        XCTAssertEqual(loading.events, [                             // 2
            next(0, false),
            next(5, true),
            next(15, false),
        ])
    }
}
```

1. Mock sharing scheduler using TestScheduler
2. Assert recorded events is matching expectations

Assert Nexts

```
func testParseResponse() {
    mockSession.requestEvents = [
        next(10, "Hello, world!".data(using: .utf8)!),
        completed(10),
    ]
    let zen = scheduler.record(source: sut.zen())
    scheduler.start()
    assert(zen).next(at: 10)                          // 1
    assert(zen) == "Hello, world!"                    // 2
}
```

1. Assert receiving next events at specific time
2. Assert first next event equals a specific value

Assert Completes

```
func testParseResponse() {
    mockSession.requestEvents = [
        next(10, "Hello, world!".data(using: .utf8)!),
        completed(10),
    ]
    let zen = scheduler.record(source: sut.zen())
    scheduler.start()
    assert(zen).complete(at: 10)                    // 1
}
```

1. Assert receiving complete event at specific time

Assert Errors

```
func testInterceptingRequestTimedOutError() {
    mockSession.requestEvents = [error(10, NSError(domain: NSURLErrorDomain,
                                                   code: NSURLErrorTimedOut,
                                                   userInfo: nil))]
    let zen = scheduler.record(source: sut.zen())
    scheduler.start()
    assert(zen).error(after: 0)                                    // 1
    assert(zen).error(with: APIClient.Error.requestTimedOut)       // 2
}
```

1. Assert receiving error event after specific number of next events.
2. Assert received error equals a specific error.

Q & A

@mostafa_amer

https://github.com/mosamer/RxSwiftMeetup-RxTest