

Agile Swift

Godfrey Nolan

RIIS LLC

Agenda

- Unit Testing intro
- Up and Running
- Swift Flavors
- Unit testing 101
- Tools of the trade
- Mocking
- User Interface testing

Unit Testing Intro

- Hello World
- Benefits
- Testing Pyramid
- UI Testing

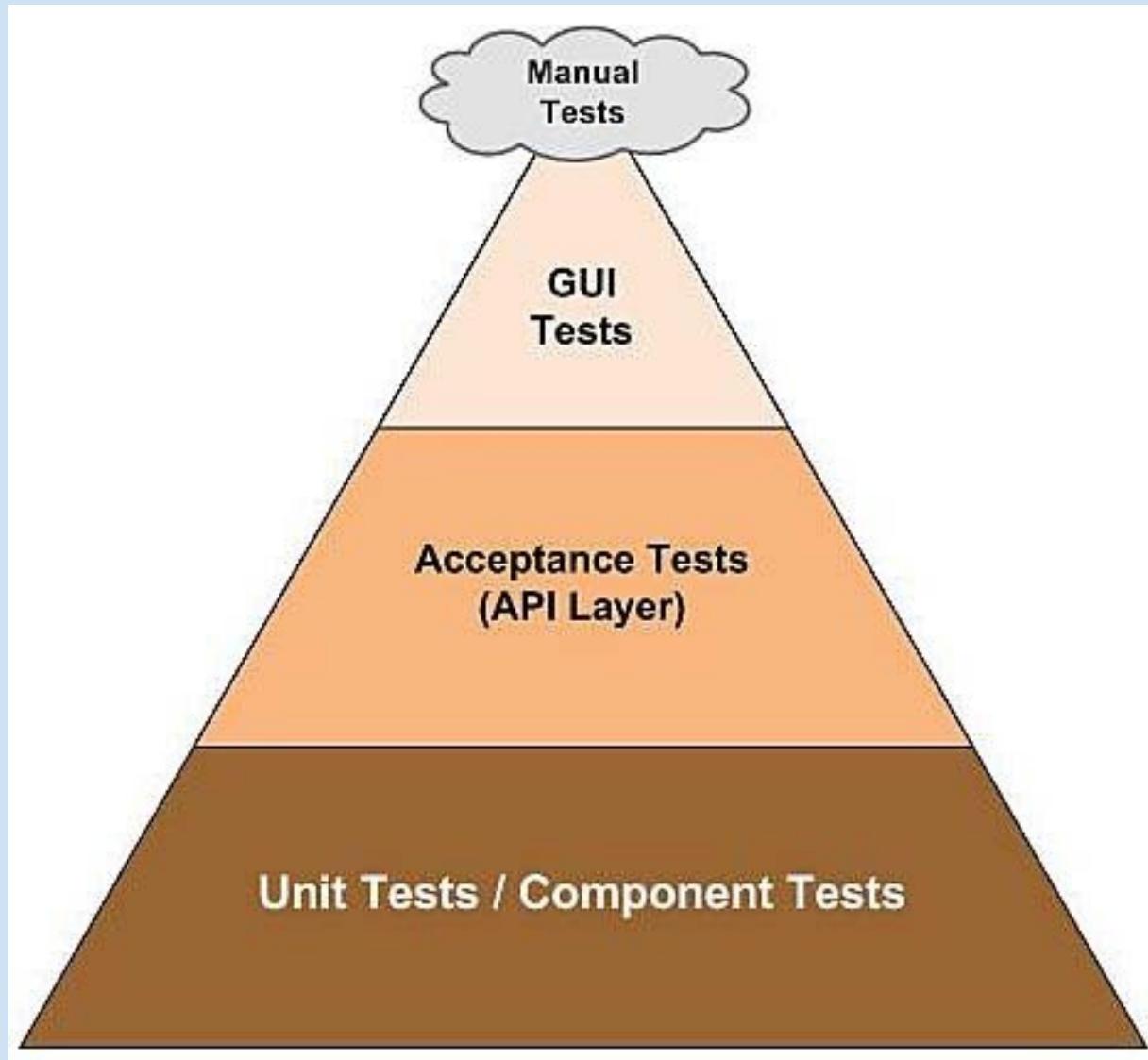
Unit Testing Intro

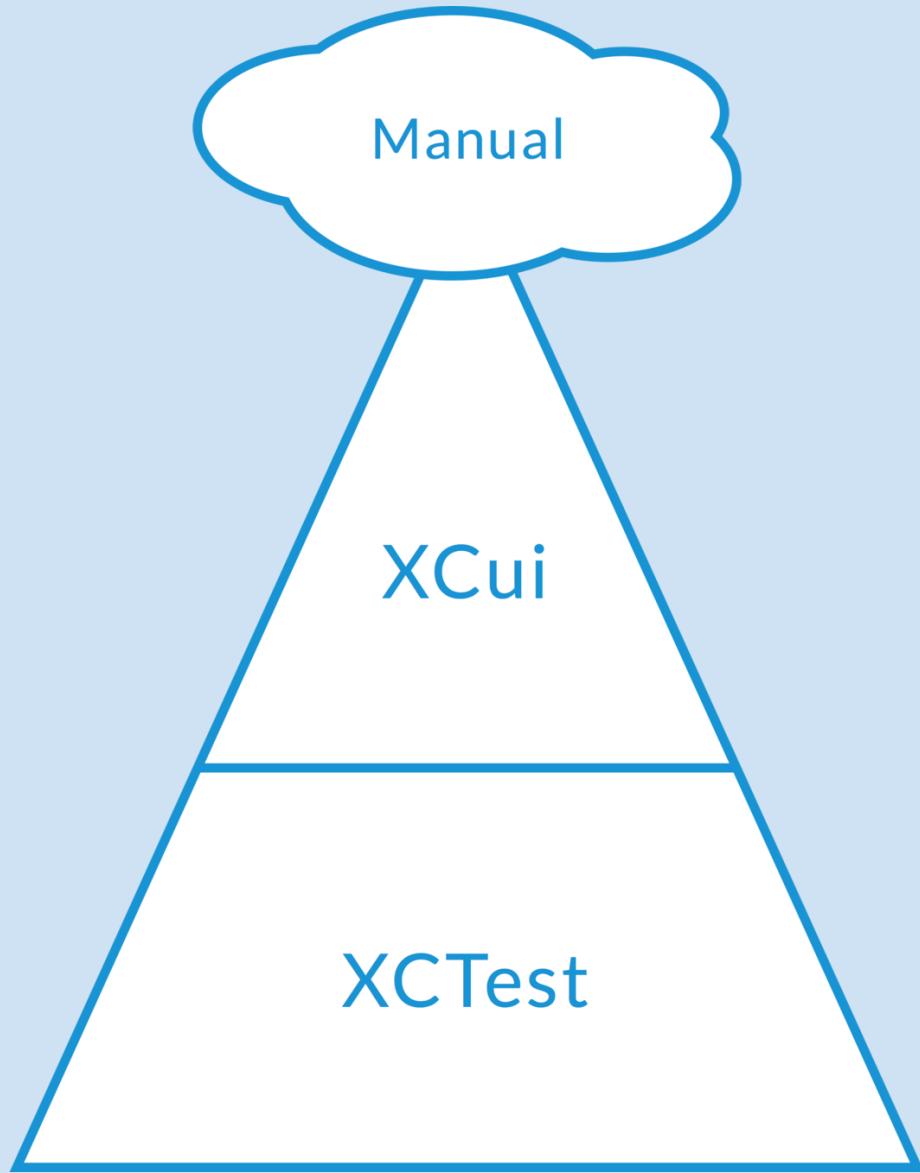
```
struct HelloWorld {  
  
    var text = "Hello, World!"  
  
}
```

```
import XCTest  
@testable import HelloWorld  
  
class HelloWorldTests: XCTestCase {  
  
    func testExample() {  
        XCTAssertEqual(HelloWorld().text, "Hello, World!")  
    }  
  
}
```

Unit Testing Benefits

- Catch more mistakes
- Confidently make more changes
- Built in regression testing
- Extend the life of your codebase





Swift 3.x Flavors

- MacOS
 - Xcode, command line and repl
- Linux
 - Ubuntu 14.04 and 16.04
 - Command line and repl
 - No GUI libraries
- Swift Playground on iPad
- More flavors to come?

Interactive Swift

- Good language learning environment
- Options
 - Playgrounds
 - Repl

A screenshot of an Xcode playground window titled "MyPlayground". The top pane shows the following Swift code:

```
//: Playground - noun: a place where people can play
import UIKit
var str = "Hello, playground"
print ("Hello, World!")
```

The bottom pane displays the output of the code execution:

```
Hello, playground
Hello, World!\n
```

A screenshot of a terminal window titled "codemash" running on a Mac OS X system. The prompt is "RIIS-MBPRO-13:codemash User\$ swift". The session shows the following interaction:

```
Welcome to Apple Swift version 3.0.2 (swiftlang-800.0.63 clang-800.0.42.1).
Type :help for assistance.
1> print ("Hello, World!")
Hello, World!
2> 3+4
$R0: Int = 7
3>
```

Command Line Swift

- Headless Swift
- First class cousin of Xcode Swift
- Package Manager
 - Demo swift package init
- Primary option on Linux
 - Docker
 - AWS Free tier
- Also works on MacOS

Package Manager

```
[RIIS-MBPRO-13:HelloWorld User$ swift package init
Creating library package: HelloWorld
Creating Package.swift
Creating .gitignore
Creating Sources/
Creating Sources/HelloWorld.swift
Creating Tests/
Creating Tests/LinuxMain.swift
Creating Tests>HelloWorldTests/
Creating Tests>HelloWorldTests>HelloWorldTests.swift
RIIS-MBPRO-13:HelloWorld User$ ]
```

Package.swift

```
import PackageDescription

let package = Package(
    name: "HelloWorld"
)
```

```
import PackageDescription

let package = Package(
    name: "DeckOfPlayingCards",
    dependencies: [
        .Package(url: "https://github.com/apple/example-package-fisheryates.git", majorVersion: 2),
        .Package(url: "https://github.com/apple/example-package-playingcard.git", majorVersion: 3),
    ]
)
```

LinuxMain.swift

```
import XCTest
@testable import HelloWorldTests

XCTMain([
    testCase(HelloWorldTests.allTests),
])
```

HelloWorldTests.swift

```
import XCTest
@testable import HelloWorld

class HelloWorldTests: XCTestCase {

    static var allTests : [(String, (HelloWorldTests) -> () throws -> Void)] {
        return [
            ("testExample", testExample),
        ]
    }

    func testExample() {
        XCTAssertEqual(HelloWorld().text, "Hello, World!")
    }

}
```

```
struct HelloWorld {

    var text = "Hello, World!"
}
```

Swift 3.x in Xcode

- More options
 - Playgrounds
 - **Xcode 8.x IDE**

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure for "Calculator".
 - Calculator:** Contains `AppDelegate.swift`, `ViewController.swift`, `Main.storyboard`, `Assets.xcassets`, `LaunchScreen.storyboard`, and `Info.plist`.
 - CalculatorModel.swift:** Selected file, highlighted with a blue background.
 - CalculatorTests:** Contains `CalculatorTests.swift` and `Info.plist`.
 - CalculatorUITests:**
 - Products:**
- Editor:** Displays the code for `CalculatorModel.swift`. The code defines a class `CalculatorModel` with four methods: `add`, `sub`, `mul`, and `div`. It includes comments indicating it was created on 12/4/16 and has copyright © 2016 example. All rights reserved.
- Build Bar:** Shows the build status as "Succeeded" at 12/4/16 at 12:55 PM.
- NotificationCenter:** Shows one warning icon.

```
//
// CalculatorModel.swift
// Calculator
//
// Created by User on 12/4/16.
// Copyright © 2016 example. All rights reserved.

import Foundation

class CalculatorModel {
    var a: Int!
    var b: Int!

    func add(_ a:Int, _ b:Int) -> Int {
        return a + b
    }

    func sub(_ a:Int, _ b:Int) -> Int {
        return a - b
    }

    func mul(_ a:Int, _ b:Int) -> Int {
        return a * b
    }

    func div(_ a:Int, _ b:Int) -> Int {
        guard b != 0 else {
            return 0
        }
        return a / b
    }
}
```

The screenshot shows a Xcode editor window with the title "CalculatorTests.swift". The file path in the title bar is "Calculator > CalculatorTests > CalculatorTests.swift". The status bar at the bottom right shows "No Selection" and three small icons. The code editor displays the following Swift test code:

```
1 //  
2 //  CalculatorTests.swift  
3 //  CalculatorTests  
4 //  
5 // Created by User on 12/4/16.  
6 // Copyright © 2016 example. All rights reserved.  
7 //  
8  
9 import XCTest  
10 @testable import Calculator  
11  
◇12 class CalculatorTests: XCTestCase {  
13  
14     var resCalc : CalculatorModel!  
15  
16     override func setUp() {  
17         super.setUp()  
18         resCalc = CalculatorModel()  
19     }  
20  
◇21     func testAdd() {  
22         XCTAssertEqual(resCalc.add(1, 1), 2)  
23         XCTAssertEqual(resCalc.add(1, 2), 3)  
24         XCTAssertEqual(resCalc.add(5, 4), 9)  
25     }  
26  
27 }  
28
```

Unit Testing 101

- Setup and Teardown
- Performance
- Assertions
- Code Coverage

Unit Testing 101

Choose options for your new project:

Product Name: Calculator

Organization Name:

Organization Identifier:

Bundle Identifier: com.riis.Calculator

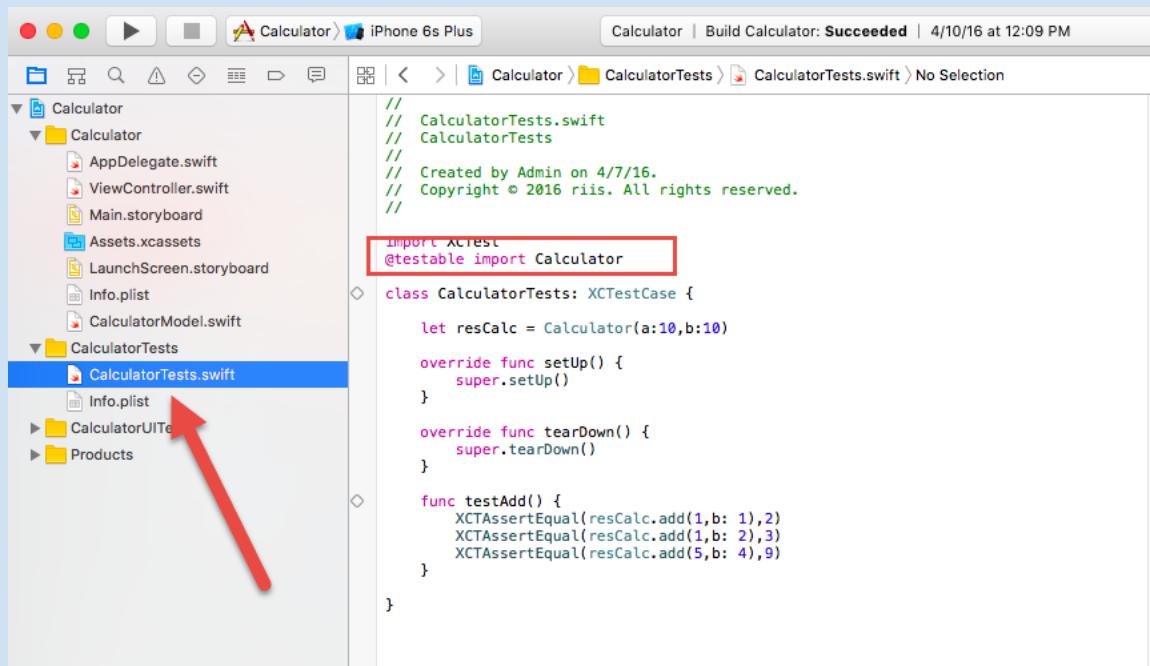
Language: Swift

Devices: iPhone

Use Core Data
 Include Unit Tests
 Include UI Tests

Cancel Previous Next

Unit Testing 101



A screenshot of the Xcode interface showing a project named "Calculator". The project navigator on the left shows files under "Calculator" and "CalculatorTests". A red arrow points from the text "CalculatorTests.swift" in the slide's heading down to the "CalculatorTests.swift" file in the project navigator. The code editor on the right displays the contents of "CalculatorTests.swift".

```
// CalculatorTests.swift
// CalculatorTests
//
// Created by Admin on 4/7/16.
// Copyright © 2016 riis. All rights reserved.

import XCTest
@testable import Calculator

class CalculatorTests: XCTestCase {

    let resCalc = Calculator(a:10,b:10)

    override func setUp() {
        super.setUp()
    }

    override func tearDown() {
        super.tearDown()
    }

    func testAdd() {
        XCTAssertEqual(resCalc.add(1,b: 1),2)
        XCTAssertEqual(resCalc.add(1,b: 2),3)
        XCTAssertEqual(resCalc.add(5,b: 4),9)
    }
}
```

Unit Testing 101

The screenshot shows the Xcode interface with a project named "BasicCalculator" for an iPhone 6s Plus. The build status is "Succeeded" at 8:31 PM.

The left sidebar shows the test navigator for "BasicCalculatorTests" with three tests: "testAdd()", "testExample()", and "testPerformanceExample()".

The main editor area contains the following Swift code:

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

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

func testAdd() {
    XCTAssertEqual(resCalc.add(1, 1), 2)
    XCTAssertEqual(resCalc.add(1, 2), 3)
    XCTAssertEqual(resCalc.add(5, 4), 9)
}

func testExample() {
    // This is an example of a functional test case.
    XCTAssertTrue(true, "Pass")
}

func testPerformanceExample() {
    self.measureBlock() {
        XCTAssertEqual(self.resCalc.add(1, 2), 3)
    }
}
```

A red box highlights the `func testPerformanceExample()` block.

A "Performance Result" callout is shown for the highlighted block, indicating:

- Metric: Time
- Result: No Baseline
- Average: 0.000s
- Baseline: No Baseline
- Max STDDEV: 10.000%

The callout also shows a histogram with a single bar at index 1, labeled "0.000 sec (296% STDEV)".

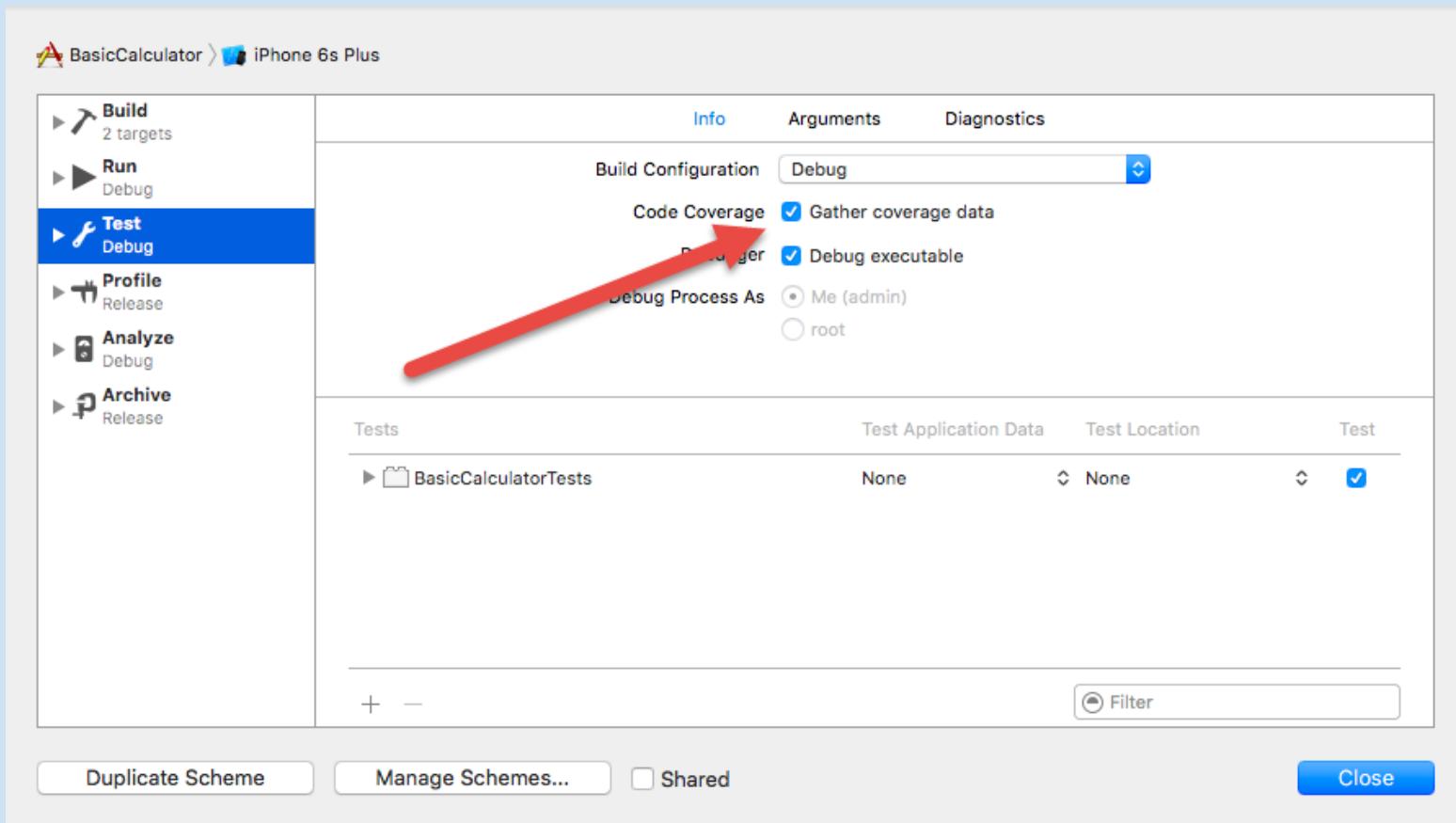
The bottom right corner displays the test session log:

```
Test session log:
/Users/admin/Library/Developer/Xcode/Logs/Test-BasicCalculator-9D-8E72-C21D58C8964D/
Session-2016-04-18
Value: 0.00 (329.35%) 20:31:12.885.
Test Suite 'BasicCalculator' passed at 2016-04-18 20:31:12.886.
Executed 1 test, with 0 failures (0 unexpected) in 0.283 (0.285) seconds
Test Suite 'Selected tests' passed at 2016-04-18 20:31:12.886.
Executed 1 test, with 0 failures (0 unexpected) in 0.283 (0.288) seconds
```

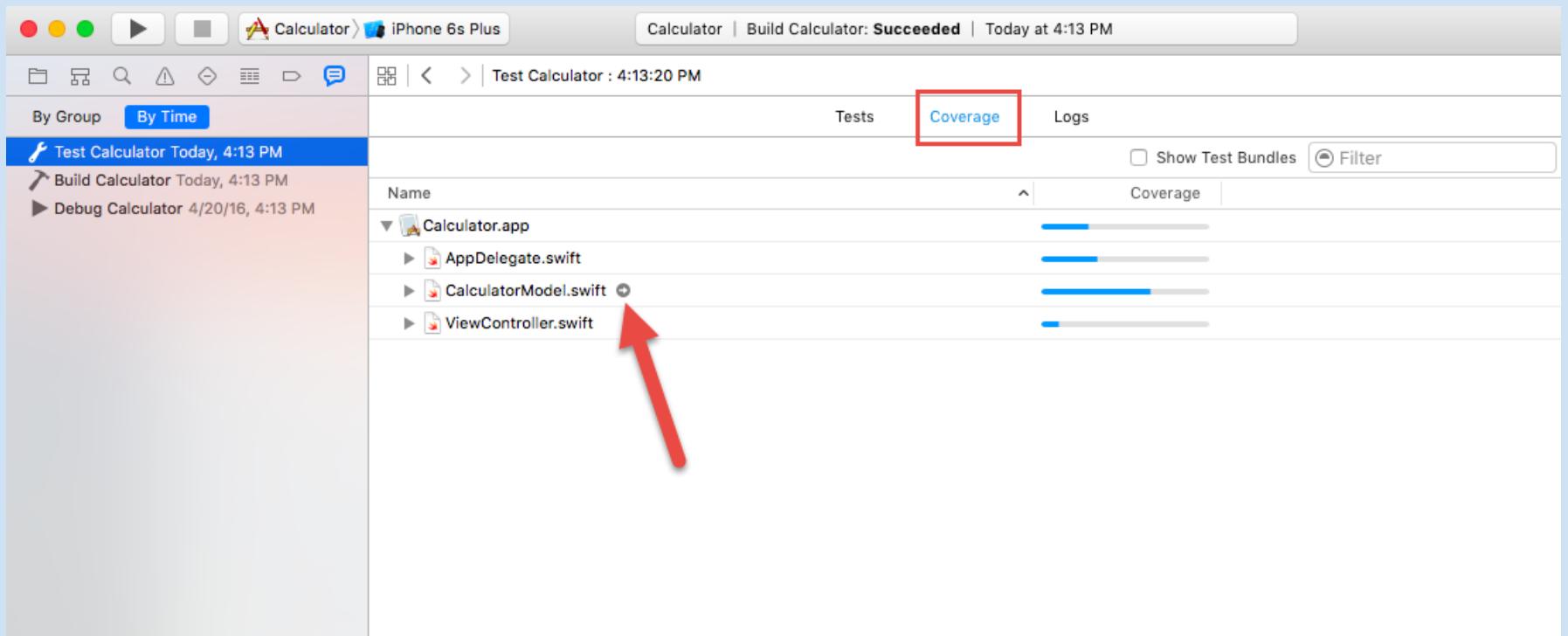
Unit Testing 101

Assertion	Description
XCTAssert	Tests that two values are the same
XCTAssertEqual	Tests that two values are equal
XCTAssertEqualWithAccuracy	Tests that two floating point values (a,b) are equal within a tolerance of c
XCTAssertFalse	Tests if a Boolean condition is false
XCTAssertTrue	Tests if a Boolean condition is true
XCTAssertGreaterThan	Tests that one value is greater than the other
XCTAssertGreaterThanOrEqual	Tests that one value is greater or equal to the other
XCTAssertLessThan	Tests that one value is less than the other
XCTAssertLessThanOrEqual	Tests that one value is less than or equal to the other
XCTAssertNil	Tests that an object is nil
XCTAssertNotEqual	Tests that two values are not equal
XCTAssertNotEqualWithAccuracy	Tests that two floating point values (a,b) are not equal within a tolerance of c
XCTAssertNotNil	Tests that an object is not nil

Unit Testing 101



Unit Testing 101



Unit Testing 101

The screenshot shows the Xcode interface with a test summary and the source code for `CalculatorModel.swift`.

Test Summary:

- Test: Test Calculator Today, 4:13 PM
- Build: Build Calculator Today, 4:13 PM
- Debug: Debug Calculator 4/20/16, 4:13 PM

Build Status: Calculator | Build Calculator: **Succeeded** | Today at 4:13 PM

Source Code (`CalculatorModel.swift`):

```
// CalculatorModel.swift
// Calculator
//
// Created by Admin on 4/16/16.
// Copyright © 2016 riis. All rights reserved.

import Foundation

class CalculatorModel {

    var a: Int
    var b: Int

    init(a:Int, b:Int){
        self.a = a
        self.b = b
    }

    func add(a:Int, b:Int) -> Int {
        return a + b
    }

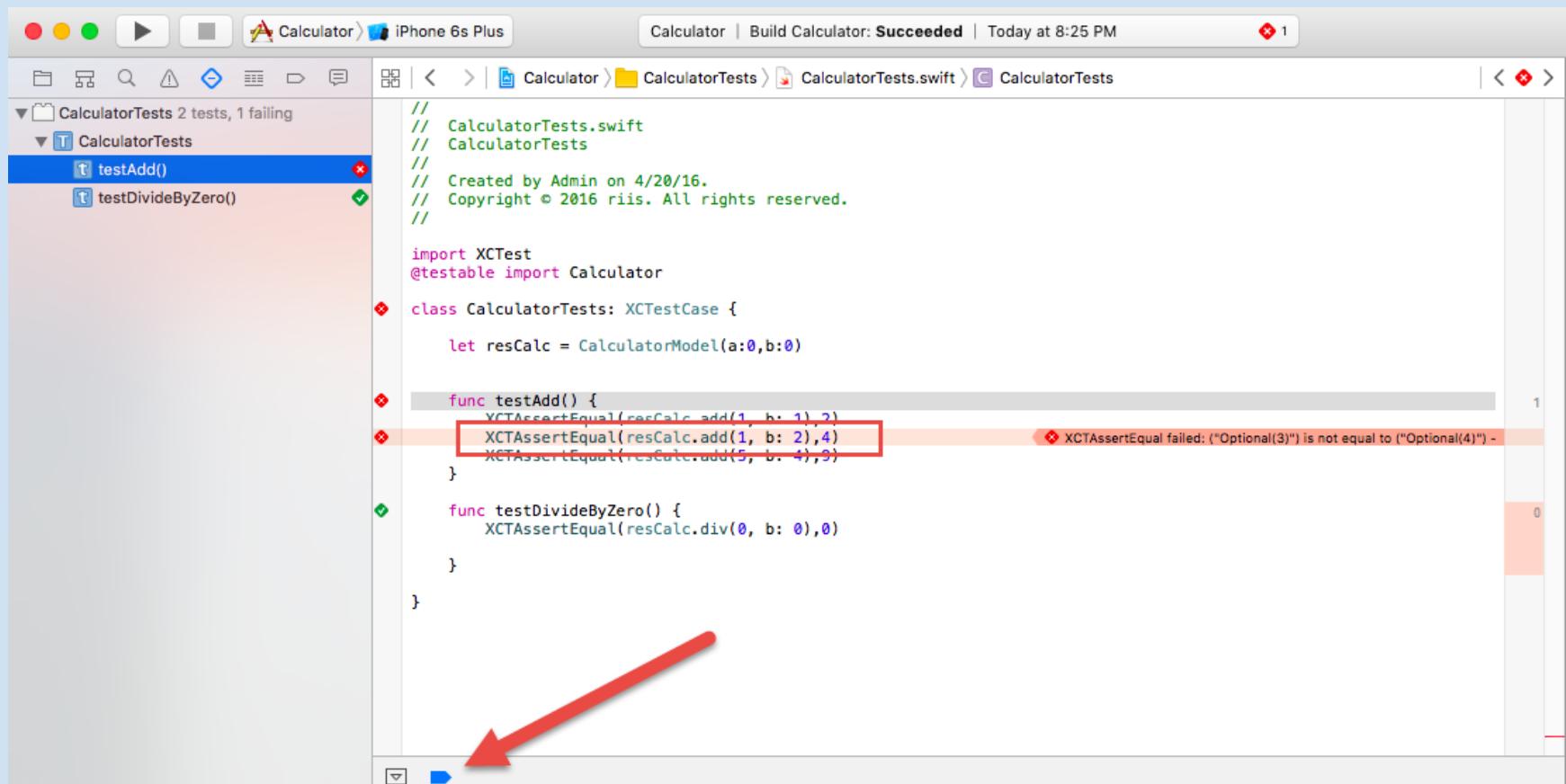
    func sub(a:Int, b:Int) -> Int {
        return a - b
    }

    func mul(a:Int, b:Int) -> Int {
        return a * b
    }

    func div(a:Int, b:Int) -> Int {
        guard b != 0 else {
            return 0
        }
        return a / b
    }
}
```

The code editor shows syntax highlighting for Swift, including green for comments and purple for keywords. A horizontal bar indicates the status of each test case: `add(_:_:)` is green (3), `sub(_:_:)` is orange (0), `mul(_:_:)` is orange (0), `div(_:_:)` is orange (1), and the remaining methods are orange (0).

Unit Testing 101



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows a single test target named "CalculatorTests". It contains two test cases: "testAdd()" (red X) and "testDivideByZero()" (green checkmark).
- Editor:** Displays the code for "CalculatorTests.swift".

```
// CalculatorTests.swift
// CalculatorTests
//
// Created by Admin on 4/20/16.
// Copyright © 2016 riis. All rights reserved.

import XCTest
@testable import Calculator

class CalculatorTests: XCTestCase {

    let resCalc = CalculatorModel(a:0,b:0)

    func testAdd() {
        XCTAssertEqual(resCalc.add(1, b: 1),2)
        XCTAssertEqual(resCalc.add(1, b: 2),4)
        XCTAssertEqual(resCalc.add(5, b: 4),9)
    }

    func testDivideByZero() {
        XCTAssertEqual(resCalc.div(0, b: 0),0)
    }
}
```
- Log Area:** A red arrow points to the bottom-left corner of the editor, where the play button icon is located.
- Status Bar:** Shows "Calculator | Build Calculator: Succeeded | Today at 8:25 PM".

Unit Testing 101

The screenshot shows the Xcode interface during a unit test session. The title bar indicates "Calculator" and "iPhone 6s Plus". The status bar shows "Testing..." and a red diamond icon with the number "1".

The left sidebar displays the "CalculatorTests" target (PID 21223) with various monitoring options like CPU, Memory, Disk, and Network. It also lists threads, with "Thread 1 Queue: com....-thread (serial)" being the active one.

The main editor area shows the `CalculatorTests.swift` file:

```
// CalculatorTests.swift
// CalculatorTests
//
// Created by Admin on 4/20/16.
// Copyright © 2016 riis. All rights reserved.

import XCTest
@testable import Calculator

class CalculatorTests: XCTestCase {

    let resCalc = CalculatorModel(a:0,b:0)

    func testAdd() {
        XCTAssertEqual(resCalc.add(1, b: 1), 2)
        XCTAssertEqual(resCalc.add(1, b: 2), 4)
        XCTAssertEqual(resCalc.add(5, b: 4), 9)
    }

    func testDivideByZero() {
        XCTAssertEqual(resCalc.div(0, b: 0), 0)
    }
}
```

A red breakpoint is set on the first line of the `testAdd()` method. A green bar at the bottom of the code editor indicates "Thread 1: breakpoint 1.1".

The bottom-left panel shows the stack trace for the current thread (Thread 1):

```
self = (CalculatorTests.CalculatorTests) 0x00007fc5f8470fa0
XCTest.XCTTestCase (XCTestCase)
_XCTTestBundleReadyWithProtocolVersion:minimumVersion:
reply received
20:27:04.714 Calculator[21223:553380]
_IDE_startExecutingTestPlanWithProtocolVersion:16
Test Suite 'Selected tests' started at 2016-04-27
20:27:04.723
Test Suite 'CalculatorTests' started at 2016-04-27
20:27:04.724
Test Case '-[CalculatorTests.CalculatorTests testAdd]' started.
(lldb)
```

The bottom-right panel shows the "All Output" tab with the same log information.

Unit Testing 101

The screenshot shows the Xcode Test Navigator interface. The title bar indicates "Calculator | Build Calculator: Succeeded | Today at 5:44 PM". The left sidebar lists recent test runs: "Test Calculator Today, 5:44 PM", "Build Calculator Today, 5:44 PM", "Test Calculator Today, 5:25 PM", "Build Calculator Today, 5:25 PM", and "Debug Calculator 4/20/16, 4:13 PM". The main area displays the "Logs" tab for the most recent test run. The log output shows the following:

```
Test Suite 'Selected tests' started at 2016-04-27 17:44:52.564
Test Suite 'CalculatorTests' started at 2016-04-27 17:44:52.565
Test Case '-[CalculatorTests.CalculatorTests testAdd]' started.
Test Case '-[CalculatorTests.CalculatorTests testAdd]' passed (0.008 seconds).
Test Case '-[CalculatorTests.CalculatorTests testDivideByZero]' started.
Test Case '-[CalculatorTests.CalculatorTests testDivideByZero]' passed (0.001 seconds).

Run test case testAdd() 0.008 seconds
Run test case testDivideByZero() 0.001 seconds
Test Case '-[CalculatorTests.CalculatorTests testAdd]' passed (0.008 seconds).
Test Case '-[CalculatorTests.CalculatorTests testDivideByZero]' started.

Testing Complete 4/27/16, 5:44 PM
No issues
```

Unit Testing 101

The screenshot shows the Xcode Test Navigator interface. The title bar indicates the project is 'Calculator' and the build status is 'Succeeded' at 'Today at 5:25 PM'. The main area displays a test run for 'Test Calculator' at 5:26:14 PM. The 'Errors Only' tab is selected in the top navigation bar. The test results show:

- Test target CalculatorTests**: A red exclamation mark icon indicates an error.
- Scheme Calculator | Destination iPhone 6s Plus**
- Run test suite Selected tests**: A red exclamation mark icon indicates an error.
- Run test suite CalculatorTests**: A red exclamation mark icon indicates an error.
- Run test case testDivideByZero()**: A red exclamation mark icon indicates an error.
- testDivideByZero() encountered an error (Lost connection to test manager service. If you believe this error represents a bug, please att...)**: A detailed error message is shown, followed by a 'more' link.
- Testing Complete 4/27/16, 5:26 PM**
- 1 error**

The bottom output pane shows the command-line logs:

```
17:25:29.526
Test Suite 'CalculatorTests' started at 2016-04-27
17:25:29.527
Test Case '-[CalculatorTests.CalculatorTests
testDivideByZero]' started.
```

Tools of the Trade

- Nimble matchers
- Cuckoo
- Slather
- SwiftFormat
- Jenkins
- SonarQube

Assertion Type	Description	Example
Equivalence	Passes if <code>actual</code> is equivalent to <code>expected</code>	<code>expect(actual).to(equal(expected))</code>
Identity	Passes if <code>actual</code> has the same pointer address as <code>expected</code>	<code>expect(actual).to(beIdenticalTo(expected))</code>
Comparison	<code>LessThan</code> , <code>LessThanOrEqual</code> , <code>GreaterThan</code> , and <code>GreaterThanOrEqual</code>	<code>expect(actual).to(beLessThan(expected))</code>
Comparison	Passes if <code>expected</code> is close to <code>actual</code> within a tolerance	<code>expect(actual).to(beCloseTo(expected, within: delta))</code>
Types/Classes	Passes if <code>instance</code> is an instance of a class	<code>expect(instance).to(beInstanceOf(aClass))</code>
Truthiness	Passes if <code>actual</code> is not nil, true, or an object with a Boolean value of true	<code>expect(actual).to(beTruthy())</code>
Error Handling	Passes if <code>somethingThatThrows()</code> throws an <code>ErrorType</code>	<code>expect{ try somethingThatThrows() }.to(throwError())</code>
Collection Membership	Passes if all of the expected values are members of <code>actual</code>	<code>expect(["whale", "dolphin", "starfish"]).to(contain("dolphin", "starfish"))</code>
Strings	Passes if <code>actual</code> contains, <code>beginsWith</code> , <code>endsWith</code> , or empty substring <code>expected</code>	<code>expect(actual).to(contain(expected))</code>
Count	Passes if <code>actual</code> collection's count is equal to <code>expected</code>	<code>expect(actual).to(haveCount(expected))</code>
Group of Matchers	Matches a value to any of a group of matchers	<code>expect(actual).to(satisfyAnyOf (beLessThan(10), beGreaterThan(20)))</code>

Tools of the Trade

```
import XCTest
import Nimble
@testable import Calculator

class CalculatorTests: XCTestCase {

    let resCalc = CalculatorModel()

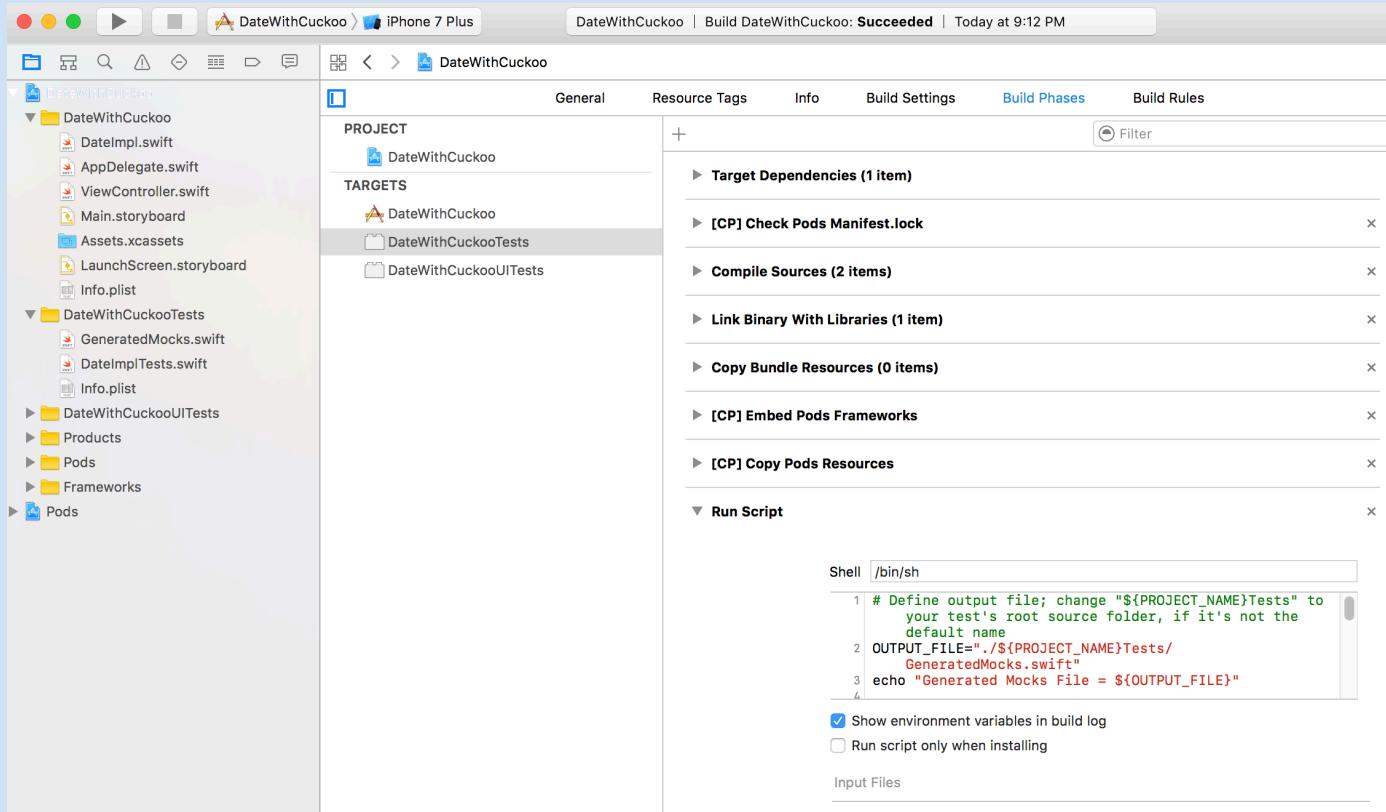
    func testAdd() {
        expect(self.resCalc.add(1, 1)) == 2
    }

    func testAddRange() {
        expect(self.resCalc.mul(4, 3)).to(satisfyAnyOf(beGreaterThan(10), beLessThan(20)))
    }
}
```

Tools of the Trade

```
$ sudo gem install cocoapods  
  
$ pod init  
  
$ vi podfile  
  
    platform :ios, '9.0'  
  
    source 'https://github.com/CocoaPods/Specs.git'  
    target 'CalculatorTests' do  
  
        use_frameworks!  
            pod 'Quick'  
            pod 'Nimble', '~> 5.0.0'  
    end  
  
$ pod install
```

Tools of the Trade



1. Create your project with model class and test class.
2. Run pod init
3. Edit the generated podfile and add pod “Cuckoo” as a test target.
4. Run pod install.
5. Close the project and reopen the workspace.
6. Click on the project folder then choose Test Target ➤ Build Phases.
7. Click + and choose New Run Script Phase.
8. Add Listing to the Run Script section, making sure to modify the input files that you want to mock.
9. Build the project.
10. Run the tests.
11. Drag and drop GeneratedMocks.swift into the test section.
12. Run the mocked tests.

Tools of the Trade

```
platform :ios, '9.0'
use_frameworks!

target 'DateWithCuckooTests' do

    pod 'Cuckoo',
        :git => 'https://github.com/SwiftKit/Cuckoo.git',
        :branch => 'swift-3.0'
end
```

Tools of the Trade

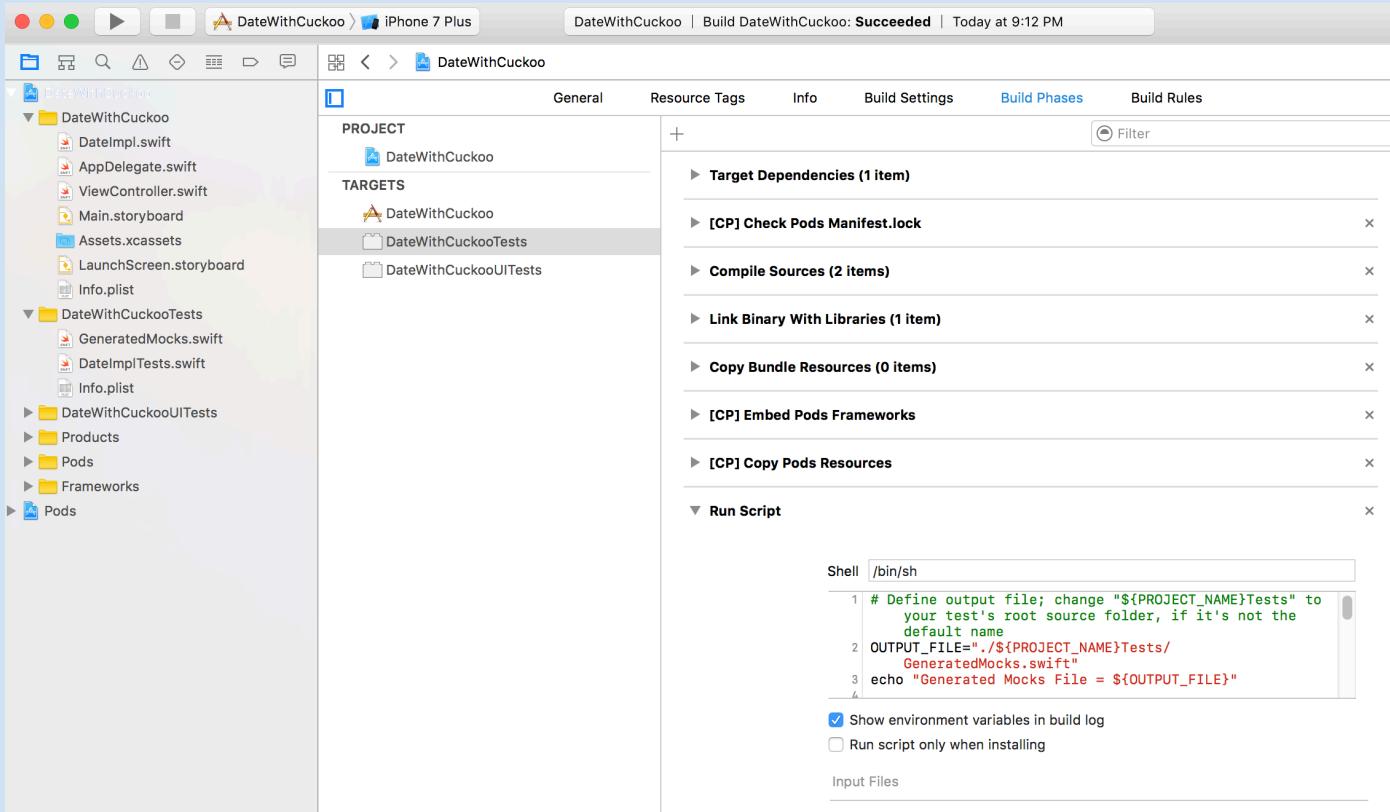
```
# Define output file; change "${PROJECT_NAME}Tests" to your test's
root source folder, if it's not the default name

OUTPUT_FILE=./${PROJECT_NAME}Tests/GeneratedMocks.swift
echo "Generated Mocks File = ${OUTPUT_FILE}"

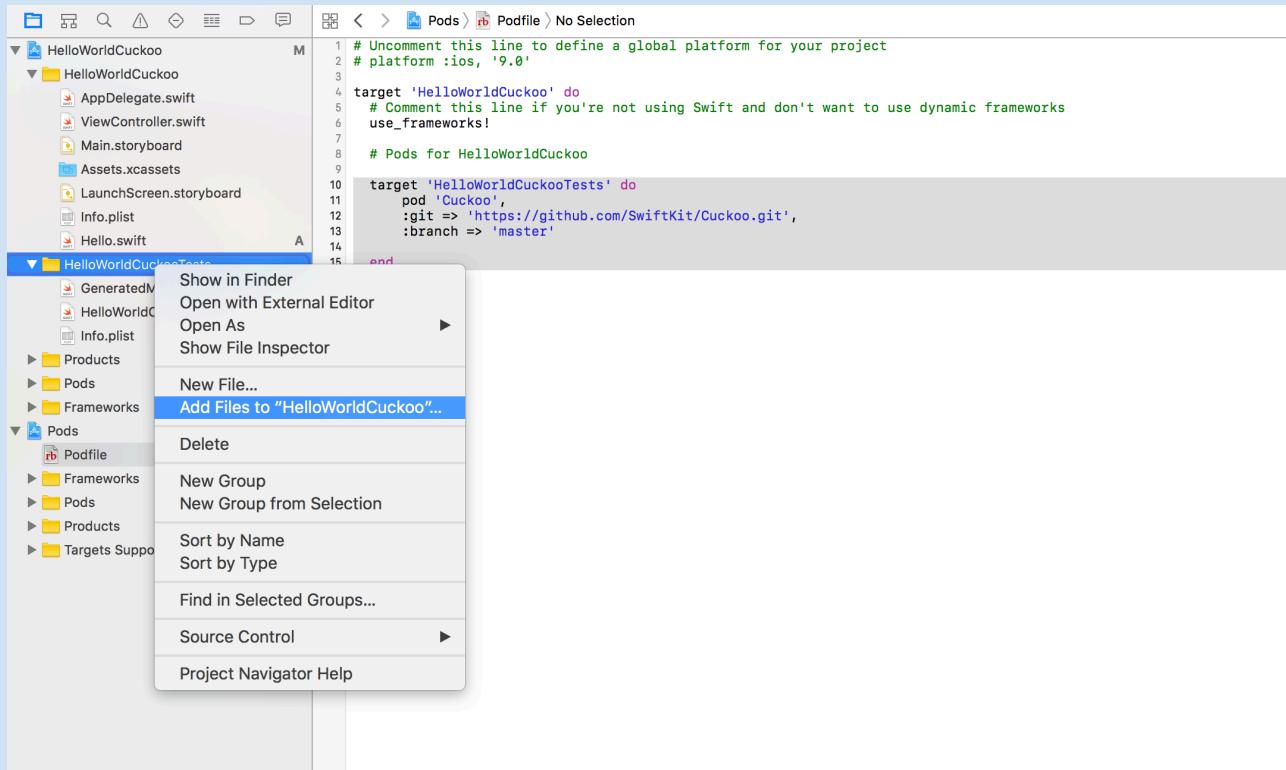
# Define input directory; change "${PROJECT_NAME}" to your project's root
source folder, if it's not the default name
INPUT_DIR=./${PROJECT_NAME}"
echo "Mocks Input Directory = ${INPUT_DIR}"

# Generate mock files; include as many input files as you'd like to create mocks for
${PODS_ROOT}/Cuckoo/run generate --testable "${PROJECT_NAME}" \
--output "${OUTPUT_FILE}" \
"${INPUT_DIR}/FileName1.swift" \
"${INPUT_DIR}/FileName2.swift" \
"${INPUT_DIR}/FileName3.swift"
# ... and so forth
```

Tools of the Trade



Tools of the Trade



Tools of the Trade

```
# Mocking takes the form when(methodIsCalled).thenReturn(aValue)

stub(mock) {
    (mock) in when(mock.getDate.get).thenReturn(dateAndTime.from(year: 2014, month: 05, day: 20) as Date)
}

XCTAssertEqual(mock.getDate, dateAndTime.from(year: 2014, month: 05, day: 20) as Date)
XCTAssertNotNil(verify(mock).getDate)
```

Tools of the Trade



Coverage for "CalculatorModel.swift" : 65.00%
(13 of 20 relevant lines covered)

Calculator/Calculator/CalculatorModel.swift

```
1 //          5x
2 //  CalculatorModel.swift           5x
3 //  Calculator                      5x
4 //                                     5x
5 //  Created by Admin on 4/16/16.    5x
6 //  Copyright © 2016 riis. All rights reserved. 5x
7 //
8
9 import Foundation
10
11 class CalculatorModel {
12
13     var operandOne: Int
14     var operandTwo: Int
15
16     init(operandOne: Int, operandTwo: Int) {
17         self.operandOne = operandOne
18         self.operandTwo = operandTwo
19     }
20
21     func add(operandOne: Int, operandTwo: Int) -> Int {
22         return operandOne + operandTwo
23     }
24
25     func sub(operandOne: Int, operandTwo: Int) -> Int {
26         return operandOne - operandTwo
27     }
28
29     func mul(operandOne: Int, operandTwo: Int) -> Int {
30         return operandOne * operandTwo
31     }
32
33     func div(operandOne: Int, operandTwo: Int) -> Int {
34
35         guard operandTwo != 0 else {
36             return 0
37         }
38         return operandOne / operandTwo
39     }
40
41 }
```

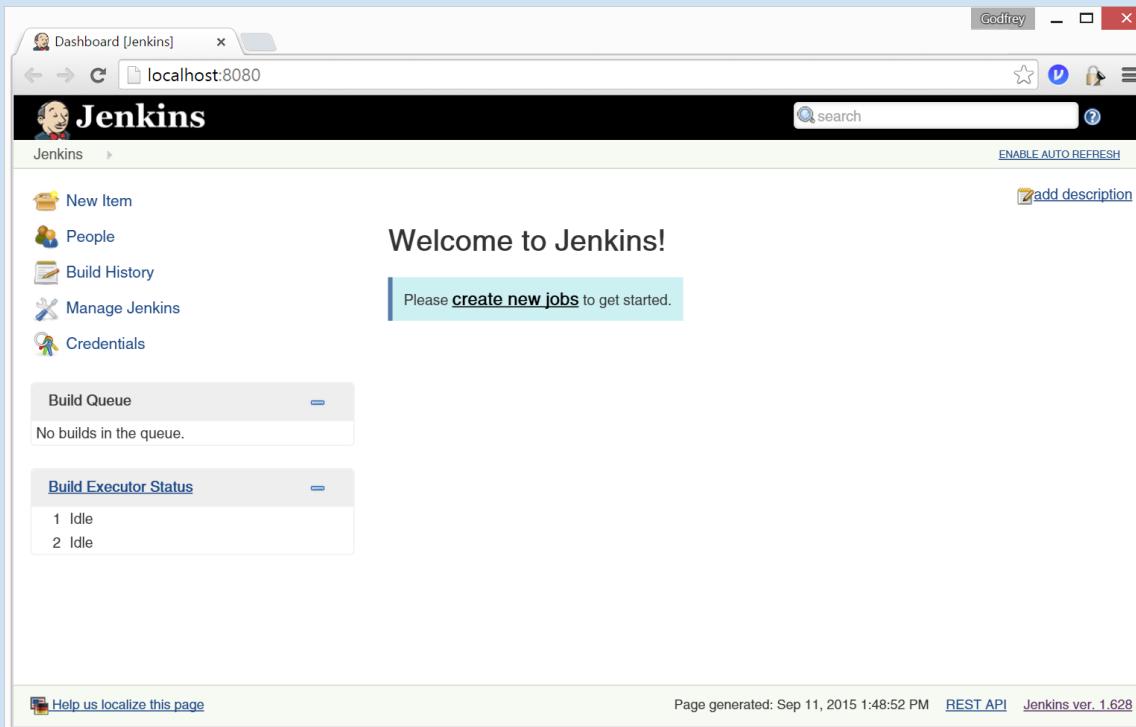
Tools of the Trade

```
$ gem install slather  
  
$ slather coverage --html --scheme XcodeSchemeName path/to/project.xcodeproj  
  
$ slather coverage --html --scheme Calculator Calculator/Calculator.xcodeproj
```

Tools of the Trade

```
# SwiftLint  
  
$ brew install swiftlint  
  
$ cd /path/to/XCode directory  
  
$ swiftlint autocorrect  
  
# SwiftFormat  
  
$ brew install swiftformat  
  
$ swiftformat --indent 4
```

Tools of the Trade



Tools of the Trade

SonarQube Dashboard for Swift :: Simple Project :: SonarQube Scanner

Version 1.0 - May 15 2016 18:54

Dashboard

- Issues
- Time Machine
- Components
- Issues Drilldown
- Design
- Libraries
- Compare

sonarqube

Lines Of Code

Swift	Files	Functions
6	1	1
	Directories	Lines
	1	11
	Classes	Statements
	0	6

Duplications

0.0%

Lines Blocks Files

0 0 0

Complexity

1.0 /function

2.0 /file

Total: 2

Events All

May 15 2016 Version 1.0

SOALE Rating A

Technical Debt Ratio 1.1%

Technical Debt 2min

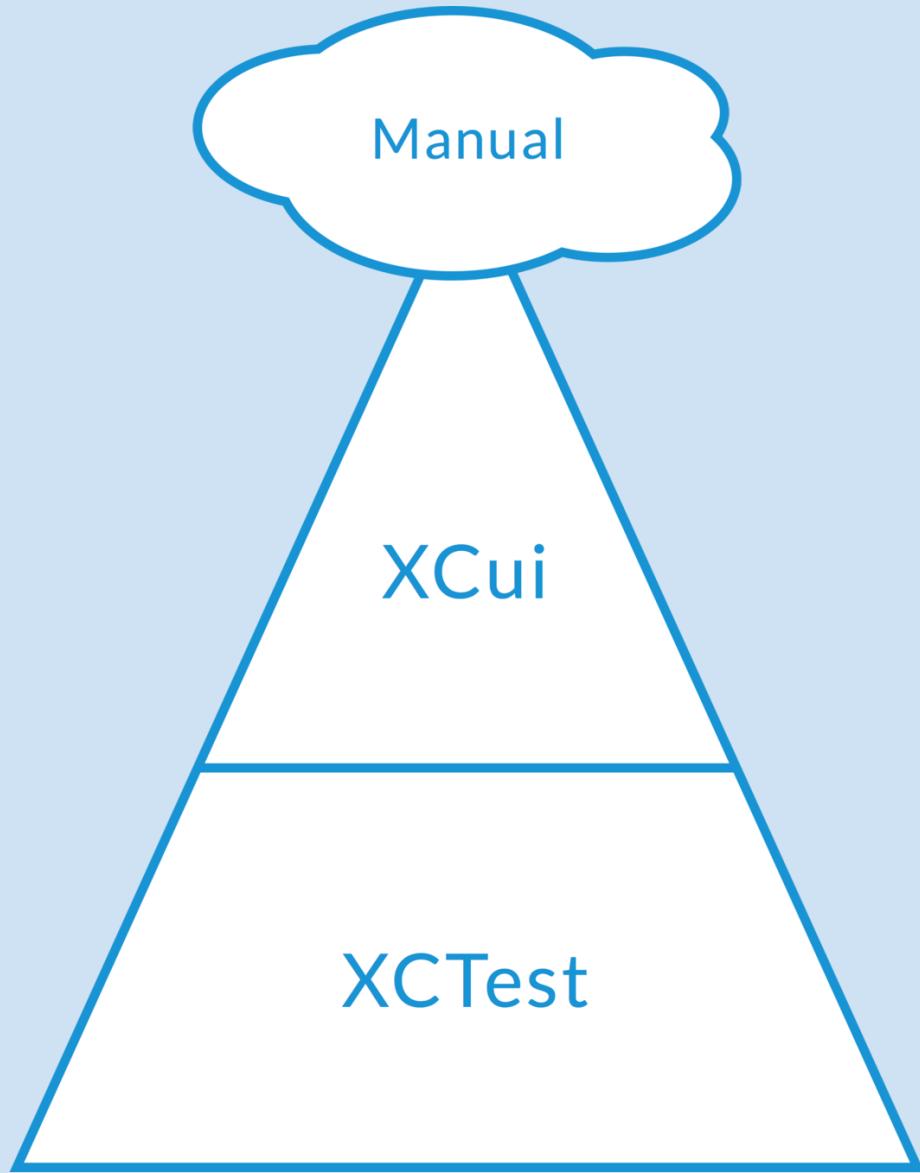
Issues 2

Severity	Count
Blocker	0
Critical	0
Major	1
Minor	1
Info	0

Profiles: Sonar way (Swift)

Configure widgets Manage dashboards

Settings Administrator Search



XCUI

- User Interface testing
- Two options
 - Recorded tests
 - Roll your own

XCUI

Choose options for your new project:

Product Name: Calculator

Team:

Organization Name: example

Organization Identifier: com.example

Bundle Identifier: com.example.Calculator

Language: Swift ▼

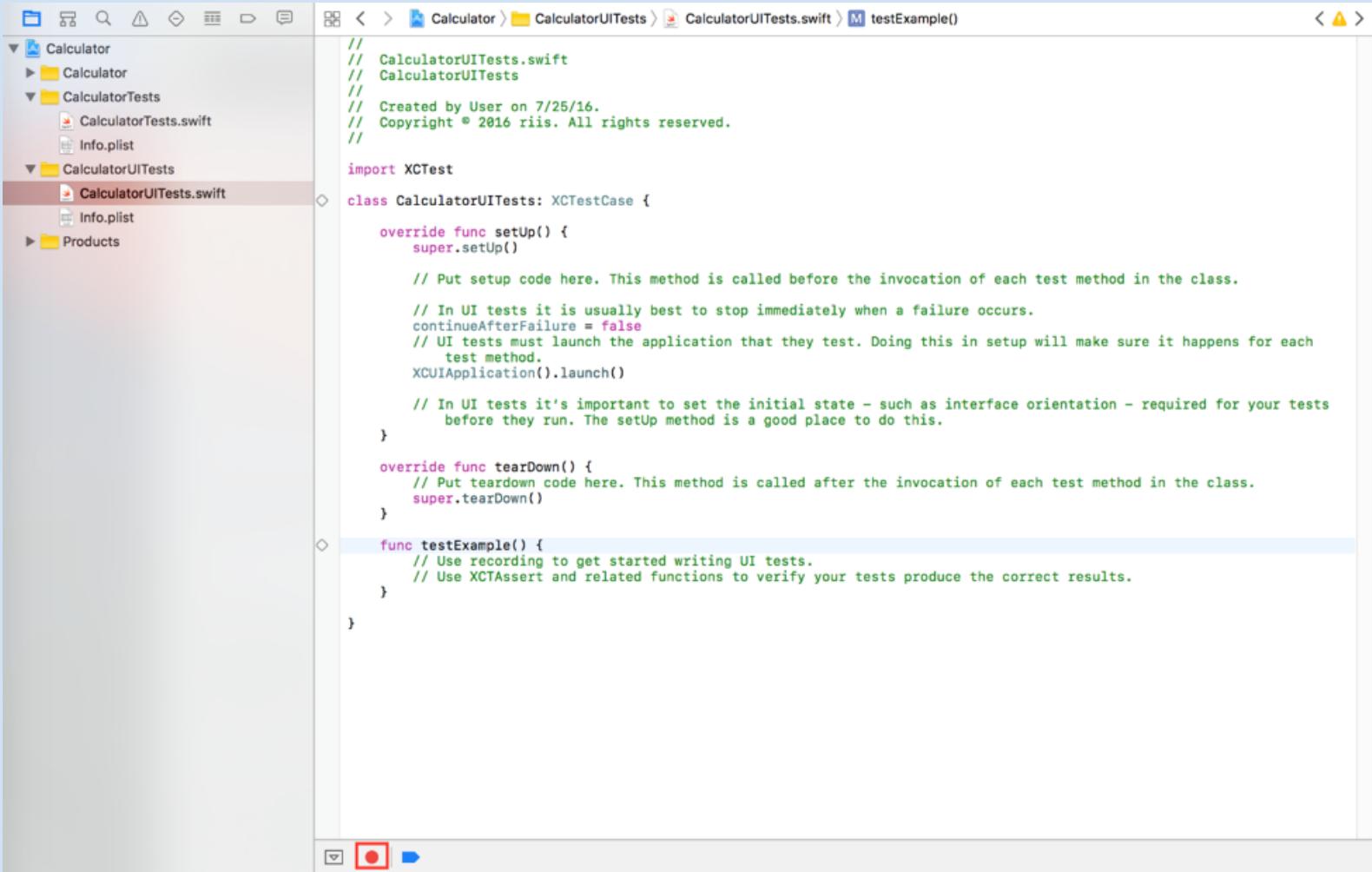
Devices: iPhone ▼

Use Core Data

Include Unit Tests

Include UI Tests

XCUI



The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under the "Calculator" project. It includes a "Calculator" target, a "CalculatorTests" target containing "CalculatorTests.swift" and "Info.plist", and a "CalculatorUITests" target containing "CalculatorUITests.swift" and "Info.plist".
- Editor Tab:** Displays the "CalculatorUITests.swift" file content.
- Text:** The code is written in Swift and defines a test case for UI testing. It includes setup and teardown methods and a test method named "testExample".

```
// CalculatorUITests.swift
// CalculatorUITests
//
// Created by User on 7/25/16.
// Copyright © 2016 riis. All rights reserved.

import XCTest

class CalculatorUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        // Put setup code here. This method is called before the invocation of each test method in the class.

        // In UI tests it is usually best to stop immediately when a failure occurs.
        continueAfterFailure = false
        // UI tests must launch the application that they test. Doing this in setup will make sure it happens for each
        // test method.
        XCUIApplication().launch()

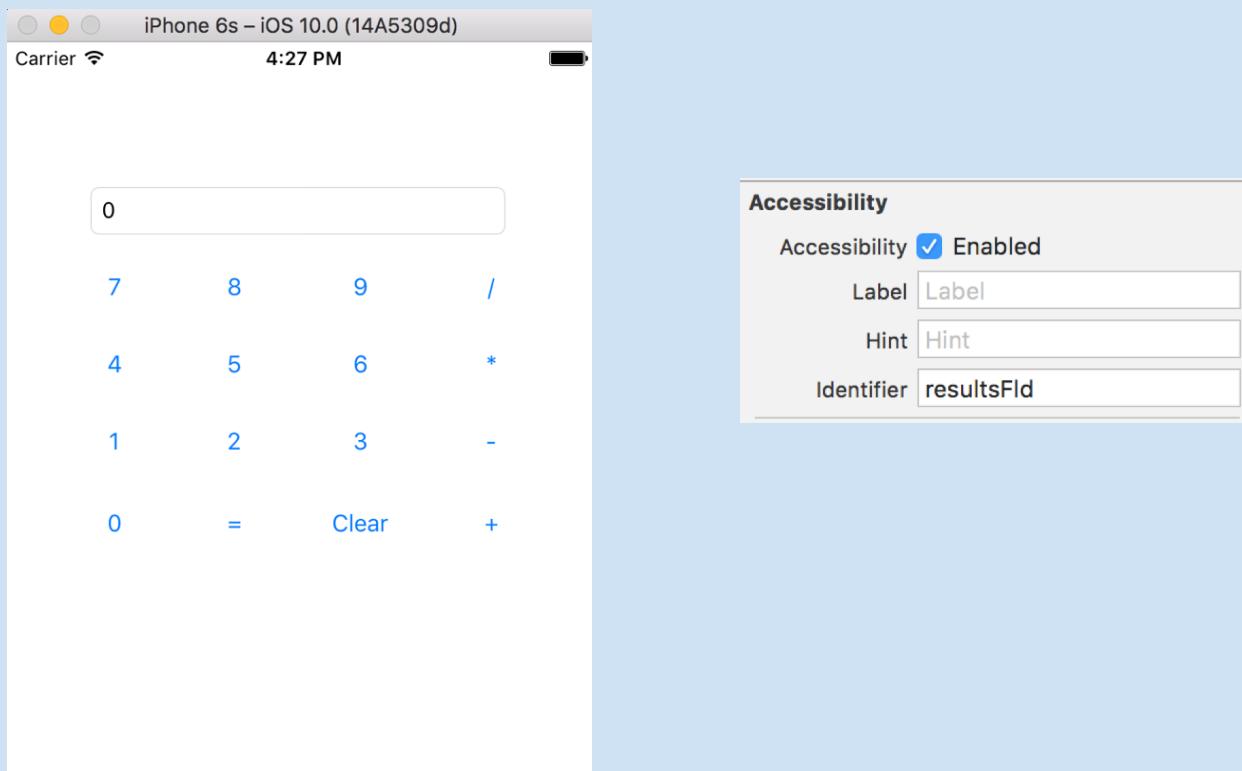
        // In UI tests it's important to set the initial state - such as interface orientation - required for your tests
        // before they run. The setUp method is a good place to do this.
    }

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

    func testExample() {
        // Use recording to get started writing UI tests.
        // Use XCTAssert and related functions to verify your tests produce the correct results.
    }
}
```

- Toolbar:** At the bottom, there are standard Xcode toolbar icons for file operations (New, Open, Save, Find, etc.) and build/run controls.

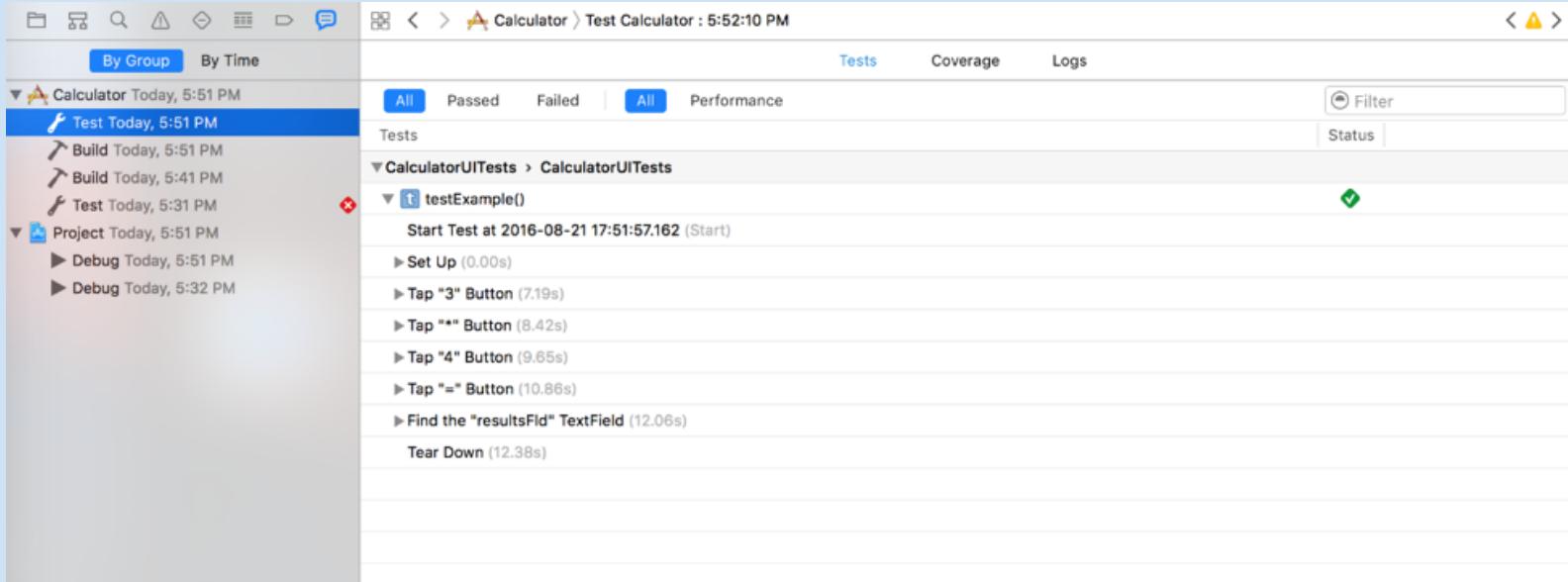
XCUI



XCUI

```
func testExample() {  
  
    let app = XCUIApplication()  
    app.buttons["3"].tap()  
    app.buttons["*"].tap()  
    app.buttons["4"].tap()  
    app.buttons[ "=" ].tap()  
  
    let resultTextField = app.textFields["resultsFld"]  
    XCTAssert(resultTextField.value! as! String == "12")  
  
}
```

XCUI



Calculator Test Case: Test Calculator : 5:32:16 PM

Tests Coverage Logs

All Recent All Messages All Issues Errors Only Filter

/Users/User/Library/Developer/Xcode/DerivedData/Calculator-azioamtiylnkumgnhtelunnpppbw/Logs/Test/0A8B9DC6-1C8A-4A83-BA35-47DBAEF8... more

Run test suite Selected tests 0 out of 1 test passed, 14.347 seconds

Run test suite CalculatorUITests.xctest 0 out of 1 test passed, 14.347 seconds

Run test suite CalculatorUITests 0 out of 1 test passed, 14.347 seconds

Run test case testExample() 14.347 seconds

Test Case '-[CalculatorUITests.CalculatorUITests testExample]' started.

```
t = 0.00s Start Test at 2016-08-21 17:32:01.738
t = 0.00s Set Up
t = 0.01s Launch com.riis.Calculator
t = 4.20s Waiting for accessibility to load
t = 7.47s Wait for app to idle
t = 8.94s Tap "3" Button
t = 8.95s Wait for app to idle
t = 9.02s Find the "3" Button
t = 9.02s Snapshot accessibility hierarchy for com.riis.Calculator
t = 9.36s Find: Descendants matching type Button
t = 9.36s Find: Elements matching predicate "'3' IN identifiers"
t = 9.37s Wait for app to idle
t = 9.44s Synthesize event
t = 9.72s Wait for app to idle
t = 10.20s Tap "*" Button
t = 10.20s Wait for app to idle
t = 10.27s Find the "*" Button
t = 10.27s Snapshot accessibility hierarchy for com.riis.Calculator
t = 10.58s Find: Descendants matching type Button
t = 10.58s Find: Elements matching predicate "'*' IN identifiers"
t = 10.59s Wait for app to idle
t = 10.66s Synthesize event
t = 10.94s Wait for app to idle
t = 11.42s Tap "4" Button
t = 11.42s Wait for app to idle
t = 11.49s Find the "4" Button
t = 11.49s Snapshot accessibility hierarchy for com.riis.Calculator
t = 11.80s Find: Descendants matching type Button
t = 11.80s Find: Elements matching predicate "'4' IN identifiers"
t = 11.80s Wait for app to idle
t = 11.87s Synthesize event
t = 12.15s Wait for app to idle
t = 12.63s Tap "=" Button
t = 12.63s Wait for app to idle
t = 12.69s Find the "=" Button
t = 12.70s Snapshot accessibility hierarchy for com.riis.Calculator
t = 13.00s Find: Descendants matching type Button
t = 13.00s Find: Elements matching predicate "'=' IN identifiers"
t = 13.01s Wait for app to idle
t = 13.07s Synthesize event
t = 13.35s Wait for app to idle
t = 13.82s Find the "resultsFld" TextField
t = 13.82s Snapshot accessibility hierarchy for com.riis.Calculator
t = 14.13s Find: Descendants matching type TextField
t = 14.13s Find: Elements matching predicate "'resultsFld' IN identifiers"
t = 14.14s Assertion Failure: CalculatorUITests.swift:40: XCTAssertTrue failed -
t = 14.14s Tear Down
```

Test Case '-[CalculatorUITests.CalculatorUITests testExample]' failed (14.347 seconds).

XCTAssertTrue failed -

XCUI - Roll your own

- XCUI has 3 component parts
 - XCUIApplication
 - XCUIElement
 - XCUIElementQuery

XCUI - Roll your own

- XCUIApplication
 - Launch the app
- XCUIElementQuery
 - Find the XCUIElement to test
- XCUIElement
 - Perform test

```
class CalculatorUITests: XCTestCase {

    override func setUp() {
        super.setUp()

        // stop if any test fails
        continueAfterFailure = false
    }

    func testExample() {

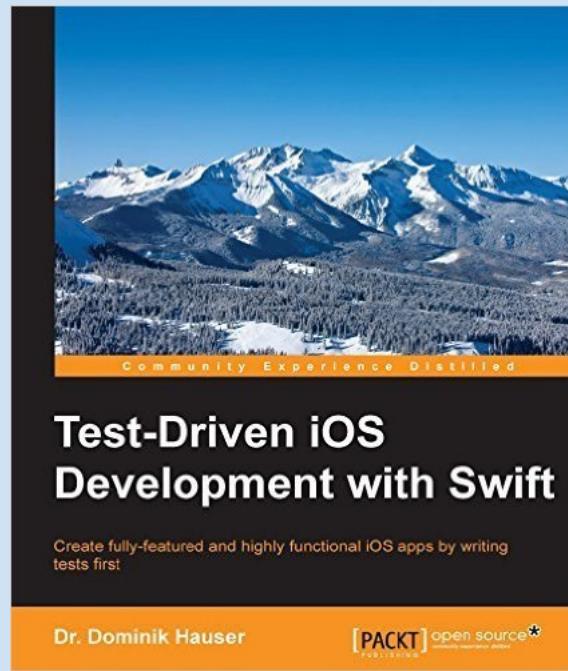
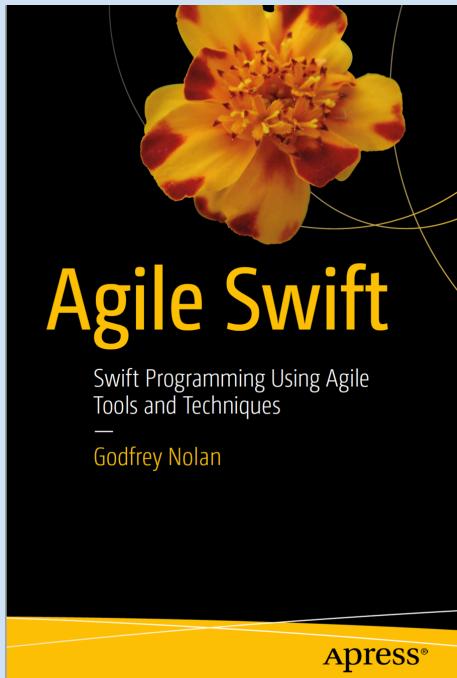
        // Arrange
        let app = XCUIApplication()

        // Act
        app.buttons["3"].tap()
        app.buttons["*"].tap()
        app.buttons["4"].tap()
        app.buttons[ "=" ].tap()

        // Assert
        XCTAssert(app.textFields["resultsFld"].value! as! String == "12")
    }

}
```

Books



Contact Details

@riisllc, @godfreynolan

godfrey@riis.com

<http://riis.com/blog>