Testing the UI of iOS apps using XCTest and XCUITest in Swift

Overview

In this tutorial, we will "create" some UI-based tests for an iOS application using XCTest testing framework along with XCUITest.

Requirements

- Xcode
- Xcode command line tools
- ^o xcode-select --install
- xcpretty
- (optional) fastlane and trainer plugin

Description

For this tutorial, we'll use a sample iOS app with UI tests by Shashikant, with minor updates as tracked in this fork.

The iOS application is quite simple: it has just a button and a text that appears whenever clicking on it.



The application has one View Controller class.

XCUITest101/ViewController.swift

```
11
// ViewController.swift
// XCUITest101
11
// Created by Shashikant Jagtap on 24/09/2018.
// Copyright © 2018 Shashikant Jagtap. All rights reserved.
11
import UIKit
class ViewController: UIViewController {
   @IBOutlet weak var welcomeText: UILabel!
   @IBAction func enterPressed(_ sender: Any) {
       welcomeText.text = "Welcome to XCUITest"
       welcomeText.isHidden = false
   }
   override func viewDidLoad() {
       super.viewDidLoad()
       welcomeText.isHidden = true
       // Do any additional setup after loading the view, typically from a nib.
    }
}
```

The project contains two tests implemented in Swift: one named "testRecorded" (which is not an actual test case) and another, the real one, named "testRefactored".

Tests use XCTest framework and XCUITest to load the application and execute the UI-baed tests.

XCUITest101UITests/XCUITest101UITests.swift

```
11
// XCUITest101UITests.swift
// XCUITest101UITests
11
// Created by Shashikant Jagtap on 24/09/2018.
// Copyright © 2018 Shashikant Jagtap. All rights reserved.
11
import XCTest
class XCUITest101UITests: XCTestCase {
   override func setUp() {
       super.setUp()
       continueAfterFailure = false
       XCUIApplication().launch()
    }
    override func tearDown() {
       super.tearDown()
    }
    func testRecorded() {
       // this is not an actual test...
       let app = XCUIApplication()
       app.otherElements.containing(.image, identifier:"wall1").element.tap()
       app.buttons["enter"].tap()
       app.staticTexts["Welcome to XCUITest"].tap()
    }
    func testRefactored() {
       let app = XCUIApplication()
       app.buttons["enter"].tap()
       XCTAssert(app.staticTexts["Welcome to XCUITest"].exists)
    }
}
```

In order to run the tests from the command line or during CI, you can use xcodebuild. By processing its output using xcpretty, a JUnit XML can be generated (build/reports/junit.xml, by default).

xcodebuild -project XCUITest101.xcodeproj/ -scheme XCUITest101 -destination 'platform=iOS Simulator,OS=13.1, name=iPhone 11 Pro Max' clean build test CODE_SIGN_IDENTITY="" CODE_SIGNING_REQUIRED=NO | xcpretty -r junit

| 🗯 Xcode File Edit View Find | l Navigate Editor | Product Debug S | Source C | ontrol Window Help | 🔄 🥖 📮 🎽 🔮 | 3 * 💷 <> | | |
|--|---|---|--|--|-----------|----------|--|--|
| 🔴 🔴 🌔 下 🔲 🚧 XCUITest101 |) 📻 iPhone 11 Pro Max | Run Test | 業R 業U | on iPhone 11 Pro Max | 🔺 1 🙁 1 | | | |
| KOUTIESTIC XCUITESTIONUTESTS XCUITESTIONUTESTS TXCUITESTIONUTESTS testRecorded() testRefactored() | Image: Constraint of the system Image: Constraint of the system 1 // 2 // XCUITest 3 // XCUITest 4 // Created 6 // Copyright 7 // B 9 import XCTes 10 Class XCUITe 12 Override | Profile Analyze Archive Build For Perform Action Build Clean Build Folder Stop Scheme Destination Test Plan | #0 #I | 2UlTest101UlTests.swift) C XCUlTest101UlTests | Tests | | | |
| | 15 cont 16 XCUIA 17 } 18 19 19 override 20 super 21 } 22 func test 23 func test 24 let a 25 app.c 26 app.t 28 } 29 func test 31 let a 32 app.t 33 XCTAs 34 } 35 > | Create Bot Application().launcl func tearDown() { c.tearDown() Recorded() { app = XCUIApplication therElements.conta puttons["enter"].tag staticTexts["Welcom Refactored() { app = XCUIApplication suttons["enter"].tag suttons["en | Create Bot plication().launch() unc tearDown() { tearDown() scorded() { p = XCUIApplication() herElements.containing(.image, identifier:"wall1").element.tap() | | | | | |

However, by default, this won't produce a JUnit XML report by itself which can, later on, be uploaded to Xray.

After running the tests and generating the JUnit XML report (e.g., junit.xml), it can be imported to Xray (either by the REST API or by using one of the CI plugins or through **Import Execution Results** action within the Test Execution).

curl -H "Content-Type: multipart/form-data" -u admin:admin -F "file=@build/reports/junit.xml" http://jiraserver. example.com/rest/raven/1.0/import/execution/junit?projectKey=CALC

A Test Execution will be created containing information about the executed scenarios.

| E E | xecution | results - | - junit.x | ml - [1572 | 2141445434 | 4] | | | | | |
|-------------|----------------|----------------|-----------|--------------|--------------|---------|--------------|--------------|---------------------------|--------------|-------|
| 🖋 Edit | Commer | lt Assign | More - | Close Issue | Reopen Issue | Admin + | | | | | |
| Details | | | | | | | | | | | |
| Type: | | Test Execu | ition | | | St | atus: | RESOLVED | ew Workflow) | | |
| Affects Ve | rsion/s: | None | | | | R | esolution: | Fixed | | | |
| Componer | nt/s: | None | | | | Fi | x Version/s: | None | | | |
| Labels: | | None | | | | | | | | | |
| Test Envir | onments: | None | | | | | | | | | |
| Test Plan: | | None | | | | | | | | | |
| Descriptior | 1 | | | | | | | | | | |
| Execution | results import | ed from extern | al source | | | | | | | | |
| Tests | | | | | | | | | | | |
| | | | | | | | | | | • Add | |
| | | | | | | | | | | • Add • | |
| Overall E | ecution Stat | JS | | | | _ | | | | | |
| 4 | 4 | | | | | | | | | | |
| PASS | FAIL | | | | | | | | | | |
| | | | | | | | | | | | |
| TOTAL TES | STS: 2 | | | | | | | | | | |
| Ŧ | Filter(s) | | | | | | | | | | |
| | Apply Rept | | | | | | | | Chan (100 A) contribution | Calura | |
| - <i>1</i> | Apply ridlik | | | | | | | | Snow 100 Pentries | Columns | • |
| | Rank | Key | \$ | Summary | Test Type | #Ree | l #Def | Assignee | Status | | |
| | 2 | CALC-513 | 8 te | stRecorded | Generic | 0 | 0 | Administrato | FAIL | | ► ••• |
| | 1 | CALC-513 | 9 te | stRefactored | Generic | 0 | 0 | Administrato | PASS | | • ••• |

Each test is mapped to a Generic Test in Jira, and the Generic Test Definition field contains the name of the class concatenated with the method name of the corresponding automated test.

The Execution Details of the Generic Test contains information about the "Test Suite" (as per JUnit format), which in this case corresponds to the fullyqualified name of the class holding the test.

| Calculator / Test Execution: CALC-5151 / Test: CALC-5139 testRefactored | | | Export Test as Text | Return to Test Execution Next |
|--|-----------------------|---|--|--|
| Execution Status PASS and PASS And PASS Starfed On: 27/Oct/19 1:57 AM | | | Assignee: Executed By: Tests environments: | Administrator Versions: Administrator Revision: |
| Comment Preview Comment | Execution Defects (0) | Create Defect Create Sub-Task Add Defects | Execution Evidence (0) | Add Evidence |
| Test Description None Test Details Test Type: Generic Definition: XCUTTest101UTTests.XCUTTest101UTTests.testRefactored | | | | |
| Results | | | | |
| Context | Error Message | | Dur | ation Status |
| TestSuite XCUITest101UITests.XCUITest101UITests | - | | | 3 sec PASS |
| Activity | | | | |

You should be able to use fastlane (docs here) to build and run your tests by using the trainer plugin.

In that case, you need to define a lane to run the tests and invoke the trainer plugin.

fastlane/Fastfile

```
default_platform(:ios)
platform :ios do
  desc "Run tests"
  lane :test do
    scan(scheme: "XCUITest101",
        output_types: "",
        fail_build: false)
        trainer(output_directory: "build/reports/")
    end
end
```

Notes

- xcpretty project seems to be not very active
- xcpretty has a known limitation that inhibits the processing of multiline error descriptions (only the header is imported as the log/output of the assertion)

References

- https://developer.apple.com/documentation/xctest
- https://developer.apple.com/library/archive/documentation/DeveloperTools/Conceptual/testing_with_xcode/chapters/09-ui_testing.html https://github.com/xcpretty/xcpretty
- https://developer.apple.com/library/archive/documentation/DeveloperTools/Conceptual/testing_with_xcode/chapters/08-automation.html# //apple_ref/doc/uid/TP40014132-CH7-SW3
- https://www.slideshare.net/ShankarAnamalla/ios-app-testing-with-xctest-and-xcuitest
- https://github.com/zanizrules/fastlane-plugin-xcresult_to_junit