Testing in Xcode

Ana Calinov, Xcode Engineer
Stuart Montgomery, XCTest Engineer
Ethan Vaughan, XCTest Engineer
Introduction to XCTest
Test plans
Continuous integration workflows
Introduction to XCTest
Test Pyramid

Unit
Test Pyramid

- User interface
- Integration
- Unit
Types of Tests in XCTest
# Types of Tests in XCTest

<table>
<thead>
<tr>
<th>Source code</th>
<th>User perspective</th>
<th>Workflow</th>
<th>Speed</th>
</tr>
</thead>
</table>


# Types of Tests in XCTest

<table>
<thead>
<tr>
<th></th>
<th>Source code</th>
<th>User perspective</th>
<th>Workflow</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Tests</td>
<td>✅</td>
<td></td>
<td></td>
<td>🐇</td>
</tr>
</tbody>
</table>
# Types of Tests in XCTest

<table>
<thead>
<tr>
<th></th>
<th>Source code</th>
<th>User perspective</th>
<th>Workflow</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unit Tests</strong></td>
<td>✅</td>
<td></td>
<td></td>
<td>🐇</td>
</tr>
<tr>
<td><strong>UI Tests</strong></td>
<td></td>
<td>✅</td>
<td>✅</td>
<td>🐢</td>
</tr>
</tbody>
</table>
## Types of Tests in XCTest

<table>
<thead>
<tr>
<th></th>
<th>Source code</th>
<th>User perspective</th>
<th>Workflow</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Tests</td>
<td>✅</td>
<td></td>
<td></td>
<td>🐇</td>
</tr>
<tr>
<td>UI Tests</td>
<td></td>
<td>✅</td>
<td>✅</td>
<td>🐌</td>
</tr>
<tr>
<td>Performance Tests</td>
<td>✅</td>
<td>✅</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Testing from Scratch

Choose options for your new project:

Product Name: MyiOSApp
Team: Add account...
Organization Name: Johnny Appleseed
Organization Identifier: com.example
Bundle Identifier: com.example.MyiOSApp
Language: Swift

- Use SwiftUI
- Use Core Data
- Use CloudKit
- Include Unit Tests
- Include UI Tests

Cancel
Previous
Next
Testing from Scratch

Choose options for your new project:

- Product Name: MyiOSApp
- Team: Add account...
- Organization Name: Johnny Appleseed
- Organization Identifier: com.example
- Bundle Identifier: com.example.MyiOSApp
- Language: Swift
  - Use SwiftUI
  - Use Core Data
  - Use CloudKit
- Include Unit Tests
- Include UI Tests

Buttons: Cancel, Previous, Next
// MyIOSAppTests.swift
// MyIOSAppTests

// Created by Johnny Appleseed on 6/29/19.
// Copyright © 2019 Johnny Appleseed. All rights reserved.

import XCTest

testable import MyIOSApp

class MyIOSAppTests: XCTestCase {

  override func 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.
  }

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

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

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()
        XCTAssertEqual(counter.value, 1, "Counter was not incremented: \(counter)"
    }
}

// How to use XCTest

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()

        XCTAssertEqual(counter.value, 1,
                        "Counter was not incremented: \(counter)"
                        )
    }
}

// How to use XCTest

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()

        XCTAssertEqual(counter.value, 1,
                        "Counter was not incremented: \(counter)")
    }

}
// How to use XCTest

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()

        XCTAssertEqual(counter.value, 1,
                       "Counter was not incremented: \(counter)"
        )
    }
}
// How to use XCTest

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()

        XCTAssertEqual(counter.value, 1,
                 "Counter was not incremented: \(counter")
    }
}

// How to use XCTest

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()

        XCTAssertEqual(counter.value, 1, "Counter was not incremented: \(counter)"
    }
}

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {
    func testIncrementCounter() {
        var counter = Counter()
        counter.increment()
        XCTAssertEqual(counter.value, 1, "Counter was not incremented: \(counter)")
    }
}
// How to use XCTest

import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

  func testIncrementCounter() {
    var counter = Counter()
    counter.increment()
    XCTAssertEqual(counter.value, 1,
      "Counter was not incremented: \(counter)"
    )
  }
}
import XCTest
@testable import MyCounterApp

class MyCounterTests: XCTestCase {

    override func setUp() {
        // Put setup code here.
    }

    override func tearDown() {
        // Put teardown code here.
    }
}
Test Execution Flow

1. setUp() → testA() → tearDown()
2. setUp() → testB() → tearDown()
3. setUp() → testC() → tearDown()
Demo
Test Organization

Unit Test Target

UI Test Target
Test Organization

Unit Test Target

- Test class A
- Test class B
- Test class C

UI Test Target

- Test class D
- Test class E
Test Organization

Unit Test Target
- Test class A
- Test class B
- Test class C

UI Test Target
- Test class D
- Test class E
Test Organization

App

Unit Test Target
- Test class A
- Test class B
- Test class C

UI Test Target
- Test class D
- Test class E
Test Organization

App

UI Test Target

Unit Test Target
Test Organization

App
  - UI Test Target
  - Unit Test Target

Framework
  - Unit Test Target
Code Coverage
```swift
var isRunning: Bool {
    guard case .running? = timer?.state else { return false }
    return true
}

var remainingTime: String {
    let time = duration.rawValue - (timer?.elapsedTime ?? 0)
    return MediationController.timeFormatter.string(from: time) ?? "?"
}

func toggle() {
    guard timer == nil else { timer!.toggle(); return }
    timer = MediationTimer()
    Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { scheduledTimer in
        if self.timer!.elapsedTime >= self.duration.rawValue {
            scheduledTimer.invalidate()
            self.timer = nil

            // Write to the health store
            let finishTimestamp = Date()
            let startTimestamp = finishTimestamp.addingTimeInterval(-self.duration.rawValue)
            let session = HealthSession(start: startTimestamp, finish: finishTimestamp)
            self.allSessions.append(session)
            self.healthStore.save(session: session) { _ in }
        }
        self.didChange.send()
    }
}

private static let timeFormatter: DateComponentsFormatter = {
    let formatter = DateComponentsFormatter()
    // Set custom formatting options
    // formatter.stringStyle = .standard
    // formatter.dateStyle = .full
    // ...
    return formatter
}
```
var isRunning: Bool {
    guard case .running? = timer?.state else { return false }
    return true
}

var remainingTime: String {
    let time = duration.rawValue - (timer?.elapsedTime ?? 0)
    return MeditiationController.timeFormatter.string(from: time) ?? "?"
}

func toggle() {
    guard timer == nil else { timer!.toggle(); return }

    timer = MeditationTimer()

    Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { scheduledTimer in
        if self.timer?.elapsedTime ?? self.duration.rawValue {
            scheduledTimer.invalidate()
            self.timer = nil

            // Write to the health store
            let finishTimestamp = Date()
            let startTimestamp = finishTimestamp.addingTimeInterval(-self.duration.rawValue)
            let session = HealthSession(start: startTimestamp, finish: finishTimestamp)
            self.allSessions.append(session)
            self.healthStore.save(session: session) { _ in }
        }
    }

    self.didChange.send()
}

private static let timeFormatter: DateComponentsFormatter = {
    let formatter = DateComponentsFormatter()
}
var isRunning: Bool {
    guard case .running? = timer?.state else { return false }
    return true
}

var remainingTime: String {
    let time = duration.rawValue - (timer?.elapsedTime ?? 0)
    return MeditationController.timeFormatter.string(from: time) ?? "?"
}

func toggle() {
    guard timer == nil else { timer!.toggle(); return }
    timer = MeditationTimer()
    Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { scheduledTimer in
        if self.timer?.elapsedTime >= self.duration.rawValue {
            scheduledTimer.invalidate()
            self.timer = nil
        }

        // Write to the health store
        let finishTimestamp = Date()
        let startTimestamp = finishTimestamp.addingTimeInterval(-self.duration.rawValue)
        let session = HealthSession(start: startTimestamp, finish: finishTimestamp)
        self.allSessions.append(session)
        self.healthStore.save(session: session) { _ in }
    }
    self.didChange.send()
}

private static let timeFormatter: DateComponentsFormatter = {
    let formatter = DateComponentsFormatter()
```swift
var isRunning: Bool {
    guard case .running? = timer?.state else { return false }
    return true
}

var remainingTime: String {
    let time = duration.rawValue - (timer?.elapsed ?? 0)
    return MeditationController.timeFormatter.string(from: time) ?? "?"
}

func toggle() {
    guard timer == nil else { timer1.toggle(); return }
    timer = MeditationTimer()
    Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { scheduledTimer in
        if self.timer!.elapsedTime >= self.duration.rawValue {
            scheduledTimer.invalidate()
            self.timer = nil
        }

        // Write to the health store
        let finishTimestamp = Date()
        let startTimestamp = finishTimestamp.addingTimeInterval(-self.duration.rawValue)
        let session = HealthSession(start: startTimestamp, finish: finishTimestamp)
        self.allSessions.append(session)
        self.healthStore.save(session: session) { _ in }
    }
    self.didChange.send()
}

private static let timeFormatter: DateComponentsFormatter = {
    let formatter = DateComponentsFormatter()
    return formatter
}
```
```swift
var isRunning: Bool {
    guard case .running? = timer?.state else { return false }
    return true
}

var remainingTime: String {
    let time = duration.rawValue - (timer?.elapsedTime ?? 0)
    return MeditationController.timeFormatter.string(from: time) ?? "?"
}

func toggle() {
    guard timer == nil else { timer!.toggle(); return }
    timer = MeditationTimer()

    Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { scheduledTimer in
        if self.timer!.elapsedTime >= self.duration.rawValue {
            scheduledTimer.invalidate()
            self.timer = nil

            // Write to the health store
            let finishTimestamp = Date()
            let startTimestamp = finishTimestamp.addingTimeInterval(-self.duration.rawValue)
            let session = HealthSession(start: startTimestamp, finish: finishTimestamp)
            self.allSessions.append(session)
            self.healthStore.save(session: session) { _ in }
        }
    }
}

self.didChange.send()

private static let timeFormatter: DateComponentsFormatter = {
    let formatter = DateComponentsFormatter()
    return formatter
}
```
```swift
var isRunning: Bool {
    guard case .running = timer?.state else { return false }
    return true
}

var remainingTime: String {
    let time = duration.rawValue - (timer?.elapsed ?? 0)
    return MeditationController.timeFormatter.string(from: time) ?? "?"
}

func toggle() {
    guard timer == nil else { timer!.toggle(); return }
    timer = MeditationTimer()
    Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true) { scheduledTimer in
        if self.timer!.elapsed >= self.duration.rawValue {
            scheduledTimer.invalidate()
            self.timer = nil
        }
        // Write to the health store
        let finishTimestamp = Date()
        let startTimestamp = finishTimestamp.addingTimeInterval(-self.duration.rawValue)
        let session = HealthSession(start: startTimestamp, finish: finishTimestamp)
        self.allSessions.append(session)
        self.healthStore.save(session: session) { _ in }
    }
}

self.didChange.send()

private static let timeFormatter: DateComponentsFormatter = {
    let formatter = DateComponentsFormatter()
    ```
Test Early, Test Often
Test Early, Test Often

Code

Test

Code Coverage
Test Early, Test Often

Diagram:
- Code
- Test
- Code Coverage
Test Early, Test Often

- Code
- Code Coverage
- Test

Diagram shows the relationship between code and code coverage, indicating that testing should occur early and often.
Test Plans

Stuart Montgomery, XCTest Engineer
Getting the Most Out of Your Tests

Run test suite multiple ways to catch more bugs
Leverage more of Xcode’s testing options
Journal

May, 2019

San Francisco

May, 2019

New York City
Journal
May, 2019
San Francisco

May, 2019
New York City
Tagebuch

Mai, 2019

VACATION_TITLE

Mai, 2019

VACATION_TITLE
Getting the Most Out of Your Tests
Running multiple ways

Different localizations
Random order
Using sanitizers
Arguments or environment variables
Editing App Launch Settings
Scheme editor, run action
Test Plans
Test Plans
Overview

Allows running tests more than once with different settings
Defines all testing variants in one place
Can share between multiple schemes
Supported in Xcode and xcodebuild for CI / Xcode Server
Easy to adopt in existing projects
Demo

Using test plans
Test Plan File

JSON file with .xctestplan extension

Contains

- Tests to run
- Test configurations

Included in project structure

Referenced by schemes
Test Configuration

Describes a single run of the plan’s tests

Has a customizable, unique name

Includes options for how to build and run tests

Inherits shared settings
Test Configuration
Available options

Command-line arguments and environment variables

Language and region

Sanitizers (ASan, TSan, UBSan) and Main Thread Checker

Simulated location

UI testing screenshots and test attachments lifetime

Randomized test execution order

Memory diagnostics (e.g. MallocStackLogging, NSZombies)
Converting a Scheme to Use Test Plans
Converting a Scheme to Use Test Plans
Converting a Scheme to Use Test Plans
Converting a Scheme to Use Test Plans
Converting a Scheme to Use Test Plans
Converting a Scheme to Use Test Plans
Converting a Scheme to Use Test Plans

This scheme will be upgraded to use Test Plans, which allows you to run your tests under different configurations.

- Create Test Plan from scheme
- Create empty Test Plan
- Choose Test Plan

Duplicate Scheme  Manage Schemes...  Convert to use Test Plans...  Close
Converting a Scheme to Use Test Plans

Convert scheme “Travel” to use Test Plans
This scheme will be upgraded to use Test Plans, which allows you to run your tests under different configurations.

- Create Test Plan from scheme
  Create a test plan with the same tests and settings as this scheme.

- Create empty Test Plan
  Create an empty test plan and add it to the scheme.

- Choose Test Plan
  Pick an existing test plan to add to the scheme.

[Options]
[Convert...]
[Cancel]
Converting a Scheme to Use Test Plans

Convert scheme “Travel” to use Test Plans

This scheme will be upgraded to use Test Plans, which allows you to run your tests under different configurations.

- Create Test Plan from scheme
  Create a test plan with the same tests and settings as this scheme.

- Create empty Test Plan
  Create an empty test plan and add it to the scheme.

- Choose Test Plan
  Pick an existing test plan to add to the scheme.

[Buttons: Cancel, Convert...]
Converting a Scheme to Use Test Plans
Potential Use Cases
Potential Use Cases

Different sanitizers

- **ASan**: Address Sanitizer
- **TSan**: Thread Sanitizer
Potential Use Cases
Different sanitizers

- ASan + UBSan
  - Address Sanitizer
  - Undefined Behavior Sanitizer

- TSan + UBSan
  - Thread Sanitizer
  - Undefined Behavior Sanitizer
Potential Use Cases
Different sanitizers

Shared Settings

- ASan + UBSan: Address Sanitizer, Undefined Behavior Sanitizer
- TSan + UBSan: Thread Sanitizer, Undefined Behavior Sanitizer
Potential Use Cases
Different sanitizers

- **ASan + UBSan**
  - Address Sanitizer
  - Undefined Behavior Sanitizer

- **TSan + UBSan**
  - Thread Sanitizer
  - Undefined Behavior Sanitizer

**Shared Settings**
- Undefined Behavior Sanitizer
Potential Use Cases
Simulated languages, locations, and locales

- English / United States
- Korean / South Korea
- Italian / Italy
## Potential Use Cases

Simulated languages, locations, and locales

<table>
<thead>
<tr>
<th>Shared Settings</th>
<th>English / United States</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Korean / South Korea</td>
<td>₩</td>
</tr>
<tr>
<td></td>
<td>Italian / Italy</td>
<td>€</td>
</tr>
</tbody>
</table>
Potential Use Cases
Simulated languages, locations, and locales

- **English / United States**
  - Currency: $
  - Localization Screenshots

- **Korean / South Korea**
  - Currency:₩
  - Localization Screenshots

- **Italian / Italy**
  - Currency: €
  - Localization Screenshots
Potential Use Cases
Simulated languages, locations, and locales

Creating Great Localized Experiences with Xcode 11
Potential Use Cases

Mix-and-match

- Memory Checking
- Concurrency
- Extra Diagnostics
Potential Use Cases
Mix-and-match

- Memory Checking
- Address Sanitizer
- Zombie Objects

- Concurrency

- Extra Diagnostics
Potential Use Cases
Mix-and-match

- Memory Checking: Address Sanitizer, Zombie Objects
- Concurrency: Thread Sanitizer, Undefined Behavior Sanitizer, Random Order
- Extra Diagnostics
Potential Use Cases
Mix-and-match

- Memory Checking
  - Address Sanitizer
  - Zombie Objects

- Concurrency
  - Thread Sanitizer
  - Undefined Behavior Sanitizer
  - Random Order

- Extra Diagnostics
  - ENABLE_LOGGING=1
  - Keep Attachments
Shared Settings

Memory Checking

Arguments

Arguments Passed On Launch

Environment Variables

Localization

Application Language

System Language

Application Region

System Region

Simulated Location

None

UI Testing

Automatic Screenshots

On, and delete if test succeeds

Localization Screenshots

Off

Attachments

Attachments

On, and delete if test succeeds

Test Execution

Execution Order

Random

Code Coverage

Code Coverage

On

Runtime Sanitization

Address Sanitizer

On, and detect stack use after return

Thread Sanitizer

Off

Undefined Behavior Sanitizer

On

Runtime API Checking

MainThread Checker

On

Memory Management

Malloc Tracing

Off

Malloc Guard Edges

Off

Guard Mallocc

Off

Zombie Objects

On

Malloc Stack Logging

Off
Continuous Integration Workflows

Ethan Vaughan, XCTest Engineer
CI Workflows

Xcode Server

Custom
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Step 3: Populate issue tracker

Step 4: Track code coverage

Build and test results

Issue Tracker

Coverage %
Build Your Own CI

1. Build tests
2. Run tests
3. Populate issue tracker
4. Track code coverage

Coverage %

- Issue Tracker
- Build and test results
- Step 3: Populate issue tracker
- Step 4: Track code coverage

Build Your Own CI
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

Step 1: Build tests
Step 2: Run tests
Step 3: Populate issue tracker
Step 4: Track code coverage

Build and test results

Coverage %

0 25 50 75 100
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %

0 25 50 75 100

Issue Tracker
Build Your Own CI

Step 1: Build tests
Step 2: Run tests
Step 3: Populate issue tracker
Step 4: Track code coverage

Issue Tracker

Build and test results

Coverage %
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %

0 25 50 75 100
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %

Issue Tracker
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

Step 1: Build tests
Step 2: Run tests

Build and test results

Issue Tracker

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Step 3: Populate issue tracker

Step 4: Track code coverage

Build and test results

Coverage %
Build Your Own CI

1. Build tests
2. Run tests
3. Populate issue tracker
4. Track code coverage

Issue Tracker

Build and test results

Coverage %
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %

0 25 50 75 100

Build and test results
xcodebuild
Running Tests with xcodebuild
Build and test together

```bash
$ xcodebuild test
   -project MyProject.xcodeproj
   -scheme MyScheme
   -destination 'platform=iOS,name=My iPhone'
```
Running Tests with xcodebuild
Build, then test

```
$ xcodebuild build-for-testing
   -project MyProject.xcodeproj
   -scheme MyScheme
   -destination 'platform=iOS,name=My iPhone'

$ xcodebuild test-without-building
   -xctestrun MyProject.xctestrun
   -destination 'platform=iOS,name=My iPhone'
```
Running Tests with xcodebuild
Build, then test

```
$ xcodebuild build-for-testing
  -project MyProject.xcodeproj
  -scheme MyScheme
  -destination 'platform=iOS,name=My iPhone'

$ xcodebuild test-without-building
  -xctestrun MyProject.xctestrun
  -destination 'platform=iOS,name=My iPhone'
```
Running Tests with xcodebuild
Build, then test

$ xcodebuild build-for-testing
   -project MyProject.xcodeproj
   -scheme MyScheme
   -destination 'platform=iOS,name=My iPhone'

$ xcodebuild test-without-building
   -xctestrun MyProject.xctestrun
   -destination 'platform=iOS,name=My iPhone'
Running Tests with xcodebuild
Build, then test

$ xcodebuild build-for-testing
   -project MyProject.xcodeproj
   -scheme MyScheme
   -destination 'platform=iOS,name=My iPhone'

$ xcodebuild test-without-building
   -xctestrun MyProject.xctestrun
   -destination 'platform=iOS,name=My iPhone'
NAME

xcodebuild.xcteestrun -- Test run parameters files for xcodebuild

DESCRIPTION

This document details the parameters contained in an xcteestrun file.

...
Testing on Multiple Destinations

$ xcodebuild test
-destination 'platform=iOS,name=My iPad'
-destination 'platform=iOS,name=My iPad mini'
-destination 'platform=iOS,name=My iPhone'
Testing on Multiple Destinations

$xcodebuild test

-destination 'platform=iOS,name=My iPad'
-destination 'platform=iOS,name=My iPad mini'
-destination 'platform=iOS,name=My iPhone'

What's New in Testing

WWDC 2018
xcodebuild and Test Plans
xcodebuild and Test Plans

List a scheme’s test plans

$ xcodebuild -project ... -scheme ... -showTestPlans
Multiple Test Plans
Multiple Test Plans
xcodebuild and Test Plans

List a scheme’s test plans

$ xcodebuild -project ... -scheme ... -showTestPlans
xcodebuild and Test Plans

List a scheme’s test plans

$ xcodebuild -project ... -scheme ... -showTestPlans

Override default test plan

$ xcodebuild test -project ... -scheme ... -testPlan 'Smoke Tests'
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Issue Tracker

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Build Your Own CI

xcodebuild build-for-testing

Step 1: Build tests

Step 2: Run tests

Step 2: Run tests

Step 3: Populate issue tracker

Step 4: Track code coverage

Build and test results

Coverage %

Issue Tracker
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Step 3: Populate issue tracker

Step 4: Track code coverage

Build and test results

xcodebuild build-for-testing

Issue Tracker

Coverage %

0
25
50
75
100
Build Your Own CI

Step 1: Build tests
xcodebuild build-for-testing

Step 2: Run tests
xcodebuild test-without-building
   -destination 'name=iPhone X'
   -destination 'name=iPad'

Build and test results

Step 3: Populate issue tracker

Step 4: Track code coverage

Issue Tracker

Coverage %
Build Your Own CI

- xcodebuild build-for-testing
- xcodebuild test-without-building
  -destination 'name=iPhone X'
  -destination 'name=iPad'

Step 1: Build tests
Step 2: Run tests
Step 3: Populate issue tracker
Step 4: Track code coverage

Coverage %
0 25 50 75 100
Result Bundles
What is a Result Bundle?

File containing the results of building and running tests
What is a Result Bundle?

File containing the results of building and running tests

- Build log
What is a Result Bundle?

File containing the results of building and running tests

- Build log
- Test report
What is a Result Bundle?

File containing the results of building and running tests

- Build log
- Test report
- Code coverage report
What is a Result Bundle?

File containing the results of building and running tests

• Build log
• Test report
• Code coverage report
• Test attachments
Producing Result Bundles

$ xcodebuild test
  -project MyProject.xcodeproj
  -scheme MyScheme
  -resultBundlePath /path/to/ResultBundle.xcresult
Build Your Own CI

Step 1: Build tests

Step 2: Run tests

Step 3: Populate issue tracker

Step 4: Track code coverage

Build and test results

xcodebuild build-for-testing

xcodebuild test-without-building
  -destination 'name=iPhone X'
  -destination 'name=iPad'
Build Your Own CI

Step 1: Build tests

```
xcodebuild build-for-testing
```

Step 2: Run tests

```
xcodebuild test-without-building
...
-resultBundlePath ResultBundle.xcresult
```

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %
Result Bundle Format
Result Bundle Format

Highly optimized, space-efficient format (~4x smaller)
Result Bundle Format

Highly optimized, space-efficient format (~4x smaller)

Viewable in Xcode
Result Bundle Format

Highly optimized, space-efficient format (~4x smaller)

Viewable in Xcode

Programmatically accessible contents
Viewing Result Bundles
Viewing Result Bundles
xcresulttool
xcresulttool

Programmatic access to result bundle contents
Programmatic access to result bundle contents

Structured data available as JSON
xcresulttool

Programmatic access to result bundle contents

Structured data available as JSON

JSON schema is publicly documented
Build Your Own CI

Step 1: Build tests

```
xcodebuild build-for-testing
```

Step 2: Run tests

```
xcodebuild test-without-building
...
-resultBundlePath ResultBundle.xcresult
```
Build Your Own CI

Step 1: Build tests

```
xcodebuild build-for-testing
```

Step 2: Run tests

```
xcodebuild test-without-building
... -resultBundlePath ResultBundle.xcresult
```
Extracting Build and Test Failures
Extracting Build and Test Failures

```
$ xcrun xcresulttool get --path ResultBundle.xcresult --format json
```
Extracting Build and Test Failures

```bash
$ xcrun xcresulttool get --path ResultBundle.xcresult --format json
```

```json
{
  "issues": {
    "errorSummaries": {
      "_values": [
        {
          "message": {
            "_value": "Use of unresolved identifier 'garbage'"
          }
        }
      ]
    }
  }
}```
Extracting Build and Test Failures

```
$ xcrun xcrun xcresulttool get --path ResultBundle.xcrresult --format json
```

```
{
  "issues": {
    "errorSummaries": {
      "_values": [
        {
          "message": {
            "_value": "Use of unresolved identifier 'garbage'
          }
        },
        ...
      ]
    }
  }
}
```
Extracting Build and Test Failures

```bash
$ xcrun xcresulttool get --path ResultBundle.xcresult --format json
```

```json
{
  "issues": {
    "testFailureSummaries": {
      "_values": [
        {
          "message": {
            "_value": "failed - ("1") is not equal to ("2")"
          },
          "testCaseName": {
            "_value": "Calculator.testAddition()"
          }
        }
      ]
    }
  }
}
```
Extracting Build and Test Failures

```
$ xcrun xcrun tool get --path ResultBundle.xcresult --format json

{
    "issues": {
        "testFailureSummaries": {
            "_values": [
                {
                    "message": {
                        "_value": "failed - ("1") is not equal to ("2")"
                    },
                    "testCaseName": {
                        "_value": "Calculator.testAddition()"
                    }
                }
            ]
        }
    }
}
```
Name: Xcode Result Types
Version: 3.18
Types:
- ActionAbstractTestSummary
  * Kind: object
  * Properties:
    + name: String?
- ActionDeviceRecord
  * Kind: object
  * Properties:
    ...

$ xcrun xcrucresulttool formatDescription get
xcresulttool(1)

NAME
   xcresulttool - view Xcode result bundle data in a human-readable or machine-parseable format.

SYNOPSIS
   xcresulttool get --path Example.xcresult [--format json | raw] [--id ID]

   ...
Build Your Own CI

Step 1: Build tests
xcodebuild build-for-testing

Step 2: Run tests
xcodebuild test-without-building
... -resultBundlePath ResultBundle.xcresult

Step 3: Populate issue tracker

Step 4: Track code coverage

Coverage %

Issue Tracker
Build Your Own CI

**Step 1:** Build tests

```
xcodebuild build-for-testing
```

**Step 2:** Run tests

```
xcodebuild test-without-building
...
-resultBundlePath ResultBundle.xcresult
```

**Step 4:** Track code coverage

```
xcresulttool get
```

- Issue Tracker

Coverage %

![Coverage Graph](graph.png)
Build Your Own CI

Step 1: Build tests
xcodebuild build-for-testing

Step 2: Run tests
xcodebuild test-without-building
... -resultBundlePath ResultBundle.xcresult

Step 4: Track code coverage
xcresulttool get

Issue Tracker

Coverage %

0 25 50 75 100
Build Your Own CI

Step 1: Build tests
- `xcodebuild build-for-testing`

Step 2: Run tests
- `xcodebuild test-without-building`
  - `--resultBundlePath ResultBundle.xcresult`

Step 4: Track code coverage
- `xcresulttool get`
Code Coverage Reports with xccov
Code Coverage Reports with xccov

```
$ xcrun xccov view --report ResultBundle.xcresult
```
## Code Coverage Reports with xccov

```shell
$ xcrun xccov view --report ResultBundle.xcresult
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld.app</td>
<td>57.14% (8/14)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.swift</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>AppDelegate.applicationDidFinishLaunching(Notification) -&gt; ()</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>AppDelegate.applicationWillTerminate(Notification) -&gt; ()</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.swift</td>
<td>62.50% (5/8)</td>
</tr>
<tr>
<td>ViewController.viewDidLoad() -&gt; ()</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>ViewController.representedObject.didset : Swift.Optional&lt;Any&gt;</td>
<td>0.00% (0/3)</td>
</tr>
</tbody>
</table>
## Code Coverage Reports with xccov

```
$ xcrun xccov view --report ResultBundle.xcresult
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld.app</td>
<td>57.14% (8/14)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.swift</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>AppDelegate.applicationDidFinishLaunching(Notification) -&gt; ()</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>AppDelegate.applicationWillTerminate(Notification) -&gt; ()</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.swift</td>
<td>62.50% (5/8)</td>
</tr>
<tr>
<td>ViewController.viewDidLoad() -&gt; ()</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>ViewController.representedObject.didset : Swift.Optional&lt;Any&gt;</td>
<td>0.00% (0/3)</td>
</tr>
</tbody>
</table>
### Code Coverage Reports with `xccov`

```bash
$ xcrun xccov view --report ResultBundle.xcresult
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HelloWorld.app</strong></td>
<td>57.14% (8/14)</td>
</tr>
<tr>
<td><code>/tmp/HelloWorld/HelloWorld/AppDelegate.swift</code></td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>AppDelegate.applicationDidFinishLaunching()</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>AppDelegate.applicationWillTerminate()</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td><code>/tmp/HelloWorld/HelloWorld/ViewController.swift</code></td>
<td>62.50% (5/8)</td>
</tr>
<tr>
<td>ViewController.viewDidLoad()</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>ViewController.representedObject.didset : Swift.Optional&lt;Any&gt;</td>
<td>0.00% (0/3)</td>
</tr>
</tbody>
</table>
**Code Coverage Reports with xccov**

```bash
$ xcrun xccov view --report ResultBundle.xcresult
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld.app</td>
<td>57.14% (8/14)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.swift</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>AppDelegate.applicationDidFinishLaunching(NSNotification) -&gt; ()</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>AppDelegate.applicationWillTerminate(NSNotification) -&gt; ()</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.swift</td>
<td>62.50% (5/8)</td>
</tr>
<tr>
<td>ViewController.viewDidLoad() -&gt; ()</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>ViewController.representedObject.didset : Swift.Optional&lt;Any&gt;</td>
<td>0.00% (0/3)</td>
</tr>
</tbody>
</table>
Comparing Code Coverage Reports
Comparing Code Coverage Reports

$ xcrun xccov diff --json Before.xcresult After.xcresult
Comparing Code Coverage Reports

```sh
taxrun xccov diff --json Before.xcresult After.xcresult
```

```json
{
  "documentLocation": "/tmp/HelloWorld/HelloWorld/AppDelegate.swift",
  "lineCoverageDelta": {
    "executableLinesDelta": 0,
    "coveredLinesDelta": 3,
    "lineCoverageDelta": 0.5
  },
  "functionDeltas": [
    {
      "name": "AppDelegate.applicationWillTerminate(Notification) -> ()",
      ...
    }
  ]
}
```
Comparing Code Coverage Reports

$ xcrun xccov diff --json Before.xcresult After.xcresult

{
  "documentLocation": "/tmp/HelloWorld/HelloWorld/AppDelegate.swift",
  "lineCoverageDelta": {
    "executableLinesDelta": 0,
    "coveredLinesDelta": 3,
    "lineCoverageDelta": 0.5
  },
  "functionDeltas": [
    {
      "name": "AppDelegate.applicationWillTerminate(Notification) -> ()",
      ...
    }
  ]
}
NAME

xcov - view Xcode coverage data in human-readable or machine-parseable format.

SYNOPSIS

xcov view [--only-targets | --files-for-target target_name | --functions-for-file name_or_path] [--json] report.xccovreport

...
Build Your Own CI

Step 1: Build tests
xcodebuild build-for-testing

Step 2: Run tests
xcodebuild test-without-building
... -resultBundlePath ResultBundle.xcresult

Step 4: Track code coverage
xcresulttool get

Coverage %

Issue Tracker
Build Your Own CI

xcodebuild build-for-testing

... -resultBundlePath ResultBundle.xcresult

xCRESULT

Issue Tracker

xcresulttool get

xccov view

Coverage %
Build Your Own CI

- xcodebuild build-for-testing
- xcodebuild test-without-building
  ... 
  -resultBundlePath ResultBundle.xcresult
- xcresulttool get
- xccov view

Coverage %

Issue Tracker
Build Your Own CI

xcodebuild build-for-testing

xcodebuild test-without-building
    ...
    -resultBundlePath ResultBundle.xcresult

xcresulttool get

xccov view

Coverage %
Summary
Write tests in Xcode using XCTest
Summary

Write tests in Xcode using XCTest

Run tests under different configurations with Test Plans
Summary

Write tests in Xcode using XCTest

Run tests under different configurations with Test Plans

Use xcodebuild, xcrresulttool, and xccov in CI
<table>
<thead>
<tr>
<th>Topic</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improving Battery Life and Performance</td>
<td>Thursday, 4:00</td>
</tr>
<tr>
<td>What’s New in Testing</td>
<td>WWDC 2018</td>
</tr>
</tbody>
</table>

More Information

developer.apple.com/wwdc19/413