SWIFTCRAFT
21-24 May 2024

Swift Platform State of the Union for Software Crafters

Jon Reid

Software Crafting

Continuous improvement

Language/platform agnostic

industrial logic

Extreme Programming

Anything worth doing
is worth doing continuously

Fast feedback

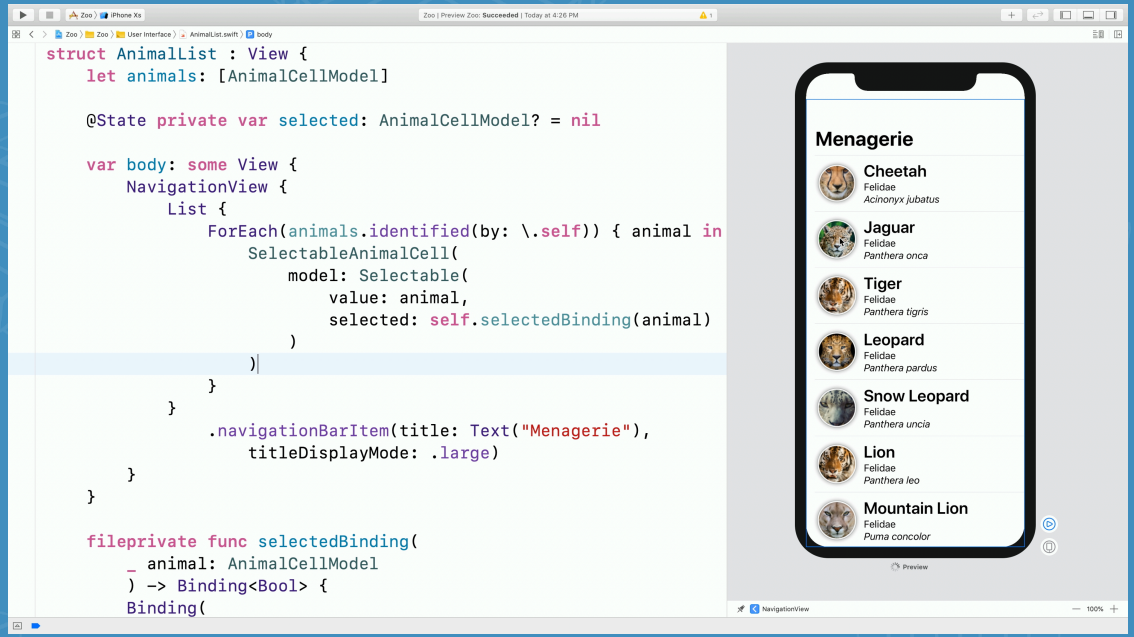
Work together

industrial logic

Principles ⇒ Practices ⇒ Tools

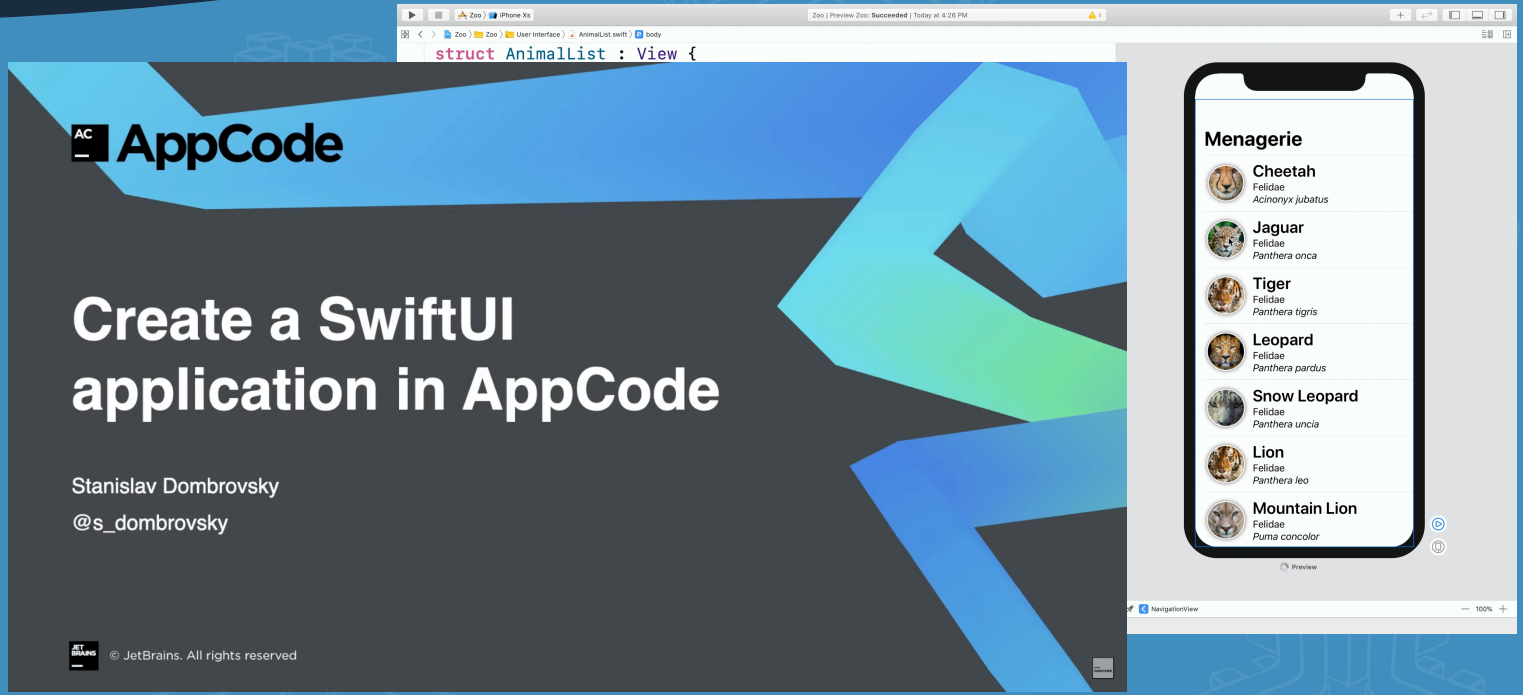
industrial logic

Does Apple Own the Entire Road?



industrial logic

Does Apple Own the Entire Road?

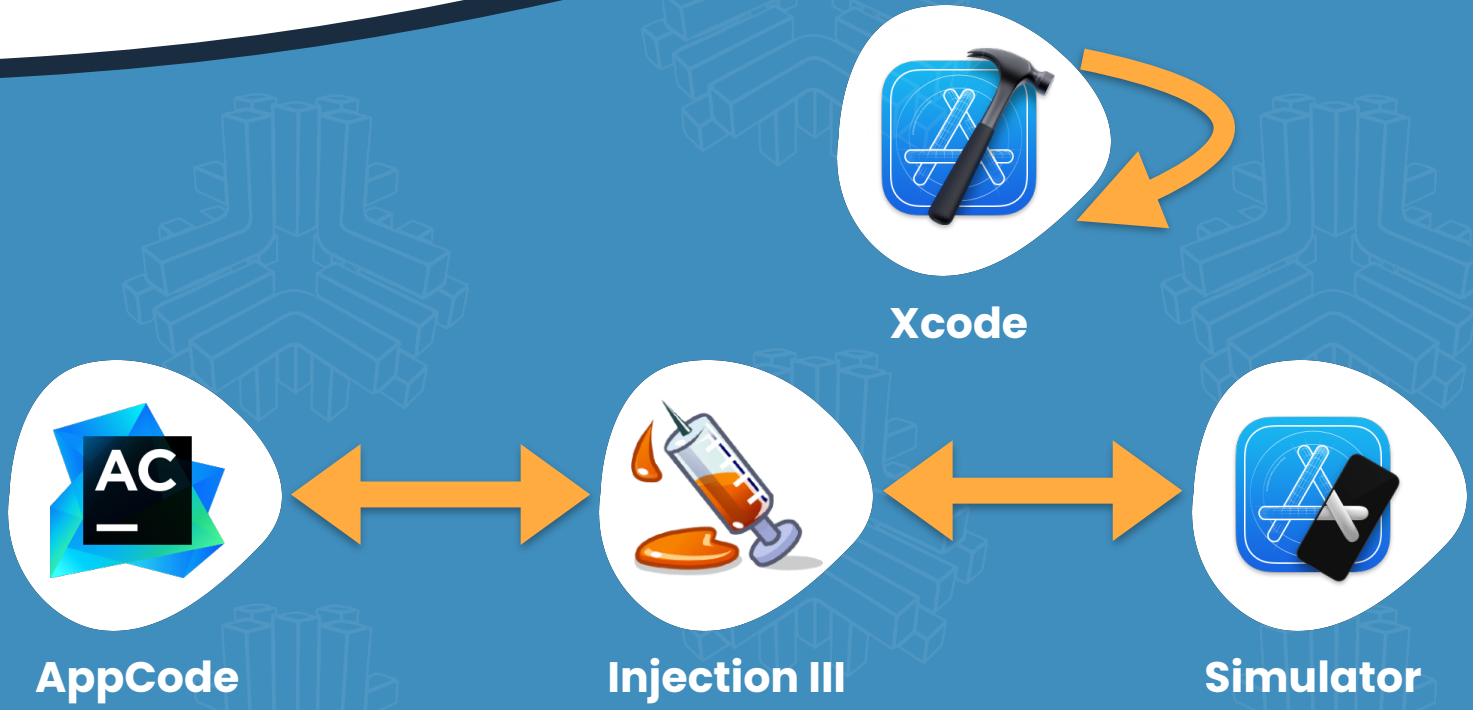


industrial logic

Does Apple Own the Entire Road?

Xcode

Does Apple Own the Entire Road?



industrial logic

It was the best of times...

A Tale of 3 IDEs

It's good to stretch beyond familiar tools.

Xcode

industrial logic

It was the best of times...

A Tale of 3 IDEs

It's good to stretch beyond familiar tools.

Xcode

AppCode

industrial logic

It was the best of times...

A Tale of 3 IDEs

It's good to stretch beyond familiar tools.

Xcode

AppCode

IntelliJ

industrial logic

Crafter Tools for Swift

1. **Dependency Analysis**
   Tools to visualize dependencies: what depends on what?
2. **Unit Testing**
   Meanwhile, elsewhere in the xUnit world: what do they have?
3. **Continuous Integration**
   Do our build systems encourage small, frequent commits?
4. **Test-Driven Development**
   What tooling supports TDD patterns?
5. **Refactoring**
   Are we out of options? I don't think so.

industrial logic

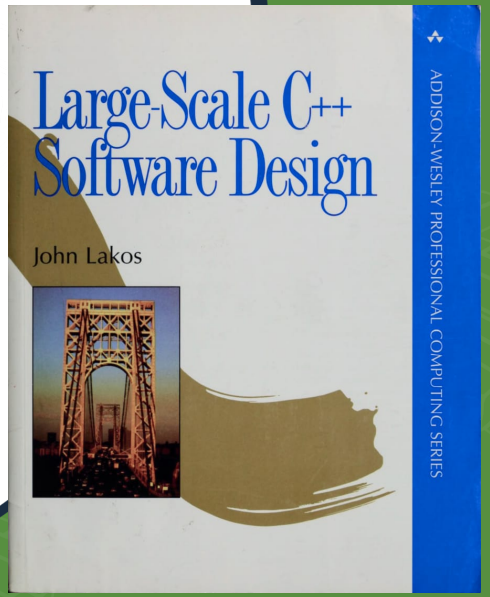
1: DEPENDENCY ANALYSIS

Tools to visualize dependencies: What depends on what?

Dependency Analysis

Physical Design

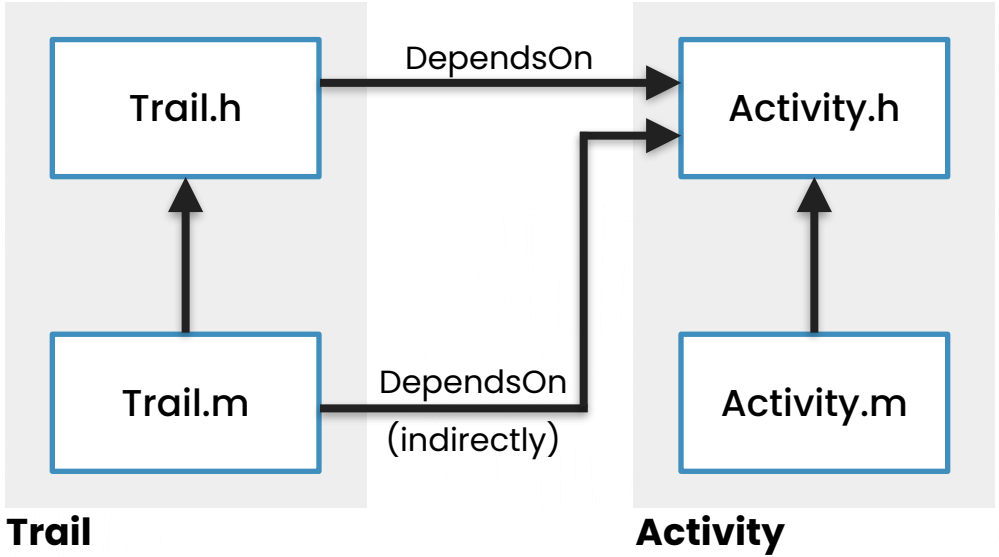
Logical design addresses only ***architectural*** issues.
Physical design addresses ***organizational*** issues.



industrial logic

Compile-Time Dependency

```
#import
  "Activity.h"
```

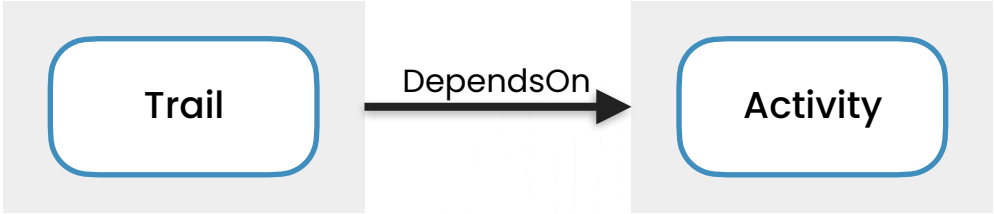



```
#import
  "Trail.h"
```

```
#import
  "Activity.h"
```

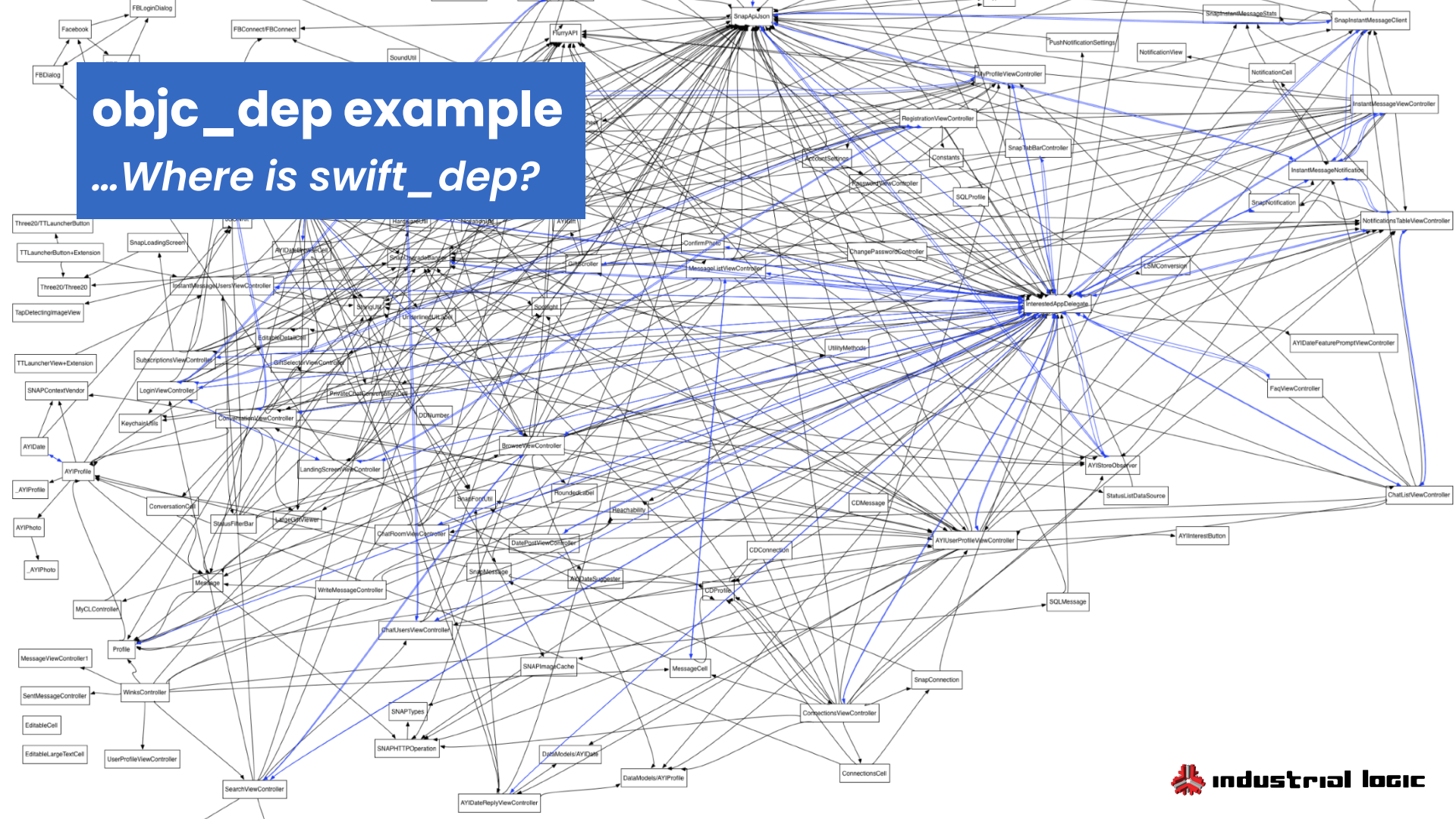

industrial logic

Behavioral Dependency



objc_dep

industrial logic


objc_dep example
...Where is swift_dep?
FBLoginDialog
Facebook
FBConnect/FBConnect
FBDialog
SoundUtil
FlurryAPI
SnapApiJson
SnapInstantMessageStats
SnapInstantMessageClient
PushNotificationSettings
NotificationView
NotificationCell
MyProfileViewController
InstantMessageViewController
RegistrationViewController
SnapTabBarController
ActionSheetSettings
Constants
PasswordViewController
SQLProfile
InstantMessageNotification
SnapNotification
NotificationsTableViewController
Three20/TTLauncherButton
TTLauncherButton+Extension
Three20/Three20
SnapLoadingScreen
TapDetectingImageView
InstantMessageUsersViewController
ConfirmPhoto
ChangePasswordController
MessageListViewController
GalleryController
CSMConversion
InterestedAppDelegate
SoundUtil
UnderlinedUILabel
Spotlight
EditableDetailCell
SubscriptionsViewController
GiftSelectionViewController
UtilityMethods
AYIDateFeaturePromptViewController
TTLauncherView+Extension
SNAPContextVendor
LoginViewController
PrivateChatConversationCell
FaqViewController
KeychainUtils
ContactPhotoViewController
CDPhoneNumber
AYIDate
AYIProfile
BrowserViewController
AYIStoreObserver
_AYIProfile
LandingScreenViewController
StatusListDataSource
ChatListViewController
ConversationCell
SnapPhotoUtil
RoundedLabel
CDMessage
AYIPhoto
StatusFilterBar
LargeGifViewer
Reachability
ChatRoomViewController
AYIUserProfileViewController
AYIInterestButton
DatePostViewController
CDConnection
_AYIPhoto
Message
SnapMessage
AYIDateSuggester
WriteMessageController
CDProfile
SQLMessage
MyCLController
ChatUsersViewController
Profile
MessageViewController1
SNAPImageCache
MessageCell
SnapConnection
SentMessageController
WinksController
SNAPTypes
ConnectionsViewController
EditableCell
EditableLargeTextCell
SNAPHTTPOperation
DataModels/AYIData
UserProfileViewController
DataModels/AYIProfile
ConnectionsCell
SearchViewController
AYIDateReplyViewController
industrial logic

2: UNIT TESTING

Meanwhile, elsewhere in the xUnit world: What do they have?

Unit Testing

Basic Parts of a Testing Framework

Test Description

Test Runner

industrial logic

Test Description: XCTest

***iOS Unit Testing by Example* page 6:**

- It lives within a subclass of `XCTestCase`.
- It isn't declared `private`.
- Its name starts with `test`.
- It takes no parameters.
- It has no return value.

```
class AssertYourSelfTests: XCTestCase {

    func testSomethingUsuallyALongName() {
        XCTAssertEqual(a, b)
    }

}
```

Test Description: JUnit

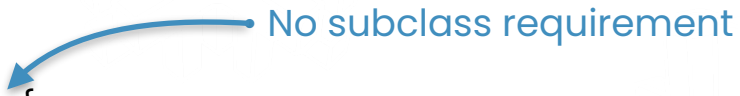
```
class AuthorNameNormalizerTests {

    @Test
    fun `initializes middle name`() {
        val normalizer = AuthorNameNormalizer()

        assertThat(
            normalizer.normalize("Henry David Thoreau"),
            equalTo("Thoreau, Henry D.")
        )
    }

}
```

Test Description: JUnit




```
class AuthorNameNormalizerTests {

    @Test
    fun `initializes middle name`() {
        val normalizer = AuthorNameNormalizer()

        assertThat(
            normalizer.normalize("Henry David Thoreau"),
            equalTo("Thoreau, Henry D.")
        )
    }

}
```

Test Description: JUnit

- No subclass requirement
- Test annotation (there are others)

```
class AuthorNameNormalizerTests {

    @Test
    fun `initializes middle name`() {
        val normalizer = AuthorNameNormalizer()

        assertThat(
            normalizer.normalize("Henry David Thoreau"),
            equalTo("Thoreau, Henry D.")
        )
    }

}
```


industrial logic

Test Description: JUnit

- No subclass requirement
- Test annotation (there are others)
- Natural language description

```
class AuthorNameNormalizerTests {

    @Test
    fun `initializes middle name`() {
        val normalizer = AuthorNameNormalizer()

        assertThat(
            normalizer.normalize("Henry David Thoreau"),
            equalTo("Thoreau, Henry D.")
        )
    }

}
```

Test Description: JUnit

```
class AuthorNameNormalizerTests {

    @Test
    fun `initializes middle name`() {
        val normalizer = AuthorNameNormalizer()

        assertThat(
            normalizer.normalize("Henry David Thoreau"),
            equalTo("Thoreau, Henry D.")
        )
    }

}
```

- No subclass requirement
- Test annotation (there are others)
- Natural language description
- Assertion without baggage

industrial logic

Test Description: swift-testing

- No subclass requirement
- Test annotation (with traits!)
- Natural language description
- Assertion without baggage

```
@Suite("AuthorNameNormalizer Tests")
struct AuthorNameNormalizerTests {

    @Test("Initializes middle name")
    func initializesMiddleName() {
        let normalizer = AuthorNameNormalizer()
        let actual = normalizer.normalize("Henry David Thoreau")
        #expect(actual == "Thoreau, Henry D.")
    }

}
```


1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
Fizz Buzz
16

Test Description

Parameterized Tests **NUnit**

```
[TestCase(3)]
[TestCase(6)]
[TestCase(9)]
public void FizzWhenDivisibleBy3(int input)
{
    var fizzBuzz = new FizzBuzz()
    var actual = fizzBuzz.evaluate(input)
    Assert.AreEqual("Fizz", actual)
}
```

Since May 2009

Test Description

Parameterized Tests **NUnit**

```
[TestCase(1, "1")]
[TestCase(2, "2")]
[TestCase(3, "Fizz")]
[TestCase(6, "Fizz")]
[TestCase(5, "Buzz")]
[TestCase(10, "Buzz")]
[TestCase(15, "FizzBuzz")]
[TestCase(30, "FizzBuzz")]
public void TestFizzBuzz(int input, string expected)
{
    var fizzBuzz = new FizzBuzz()
    var actual = fizzBuzz.evaluate(input)
    Assert.AreEqual(expected, actual)
}
```

Test Description

Parameterized Tests **EGTest**

```
func testFizzBuzz() {
    check([
        eg((1), expect: "1"),
        eg((2), expect: "2"),
        eg((3), expect: "Fizz"),
        eg((6), expect: "Fizz"),
        eg((5), expect: "Buzz"),
        eg((10), expect: "Buzz"),
        eg((15), expect: "FizzBuzz"),
        eg((30), expect: "FizzBuzz"),
    ]) { example in
        let fizzBuzz = FizzBuzz()
        let actual = fizzBuzz.evaluate(example.input.1)
        EGAssertEqual(actual, example)
   }
}
```

Test Description

Parameterized Tests XCTestParametrizedMacro

```
@Parametrize(input: [3, 6, 9])
func testFizz(input: Int) {
    let fizzBuzz = FizzBuzz()
    let actual = fizzBuzz.evaluate(input)
    XCTAssertEqual(actual, "Fizz")
}
```

Test Description

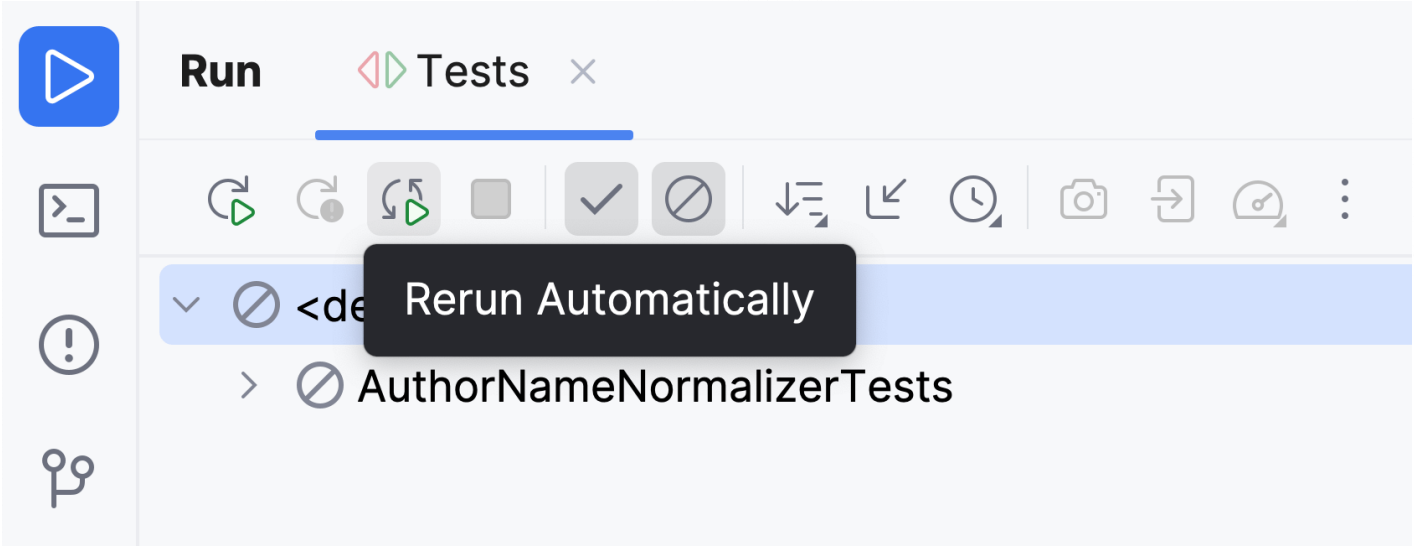
Parameterized Tests **swift-testing**

```
@Parametrize(input: [3, 6, 9])
func testFizz(input: Int) {
    let fizzBuzz = FizzBuzz()
    let actual = fizzBuzz.evaluate(input)
    XCTAssertEqual(actual, "Fizz")
}
```

```
@Test(arguments: [3, 6, 9])
func fizz(input: Int) {
    let fizzBuzz = FizzBuzz()
    let actual = fizzBuzz.evaluate(input)
    #expect(actual == "Fizz")
}
```

Test Runner

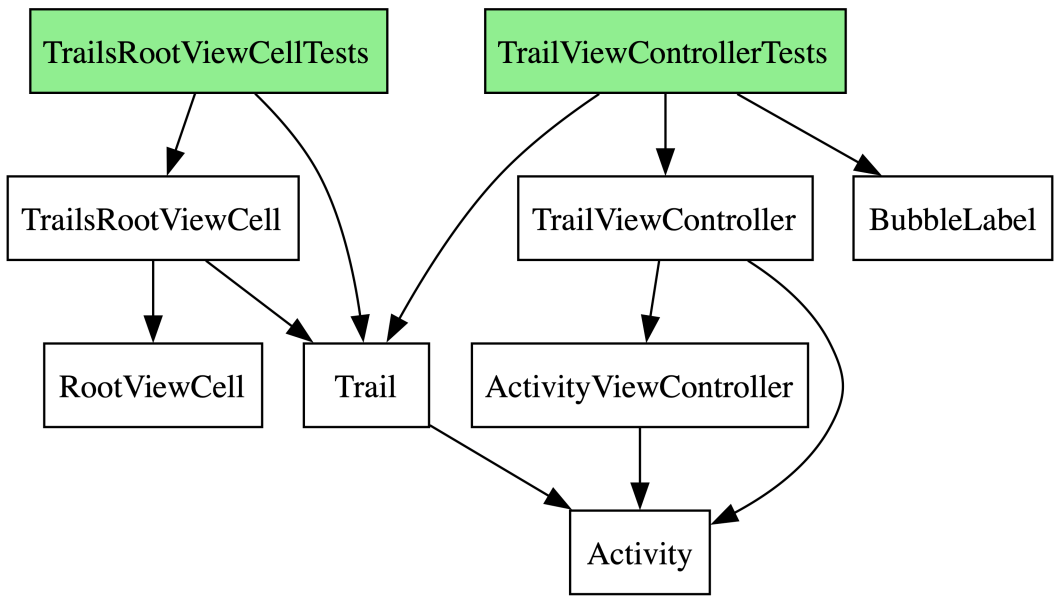
Continuous Testing **JetBrains IDEs**



industrial logic

Test Runner

If we had dependency analysis...



industrial logic

Test Runner

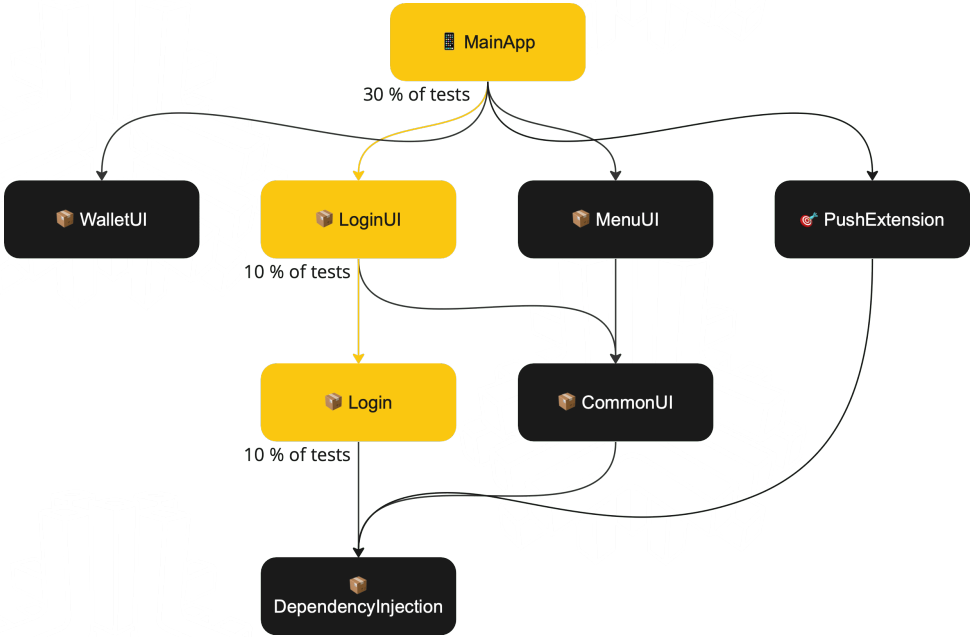
Continuous Minimal Testing **Infinitest**

Every time you change some code and IntelliJ recompiles it, infinitest computes *which tests need to be executed* and runs them.

industrial logic

Test Runner

Module Minimal Testing XcodeSelectiveTesting



Wishlist:

- File minimal
- Continuous

industrial logic

Test Runner

Faster Feedback **Jest**

Run any failing tests first!

industrial logic

Test Runner

Random Execution Order **XCTest**

Working in RSpec since Jan 2012

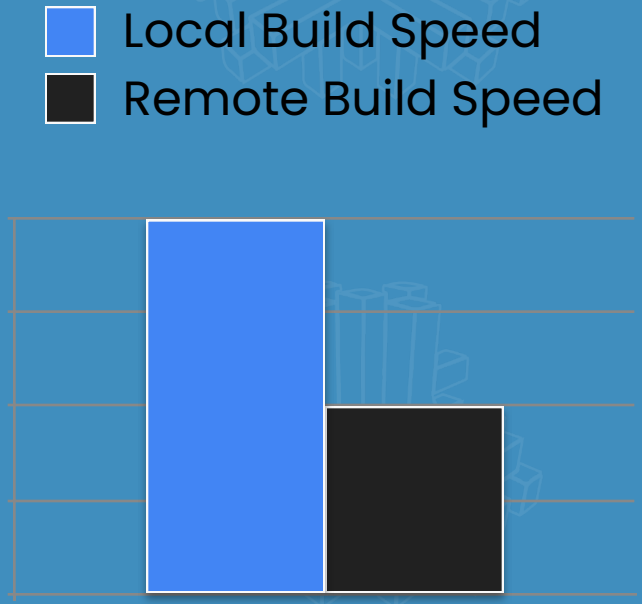
industrial logic

3: CONTINUOUS INTEGRATION

Do our build systems encourage small, frequent commits?


WAITING FOR
CI BUILD...

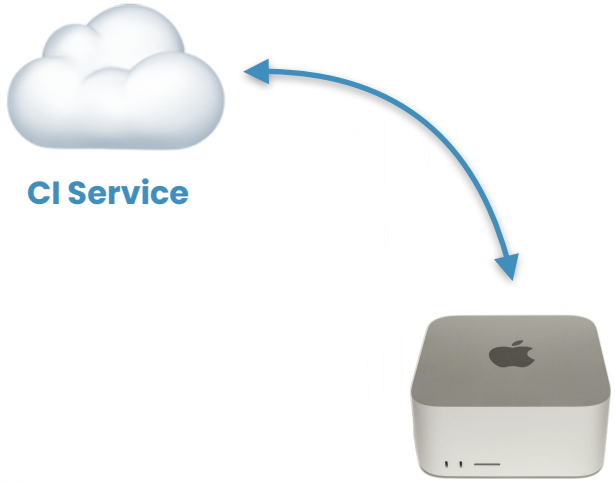
Build Speed Discrepancy



industrial logic

Continuous Integration

Use Your Own Build Hardware



industrial logic


STILL WAITING...

CI Is a Practice *Not a Tool*

Principles ⇒ Practices ⇒ Tools

industrial logic

Continuous Integration

Martin Fowler

Local environment

Central repo and environment

Teammates

industrial logic

Continuous Integration

Martin Fowler

We pull from the central store to ensure our local copy is up-to-date

Local environment

Central repo and environment

Teammates

industrial logic

Continuous Integration

Martin Fowler

We make our changes to the codebase, until we're ready and tests are green...

Local environment

Central repo and environment

Teammates

industrial logic

Continuous Integration

Martin Fowler

We make our changes to the codebase, until we're ready and tests are green...

Local environment

Central repo and environment

...meanwhile, colleagues update the central repo with their commits

Teammates

industrial logic

Continuous Integration

Martin Fowler

We pull collaborator's changes from the central store and combine them with our changes

Local environment

Central repo and environment

Teammates

industrial logic

Continuous Integration

Martin Fowler

We rebuild and check tests are green

Local environment

Central repo and environment

Teammates

industrial logic

Continuous Integration

Martin Fowler

We push our changes
to the central repo

Local environment

Central repo and
environment

Teammates

industrial logic

Continuous Integration

Martin Fowler

A service builds the product
on a central environment

Local environment

Central repo and
environment

Teammates

industrial logic

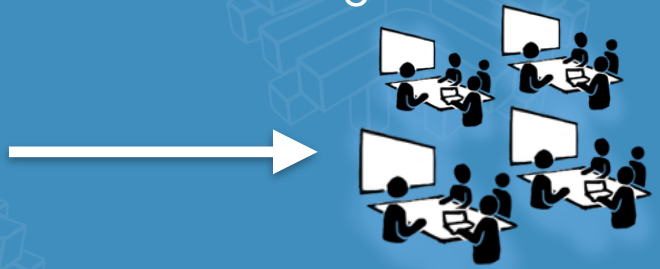
Continuous Integration

Martin Fowler

Our colleagues will combine their changes when they prepare to push their next changes

Local environment

Central repo and environment



Teammates

industrial logic

Continuous Integration

Martin Fowler

The central system notifies us that all was well with the central build

Our colleagues will combine their changes when they prepare to push their next changes

Local environment

Central repo and environment

Teammates

industrial logic

4: TEST-DRIVEN DEVELOPMENT

What tooling supports TDD patterns?

Frame First

by Jon Reid

Caller Creates

Click on undefined type to create it.

Click on undefined method to create it.

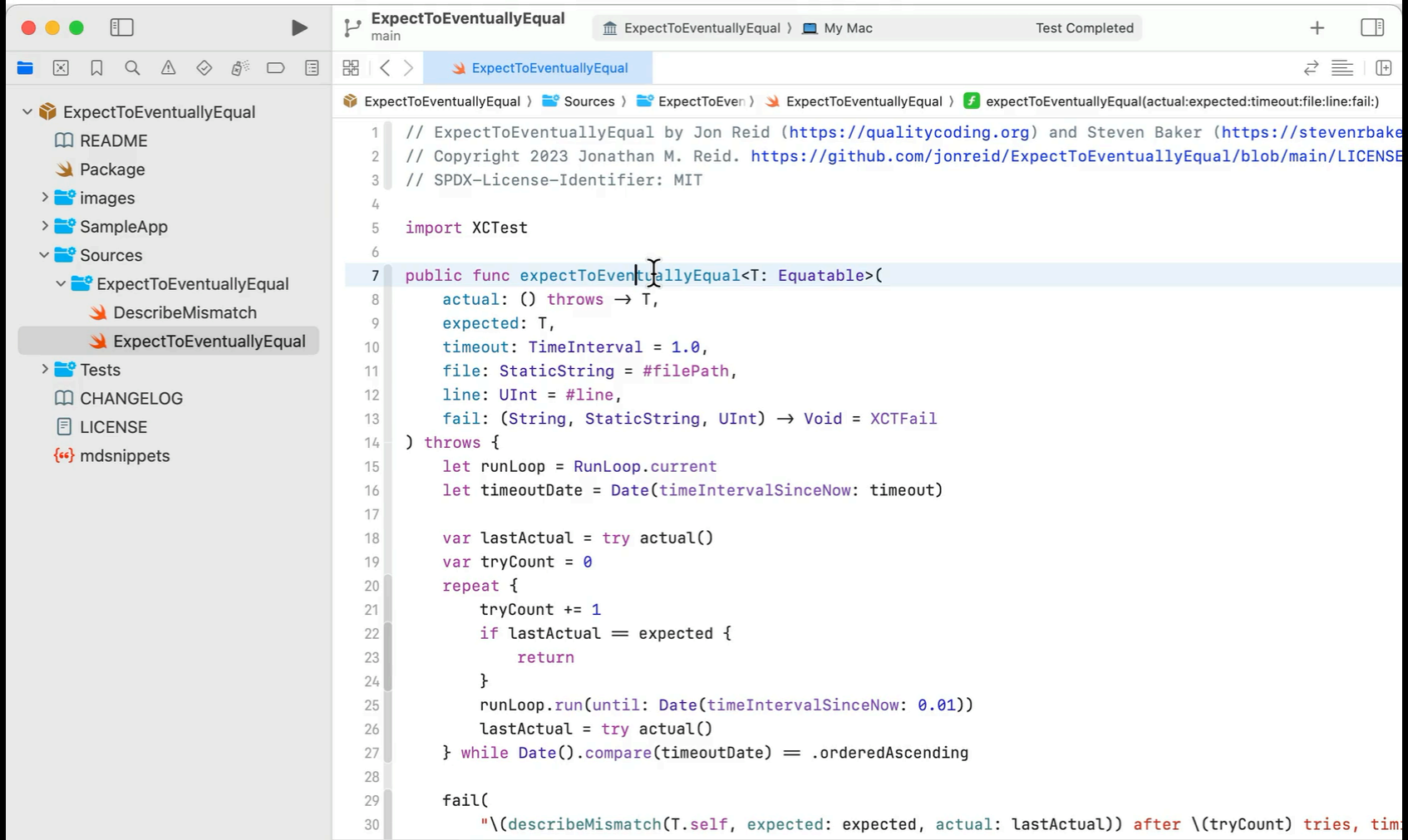
Stay focused on
caller's point-of-view

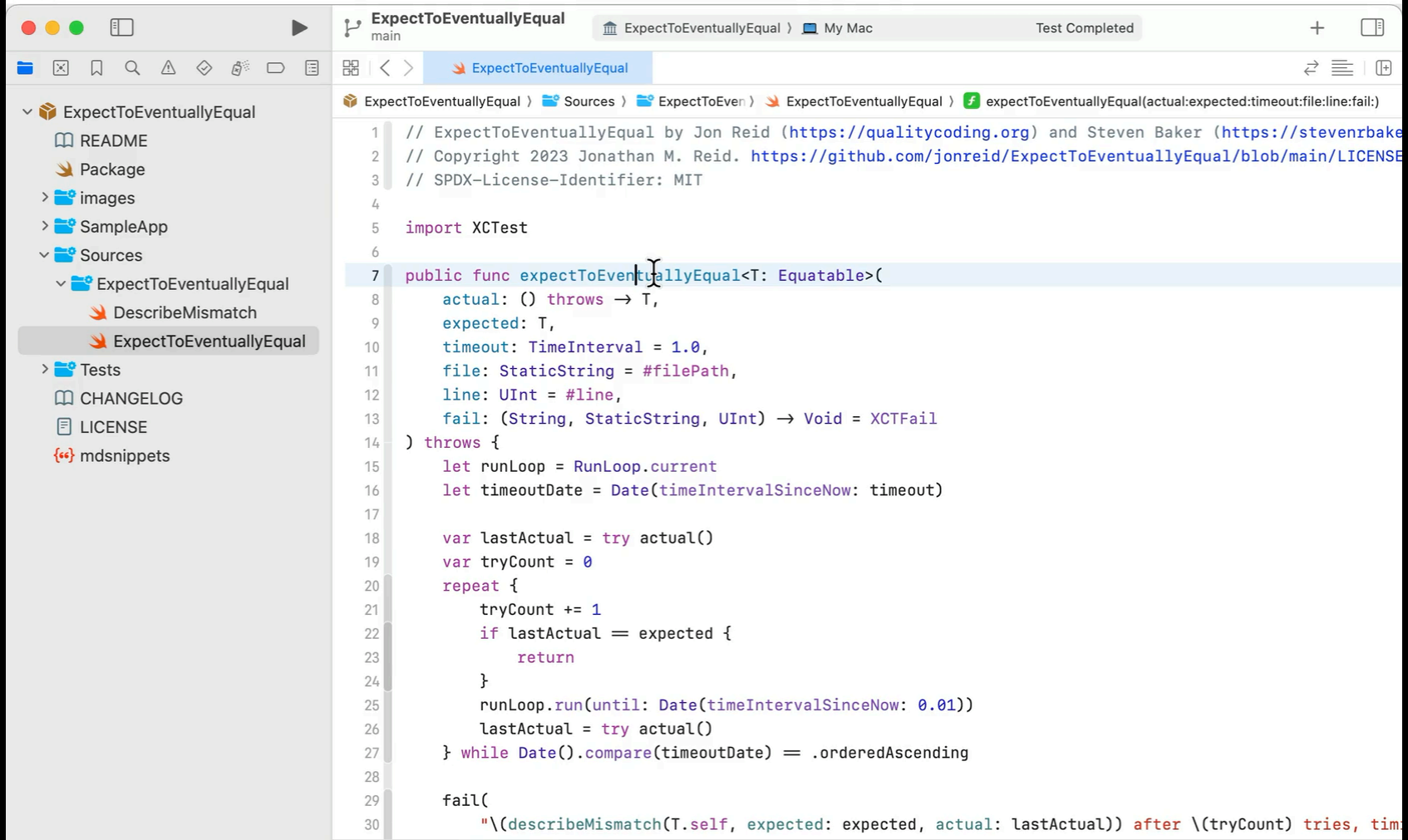
Available as Eclipse "Quick Fix" since... 2002?

5: REFACTORING

Are we out of options?

I don't think so.


Xcode File Edit View Find Navigate Editor Product Debug Integrate Window Help
Tue May 7 10:37 AM
ExpectToEventuallyEqual
main
ExpectToEventuallyEqual 〉 My Mac
Test Completed
ExpectToEventuallyEqual
ExpectToEventuallyEqual 〉 Sources 〉 ExpectToEven 〉 ExpectToEventuallyEqual 〉 expectToEventuallyEqual(actual:expected:timeout:file:line:fail:)
ExpectToEventuallyEqual
README
Package
images
SampleApp
Sources
ExpectToEventuallyEqual
DescribeMismatch
ExpectToEventuallyEqual
Tests
CHANGELOG
LICENSE
mdsnippets
1 // ExpectToEventuallyEqual by Jon Reid (https://qualitycoding.org) and Steven Baker (https://stevenrbake
2 // Copyright 2023 Jonathan M. Reid. https://github.com/jonreid/ExpectToEventuallyEqual/blob/main/LICENSE
3 // SPDX-License-Identifier: MIT
4
5 import XCTest
6
7 public func expectToEventuallyEqual<T: Equatable>(
8 actual: () throws → T,
9 expected: T,
10 timeout: TimeInterval = 1.0,
11 file: StaticString = #filePath,
12 line: UInt = #line,
13 fail: (String, StaticString, UInt) → Void = XCTFail
14) throws {
15 let runLoop = RunLoop.current
16 let timeoutDate = Date(timeIntervalSinceNow: timeout)
17
18 var lastActual = try actual()
19 var tryCount = 0
20 repeat {
21 tryCount += 1
22 if lastActual == expected {
23 return
24 }
25 runLoop.run(until: Date(timeIntervalSinceNow: 0.01))
26 lastActual = try actual()
27 } while Date().compare(timeoutDate) == .orderedAscending
28
29 fail(
30 "\(describeMismatch(T.self, expected: expected, actual: lastActual)) after \(tryCount) tries, tim
Filter
Line: 7 Col: 25


Xcode File Edit View Find Navigate Editor Product Debug Integrate Window Help
Tue May 7 10:37 AM
ExpectToEventuallyEqual
main
ExpectToEventuallyEqual 〉 My Mac
Test Completed
ExpectToEventuallyEqual
ExpectToEventuallyEqual 〉 Sources 〉 ExpectToEven 〉 ExpectToEventuallyEqual 〉 expectToEventuallyEqual(actual:expected:timeout:file:line:fail:)
ExpectToEventuallyEqual
README
Package
images
SampleApp
Sources
ExpectToEventuallyEqual
DescribeMismatch
ExpectToEventuallyEqual
Tests
CHANGELOG
LICENSE
mdsnippets
1 // ExpectToEventuallyEqual by Jon Reid (https://qualitycoding.org) and Steven Baker (https://stevenrbake
2 // Copyright 2023 Jonathan M. Reid. https://github.com/jonreid/ExpectToEventuallyEqual/blob/main/LICENSE
3 // SPDX-License-Identifier: MIT
4
5 import XCTest
6
7 public func expectToEventuallyEqual<T: Equatable>(
8 actual: () throws → T,
9 expected: T,
10 timeout: TimeInterval = 1.0,
11 file: StaticString = #filePath,
12 line: UInt = #line,
13 fail: (String, StaticString, UInt) → Void = XCTFail
14) throws {
15 let runLoop = RunLoop.current
16 let timeoutDate = Date(timeIntervalSinceNow: timeout)
17
18 var lastActual = try actual()
19 var tryCount = 0
20 repeat {
21 tryCount += 1
22 if lastActual == expected {
23 return
24 }
25 runLoop.run(until: Date(timeIntervalSinceNow: 0.01))
26 lastActual = try actual()
27 } while Date().compare(timeoutDate) == .orderedAscending
28
29 fail(
30 "\(describeMismatch(T.self, expected: expected, actual: lastActual)) after \(tryCount) tries, tim
Filter
Line: 7 Col: 25

Recommended Key Bindings for Xcode

^⌘N

Rename

Conflicts with “New Workspace” so delete that.

^⌘V

Extract All Occurrences

Don’t map to “Extract to Variable” which extracts only one occurrence.

^⌘M

Extract to Method

Often doesn’t work, but now you can try it quickly to see.

⌘⌫

Delete Line

Conflicts with “Delete to Beginning of Line” but easier than triple-clicking.

industrial logic

Automated Refactoring

Xcode

- Rename
- Extract to Function
- Extract to Method
- Extract to Variable
- Extract All Occurrences

AppCode (Swift)

- Rename
- Change Signature
- Extract Variable
- Extract Closure
- Extract Function (Method)
- Inline
- Safe Delete

industrial logic

Automated Refactoring

AppCode (Swift)

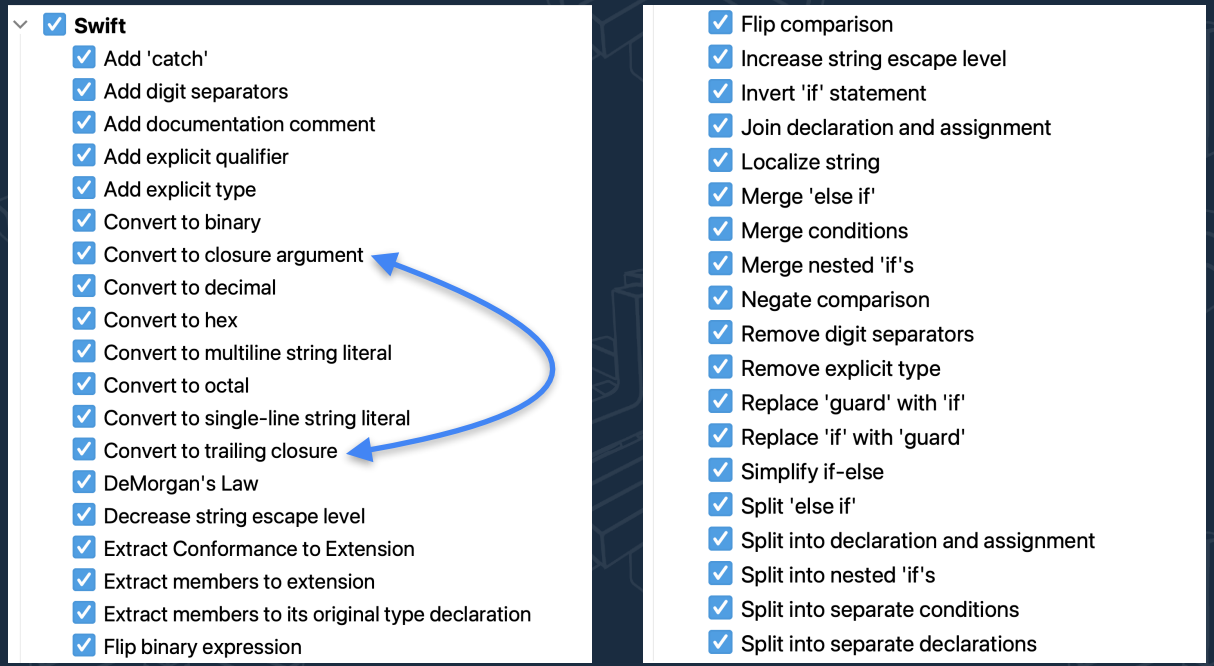
- Rename
- Change Signature
- Extract Variable
- Extract Closure
- Extract Function (Method)
- Inline
- Safe Delete

AppCode (additional for Objective-C)

- Convert to Method
- Convert to Block
- Convert to Property
- Convert to Instance Variable
- Move Member
- Extract Block
- Extract Category
- Extract Constant
- Extract Define
- Extract Instance Variable
- Extract Parameter
- Extract Property
- Extract Protocol
- Extract Subclass
- Extract Superclass
- Extract Typedef
- Pull Members Up
- Push Members Down

industrial logic

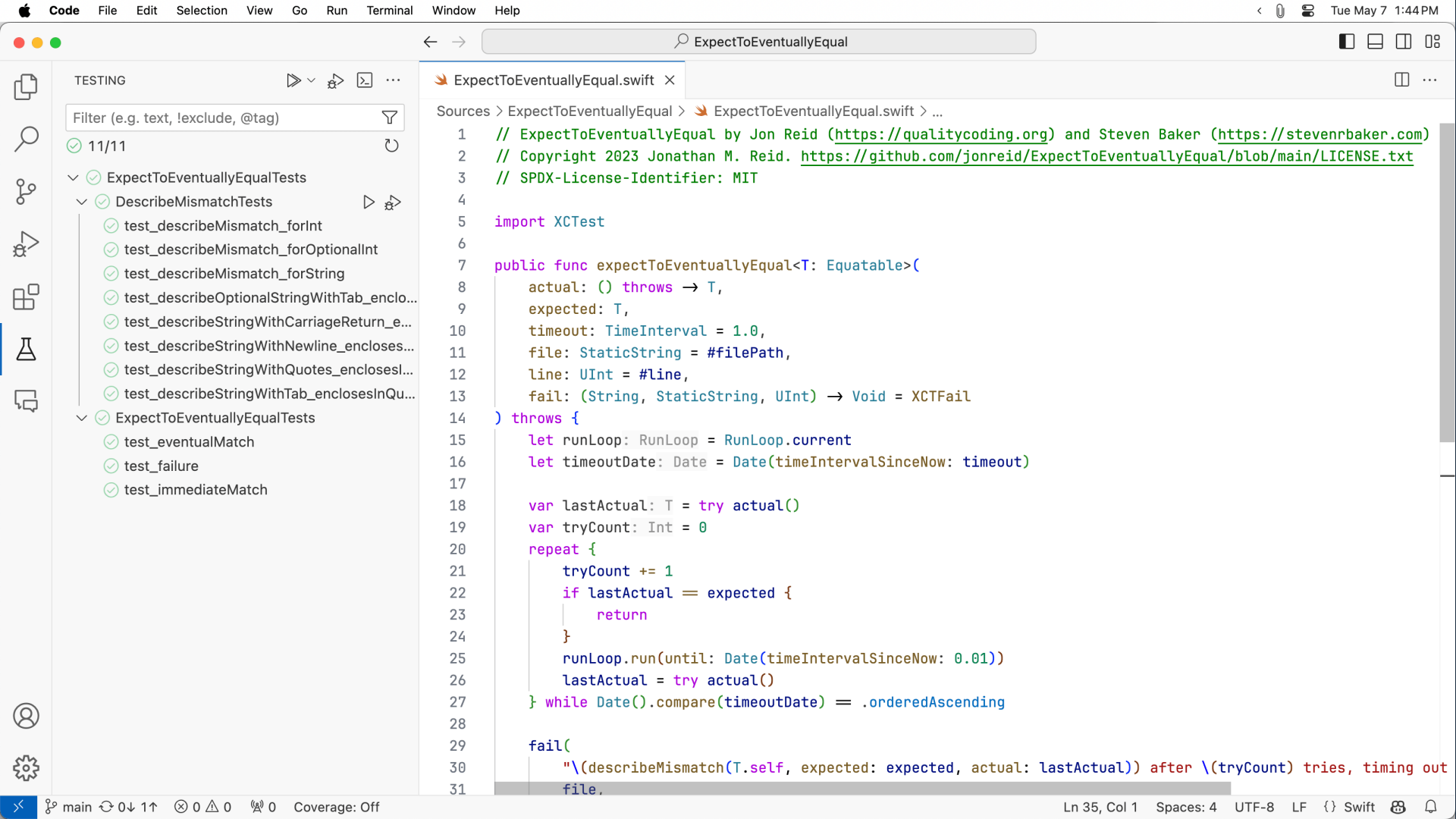
AppCode Intentions



industrial logic

DeMorgan's Law

industrial logic


Code File Edit Selection View Go Run Terminal Window Help
Tue May 7 1:44 PM
ExpectToEventuallyEqual
TESTING
Filter (e.g. text, !exclude, @tag)
11/11
ExpectToEventuallyEqualTests
DescribeMismatchTests
test_describeMismatch_forInt
test_describeMismatch_forOptionalInt
test_describeMismatch_forString
test_describeOptionalStringWithTab_enclo...
test_describeStringWithCarriageReturn_e...
test_describeStringWithNewline_encloses...
test_describeStringWithQuotes_enclosesI...
test_describeStringWithTab_enclosesInQu...
ExpectToEventuallyEqualTests
test_eventualMatch
test_failure
test_immediateMatch
ExpectToEventuallyEqual.swift
Sources > ExpectToEventuallyEqual > ExpectToEventuallyEqual.swift > ...
1 // ExpectToEventuallyEqual by Jon Reid (https://qualitycoding.org) and Steven Baker (https://stevenrbaker.com)
2 // Copyright 2023 Jonathan M. Reid. https://github.com/jonreid/ExpectToEventuallyEqual/blob/main/LICENSE.txt
3 // SPDX-License-Identifier: MIT
4
5 import XCTest
6
7 public func expectToEventuallyEqual<T: Equatable>(
8 actual: () throws → T,
9 expected: T,
10 timeout: TimeInterval = 1.0,
11 file: StaticString = #filePath,
12 line: UInt = #line,
13 fail: (String, StaticString, UInt) → Void = XCTFail
14) throws {
15 let runLoop: RunLoop = RunLoop.current
16 let timeoutDate: Date = Date(timeIntervalSinceNow: timeout)
17
18 var lastActual: T = try actual()
19 var tryCount: Int = 0
20 repeat {
21 tryCount += 1
22 if lastActual == expected {
23 return
24 }
25 runLoop.run(until: Date(timeIntervalSinceNow: 0.01))
26 lastActual = try actual()
27 } while Date().compare(timeoutDate) == .orderedAscending
28
29 fail(
30 "\(describeMismatch(T.self, expected: expected, actual: lastActual)) after \(tryCount) tries, timing out
31 file,
main 0↓ 1↑ 0 0 0 Coverage: Off
Ln 35, Col 1 Spaces: 4 UTF-8 LF Swift

Language Server Protocol

industrial logic

EXTENSIONS

Search Extensions in Marketplace

INSTALLED 9

Installed

l pair programmer
GitHub

GitHub Copilot Chat 33ms
AI chat features powered by Copilot
GitHub

Pylance
A performant, feature-rich language ser...
Microsoft

Python
Python language support with extensio...
Microsoft

Python Debugger
Python Debugger extension using debu...
Microsoft

Swift 16ms
Swift Language Support for Visual Studi...
Swift Server Work Group

Test Adapter Converter 2ms
Converter extension from the Test Adap...
Microsoft

Test Explorer UI 13ms
Run your tests in the Sidebar of Visual ...
Holger Benl

RECOMMENDED 6

ExpectToEventuallyEqual.swift | *Extension: Swift*

Swift v1.9.0

Swift Server Work Group | 256,447 | ★★★★★ (24)

Swift Language Support for Visual Studio Code.

Disable | Uninstall

This extension is enabled globally.

DETAILS FEATURES CHANGELOG DEPENDENCIES

Swift for Visual Studio Code

This extension adds language support for Swift to Visual Studio Code. It supports:

- Code completion
- Jump to definition, peek definition, find all references, symbol search
- Error annotations and apply suggestions from errors
- Automatic generation of launch configurations for debugging with CodeLLDB
- Automatic task creation
- Package dependency view
- Test Explorer view

Swift support uses SourceKit LSP for the language server to power code completion and LLDB to enable debugging.

The extension is developed by members of the Swift Community and maintained by the SSWG. The aim is to provide a first-class, feature complete extension to mak developing Swift applications on all platforms a seamless experience.

Categories

Programming Languages
Debuggers

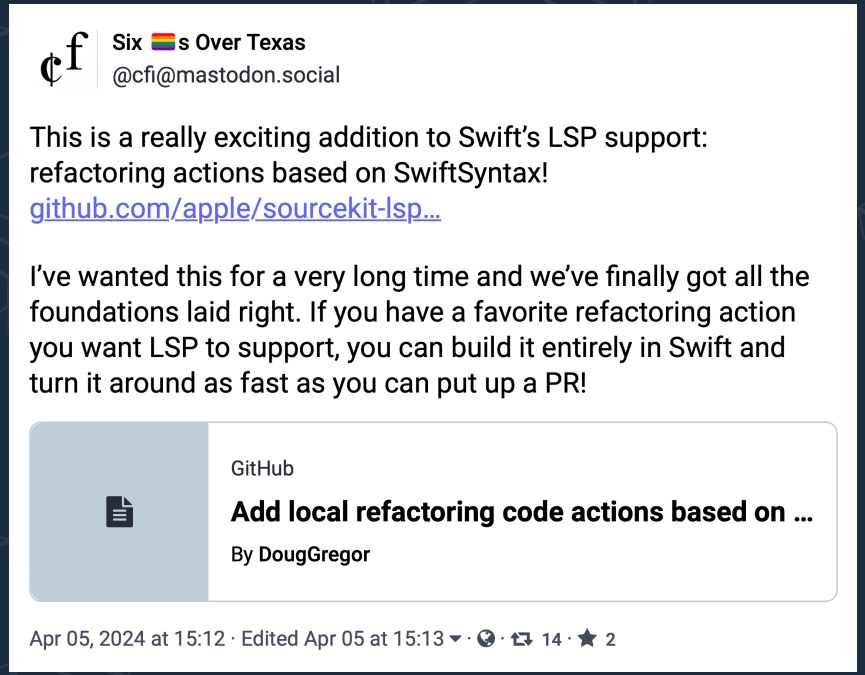
Resources

Marketplace
Issues
Repository
License
Swift Server Work Group

More Info

Published	2021-12-23, 17:36:51
Last	2024-04-17,

A Ray of Hope!



¢f **Six 🏳️‍🌈s Over Texas**
@cfi@mastodon.social

This is a really exciting addition to Swift's LSP support: refactoring actions based on SwiftSyntax!
github.com/apple/sourcekit-lsp…

I've wanted this for a very long time and we've finally got all the foundations laid right. If you have a favorite refactoring action you want LSP to support, you can build it entirely in Swift and turn it around as fast as you can put up a PR!

GitHub

Add local refactoring code actions based on …

By **DougGregor**

Apr 05, 2024 at 15:12 · Edited Apr 05 at 15:13 · 14 · 2

Add local refactoring code actions based on swift-syntax #1169

Closed **DougGregor** wants to merge 6 commits into `apple:main` from `DougGregor:syntax-refactoring-actions`

Conversation 100 | Commits 6 | Checks 0 | Files changed 11

DougGregor commented on Apr 5 • edited — Member

Introduce new local refactoring code actions based on the Swift syntax tree, without going through SourceKit.

This change includes a number of new refactoring code actions. Most of them adapt the syntax refactorings from the `SwiftRefactor` module of swift-syntax:

- Add digit separators to an integer literal, e.g., `100000` -> `1_000_000`.
- Remove digit separators from an integer literal, e.g., `1_000_000` -> `1000000`. (from SwiftRefactor)
- Format a raw string literal, e.g., `"Hello \#(world)"` -> `##"Hello \#(world)"##`
- Migrate to new if let syntax, e.g., `if let x = x { ... }` -> `if let x { ... }`
- Replace opaque parameters with generic parameters, e.g., `func f(p: some P)` --> `func f<T1: P>(p: T1)`.

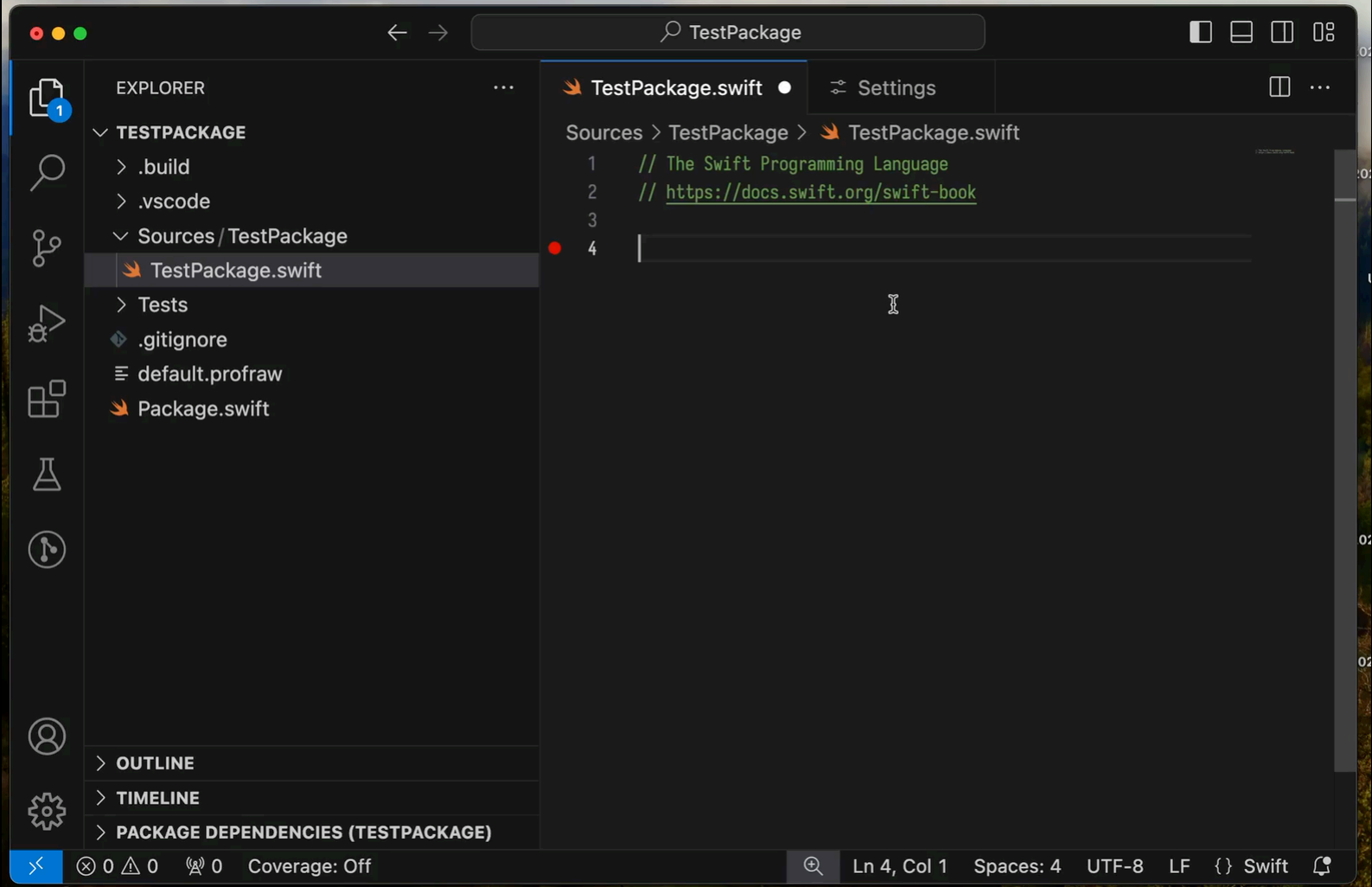
This is generally easy to do, requiring one conformance to provide a name for the refactoring:

```
extension AddSeparatorsToIntegerLiteral: SyntaxRefactoringCodeActionProvider {
  public static var title: String { "Add digit separators" }
}
```

There are also a few new refactoring code actions implemented here. These could be sunk down into SwiftRefactor to be made more widely available, or stay here if they're too IDE-centric to live in swift-refactor:

- Convert an integer literal from one base to another (e.g., `172387` -> `0x2a163`).
- Convert JSON into a Codable struct.
- Apply Demorgan's law, e.g., `!(a || b)` --> `!a && !b`.

There is a small amount of overlap between the refactoring code actions added here and those SourceKit provides. For example,


TestPackage
EXPLORER
TESTPACKAGE
.build
.vscode
Sources / TestPackage
TestPackage.swift
Tests
.gitignore
default.profraw
Package.swift
OUTLINE
TIMELINE
PACKAGE DEPENDENCIES (TESTPACKAGE)
TestPackage.swift
Settings
Sources > TestPackage > TestPackage.swift
1 // The Swift Programming Language
2 // https://docs.swift.org/swift-book
3
4
0 0 0 Coverage: Off
Ln 4, Col 1 Spaces: 4 UTF-8 LF Swift

How to Refactor Without Tools?

Principles ⇒ Practices ⇒ Tools

industrial logic

How to Refactor Without Tools?

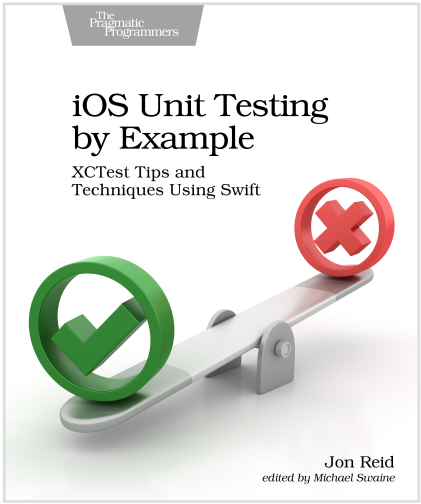
industrial logic

About *Jon Reid*

Senior Consultant, Industrial Logic

Better Software Sooner

industrial logic



Quality Coding

industrial logic

IDEs: End of Story?

AppCode

Xcode

Tools Outside Apple

SwiftLint

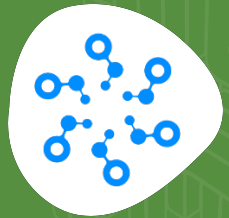
SwiftFormat

Tuist

Muter

Sourcery

Injection III



Periphery

ViewInspector

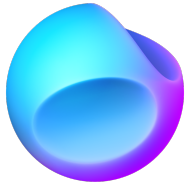
ApprovalTests

industrial logic

Keep an Eye on These New IDEs

Check the Fleet Features Matrix for Swift

Visual Studio Code



JetBrains Fleet

industrial logic

What Can *You* Do?

1. **Craft the Tools You Need**
   You may start solo, but it will thrive as a community effort.

2. **Try Other Environments**
   Learn a new language/IDE combination (especially by JetBrains).

3. **Connect with Crafters**
   Join XP or Software Crafting meet-ups. Or make your own!

industrial logic

Connect with Crafters

Meetup

- London Software Craftsmanship
- PhillyXP
- Seattle Software Crafters

...Far too many to list!

SoCraTes
BE, DE, FR, IT, UK

Global Day
Of Coderetreat

Form your own
book club

industrial logic

Principles ⇒ Practices ⇒ Tools