ASYNCHRONOUS PROGRAMMING WITH OPERATIONS IN SWIFT

Antoine van der Lee / @twannl

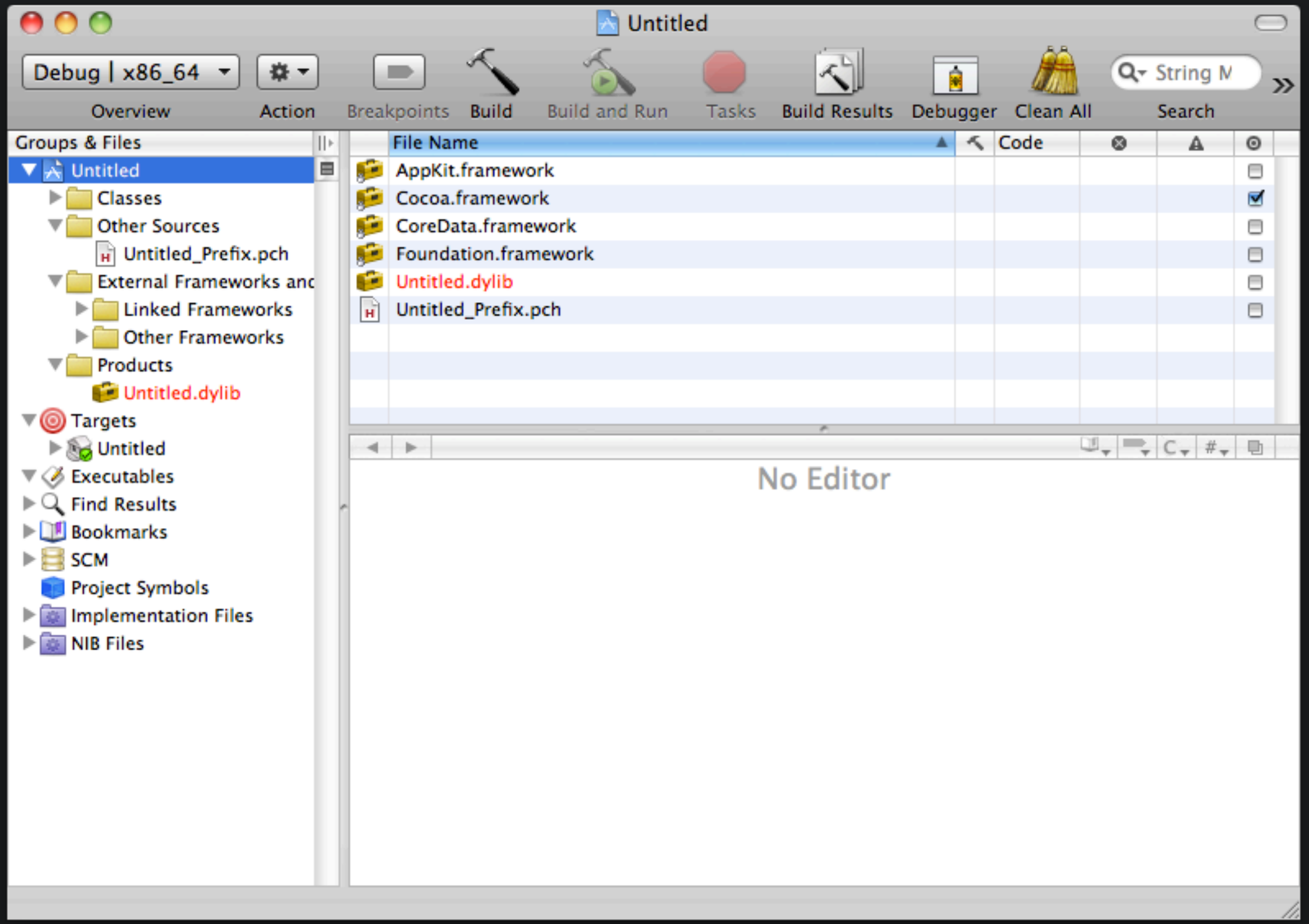
CocoaHeadsNL, March 18th 2020, Spanbroek, Netherlands

iPhone 3G


iPhone


AT&T 3G
9:42 AM
SMS
Text
Tuesday
9
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Clock
Calculator
Notes
Settings
iTunes
App Store
Phone
Mail
Safari
iPod


iPhone


Untitled
Debug | x86_64
Overview
Action
Breakpoints
Build
Build and Run
Tasks
Build Results
Debugger
Clean All
String M
Search
Groups & Files
Untitled
Classes
Other Sources
Untitled_Prefix.pch
External Frameworks and
Linked Frameworks
Other Frameworks
Products
Untitled.dylib
Targets
Untitled
Executables
Find Results
Bookmarks
SCM
Project Symbols
Implementation Files
NIB Files
File Name
Code
AppKit.framework
Cocoa.framework
CoreData.framework
Foundation.framework
Untitled.dylib
Untitled_Prefix.pch
No Editor

Class

Operation

An abstract class that represents the code and data associated with a single task.

Declaration

```
class Operation : NSObject
```

SDKs

iOS 2.0+

macOS 10.5+

Mac Catalyst 13.0+

tvOS 9.0+

watchOS 2.0+

Class

Operation

An abstract class that represents the code and data associated with a single task.

SDKs

iOS 2.0+

macOS 10.5+

Mac Catalyst 13.0+

tvOS 9.0+

watchOS 2.0+

Declaration

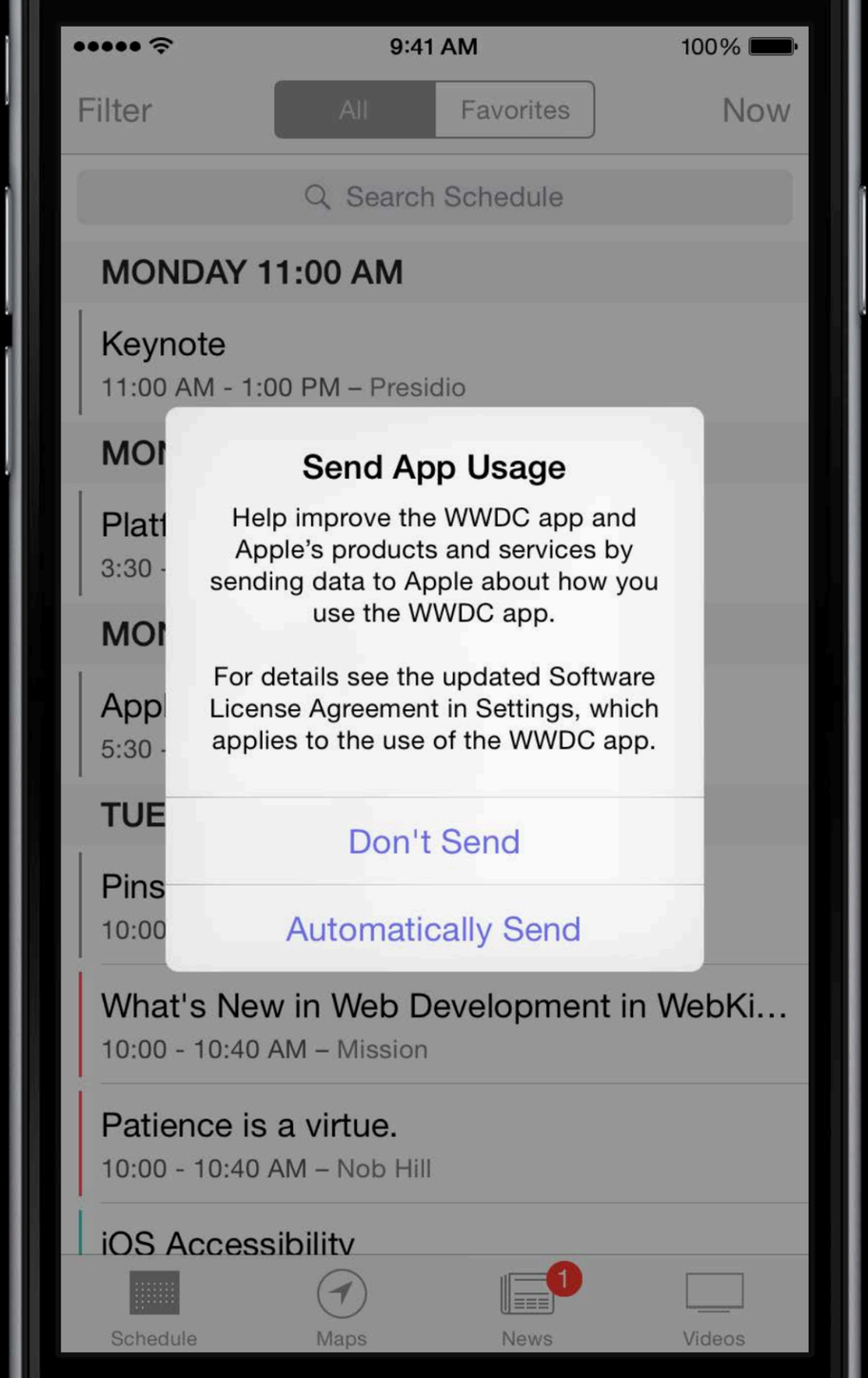
```
class Operation : NSObject
```

Advanced NSOperations

Exploring the WWDC app

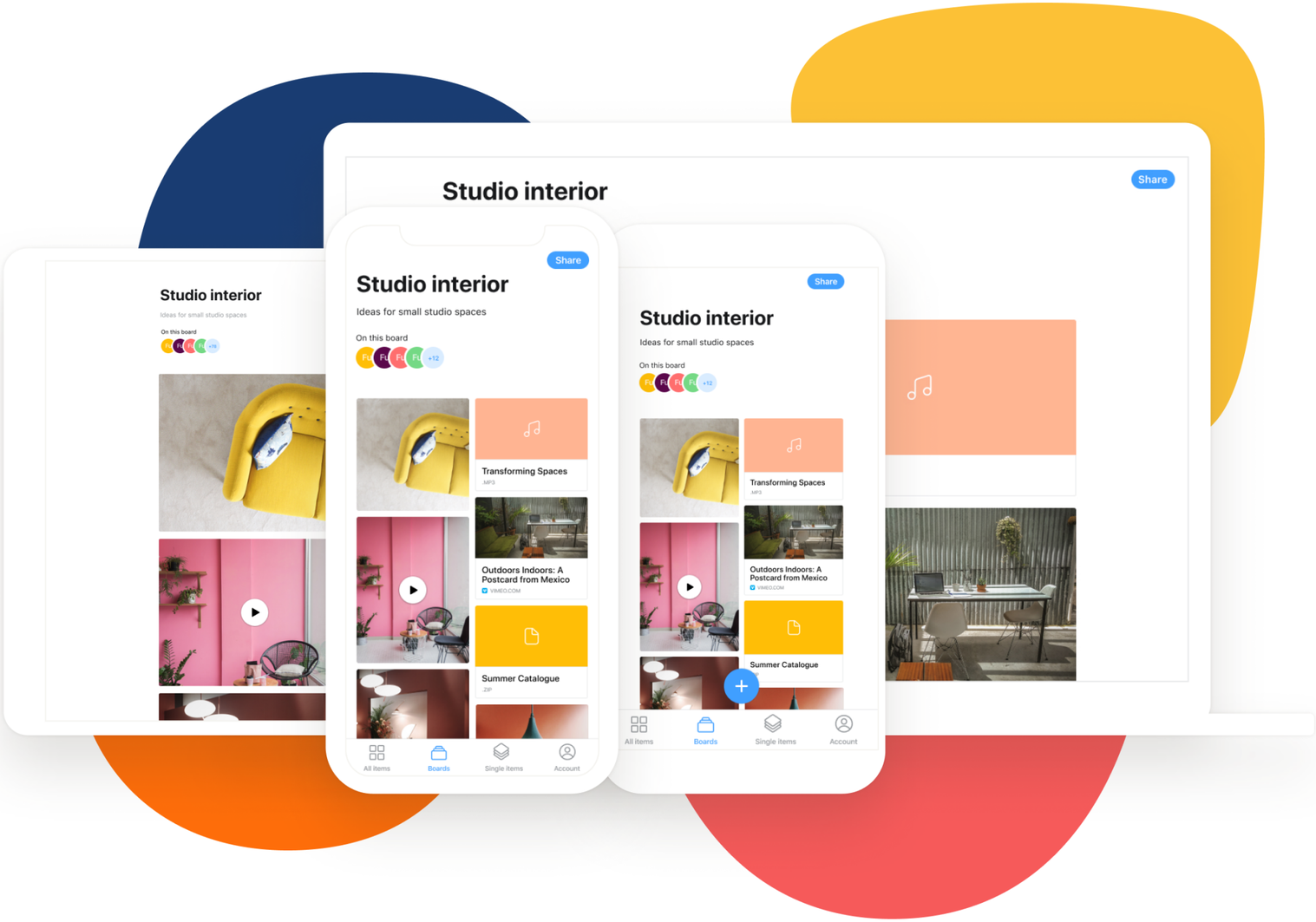
Session 226

Philippe Hausler Foundation Engineer
Dave DeLong Frameworks Evangelist


9:41 AM
100%
Filter
All
Favorites
Now
Search Schedule
MONDAY 11:00 AM
Keynote
11:00 AM - 1:00 PM – Presidio
Send App Usage
Help improve the WWDC app and Apple's products and services by sending data to Apple about how you use the WWDC app.
For details see the updated Software License Agreement in Settings, which applies to the use of the WWDC app.
Don't Send
Automatically Send
What's New in Web Development in WebKi…
10:00 - 10:40 AM – Mission
Patience is a virtue.
10:00 - 10:40 AM – Nob Hill
Schedule
Maps
News
1
Videos

Save everything that inspires your ideas

Collect by WeTransfer



COLLECT BY WETRANSFER

48 OPERATIONS

An operation is a solution for writing asynchronous code that is testable and responsible for a single part of functionality.

CONVINCE YOU

THE NEXT COOL THING

Why operations?

SINGLE RESPONSIBILITY

```
final class ContentImportOperation: Operation {
    override func main() {
        // .. import the content
    }
}

final class ContentUploadOperation: Operation {
    override func main() {
        // .. upload the content
    }
}
```

EASY TO TEST

```
final class ContentImportOperationTests: Operation {
    override func main() {
        // .. import the content
    }
}

final class ContentUploadOperationTests: Operation {
    override func main() {
        // .. upload the content
    }
}
```

CHAINABLE

```
let importOperation = ContentImportOperation()
let uploadOperation = UploadContentOperation()
uploadOperation.addDependency(importOperation)
```

CHAINABLE & REUSABLE

```
let importOperation = ContentImportOperation()
let uploadOperation = UploadContentOperation()
uploadOperation.addDependency(importOperation)

let copyOperation = ContentCopyOperation()
let uploadOperation = UploadContentOperation()
uploadOperation.addDependency(copyOperation)
```

PART OF FOUNDATION

```
import Foundation
```

BUY WHY NOT..

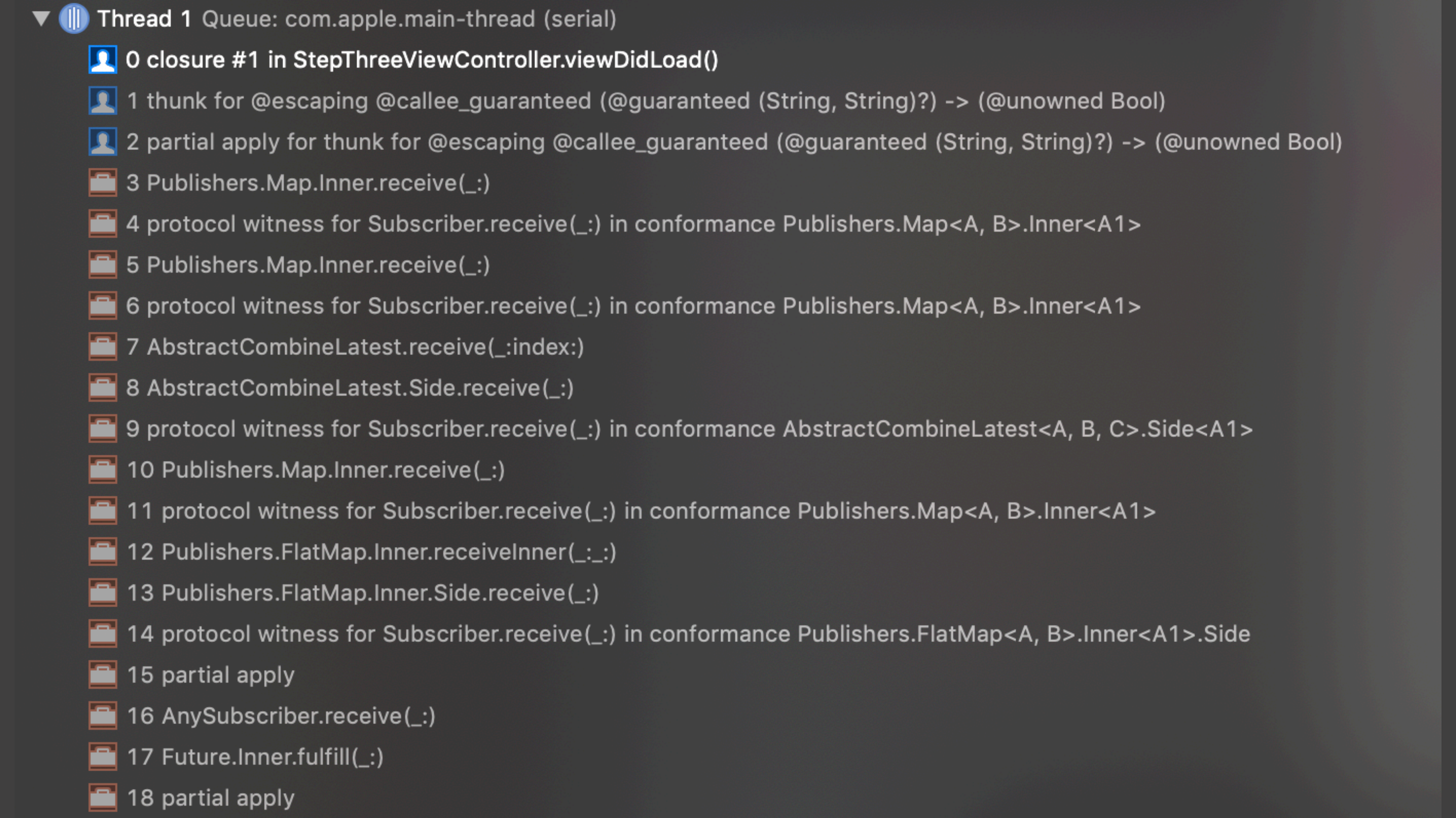
~~import Foundation~~

```
import RxSwift
```

BUY WHY NOT..

~~import Foundation~~

```
import Combine
```


Thread 1 Queue: com.apple.main-thread (serial)
0 closure #1 in StepThreeViewController.viewDidLoad()
1 thunk for @escaping @callee_guaranteed (@guaranteed (String, String)?) -> (@unowned Bool)
2 partial apply for thunk for @escaping @callee_guaranteed (@guaranteed (String, String)?) -> (@unowned Bool)
3 Publishers.Map.Inner.receive(_:)
4 protocol witness for Subscriber.receive(_:) in conformance Publishers.Map<A, B>.Inner<A1>
5 Publishers.Map.Inner.receive(_:)
6 protocol witness for Subscriber.receive(_:) in conformance Publishers.Map<A, B>.Inner<A1>
7 AbstractCombineLatest.receive(_:index:)
8 AbstractCombineLatest.Side.receive(_:)
9 protocol witness for Subscriber.receive(_:) in conformance AbstractCombineLatest<A, B, C>.Side<A1>
10 Publishers.Map.Inner.receive(_:)
11 protocol witness for Subscriber.receive(_:) in conformance Publishers.Map<A, B>.Inner<A1>
12 Publishers.FlatMap.Inner.receiveInner(_:_:)
13 Publishers.FlatMap.Inner.Side.receive(_:)
14 protocol witness for Subscriber.receive(_:) in conformance Publishers.FlatMap<A, B>.Inner<A1>.Side
15 partial apply
16 AnySubscriber.receive(_:)
17 Future.Inner.fulfill(_:)
18 partial apply

```
pod 'RxSwift'
```

```
IPHONEOS_DEPLOYMENT_TARGET = 13.0
```

```swift
import Combine

@available(iOS 13.0, *)
class ContentUploadPublisher<S: Subscriber>: Subscription where S.Input == URL {
    // ..
}
```

OPERATIONS

Playground time!

Tie back

- Single Responsibility
- Easy to test
- Part of Foundation

Swift for Good

www.swiftforgood.com

Chapter 13

Operations

By Antoine van der Lee

www.swiftforgood.com

SwiftLee

avanderlee.com

THANKS

@TWANNL

AVANDERLEE.COM