What’s New in Testing
Session 403

Honza Dvorsky, Xcode Engineer
Ethan Vaughan, Xcode Engineer
Code coverage
Test selection and ordering
Parallel testing
Code coverage
Test selection and ordering
Parallel testing
Code Coverage

9.3
Code Coverage

Xcode 9.3
Code Coverage

Xcode 9.3

Performance and accuracy
Code Coverage

Xcode 9.3

Performance and accuracy

Target selection
Code Coverage

Xcode 9.3

Performance and accuracy

Target selection

xccov
Code Coverage

Xcode 9.3

Performance and accuracy

Target selection

xccov

Source editor
Performance and Accuracy
Time to Load
In seconds
Time to Load
In seconds

Xcode 9.3

Xcode 9
Time to Load
In seconds

- Xcode 9.3
- Xcode 9
Time to Load
In seconds

- Xcode 9: 6.5 seconds
Time to Load
In seconds

Xcode 9.3: 0.3 seconds
Xcode 9: 6.5 seconds
```cpp
class Store {
private:
    int count;

public:
    void increment() {
        count += 1;
    }

    int currentCount() {
        return count;
    }
};
```
```cpp
class Store {
private:
    int count;

public:
    void increment() {
        count += 1;
    }

    int currentCount() {
        return count;
    }
};
```
Target Selection
Target Selection

Code Coverage
• Enabled for all targets
• Enabled for selected targets
• Disabled
Does time in window need to match top bar?
XCCOV

Command line tool

Output formats
• Human-readable
• Machine-parseable (JSON)

View coverage data
Coverage Data
Coverage Data

- Coverage report
- Line coverage percentages

xccovreport
Coverage Data

- coverage report: xccovreport
- line coverage percentages
- coverage archive: xccovarchive
- raw execution counts
Coverage Data
Coverage Data

Derived data
Coverage Data

Derived data

Result bundle

• xcodebuild -resultBundlePath ...
xcrun xccov view HelloWorld.xccovreport

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld.app</td>
<td>57.89% (11/19)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.m</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>-[AppDelegate applicationWillTerminate:]</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>-[AppDelegate applicationDidFinishLaunching:]</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.m</td>
<td>50.00% (5/10)</td>
</tr>
<tr>
<td>-[ViewController setRepresentedObject:]</td>
<td>0.00% (0/5)</td>
</tr>
<tr>
<td>-[ViewController viewDidLoad]</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/main.m</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>main</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>HelloWorldTests.xctest</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorldTests/HelloWorldTests.m</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>-[HelloWorldTests testExample]</td>
<td>100.00% (4/4)</td>
</tr>
</tbody>
</table>
xcrun xccov view HelloWorld.xccovreport

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld.app</td>
<td>57.89% (11/19)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.m</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>-[AppDelegate applicationWillTerminate:]</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>-[AppDelegate applicationDidFinishLaunching:]</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.m</td>
<td>50.00% (5/10)</td>
</tr>
<tr>
<td>-[ViewController setRepresentedObject:]</td>
<td>0.00% (0/5)</td>
</tr>
<tr>
<td>-[ViewController viewDidLoad]</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/main.m</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>main</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>HelloWorldTests.xctest</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorldTests/HelloWorldTests.m</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>-[HelloWorldTests testExample]</td>
<td>100.00% (4/4)</td>
</tr>
<tr>
<td>Name</td>
<td>Coverage</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>HelloWorld.app</td>
<td>57.89% (11/19)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.m</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>-[AppDelegate applicationWillTerminate:]</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>-[AppDelegate applicationDidFinishLaunching:]</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.m</td>
<td>50.00% (5/10)</td>
</tr>
<tr>
<td>-[ViewController setRepresentedObject:]</td>
<td>0.00% (0/5)</td>
</tr>
<tr>
<td>-[ViewController viewDidLoad:]</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/main.m</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>main</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>HelloWorldTests.xctest</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorldTests/HelloWorldTests.m</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>-[HelloWorldTests testExample]</td>
<td>100.00% (4/4)</td>
</tr>
</tbody>
</table>
xcrun xccov view HelloWorld.xccovreport

<table>
<thead>
<tr>
<th>Name</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelloWorld.app</td>
<td>57.89% (11/19)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/AppDelegate.m</td>
<td>50.00% (3/6)</td>
</tr>
<tr>
<td>-[AppDelegate applicationWillTerminate:]</td>
<td>0.00% (0/3)</td>
</tr>
<tr>
<td>-[AppDelegate applicationDidFinishLaunching:]</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/ViewController.m</td>
<td>50.00% (5/10)</td>
</tr>
<tr>
<td>-[ViewController setRepresentedObject:]</td>
<td>0.00% (0/5)</td>
</tr>
<tr>
<td>-[ViewController viewDidLoad:]</td>
<td>100.00% (5/5)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorld/main.m</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>main</td>
<td>100.00% (3/3)</td>
</tr>
<tr>
<td>HelloWorldTests.xctest</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>/tmp/HelloWorld/HelloWorldTests/HelloWorldTests.m</td>
<td>100.00% (21/21)</td>
</tr>
<tr>
<td>-[HelloWorldTests testExample]</td>
<td>100.00% (4/4)</td>
</tr>
</tbody>
</table>
xcrun xccov view --json HelloWorld.xccovreport

{
  "coveredLines": 32,
  "lineCoverage": 0.8,
  "targets": [
    {
      "coveredLines": 11,
      "lineCoverage": 0.57894736842105265,
      "files": [
        {
          "coveredLines": 3,
          "lineCoverage": 0.5,
          "path": "tmpHelloWorldHelloWorldAppDelegate.m",
          "functions": [
            {
              "coveredLines": 0,
            }
          ]
        }
      ]
    }
  ]
}
xcrun xccov view --json HelloWorld.xccovreport

{
  "coveredLines": 32,
  "lineCoverage": 0.8,
  "targets": [
    {
      "coveredLines": 11,
      "lineCoverage": 0.57894736842105265,
      "files": [
        {
          "coveredLines": 3,
          "lineCoverage": 0.5,
          "path": "tmpHelloWorldHelloWorldAppDelegate.m",
          "functions": [
            {
              "coveredLines": 0,
            }
          ]
        }
      ]
    }
  ]
}
xcrun xccov view --json HelloWorld.xccovreport

{
  "coveredLines": 32,
  "lineCoverage": 0.8,
  "targets": [
    {
      "coveredLines": 11,
      "lineCoverage": 0.57894736842105265,
      "files": [
        {
          "coveredLines": 3,
          "lineCoverage": 0.5,
          "path": "tmpHelloWorldHelloWorldAppDelegate.m",
          "functions": [
            {
              "coveredLines": 0,
            }]
        }
      ]
    }
  ]
}
NAME

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

SYNOPSIS

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

xccov view [--file-list | --file path] archive.xccovarchive

DESCRIPTION

When tests are run with code coverage enabled, Xcode generates two files that contain its representation of coverage data. These are the coverage report, with extension xccovreport, and the coverage archive, with extension xccovarchive. The coverage report contains line coverage percentages for each target, source file, and function/method that has coverage information. The coverage archive contains the raw
Source Editor
protocol Server {
    func sendMessage(_ message: Message) throws -> MessageReceipt
    func getMessages() throws -> [Message]
}

extension Server {
    func getRecentMessages() throws -> [Message] {
        let messages = try getMessages()
        return Array(messages.dropFirst(max(messages.count / 2, 0)))
    }
}

class LocalServer: Server {

    private var storage: [Message] = []

    init() {
    }

    func sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }

    func getMessages() throws -> [Message] {
        return storage
    }
protocol Server {
    func sendMessage(_ message: Message) throws -> Message
    func getMessages() throws -> [Message]
}

extension Server {
    func getRecentMessages() throws -> [Message] {
        let messages = try getMessages()
        return Array(messages.dropFirst(max(messages.count, 1)))
    }
}
import Foundation

protocol Server {
    func sendMessage(_ message: Message) throws -> MessageReceipt
    func getMessages() throws -> [Message]
}

extension Server {
    func getRecentMessages() throws -> [Message] {
        let messages = try getMessages()
        return Array(messages.dropFirst(max(messages.count / 2, 0)))
    }
}

class LocalServer: Server {

    private var storage: [Message] = []

    init() {}{

    func sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }

    func getMessages() throws -> [Message] {
        return storage
    }
Demo
Code coverage

Test selection and ordering

Parallel testing
Code coverage
Test selection and ordering
Parallel testing
Test Selection

Not all tests are equal

• 1,000 unit tests
• 10 UI tests

Different tests in different situations

• Pre-commit
• Nightly
Test Selection with Schemes
Test Selection with Schemes

“Tests to skip”

• Automatically includes new tests
Test Selection with Schemes

“Tests to skip”
• Automatically includes new tests

“Tests to run”
• Explicit opt-in for tests
class LocalServer: Server {

    private var storage: [Message] = []

    init() {
    }

    func sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }

    func getMessages() throws -> [Message] {
        return storage
    }
}
localServer: Server {

    private var storage: [Message] = []

    init() {
    }

c    sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }
```swift
localServer: Server {
  
  var storage: [Message] = []

  init() {
  }

  func sendMessage(_: Message) throws -> MessageReceipt {
    storage.append(message)
    return MessageReceipt(delivered: true, deliveryDate: Date())
  }
}
```
Test Ordering
Test Ordering

Alphabetical
• Default
• Deterministic
• Can hide implicit dependencies
Implicit Dependencies Between Tests
Implicit Dependencies Between Tests

test A

test B

test C
Implicit Dependencies Between Tests

test A

test B

test C
Implicit Dependencies Between Tests

test A

test B

test C
Implicit Dependencies Between Tests

test A

- test B

- test C
Implicit Dependencies Between Tests

test A

test B

test C
Implicit Dependencies Between Tests

test B

test A

test C
Implicit Dependencies Between Tests

- test B
- test A
- test C
Test Ordering

Alphabetical
- Default
- Deterministic
- Can hide implicit dependencies
Test Ordering

Alphabetical
• Default
• Deterministic
• Can hide implicit dependencies

Randomized
• Nondeterministic
• Can uncover implicit dependencies
```swift
import Foundation

class LocalServer: Server {

    private var storage: [Message] = []

    init() {
    }

    func sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }

    func getMessages() throws -> [Message] {
        return storage
    }
}
```
class LocalServer: Server {

    private var storage: [Message] = []

    init() {
    }

    func sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }

    func getMessages() throws -> [Message] {
    }
class LocalServer: Server {

    private var storage: [Message] = []

    init() {}  

    func sendMessage(_ message: Message) throws -> MessageReceipt {
        storage.append(message)
        return MessageReceipt(delivered: true, deliveryDate: Date())
    }

    func getMessages() throws -> [Message] {
    }
Code coverage
Test selection and ordering
Parallel testing
Code coverage
Test selection and ordering
Parallel testing
Development Cycle

Write → Debug → Test → Push
Waiting for Tests to Finish
Waiting for Tests to Finish
Parallel Destination Testing

Ability to run all of your tests on multiple destinations simultaneously

```
xcodebuild test
  -destination 'platform=iOS,name=iPhone X'
  -destination 'platform=iOS,name=iPad'
```
Parallel Destination Testing

iPhone X → iPad

Time → 8.0
Parallel Destination Testing

- iPhone X
- iPad

Time

9.0
Parallel Destination Testing

Limitations
Parallel Destination Testing

Limitations

Only beneficial if testing on multiple destinations
Parallel Destination Testing

Limitations

Only beneficial if testing on multiple destinations

Only available from xcodebuild
Parallel Distributed Testing
Parallel Distributed Testing

Execute tests in parallel on a single destination

Time
Parallel Distributed Testing

Execute tests in parallel on a single destination

Time
Testing Architecture
Testing Architecture

Unit tests

```go
func testFoo() { ... }
```
Testing Architecture

Unit tests

```go
func testFoo() { ... }
```
Testing Architecture

Unit tests

func testFoo() {
... 
}
Testing Architecture

Unit tests

```go
func testFoo() { ... }
```
Testing Architecture

Unit tests

Your App

Test Bundle
Testing Architecture

Unit tests
Testing Architecture

UI tests

UI Test Runner
Testing Architecture

UI tests

UI Test Runner
Testing Architecture

UI tests

UI Test Runner

Your App
Testing Architecture

UI tests

UI Test Runner

Your App
Testing Architecture

UI tests

UI Test Runner ➔ Click ➔ Your App
Testing Architecture

UI tests

Advanced Testing and Continuous Integration

WWDC 2016
Overview
Parallel Distributed Testing

Xcode
Overview
Parallel Distributed Testing
Xcode
Overview
Parallel Distributed Testing
Overview
Parallel Distributed Testing
Overview
Parallel Distributed Testing

Xcode
Overview
Parallel Distributed Testing
Overview
Parallel Distributed Testing
Classes Execute in Parallel
Classes Execute in Parallel

Time

R1

R2

R3
Classes Execute in Parallel

Motivation
Classes Execute in Parallel

Motivation

Hidden dependencies between tests in a class
Classes Execute in Parallel

Motivation

Hidden dependencies between tests in a class

Avoid unnecessary \texttt{setUp} and \texttt{tearDown} computation
Parallel Testing on Simulator
Parallel Testing on Simulator

Original Simulator
Parallel Testing on Simulator

Original Simulator

Clone 1

Clone 2

Clone 3
Parallel Testing on Simulator

Original Simulator  →  Clone 1  →  Clone 2  →  Clone 3
Parallel Testing on Simulator
Parallel Testing on Simulator

Implications
Parallel Testing on Simulator

Implications

Original simulator is not used during testing
Parallel Testing on Simulator

Implications

Original simulator is not used during testing

Separate data containers
<table>
<thead>
<tr>
<th></th>
<th>Unit Tests</th>
<th>UI Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Simulator (iOS and tvOS)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Demo Recap
Demo Recap

Enabling parallelization
Demo Recap

Enabling parallelization

Viewing results in the test log and test report
Demo Recap

Enabling parallelization

Viewing results in the test log and test report

Multiple instances of a Mac app running unit tests
Demo Recap

Enabling parallelization
Viewing results in the test log and test report
Multiple instances of a Mac app running unit tests
Multiple simulator clones running UI tests
xcodebuild
Override the number of workers

-xcodebuild

-parallel-testing-worker-count n
xcodebuild

Override the number of workers

-parallel-testing-worker-count n

Force parallel testing on or off

-parallel-testing-enabled YES | NO
Tips and Tricks
Tips and Tricks

Consider splitting a long running class into two classes.
Classes Execute in Parallel

- $R_1$
- $R_2$
- $R_3$

Time

- C
- C
- C
- C
- C
Classes Execute in Parallel

R1

R2

R3

Time
Tips and Tricks

Consider splitting a long running class into two classes
Tips and Tricks

Consider splitting a long running class into two classes

Put performance tests into their own bundle, with parallelization disabled
Tips and Tricks

Consider splitting a long running class into two classes
Put performance tests into their own bundle, with parallelization disabled
Understand which tests are not safe for parallelization
Tips and Tricks

Consider splitting a long running class into two classes

Put performance tests into their own bundle, with parallelization disabled

Understand which tests are not safe for parallelization
Summary
Summary

Code coverage
Summary

Code coverage

Test selection and ordering
Summary

Code coverage
Test selection and ordering
Parallel testing
More Information

https://developer.apple.com/wwdc18/403

<table>
<thead>
<tr>
<th>Event</th>
<th>Lab</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator, Testing, and Continuous Integration Lab</td>
<td>Technology Lab 8</td>
<td>Wednesday 12:00PM</td>
</tr>
<tr>
<td>Simulator, Testing, and Continuous Integration Lab</td>
<td>Technology Lab 8</td>
<td>Friday 9:00AM</td>
</tr>
</tbody>
</table>