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

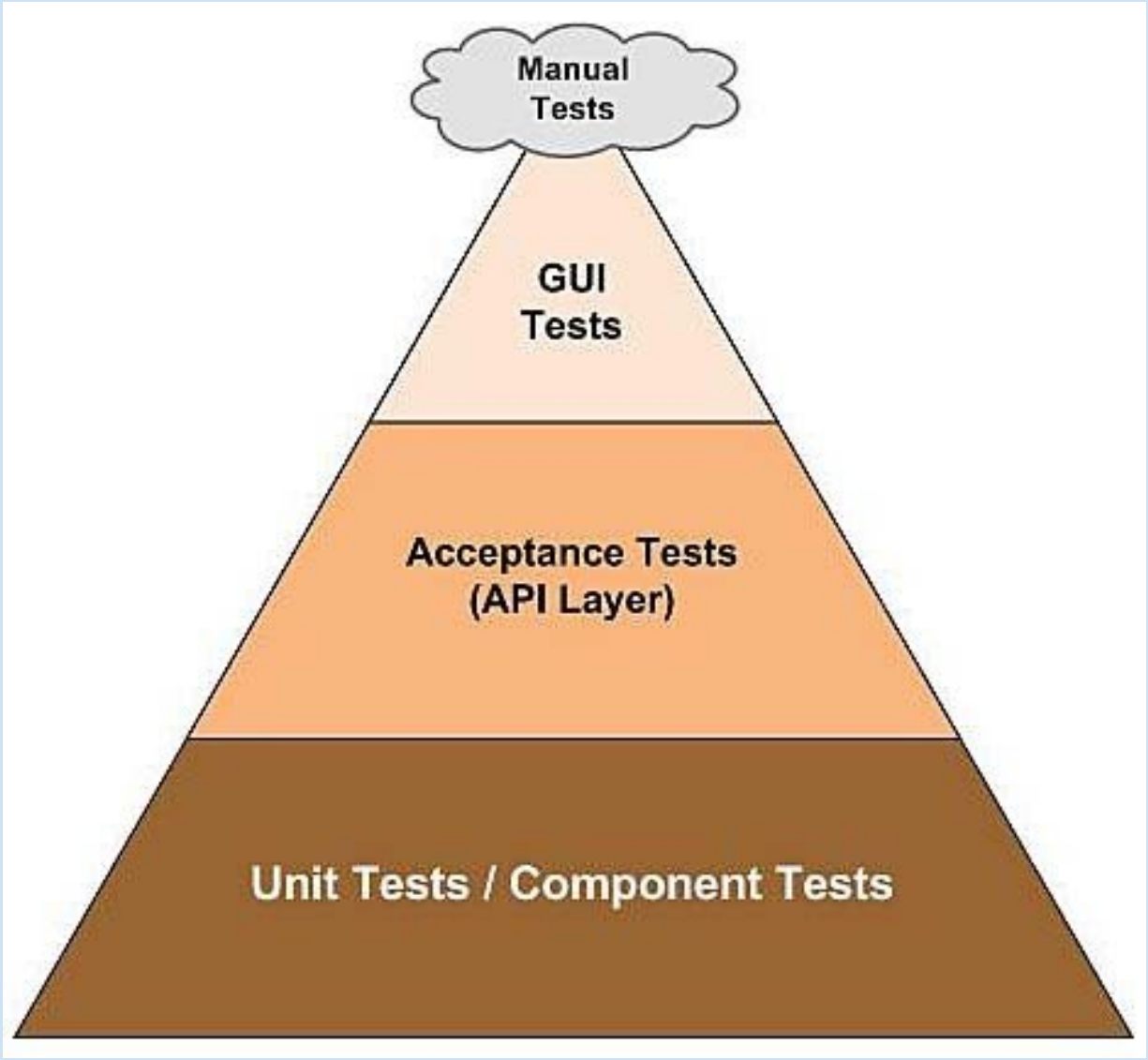
Unit Testing Intro

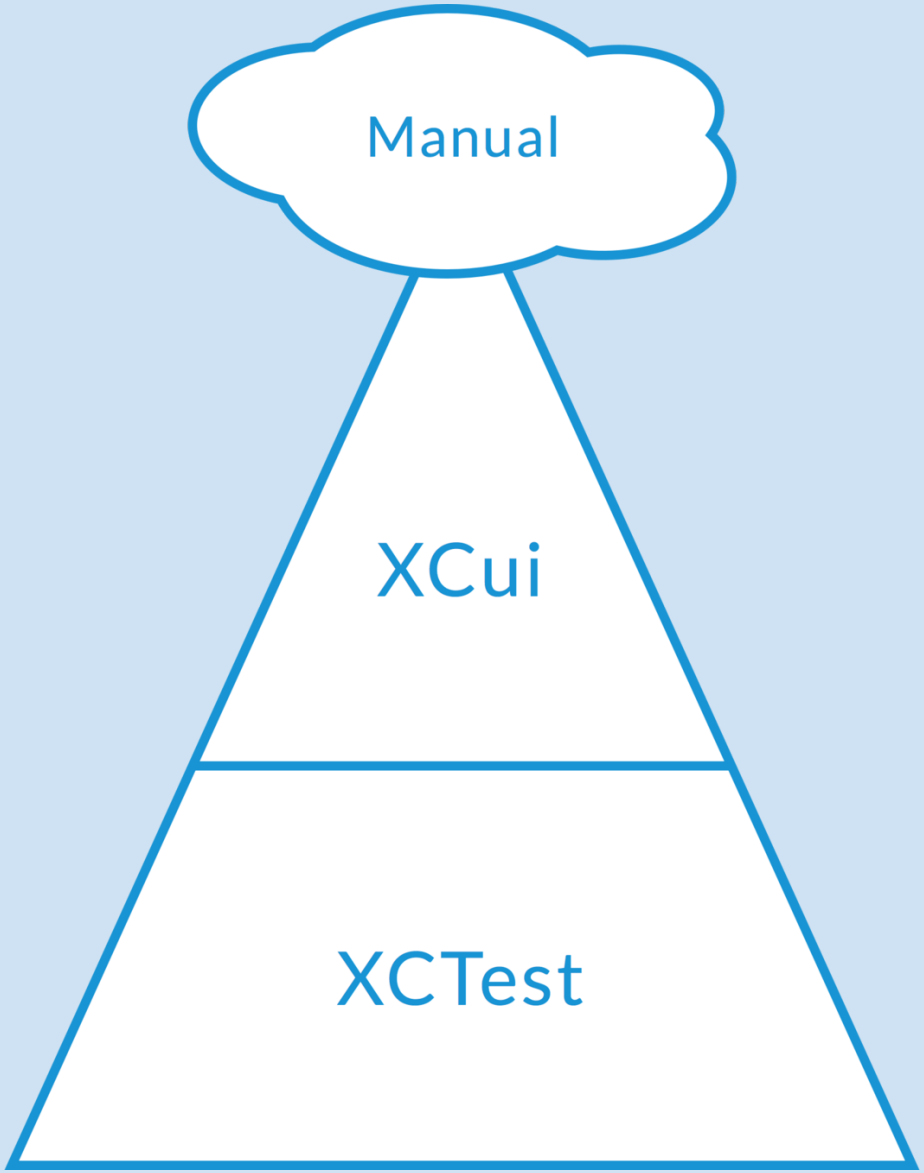
```
struct HelloWorld {

    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


Manual
Tests
GUI
Tests
Acceptance Tests
(API Layer)
Unit Tests / Component Tests

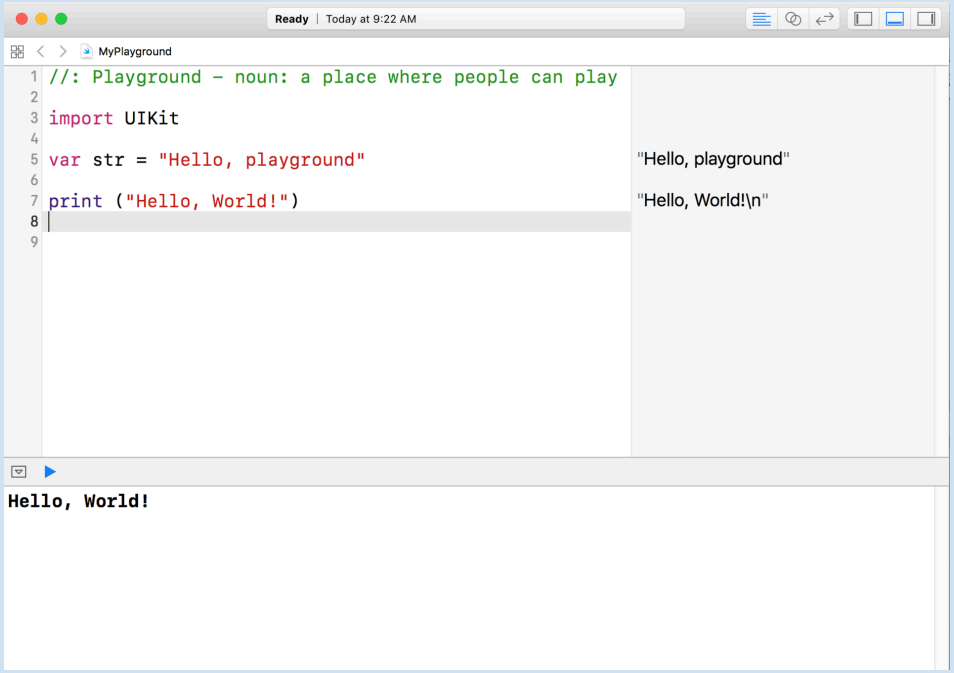

Manual
XCui
XCTest

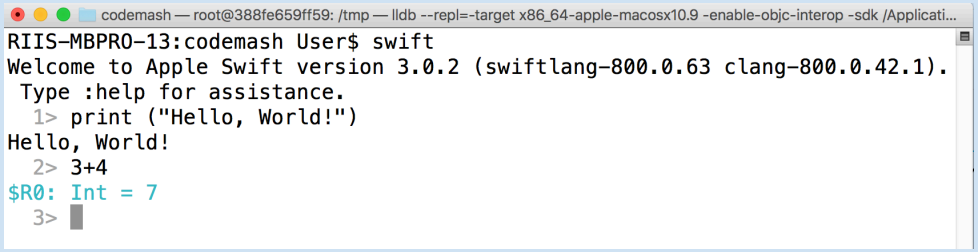
Swift 3.x Flavors

- 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


Ready | Today at 9:22 AM
MyPlayground
1 //: Playground - noun: a place where people can play
2
3 import UIKit
4
5 var str = "Hello, playground"
6
7 print ("Hello, World!")
8
9
"Hello, playground"
"Hello, World!\n"
Hello, World!


codemash — root@388fe659ff59: /tmp — lldb --repl=-target x86_64-apple-macosx10.9 -enable-objc-interop -sdk /Applicati...
RIIS-MBPRO-13:codemash User$ swift
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),
])
```

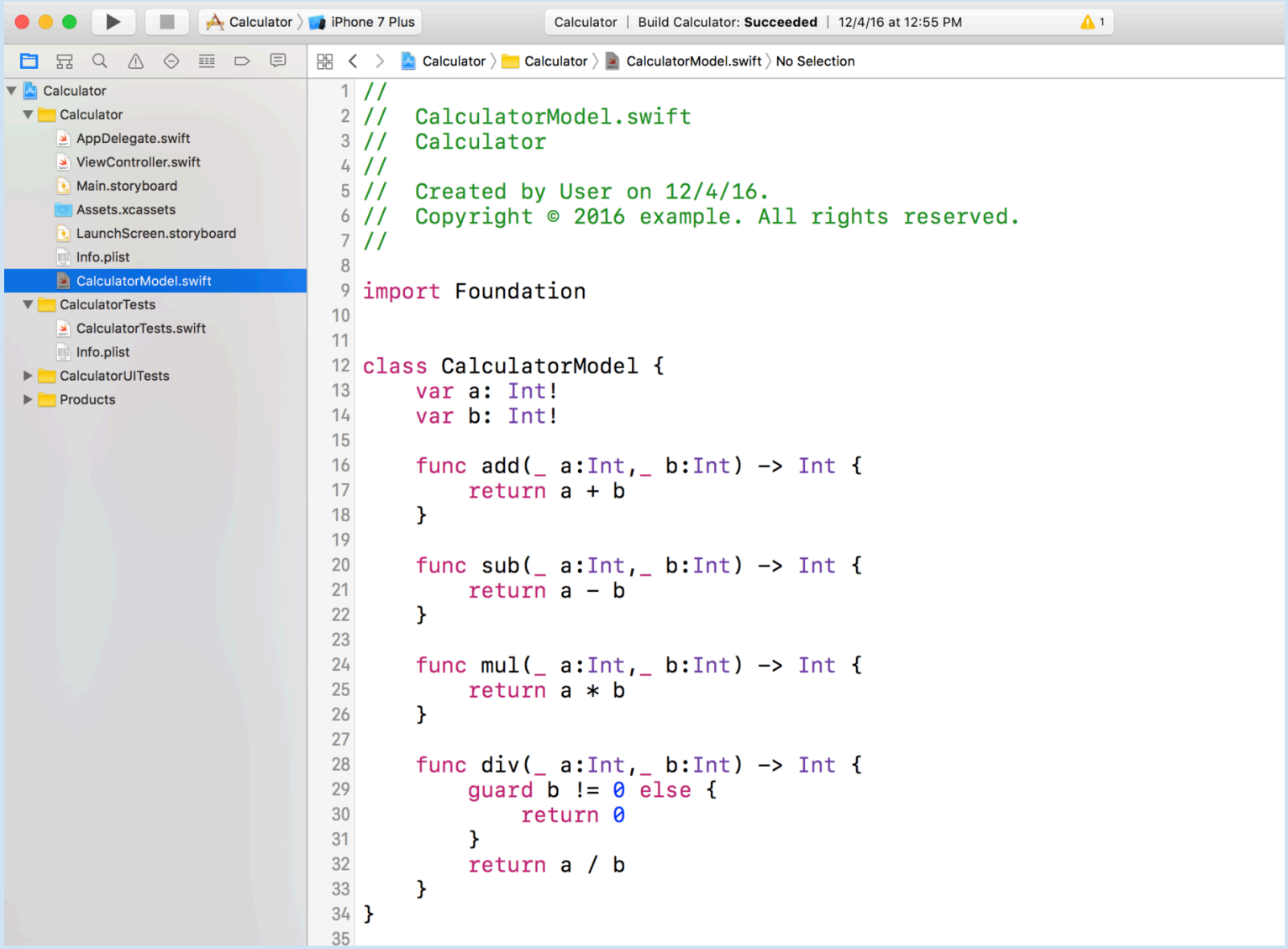
HelloWorldTests.swift

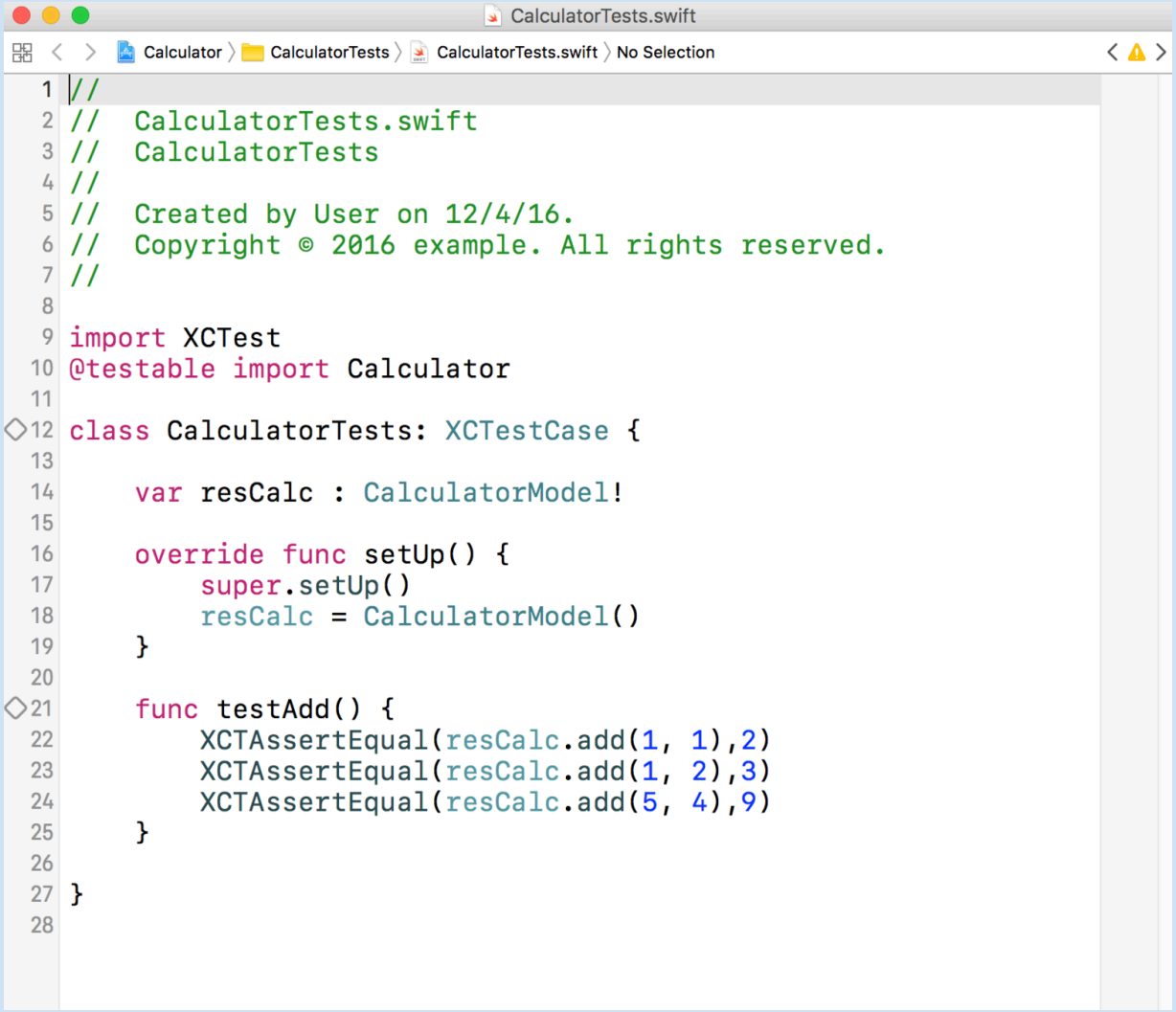
```
import XCTest
@testable import HelloWorld

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**

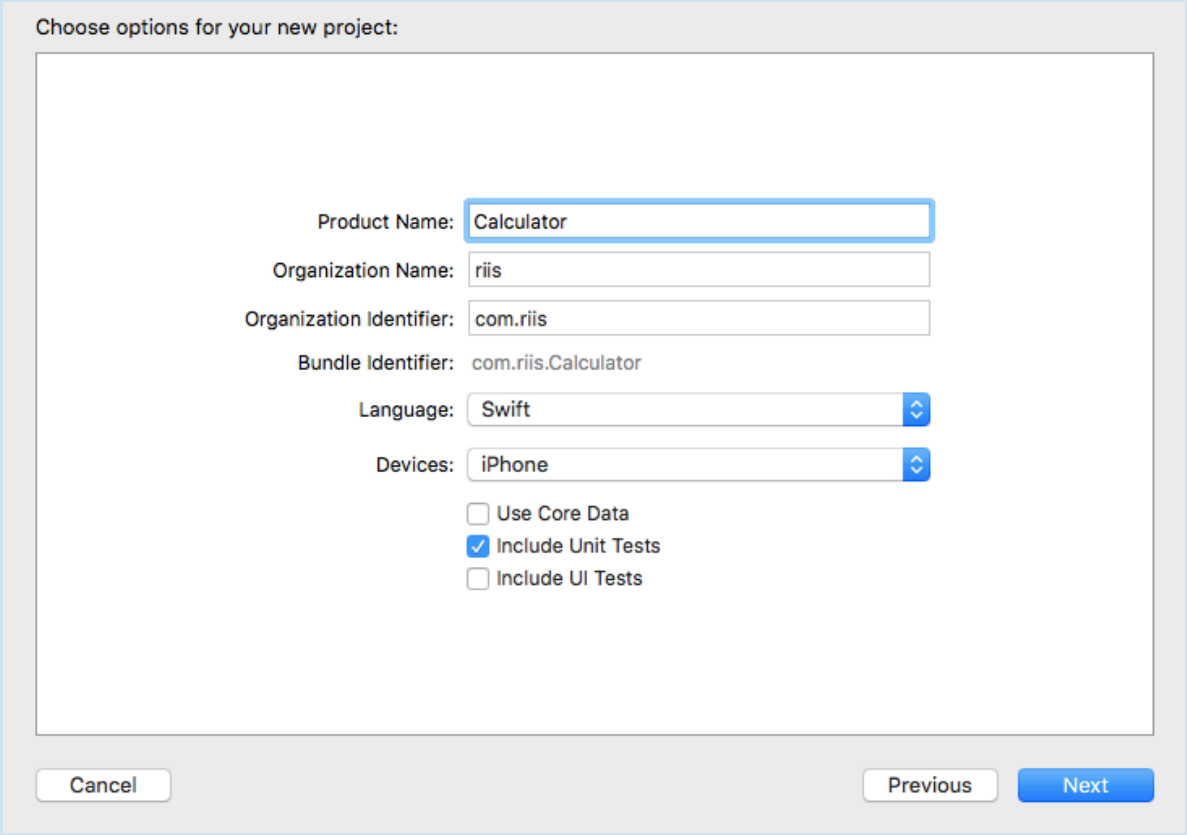

Calculator 〉 iPhone 7 Plus
Calculator | Build Calculator: Succeeded | 12/4/16 at 12:55 PM
1
Calculator 〉 Calculator 〉 CalculatorModel.swift 〉 No Selection
Calculator
Calculator
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
CalculatorModel.swift
CalculatorTests
CalculatorTests.swift
Info.plist
CalculatorUITests
Products
1 //
2 // CalculatorModel.swift
3 // Calculator
4 //
5 // Created by User on 12/4/16.
6 // Copyright © 2016 example. All rights reserved.
7 //
8
9 import Foundation
10
11
12 class CalculatorModel {
13 var a: Int!
14 var b: Int!
15
16 func add(_ a:Int,_ b:Int) -> Int {
17 return a + b
18 }
19
20 func sub(_ a:Int,_ b:Int) -> Int {
21 return a - b
22 }
23
24 func mul(_ a:Int,_ b:Int) -> Int {
25 return a * b
26 }
27
28 func div(_ a:Int,_ b:Int) -> Int {
29 guard b != 0 else {
30 return 0
31 }
32 return a / b
33 }
34 }
35


CalculatorTests.swift
Calculator
CalculatorTests
CalculatorTests.swift
No Selection
//
// CalculatorTests.swift
// CalculatorTests
//
// Created by User on 12/4/16.
// Copyright © 2016 example. All rights reserved.
//
import XCTest
@testable import Calculator
class CalculatorTests: XCTestCase {
var resCalc : CalculatorModel!
override func setUp() {
super.setUp()
resCalc = CalculatorModel()
}
func testAdd() {
XCTAssertEqual(resCalc.add(1, 1),2)
XCTAssertEqual(resCalc.add(1, 2),3)
XCTAssertEqual(resCalc.add(5, 4),9)
}
}

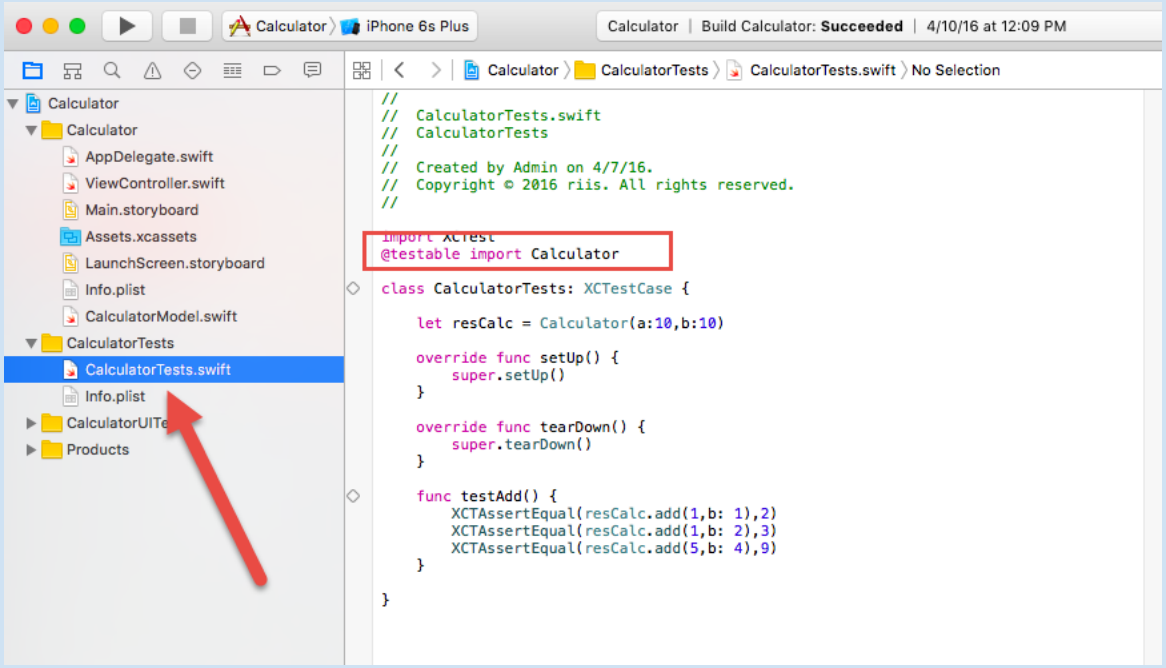
Unit Testing 101

- Setup and Teardown
- Performance
- Assertions
- Code Coverage

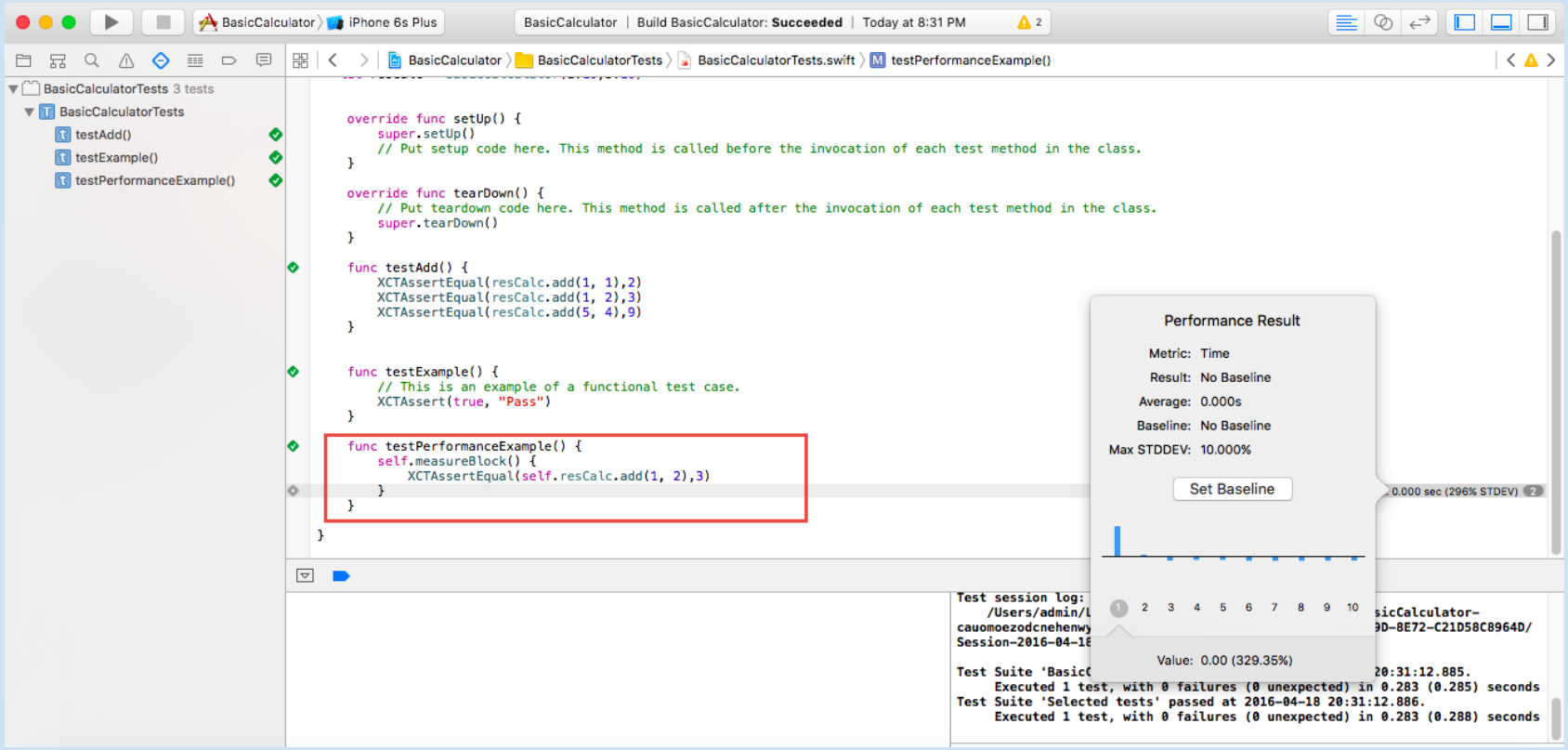
Unit Testing 101



Unit Testing 101



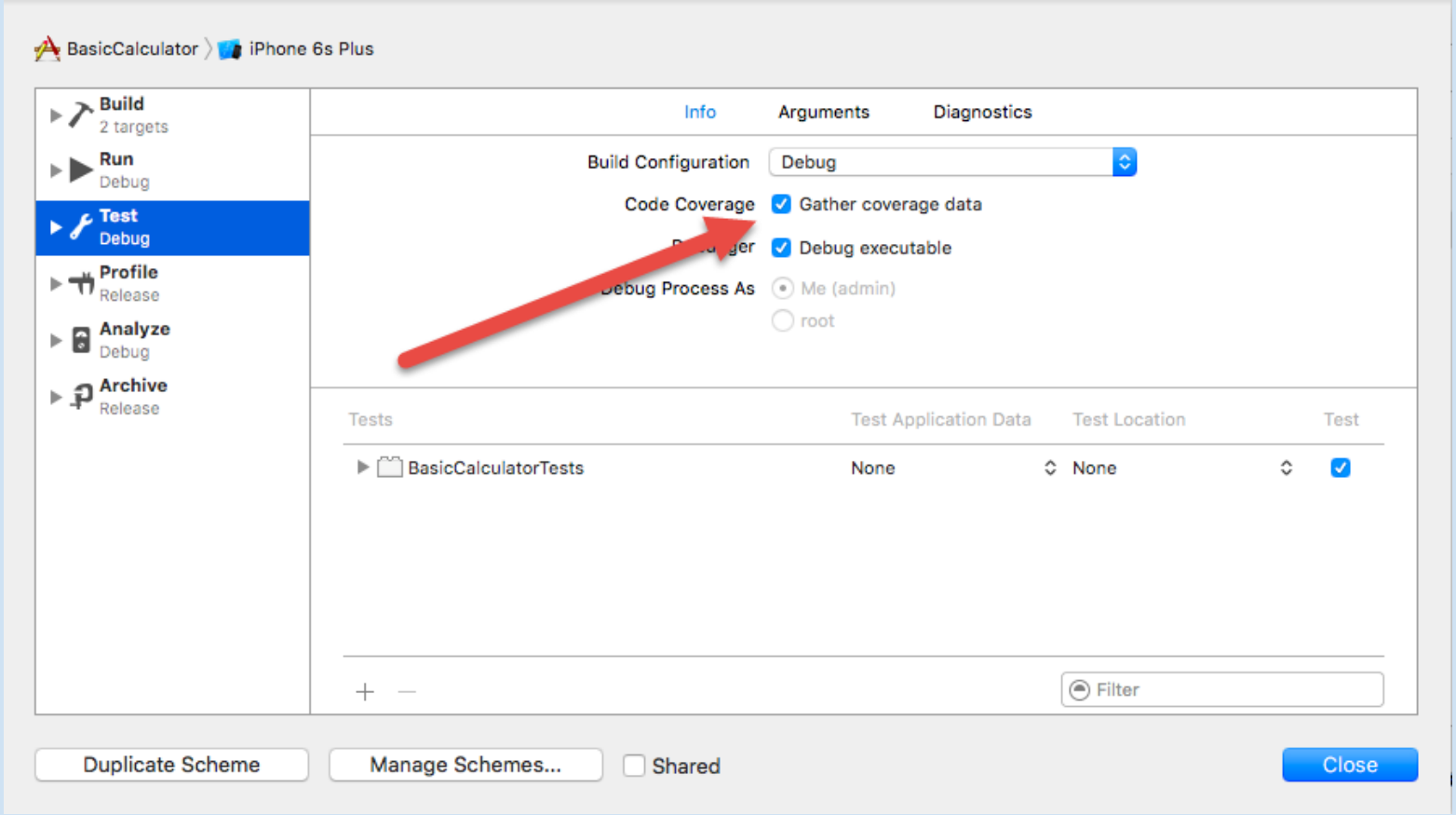
Unit Testing 101



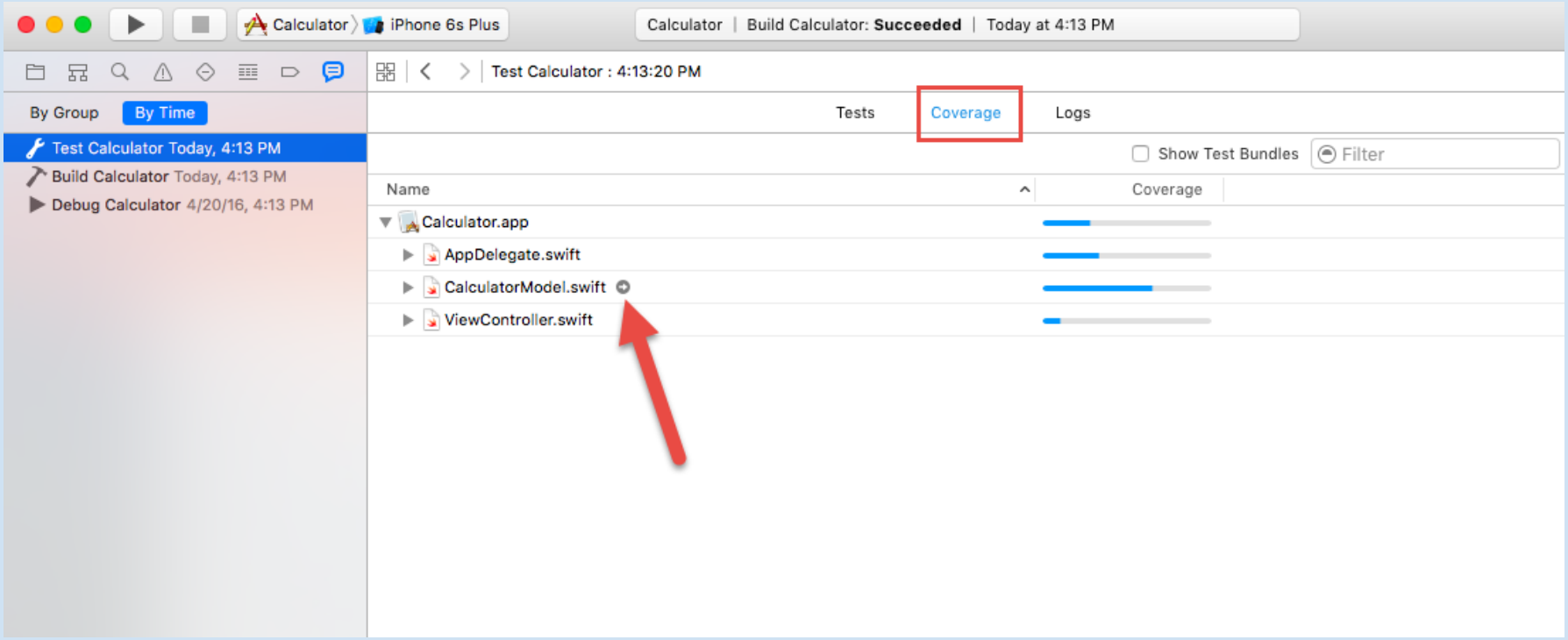
Unit Testing 101

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

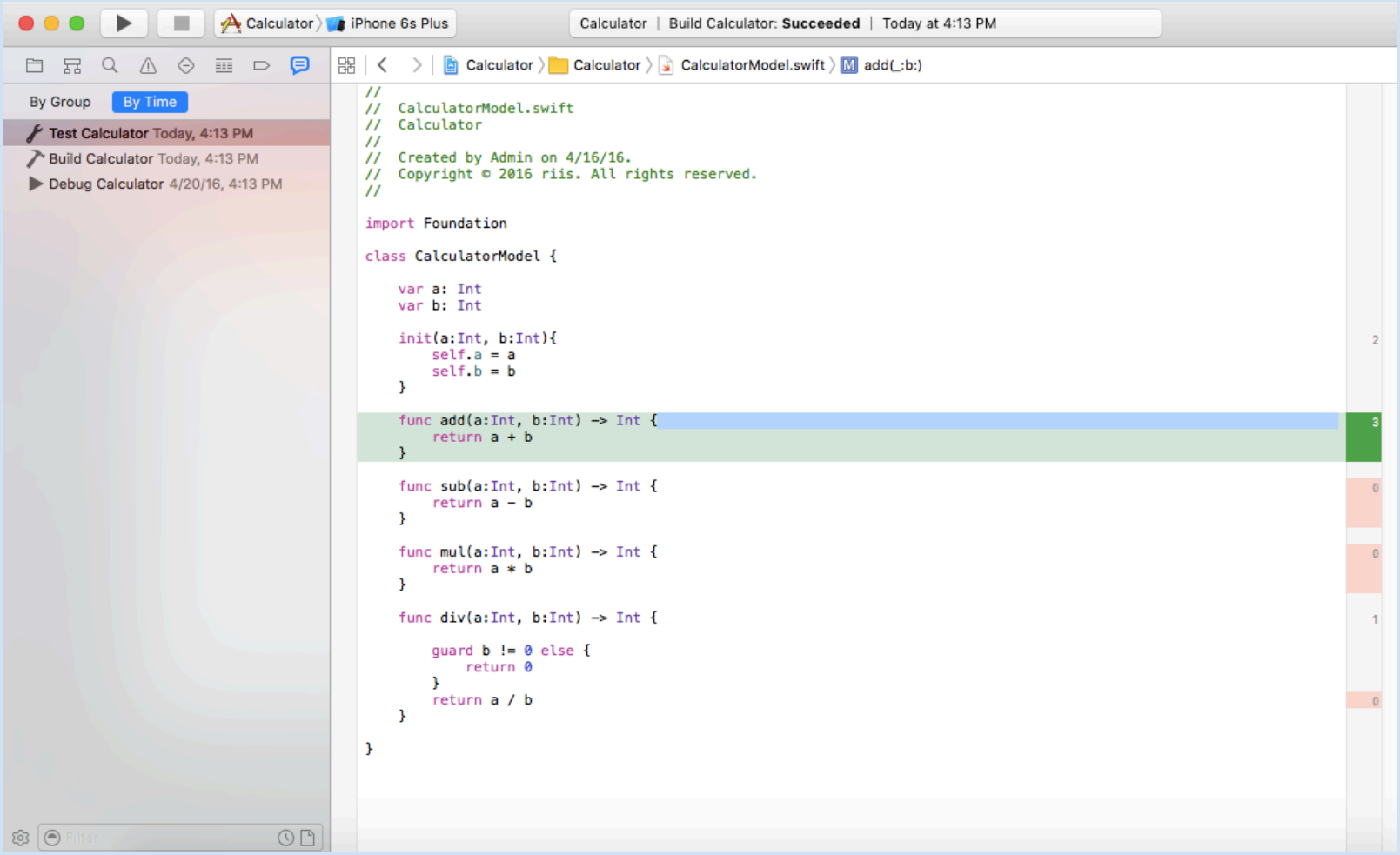
Unit Testing 101



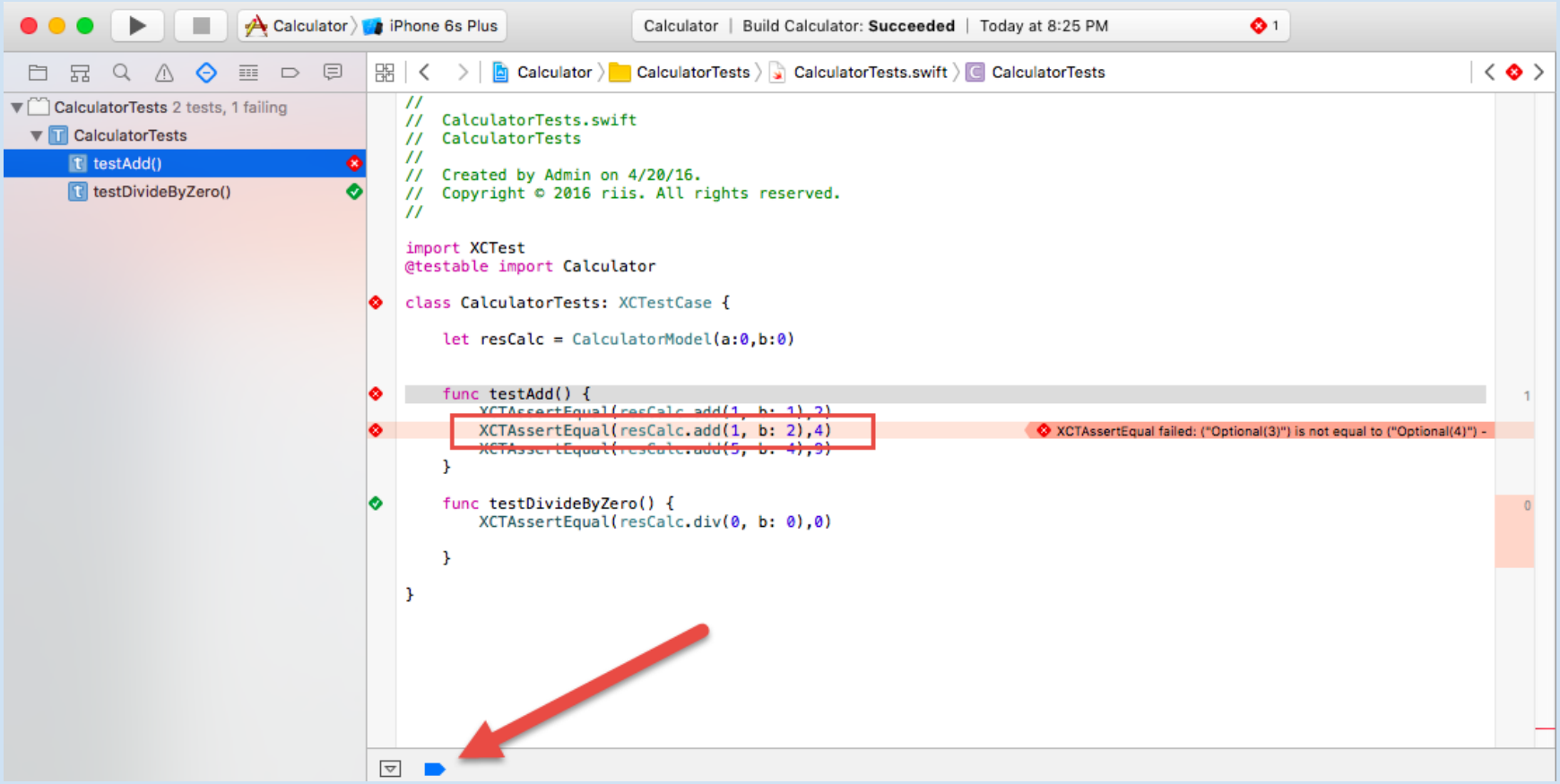
Unit Testing 101



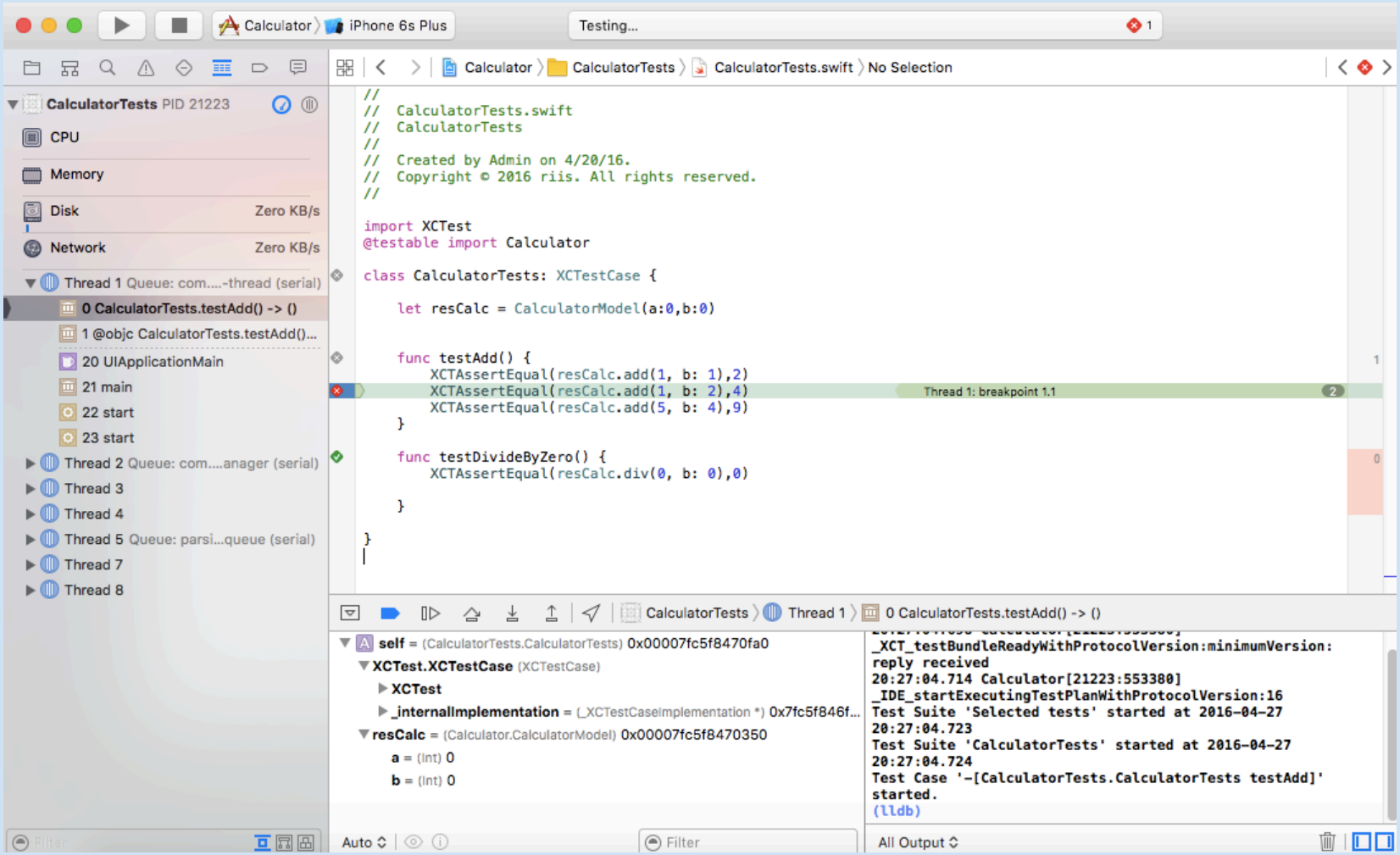
Unit Testing 101



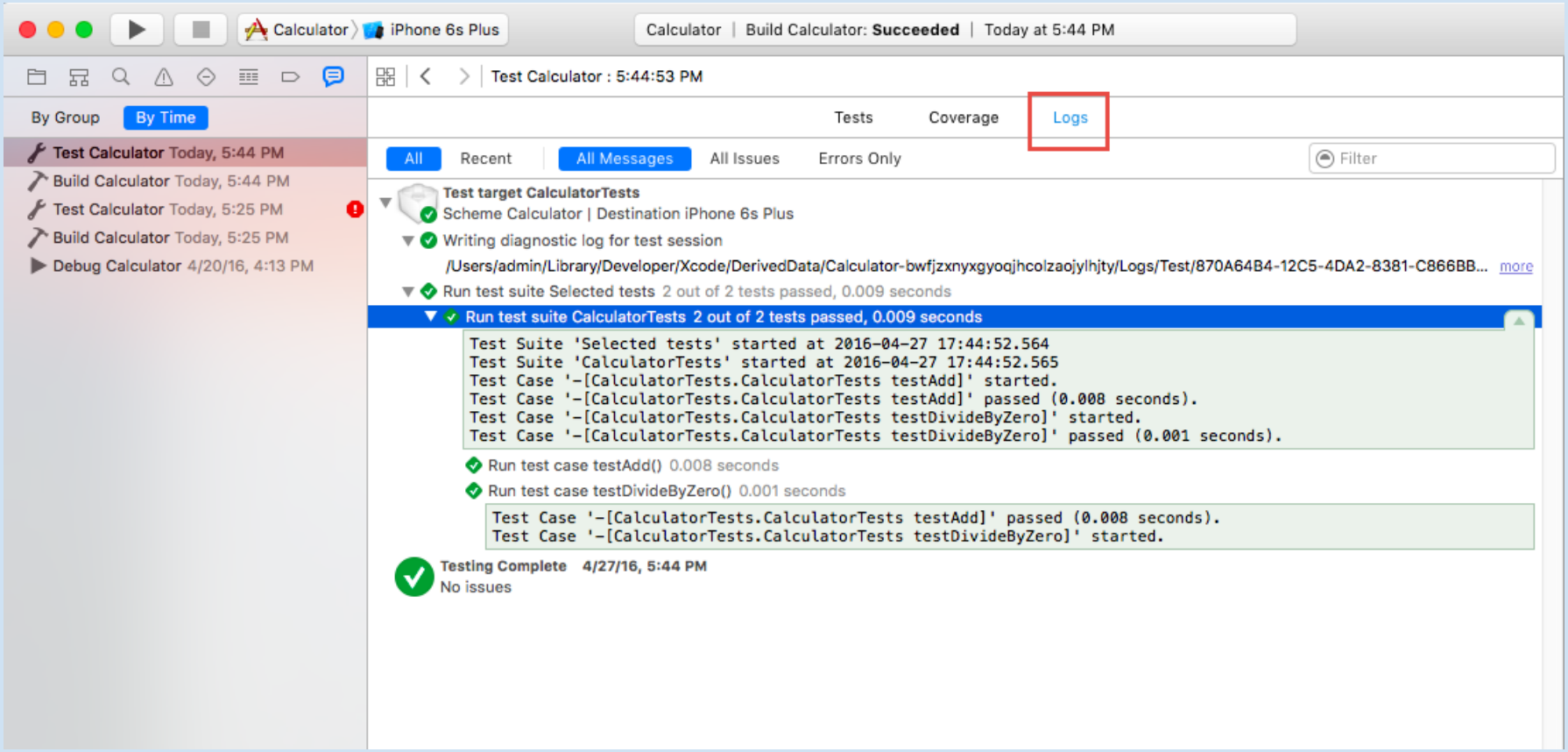
Unit Testing 101



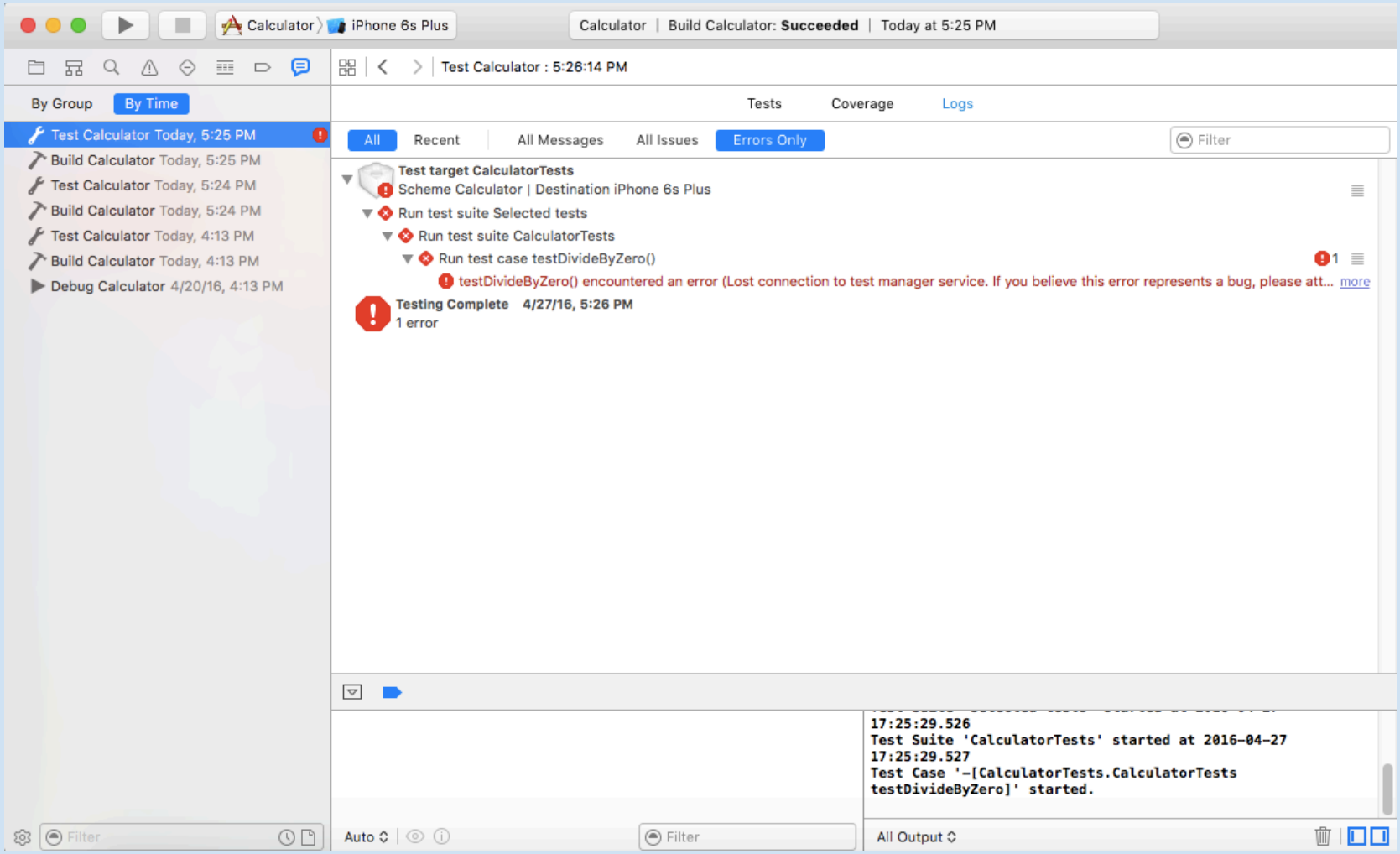
Unit Testing 101



Unit Testing 101



Unit Testing 101



Tools of the Trade

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

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

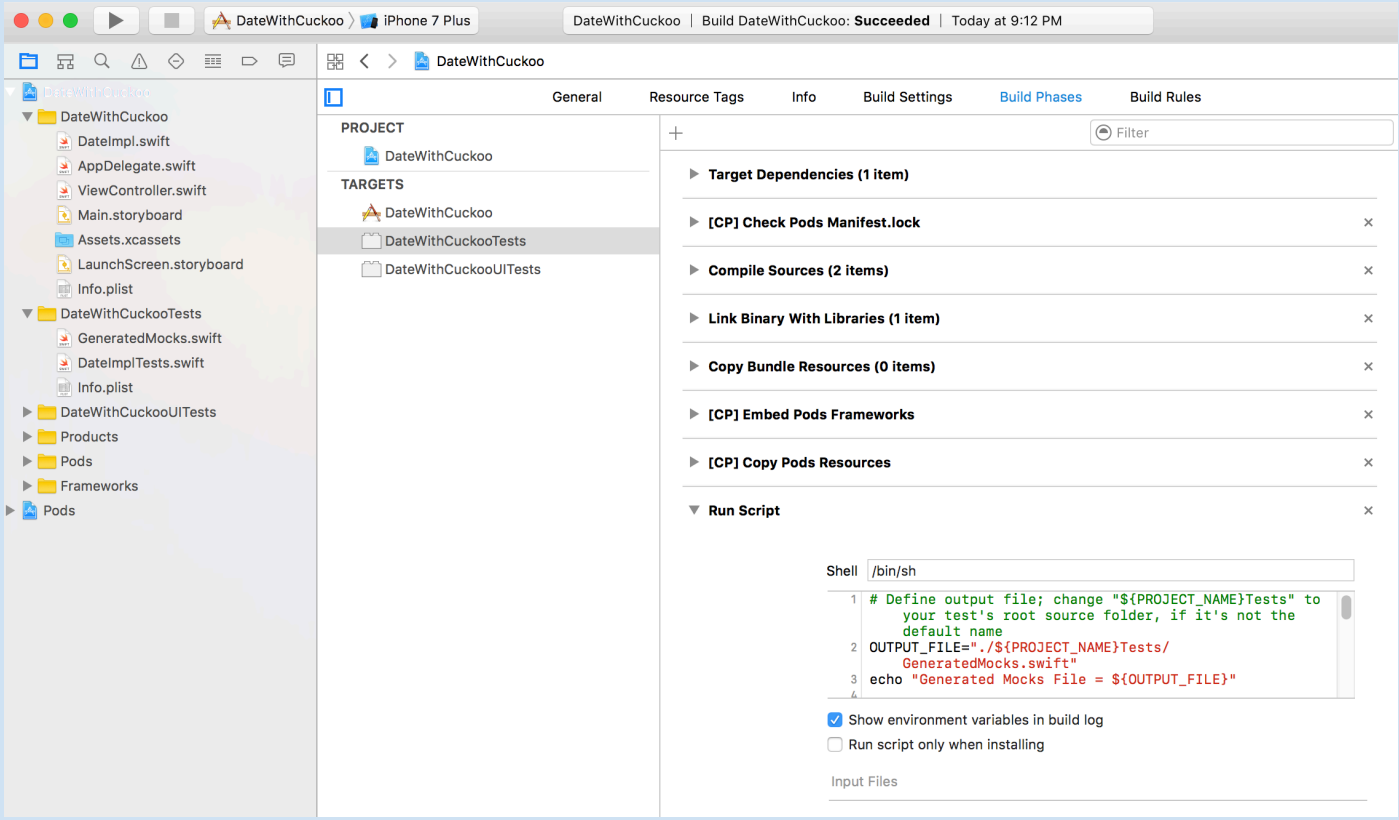
Tools of the Trade

```
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.

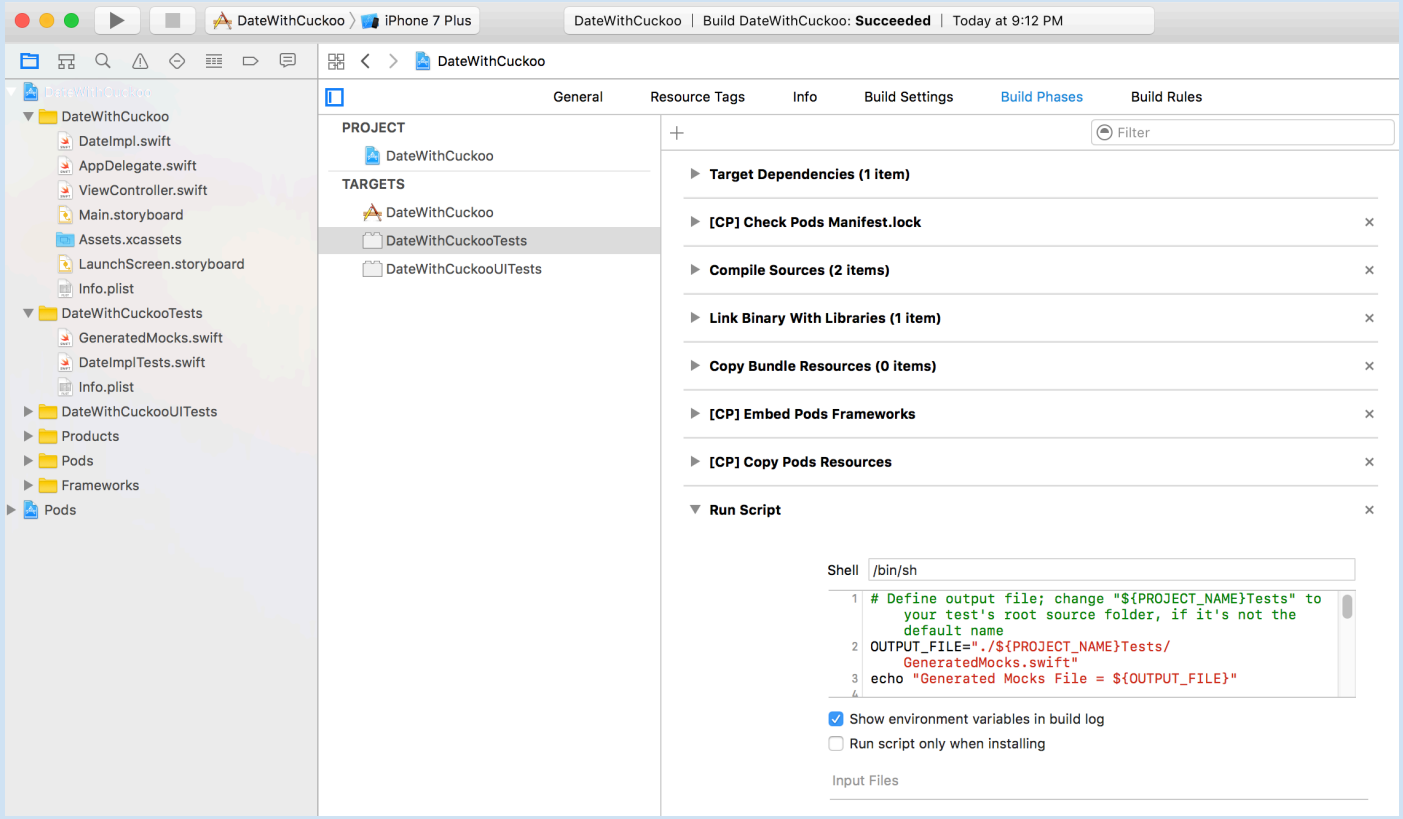
Tools of the Trade

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

target 'DateWithCuckooTests' do

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

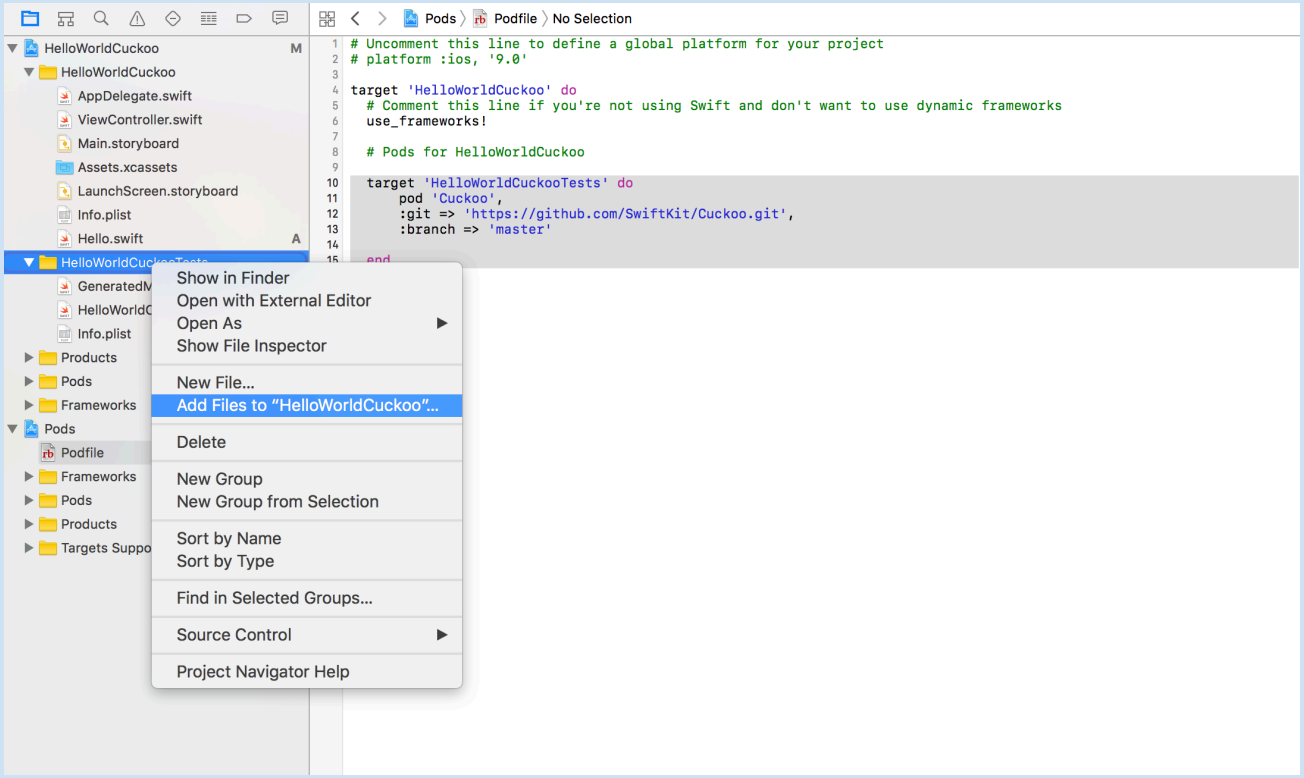
Tools of the Trade

```
# 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



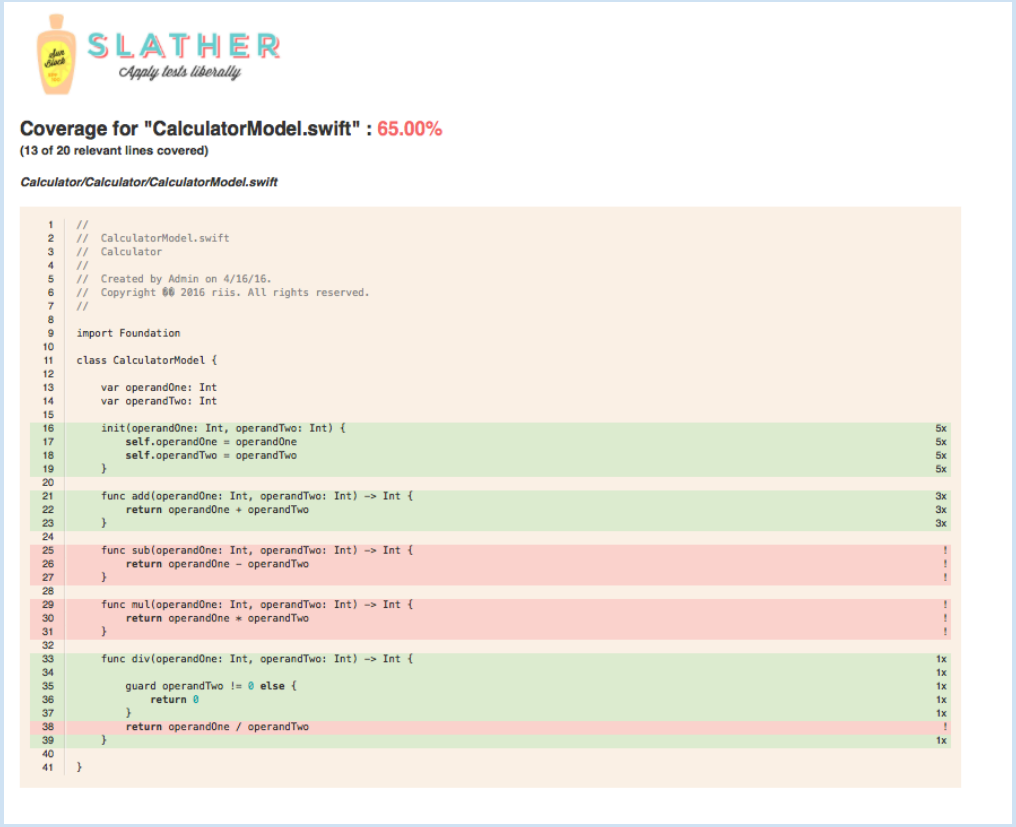
Tools of the Trade



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



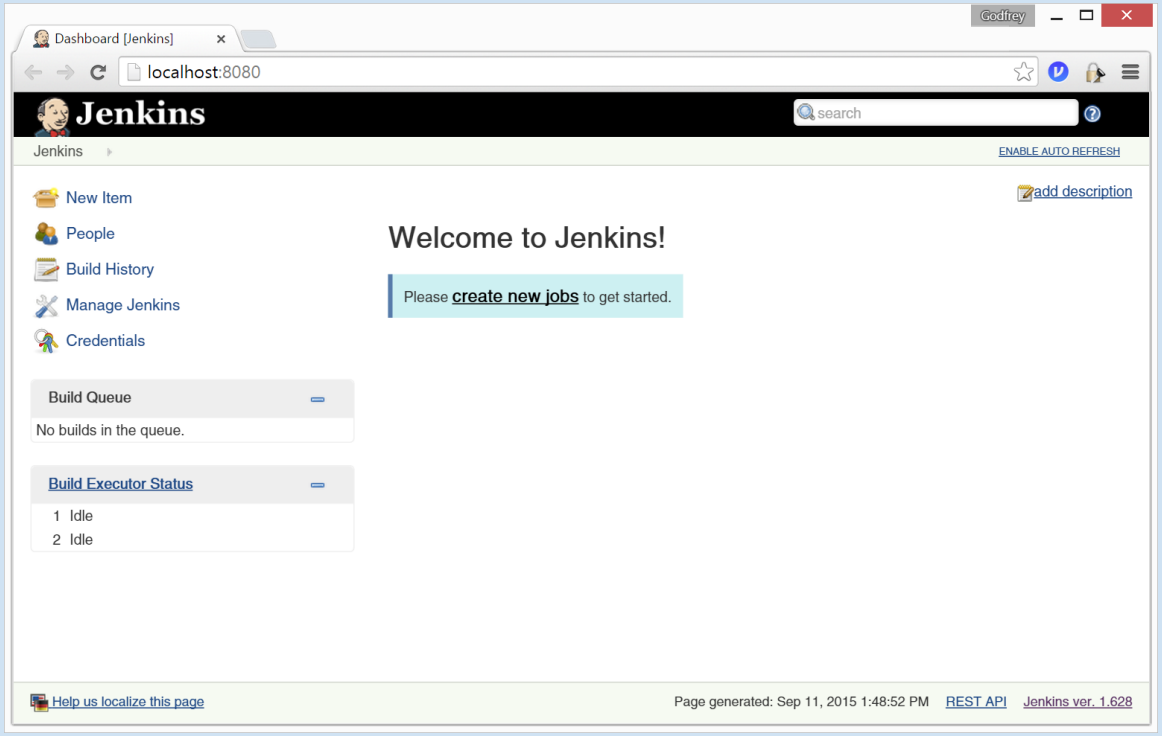
Tools of the Trade

```
$ gem install slather

$ slather coverage --html --scheme XcodeSchemeName path/to/project.xcodeproj

$ slather coverage --html --scheme Calculator Calculator/Calculator.xcodeproj
```

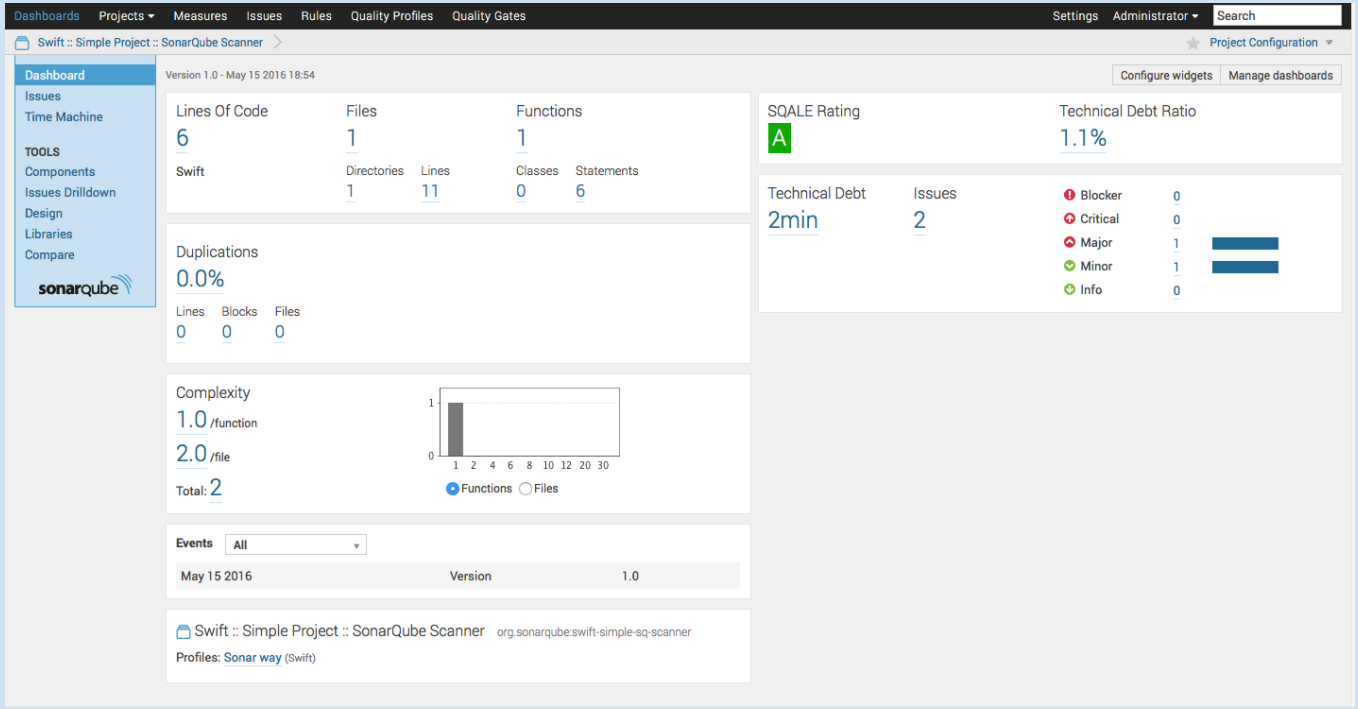
Tools of the Trade

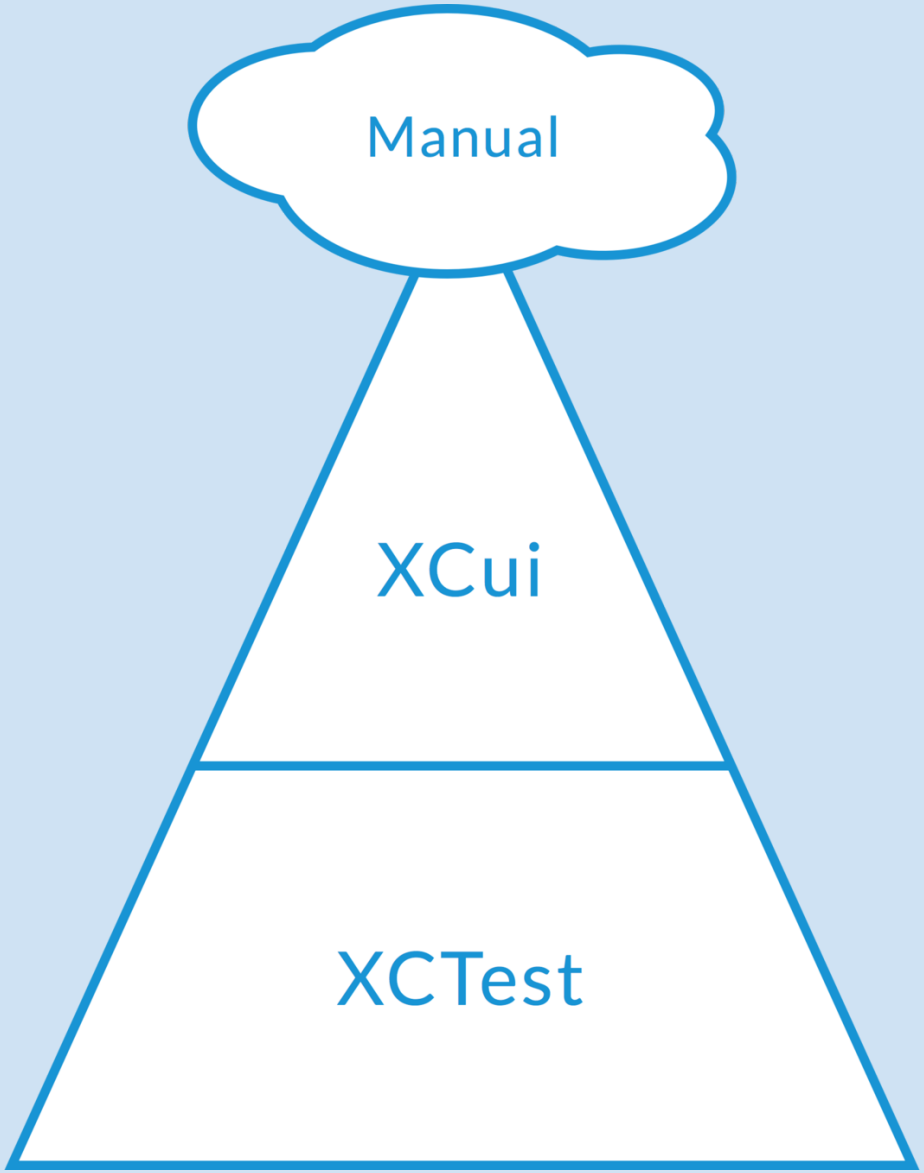
```
# SwiftLint

$ 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

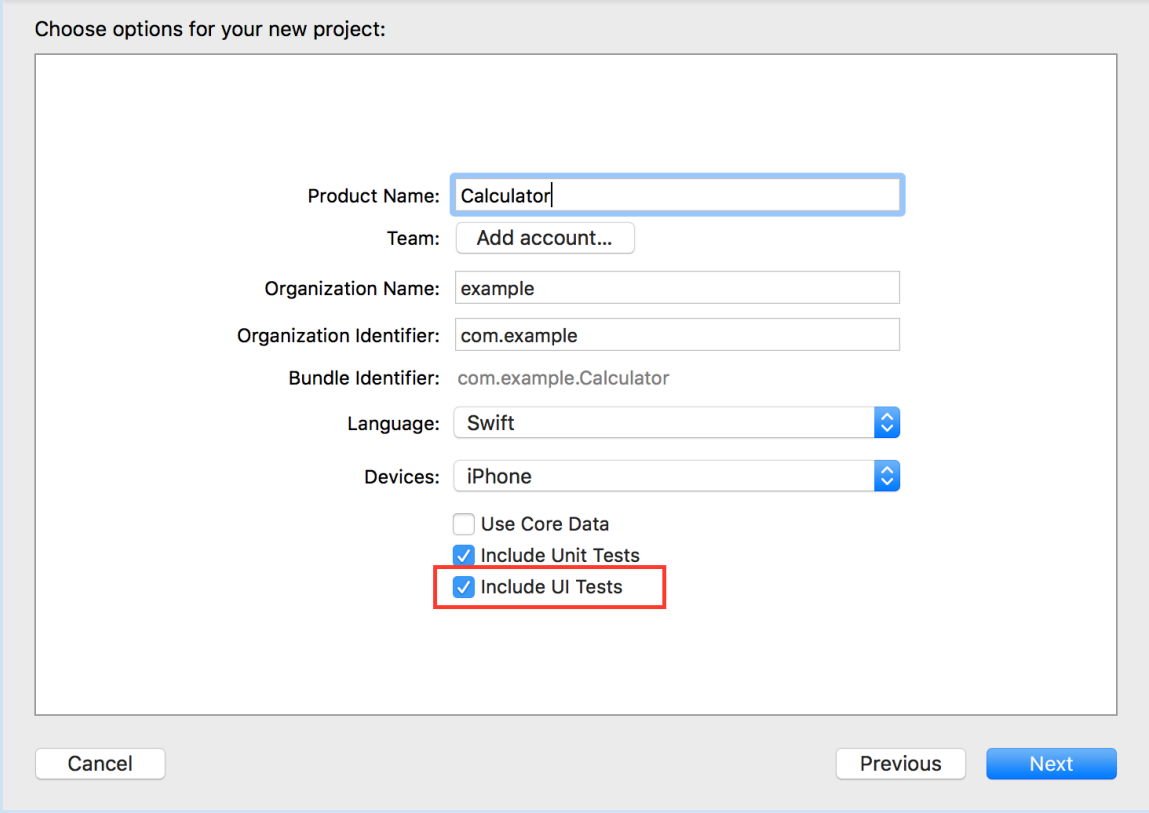



Manual
XCui
XCTest

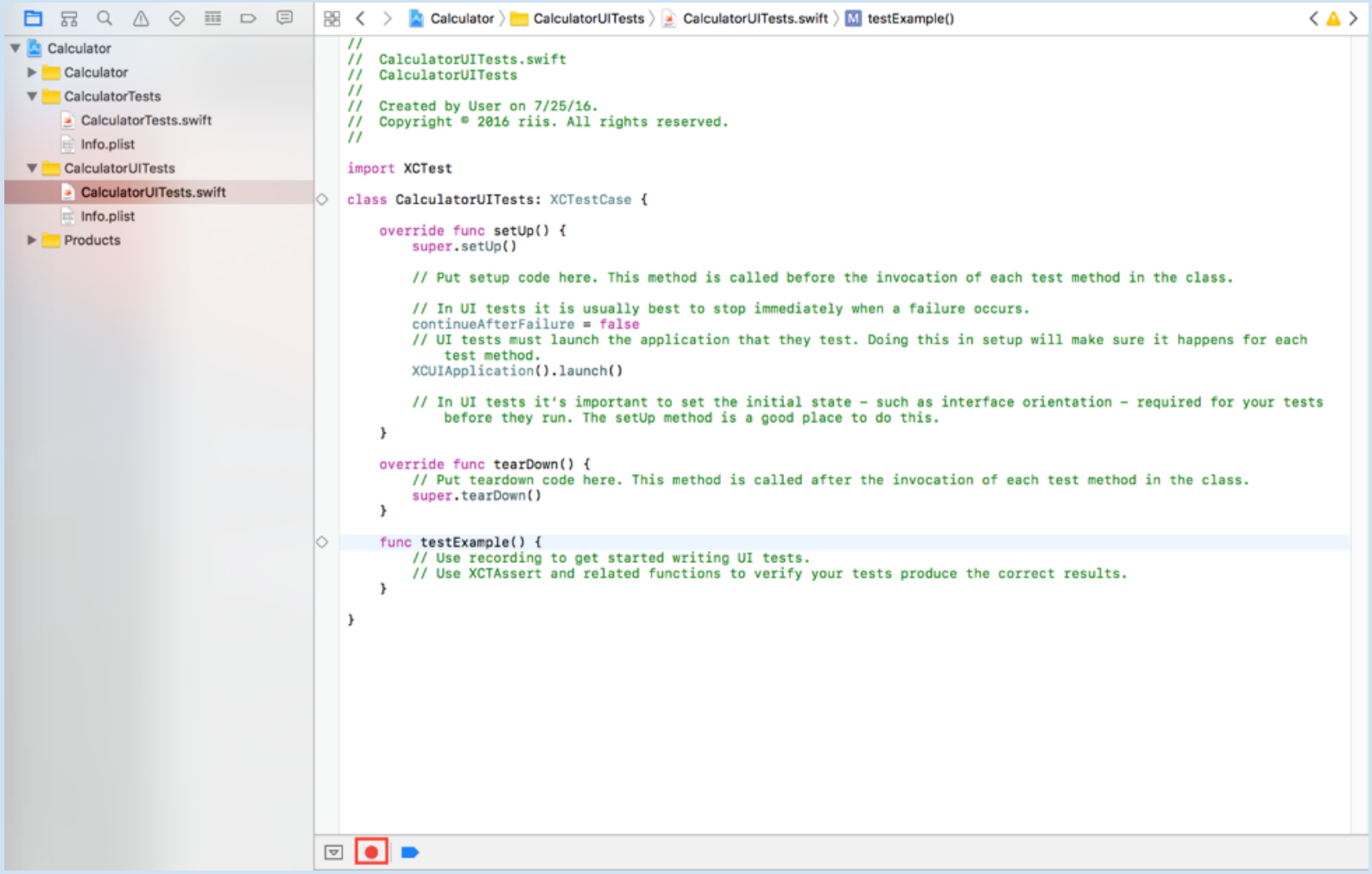
XCUI

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

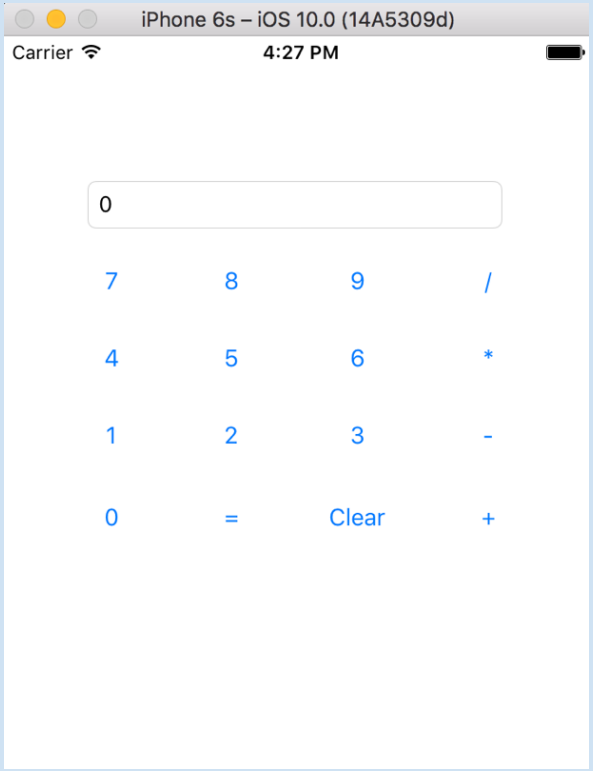
XCUI

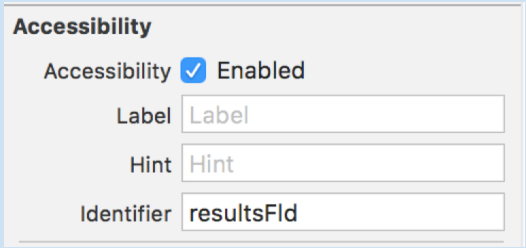

Choose options for your new project:
Product Name: Calculator
Team: Add account...
Organization Name: example
Organization Identifier: com.example
Bundle Identifier: com.example.Calculator
Language: Swift
Devices: iPhone
Use Core Data
Include Unit Tests
Include UI Tests
Cancel
Previous
Next

XCUI


Calculator
Calculator
CalculatorTests
CalculatorTests.swift
Info.plist
CalculatorUITests
CalculatorUITests.swift
Info.plist
Products
Calculator
CalculatorUITests
CalculatorUITests.swift
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.
}
}

XCUI

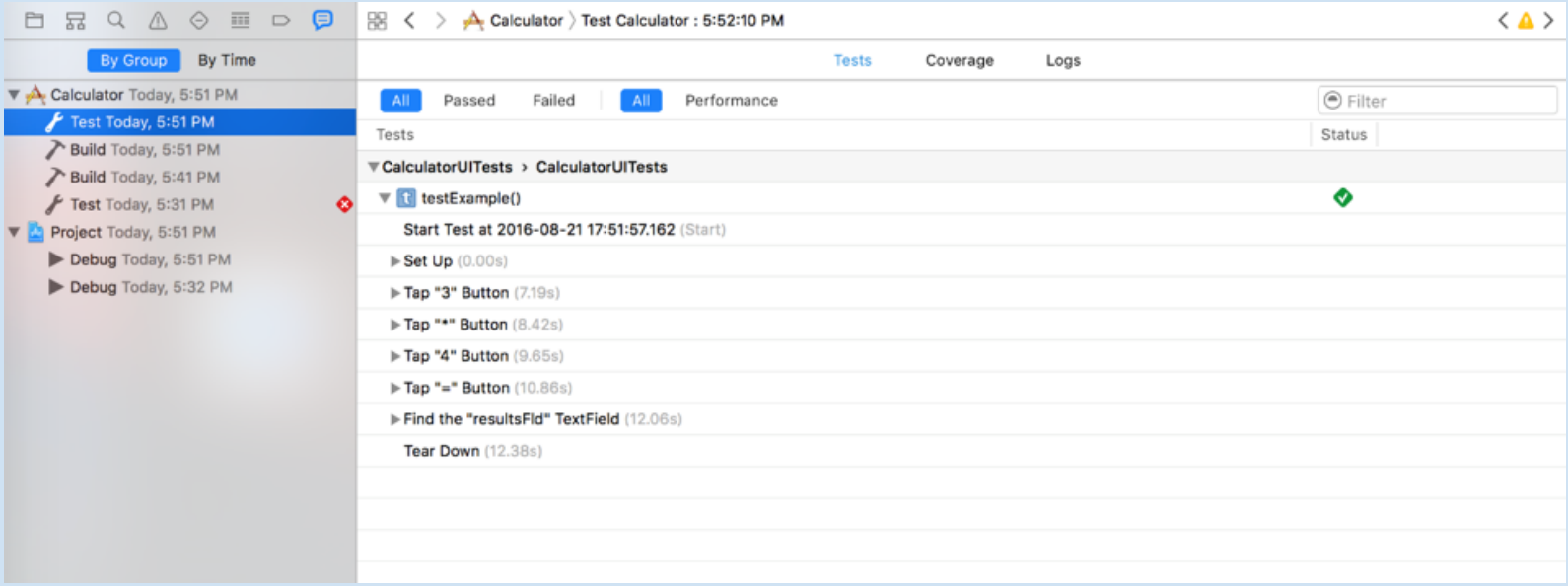

iPhone 6s – iOS 10.0 (14A5309d)
Carrier
4:27 PM
0
7
8
9
/
4
5
6
*
1
2
3
-
0
=
Clear
+

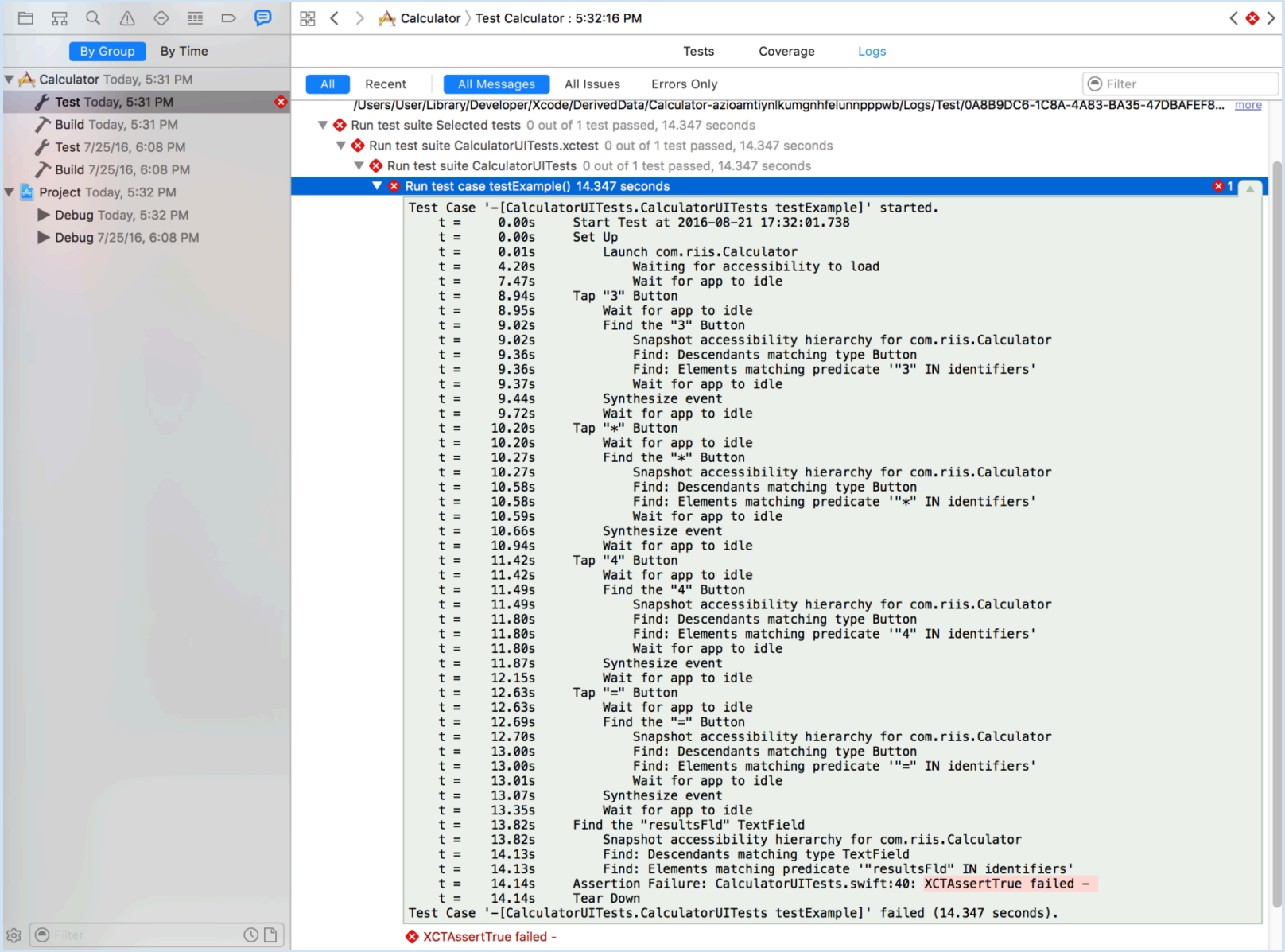

Accessibility
Accessibility
Enabled
Label
Label
Hint
Hint
Identifier
resultsFld

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


By Group
By Time
Calculator Today, 5:51 PM
Test Today, 5:51 PM
Build Today, 5:51 PM
Build Today, 5:41 PM
Test Today, 5:31 PM
Project Today, 5:51 PM
Debug Today, 5:51 PM
Debug Today, 5:32 PM
Calculator 〉 Test Calculator : 5:52:10 PM
Tests
Coverage
Logs
All
Passed
Failed
All
Performance
Filter
Tests
Status
CalculatorUITests › CalculatorUITests
testExample()
Start Test at 2016-08-21 17:51:57.162 (Start)
Set Up (0.00s)
Tap "3" Button (7.19s)
Tap "*" Button (8.42s)
Tap "4" Button (9.65s)
Tap "=" Button (10.86s)
Find the "resultsFld" TextField (12.06s)
Tear Down (12.38s)


By Group
By Time
Calculator Today, 5:31 PM
Test Today, 5:31 PM
Build Today, 5:31 PM
Test 7/25/16, 6:08 PM
Build 7/25/16, 6:08 PM
Project Today, 5:32 PM
Debug Today, 5:32 PM
Debug 7/25/16, 6:08 PM
Calculator 〉 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-azioamtiynlkumgnhfelunnpppwb/Logs/Test/0A8B9DC6-1C8A-4A83-BA35-47DBAFEF8...
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
1
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 -
Filter

XCUI - Roll your own

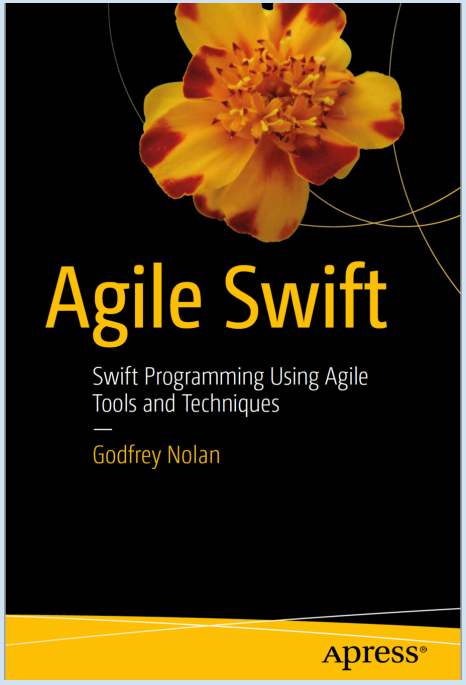
- XCUI has 3 component parts
  - XCUIApplication
  - XCUIElement
  - XCUIElementQuery

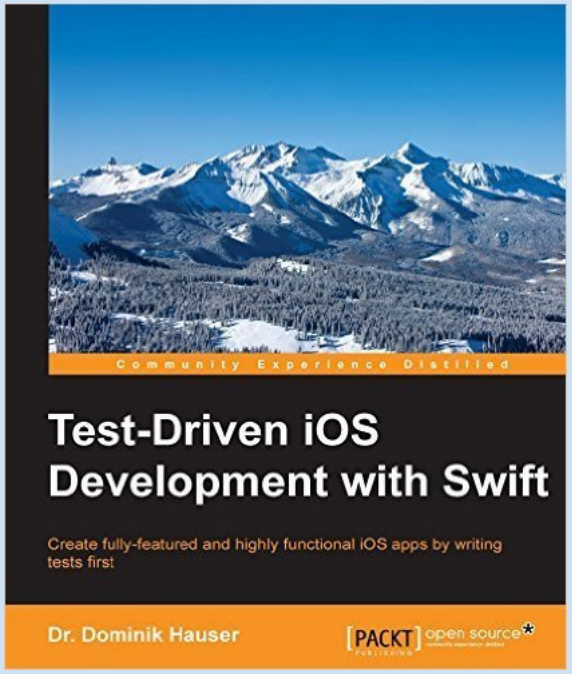
XCUI - Roll your own

- 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


Agile Swift
Swift Programming Using Agile Tools and Techniques
Godfrey Nolan
Apress


Community Experience Distilled
Test-Driven iOS Development with Swift
Create fully-featured and highly functional iOS apps by writing tests first
Dr. Dominik Hauser
PACKT open source

Contact Details

@riisllc, @godfreynolan
godfrey@riis.com
http://riis.com/blog