

What is The Test

Feat. UniTest, UITest

Test?

개발자가 작성한 코드가 의도한대로 작동하는지 확인하는 과정

Test?

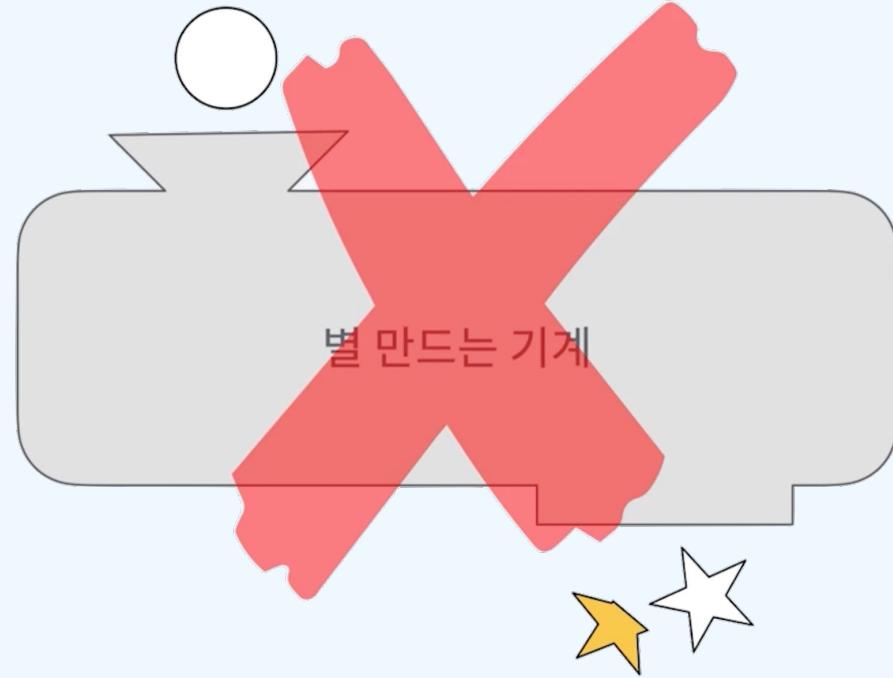
개발자가 작성한 코드가 **의도한대로 작동**하는지 확인하는 과정

- UI 표시
- UI Layout 설정
- API request
- UserDefaults 저장하기/ 가져오기
- 등등

Test?



Test?

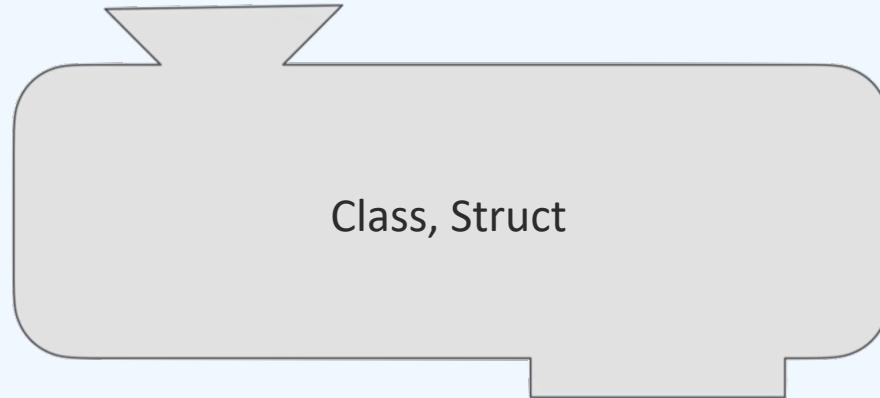


Test?



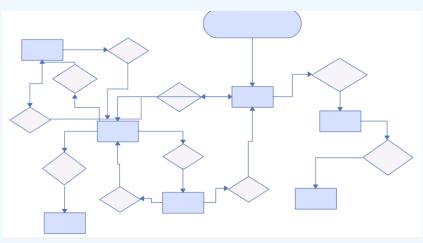
Test?

View의 onAppear



func

Test 종류



Unit Test

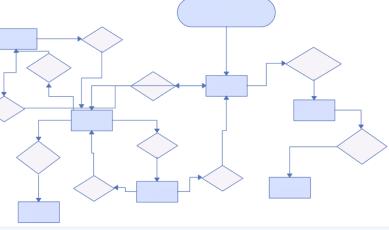
특정 함수, 메소드의 동작에 대한
테스트



UI Test

UI 표시 or UI Action에 대한
테스트

Test 종류



Unit Test

특정 함수, 메소드의 동작에 대한
테스트

Scene을 테스트 대상으로 실행

Class ProfileViewModel: XCTestCase {}



UI Test

UI 표시 or UI Action에 대한
테스트

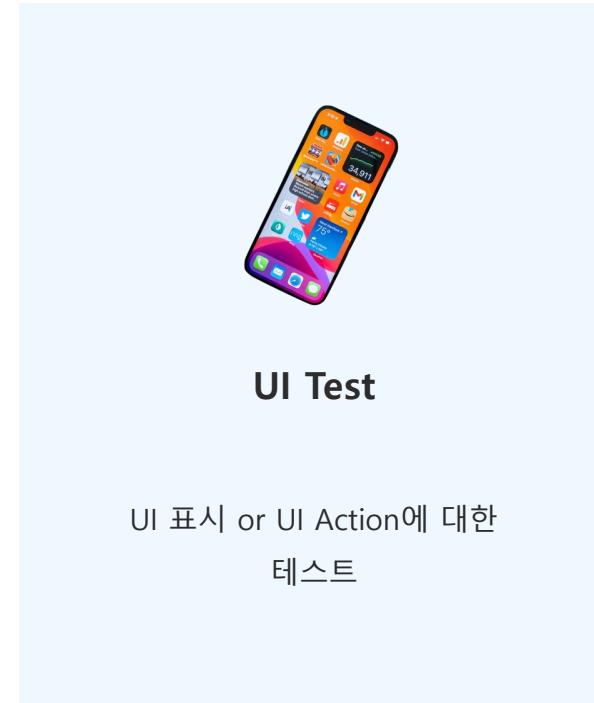
하나의 앱을 테스트 대상으로 실행

XCUIApplication().launch()

Test 종류



Scene을 테스트 대상으로 실행
Class ProfileViewModel: XCTestCase {}

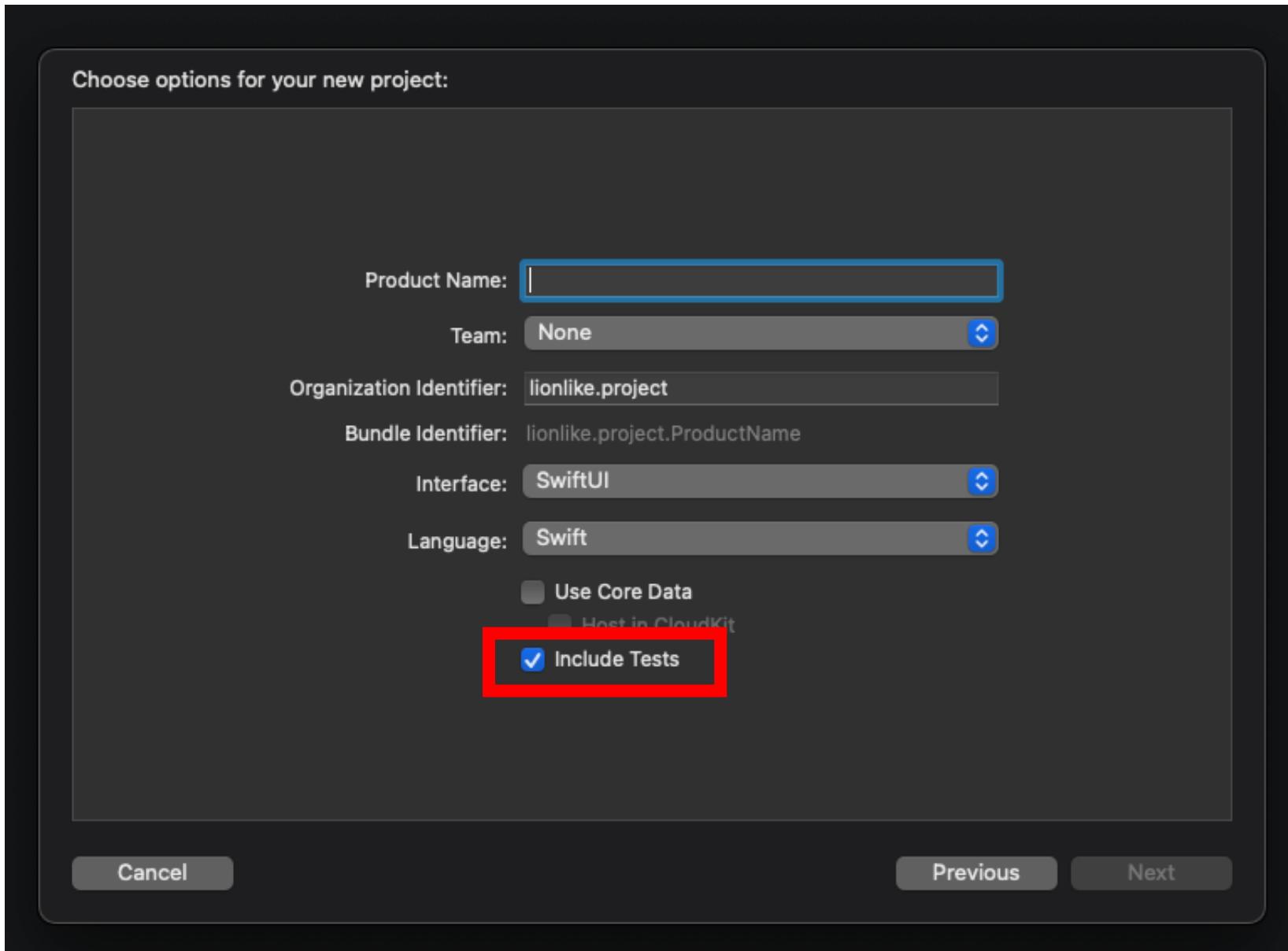


하나의 앱을 테스트 대상으로 실행
XCUIApplication().launch()

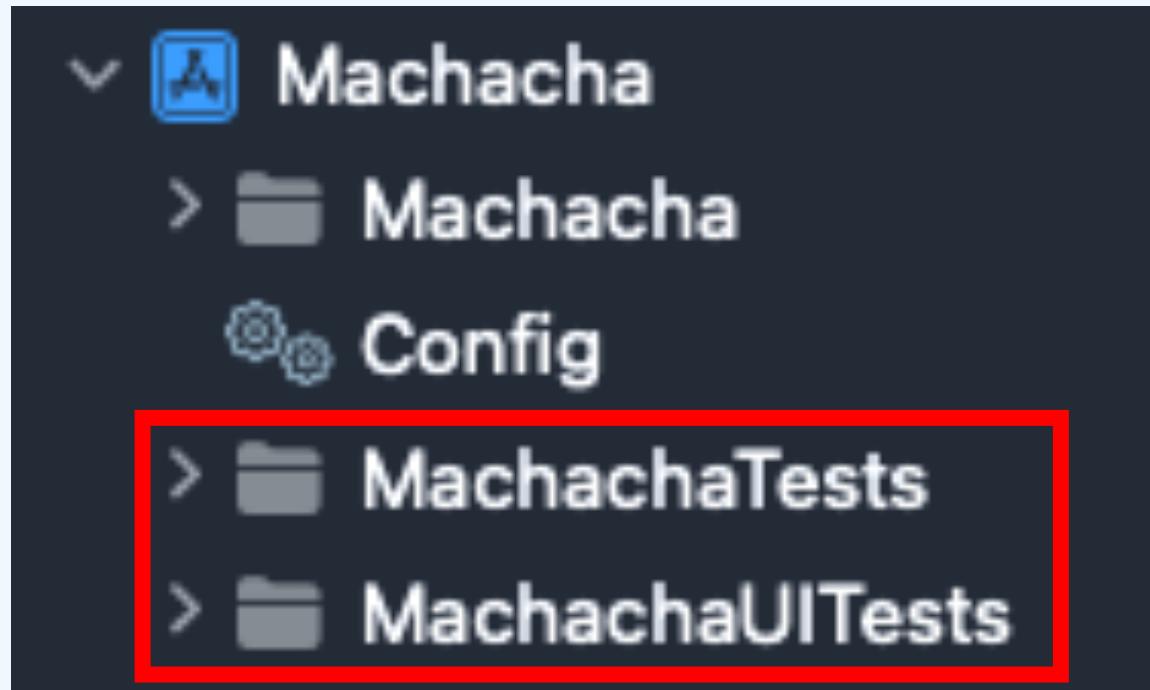
The screenshot shows the Apple Developer Documentation website with the following details:

- Header:** Apple Developer, Discover, Design, Develop, Distribute, Support, Account, and a search icon.
- Breadcrumbs:** Documentation > XCTest.
- Language and API Changes:** Language: Swift ▾ API Changes: Show ▾
- Section Headers:** Framework, XCTest.
- Text:** Create and run unit tests, performance tests, and UI tests for your Xcode project.
- Availability:** Xcode 5.0+.
- Overview:** Use the XCTest framework to write unit tests for your Xcode projects that integrate seamlessly with Xcode's testing workflow.
- Text (continued):** Tests assert that certain conditions are satisfied during code execution, and record test failures (with optional messages) if those conditions aren't satisfied. Tests can also measure the performance of blocks of code to check for performance regressions, and can interact with an application's UI to validate user interaction flows.
- On This Page:** Overview ⓘ, Topics ⓘ.

XCTest > UniTest



XCTest > UniTest



XCTest > UniTest

```
1 //  
2 //  MachachaTests.swift  
3 //  MachachaTests  
4 //  
5 //  Created by geonhyeong on 2023/01/17.  
6 //  
7  
8 import XCTest  
9 @testable import Machacha  
10  
◇ final class MachachaTests: XCTestCase {  
11  
13     override func setUpWithError() throws {  
14         // Put setup code here. This method is called before the invocation of each test method in the class.  
15     }  
16  
17     override func tearDownWithError() throws {  
18         // Put teardown code here. This method is called after the invocation of each test method in the class.  
19     }  
20  
◇     func testExample() throws {  
21         // This is an example of a functional test case.  
22         // Use XCTAssert and related functions to verify your tests produce the correct results.  
23         // Any test you write for XCTest can be annotated as throws and async.  
24         // Mark your test throws to produce an unexpected failure when your test encounters an uncaught error.  
25         // Mark your test async to allow awaiting for asynchronous code to complete. Check the results with assertions afterwards.  
26     }  
27  
◇     func testPerformanceExample() throws {  
28         // This is an example of a performance test case.  
29         self.measure {  
30             // Put the code you want to measure the time of here.  
31         }  
32     }  
33 }  
34 }  
35 }  
36 }
```

XCTest > UniTest

```
1 //  
2 //  MachachaTests.swift  
3 //  MachachaTests  
4 //  
5 //  Created by geonhyeong on 2023/01/17.  
6 //  
  
import XCTest  
@testable import Machacha  
  
◇ final class MachachaTests: XCTestCase {  
12  
13     override func setUpWithError(  
14         // Put setup code here. T  
15     )  
16  
17     override func tearDownWithError(  
18         // Put teardown code here  
19     )  
20  
◇     func testExample() throws {  
22         // This is an example of  
23         // Use XCTAssert and rela  
24         // Any test you write for  
25         // Mark your test throws  
26         // Mark your test async t  
27     }  
28  
◇     func testPerformanceExample() throws {  
30         // This is an example of a performance test case.  
31         self.measure {  
32             // Put the code you want to measure the time of here.  
33         }  
34     }  
35  
36 }
```

import XCTest
@testable import Machacha

XCTest > UniTest

```
1 //  
2 // MachachaTests.swift  
3 // MachachaTests  
4 //  
5 // Created by geonhyeong on 2023/01/17.  
6 //  
7  
8 import XCTest  
9 @testable import Machacha  
10  
11 final class MachachaTests: XCTestCase {  
12  
13     override func setUpWithError() throws {  
14         // Put setup code here. This method is called before the invocation of each test method in the class.  
15     }  
16  
17     override func tearDownWithError() throws {  
18         // Put teardown code here  
19     }  
20  
21     func testExample() throws {  
22         // This is an example of a functional test case.  
23         // Use XCTAssertEqual and related functions to verify your tests produce the correct results.  
24         // Any test you write for XCTest can be annotated as throws and async.  
25         // Mark your test throws to produce an unexpected failure when your test encounters an uncaught error.  
26         // Mark your test async to allow awaiting for asynchronous code to complete. Check the results with assertions afterwards.  
27     }  
28  
29     func testPerformanceExample() throws {  
30         // This is an example of a performance test case.  
31         self.measure {  
32             // Put the code you want to measure the time of here.  
33         }  
34     }  
35 }  
36 }
```

final class MachachaTests: XCTestCase {

Class ProfileViewModelTest

XCTest > UniTest

```
1 //  
2 //  MachachaTests.swift  
3 //  MachachaTests  
4 //  
5 //  Created by geonhyeong on 2023/01/17.  
6 //  
7  
8 import XCTest  
9 @testable import Machacha  
10  
◇ final class MachachaTests: XCTestCase {  
11  
12     override func setUpWithError() throws {  
13         // Put setup code here. This method is called before the invocation of each test method in the class.  
14     }  
15     override func tearDownWithError() throws {  
16         // Put teardown code here. This method is called after the invocation of each test method in the class.  
17     }  
18 }  
19  
◇ func testPerformanceExample() throws {  
20     // This is an example of a performance test case.  
21     self.measure {  
22         // Put the code you want to measure the time of here.  
23     }  
24 }  
25  
26 // MARK: - Test Async To Allow Handling Of Asynchronous Code To Complete. Check The Results With Assertions At Conclusion.  
27 }  
28  
29  
30 }
```

XCTest > UniTest

```
1 //  
2 // MachachaTests.swift  
3 // MachachaTests  
4 //  
5 // Created by geonhyeong on 2023/01/17.  
6 //  
7  
8 func testExample() throws {  
9     // This is an example of a functional test case.  
10    // Use XCTAssert and related functions to verify your tests produce the correct results.  
11    // Any test you write for XCTest can be annotated as throws and async.  
12    // Mark your test throws to produce an unexpected failure when your test encounters an uncaught error.  
13    // Mark your test async to allow awaiting for asynchronous code to complete. Check the results with assertions afterwards.  
14    //  
15    //  
16 }  
17  
18    // Put teardown code here. This method is called after the invocation of each test method in the class.  
19 }  
20  
21 func testExample() throws {  
22     // This is an example of a functional test case.  
23     // Use XCTAssert and related functions to verify your tests produce the correct results.  
24     // Any test you write for XCTest can be annotated as throws and async.  
25     // Mark your test throws to produce an unexpected failure when your test encounters an uncaught error.  
26     // Mark your test async to allow awaiting for asynchronous code to complete. Check the results with assertions afterwards.  
27 }  
28  
29 func testPerformanceExample() throws {  
30     // This is an example of a performance test case.  
31     self.measure {  
32         // Put the code you want to measure the time of here.  
33     }  
34 }  
35  
36 }
```

XCTest > UniTest

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with BookReviewTests as the selected target.
- Editor:** Displays the code for `BookReviewTests.swift`. A red box highlights line 20, which contains the declaration of the `testExample()` method.
- Utilities:** The right-hand panel shows the file's properties:
 - Identity and Type:** Name: `BookReviewTests.swift`, Type: `Default - Swift Source`, Location: `Relative to Group`.
 - On Demand Resource Tags:** Only resources are taggable.
 - Target Membership:** BookReviewTests is checked.
 - Text Settings:** Text Encoding: `No Explicit Encoding`, Line Endings: `No Explicit Line Endings`, Indent Using: `Spaces`, Widths: Tab 4, Indent 4, Wrap lines checked.

```
// BookReviewTests
// Created by Eunyeong Kim on 2021/07/21.

import XCTest
@testable import BookReview

class BookReviewTests: XCTestCase {

    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the class.
    }

    override func tearDownWithError() throws {
        // Put teardown code here. This method is called after the invocation of each test method in the class.
    }

    func testExample() throws {
        // This is an example of a functional test case.
        // Use XCTAssert and related functions to verify your tests produce the correct results.
    }

    func testPerformanceExample() throws {
        // This is an example of a performance test case.
        self.measure {
            // Put the code you want to measure the time of here.
        }
    }
}
```

XCTest > UniTest

The screenshot shows the Xcode project navigator on the left with 'Machacha' and 'MachachaTests' selected. The 'Link Binary With Libraries' section for 'MachachaTests' is highlighted with a red box. Inside this box, five frameworks are listed: FirebaseFunctions, GTMSessionFetcherCore, GoogleSignIn, FirebaseFirestore, and FirebaseAuth. A second red box highlights the GTMSessionFetcherCore entry.

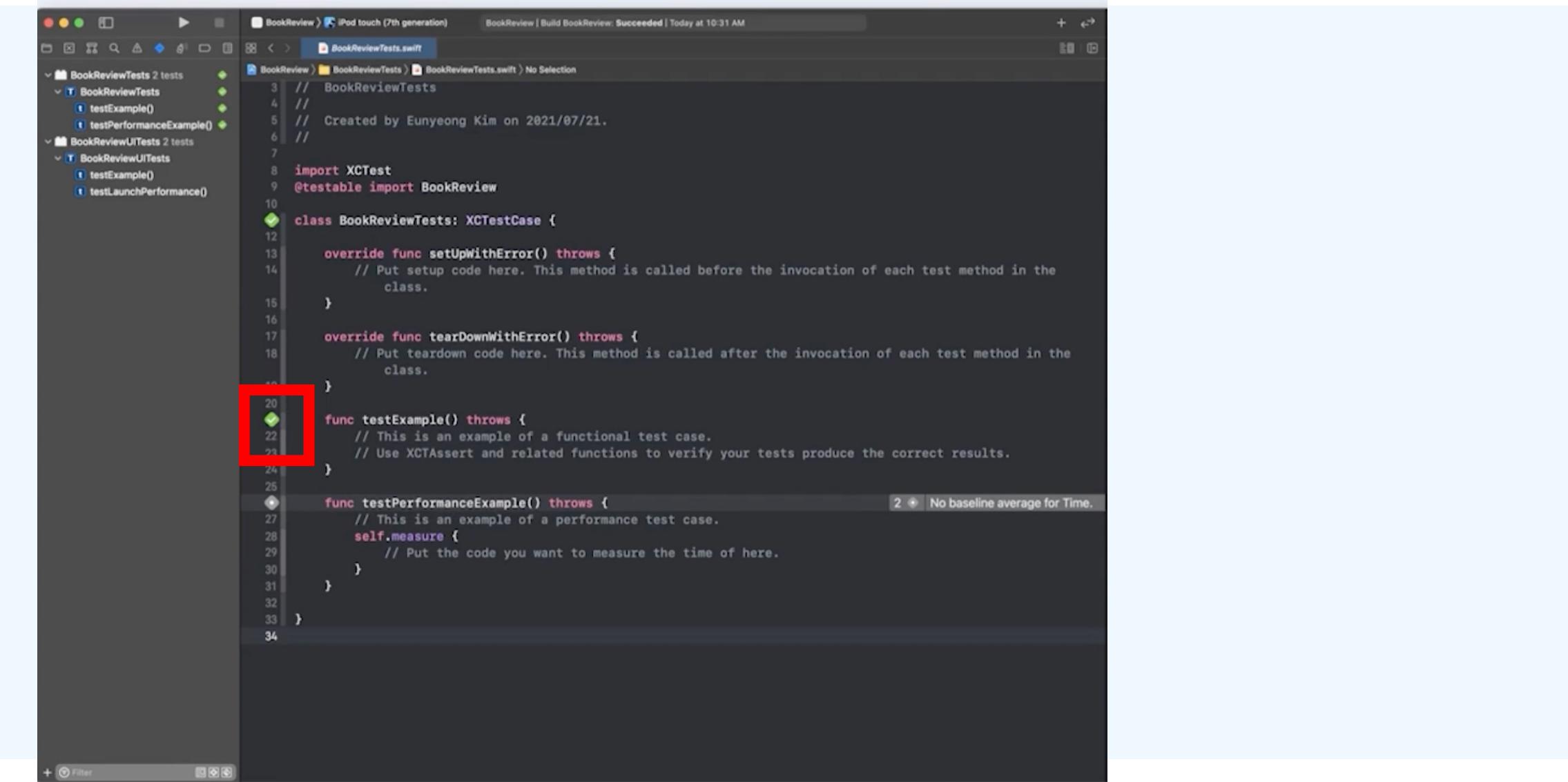
Name	Filters	Status
FirebaseFunctions	Always Used	Required
GTMSessionFetcherCore	Always Used	Required
GoogleSignIn	Always Used	Required
FirebaseFirestore	Always Used	Required
FirebaseAuth	Always Used	Required

Drag to reorder linked binaries

- > Target Dependencies (1 item)
- > Run Build Tool Plug-ins (0 items)
- > Compile Sources (1 item)
- > Copy Bundle Resources (0 items)

GTMSessionFetcherCore: <https://github.com/google/gtm-session-fetcher>

XCTest > UniTest



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows two test targets:
 - BookReviewTests:** Contains two tests: `testExample()` and `testPerformanceExample()`.
 - BookReviewUITests:** Contains two tests: `testExample()` and `testLaunchPerformance()`.
- Editor:** Displays the `BookReviewTests.swift` file content.

```
// BookReviewTests
// Created by Eunyeong Kim on 2021/07/21.

import XCTest
@testable import BookReview

class BookReviewTests: XCTestCase {

    override func setUpWithError() throws {
        // Put setup code here. This method is called before the invocation of each test method in the class.
    }

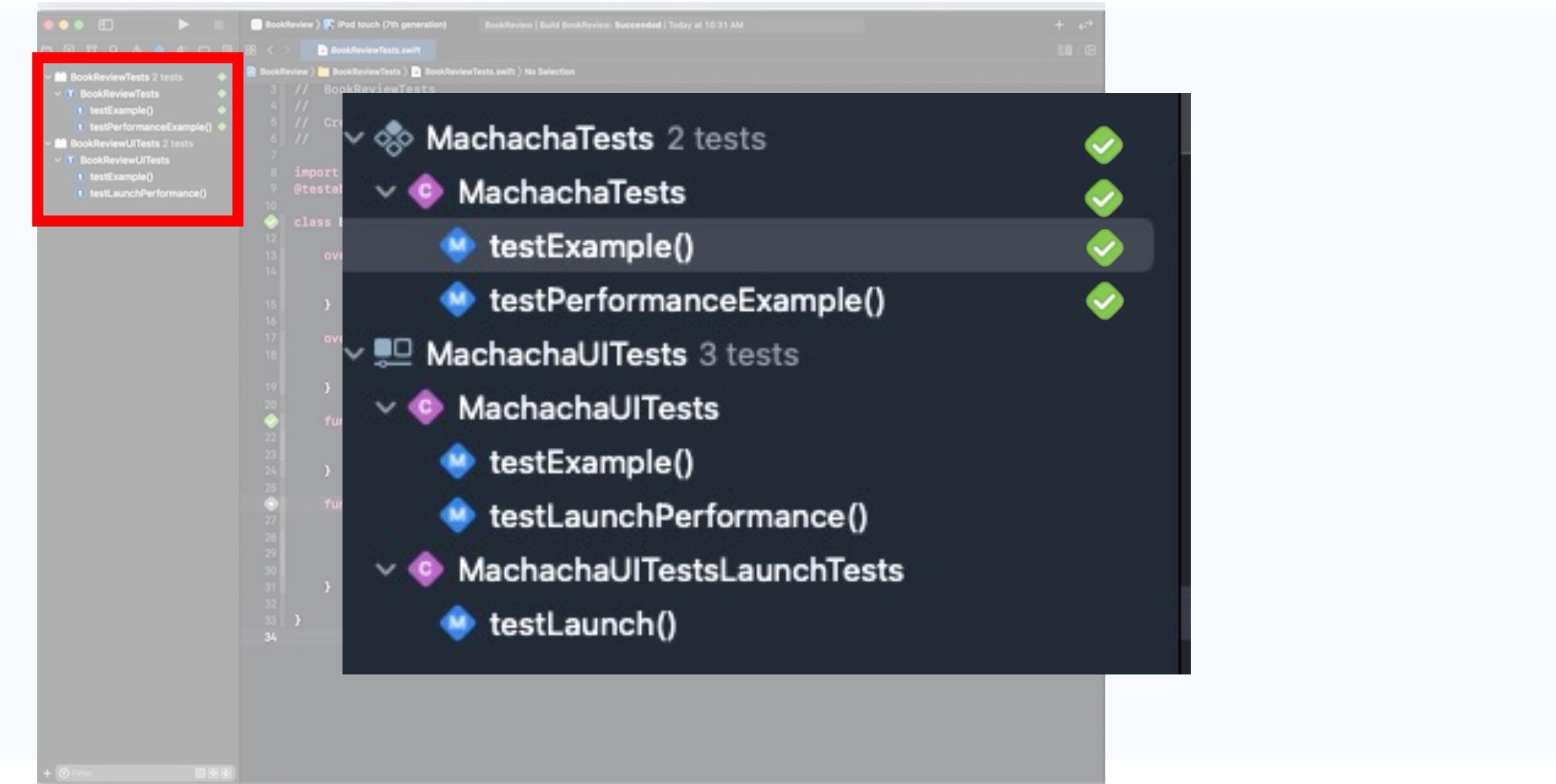
    override func tearDownWithError() throws {
        // Put teardown code here. This method is called after the invocation of each test method in the class.
    }

    func testExample() throws {
        // This is an example of a functional test case.
        // Use XCTAssert and related functions to verify your tests produce the correct results.
    }

    func testPerformanceExample() throws {
        // This is an example of a performance test case.
        self.measure {
            // Put the code you want to measure the time of here.
        }
    }
}
```
- Build Bar:** Shows the build status: `BookReview | Build BookReview: Succeeded | Today at 10:31 AM`.
- Bottom Bar:** Includes a "Filter" button and other standard Xcode navigation icons.

A red box highlights the `func testExample() throws {` line, indicating it is the current selection or the next step in the workflow.

XCTest > UniTest



```
import XCTest
@testable import Machacha

final class ProfileViewTests: XCTestCase {
    var sut: ProfileView! // 테스트 대상(일반적으로 sut이라 불린다), 테스트이기 때문에 ?보다는 !로 빠르게 버그를 찾는게 좋다
    var viewModel: ProfileViewModel!

    // 테스트가 실행될때마다 실행됨
    override func setUpWithError() throws {
        sut = ProfileView()
        viewModel = ProfileViewModel()
    }

    // 테스트 메소드들이 끝날때마다 실행
    override func tearDownWithError() throws {
        sut = nil
        viewModel = nil
    }

    // 임의의 test func을 작성
    func testExample() {
        let view = sut.environmentObject(viewModel)
        let body = view.body
        XCTAssertNil(body, "body가 존재하지 않음")
    }

    // 임의의 test func을 작성
    func ProfileViewModel의_fetchImage함수_Test() async {
        do {
            let result = try await viewModel.fetchImage(foodCartId: "InzqNwgl15TytWN0dIZz", imageName: "text.jpg")
            XCTAssertNotNil(result)
        } catch {
            XCTFail("이미지 불러오면서 에러발생, \(error)")
        }
    }
}
```

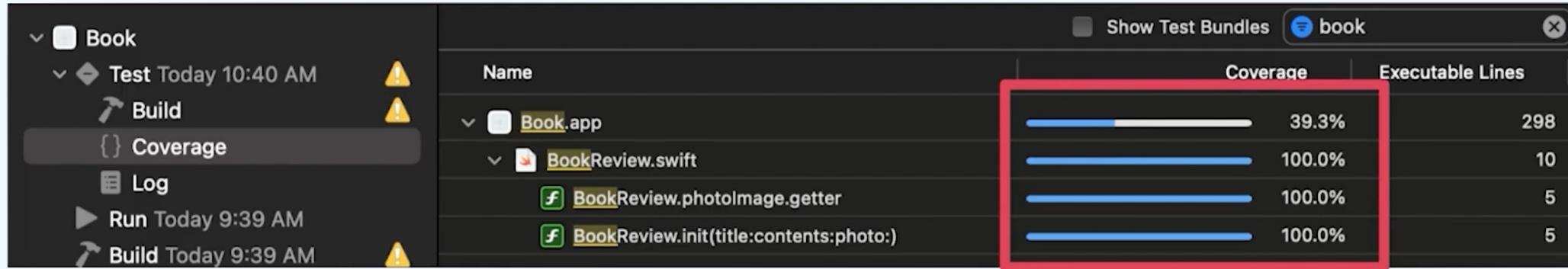
Test Coverage?

Xcode Project에서 몇 %의 코드에 대해서 테스트가 작성되어 있는지 나타낸 숫자

Test Coverage의 사용

App의 안정성을 확인하는 기준

XCTest > UniTest

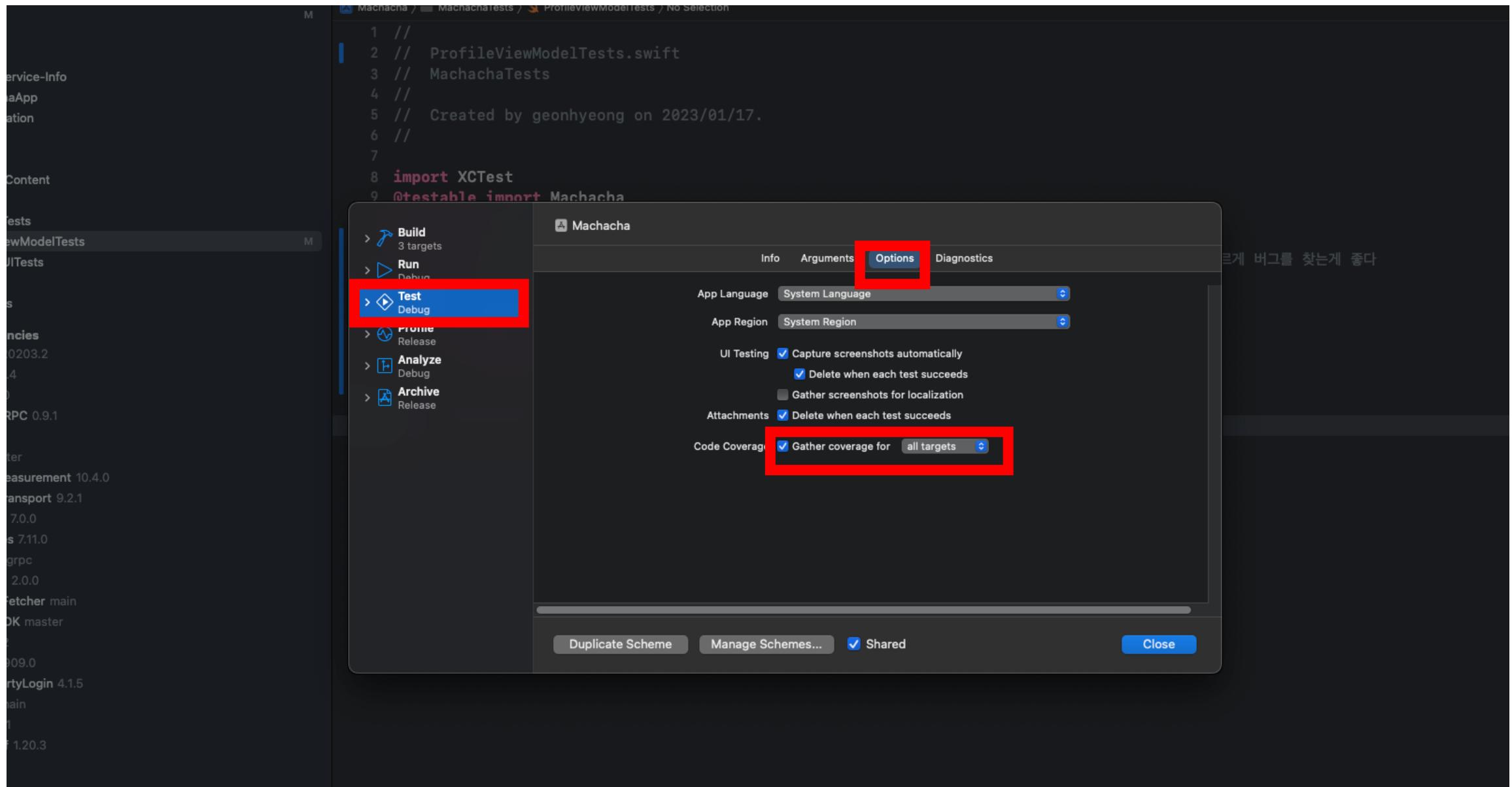


XCTest > UniTest

The screenshot shows the Xcode interface with a project named "Machacha". The current file is "ProfileViewModelTests.swift". A context menu is open over the first line of code, which defines the class "ProfileViewModelTests". The menu includes options like "Edit Scheme...", "New Scheme...", and "Manage Schemes...".

```
1 //  
2 // ProfileViewModelTests  
3 // MachachaTests  
4 //  
5 // Created by geonhyeong on 2023/01/17.  
6 //  
7  
8 import XCTest  
9 @testable import Machacha  
10  
◇ final class ProfileViewModelTests: XCTestCase {  
11     var sut: ProfileViewModel! // 테스트 대상(일반적으로 sut이라 불린다), 테스트이기 때문에 ?보다는 !로 빠르게 버그를 찾는게 좋다  
12  
13     // 테스트가 실행될때마다 실행됨  
14     override class func setUp() {  
15         super.setUp()  
16     }  
17 }  
18 }  
19 }  
20 }
```

XCTest > UniTest



XCTest > UniTest

The screenshot shows the Xcode interface for a project named "Machacha". The main window title is "Machacha main". The top navigation bar includes "Machacha" and "iPhone 14 Pro". The status bar indicates "Building | 520/2065" and "▲ 1".

The left sidebar displays a tree view of recent builds and tests. A red box highlights the "Coverage" item under the first "Build" entry.

The right pane shows a detailed build log for "Build Machacha - Log". The log tab is selected. The log table has columns: All, Recent, All Messages, All Issues, and Errors Only. The "All" column is currently active. The log lists numerous compilation tasks, mostly successful (green checkmarks), with some minor errors (grey question marks) and warnings (yellow exclamation marks). Examples of listed files include "Compile xds_cluster_manager.cc (x86_64)", "Compile xds_cluster_impl.cc (x86_64)", and "Compile typed_struct.upbdefs.c (x86_64)".

XCTest > UI Test

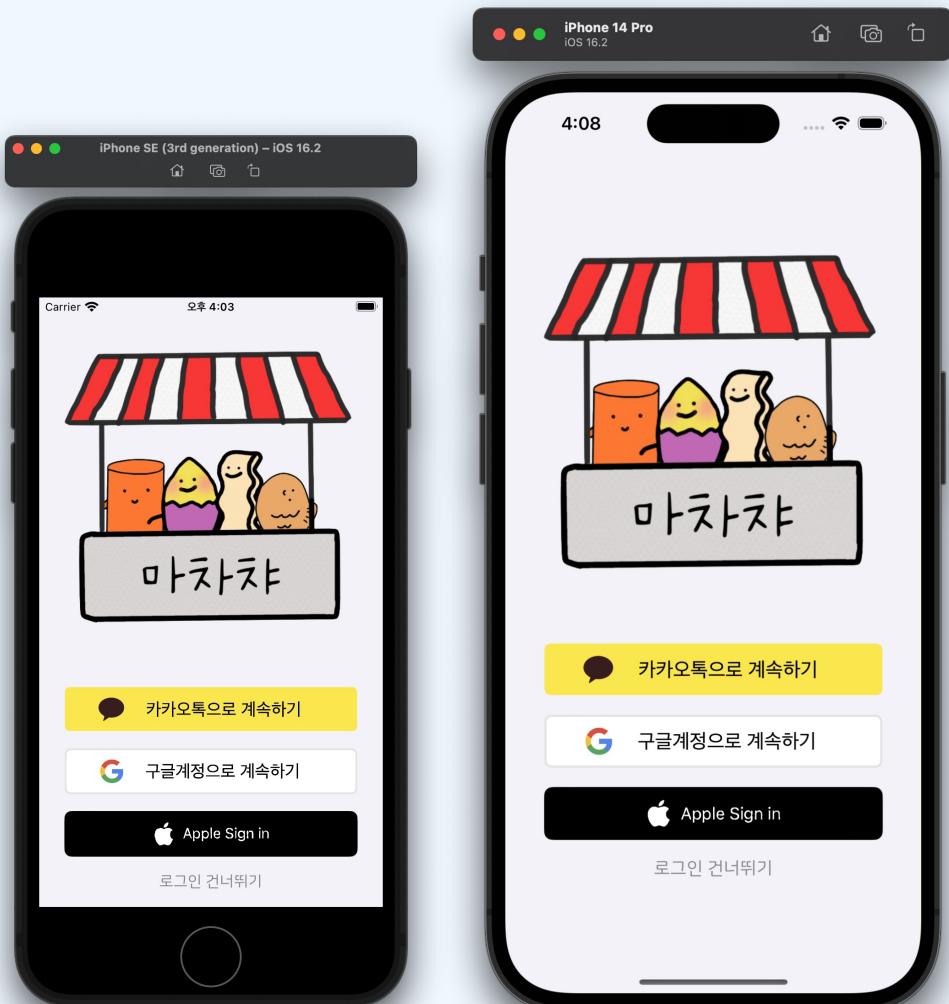


Class ProfileViewModel: XCTestCase {}



XCUIApplication().launch()

XCTest > UI Test



UI Test

UI Component의 표시와 동작이
의도한 대로 잘 작동되는지 확인하는 Test

- XCUIApplication().launch()

Class

XCUIApplication

A proxy that can launch, monitor, and terminate a test application.

(Xcode 7.0+)

Declaration

```
class XCUIApplication : XCUIElement
```

Overview

Use this class to launch, monitor, and terminate your app in a UI test. Use `wait(for:timeout:)` to launch your app and wait for it to reach an expected state before you check test conditions.

Topics

Creating an Application Proxy

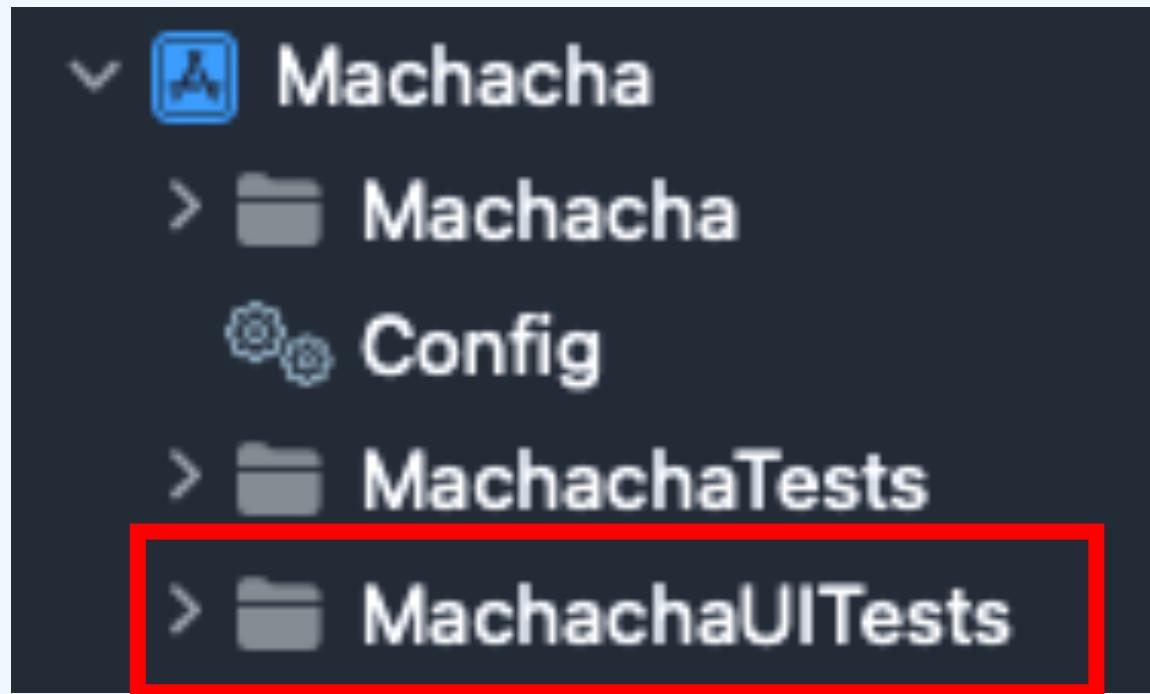
`init()`

Creates a proxy for the application as the *Target Application* in Xcode's target settings.

`init(bundleIdentifier: String)`

Creates a proxy for the application with the specified bundle identifier.

XCTest > UI Test



XCTest > UI Test

The screenshot shows the Xcode interface with the project navigation bar at the top. Below it is the project structure sidebar, which includes sections for Machacha (Info, GoogleService-Info, MachachaApp, Configuration, Sources, Assets, Preview Content, Config), MachachaTests (ProfileViewModelTests), MachachaUITests (MachachaUITests, MachachaUITestsLaunchTests), Products, Frameworks, and Package Dependencies. Under Package Dependencies, there is a list of various frameworks and their versions.

The main editor area displays the source code for `MachachaUITests.swift`. The code defines a `MachachaUITests` class that inherits from `XTestCase`. It includes implementations for `setUpWithError()`, `tearDownWithError()`, `testExample()`, and `testLaunchPerformance()`. The code uses Swift's `#available` directive to check for specific iOS versions and measure application launch metrics.

```
1 //  
2 // MachachaUITests.swift  
3 // MachachaUITests  
4 //  
5 // Created by geonhyeong on 2023/01/17.  
6 //  
7  
8 import XCTest  
9  
◇ final class MachachaUITests: XCTestCase {  
11  
12     override func setUpWithError() throws {  
13         // Put setup code here. This method is called before the invocation of each test method in the class.  
14  
15         // In UI tests it is usually best to stop immediately when a failure occurs.  
16         continueAfterFailure = false  
17  
18         // In UI tests it's important to set the initial state - such as interface orientation - required for your tests before they run. The setUp  
19     }  
20  
21     override func tearDownWithError() throws {  
22         // Put teardown code here. This method is called after the invocation of each test method in the class.  
23     }  
24  
◇ func testExample() throws {  
26     // UI tests must launch the application that they test.  
27     let app = XCUIApplication()  
28     app.launch()  
29  
30     // Use XCTAssert and related functions to verify your tests produce the correct results.  
31 }  
32  
◇ func testLaunchPerformance() throws {  
34     if #available(macOS 10.15, iOS 13.0, tvOS 13.0, watchOS 7.0, *) {  
35         // This measures how long it takes to launch your application.  
36         measure(metrics: [XCTApplicationLaunchMetric()]) {  
37             XCUIApplication().launch()  
38         }  
39     }  
40 }  
41  
42 }
```

```
8 import XCTest
9
10 ◇ final class MachachaUITests: XCTestCase {
11     var app: XCUIApplication! // 1. UniTest에서 sut으로 표시되었던 테스트 대상
12
13     // 테스트가 실행될때마다 실행됨
14     override func setUp() {
15         super.setUp()
16
17         continueAfterFailure = false // 2. 실패해도 계속 동작할 것인지? false(하나라도 실패가 생기면 종료)
18
19         app = XCUIApplication() // 3. XCUIApplication을 대입후,
20         app.launch() // 4. 앱을 런치 시켜준다
21     }
22
23     override func tearDown() {
24         super.tearDown()
25
26         app = nil // 5. 앱을 nil로 초기화
27     }
28
29     // 6. test할 화면을 함수로 만들기
30     ◇ func test_navigation의_Title이_즐겨찾기로_설정되어있다() {
31         let existNavigationBar = app.navigationBars["즐겨찾기"].exists // 7. NavigationBar의 제목이 즐겨찾기가 존재하는지 확인
32         XCTAssertTrue(existNavigationBar)
33     }
34 }
35
```

```
final class MachachaUITests: XCTestCase {
    override func setUp() {
        app = XCUIApplication() // 3. XCUIApplication을 대입후,
        app.launch() // 4. 앱을 런치 시켜준다
    }

    override func tearDown() {
        super.tearDown()

        app = nil // 5. 앱을 nil로 초기화
    }

    // 6. test할 화면을 함수로 만들기
    func test_navigation의_Title이_즐겨찾기로_설정되어있다() {
        let existNavigationBar = app.navigationBars["즐겨찾기"].exists // 7. NavigationBar의 제목이 즐겨찾기가 존재하는지 확인
        XCTAssertTrue(existNavigationBar)
    }
}
```

iPhone 14 Pro
4:08
카카오톡으로 계속하기
구글계정으로 계속하기
Apple Sign In
로그인 건너뛰기

```
2023-02-02 16:58:55.095040+0900 MachachaUITests-Runner[44693:29165669] [SceneConfiguration] Info.plist contained no UIScene configuration dictionary (looking for configuration named "(no name)")
2023-02-02 16:58:55.103740+0900 MachachaUITests-Runner[44693:29165669] Running tests...
Test Suite 'MachachaUITests' started at 2023-02-02 16:59:01.459
Test Case '-[MachachaUITests.MachachaUITests test_navigation의_Title이_즐겨찾기로_설정되어있다]' started.
t = 0.00s Start Test at 2023-02-02 16:59:01.463
t = 0.18s Set Up
t = 0.19s Open lionlike.project.Machacha
t = 0.26s Launch lionlike.project.Machacha
t = 10.61s Setting up automation session
t = 26.68s Wait for lionlike.project.Machacha to idle
t = 28.02s Checking existence of ` 즐겨찾기` NavigationBar
/Users/geonhyeong/Desktop/finalproject-machacha/src/frontend/user/MachachaUITests/MachachaUITests.swift:32: error: -[MachachaUITests.MachachaUITests test_navigation의 Title이_즐겨찾기로_설정되어있다] : XCTAssertTrue failed
t = 28.76s Tear Down
Test Case '-[MachachaUITests.MachachaUITests test_navigation의_Title이_즐겨찾기로_설정되어있다]' failed (28.971 seconds).
Test Suite 'MachachaUITests' failed at 2023-02-02 16:59:30.433.
Executed 1 test, with 1 failure (0 unexpected) in 28.971 (28.974) seconds
```

```
◆ final class MachachaUITests: XCTestCase {
23     override func tearDown() {
26         app = nil // 5. 앱을 nil로 초기화
27     }
28
29     // 6. test할 화면을 함수로 만들기
30 //    func test_navigation의_Title이_즐겨찾기로_설정되어있다() {
31 //        let existNavigationBar = app.navigationBars["즐겨찾기"].exists // 7. NavigationBar의 제목이 즐겨찾기가 존재하는지 확인
32 //        XCTAssertTrue(existNavigationBar)
33 //    }
34 //
35     // 7. 손쉽게 작성
◆     func test_test() {
37
38
39     }
40 }
41
```

```
✖ final class MachachaUITests: XCTestCase {
23     override func tearDown() {
26         app = nil // 5. 앱을 nil로 초기화
27     }
28
29     // 6. test할 화면을 함수로 만들기
30 //    func test_navigation의_Title이_즐겨찾기로_설정되어있다() {
31 //        let existNavigationBar = app.navigationBars["즐겨찾기"].exists // 7. NavigationBar의 제목이 즐겨찾기가 존재하는지 확인
32 //        XCTAssertTrue(existNavigationBar)
33 //    }
34 //
35
36     // 7. 녹화버튼을 눌러 손쉽게 작성
37 ◇ func test_test() {
38     let app = XCUIApplication()
39     app.buttons["로그인 건너뛰기"].tap()
40     app.buttons["내정보"].tap()
41     app.scrollViews.otherElements.buttons["1, 즐겨찾기"].tap()
42 }
43 }
44 |
```

마무리

BDD(Behavior Driven Develop, 시나리오를 기반으로 테스트 케이스를 작성하는 방법)
를 따르는 UniTest/UITest 작성해보기

THANK
YOU