

10-Day iOS Development Bootcamp

Swift Basics to App Publishing

Ravi Shankar

iOS/Swift/SwiftUI Trainer and Developer

 25+ years in tech industry

 9 years focussed on iOS/Swift

 Freelance iOS Developer

 Co-author of Udemy Course

Published Apps Across Apple Platforms

 iPhone

 Mac

 Apple Watch

Aspiring Indie App Developer

Working towards success in independent app development

10-Day iOS Development Training

<> Swift Fundamentals

- Swift Basics
- Collections
- Functions & Closures
- Object-Oriented Programming

📁 Data & Networking

- Data Storage Options
- Core Data
- Networking & API Integration
- Concurrency & Multitasking

📱 iOS & UI Development

- iOS Core Concepts
- UIKit Essentials
- SwiftUI Fundamentals
- Navigation & Multi-View Apps

⚙️ Advanced Topics

- Design Patterns (MVC, MVVM)
- Performance Optimization
- Debugging & Profiling
- Testing (Unit & UI)

"From Swift basics to publishing your app, we'll cover it all!"

Our Training Approach

Hands-on Coding

Extensive practical exercises and real-world projects

Coding Challenges

Regular challenges to test and reinforce your skills

MCQ Assessments

Comprehensive online testing of key concepts

Individual App Assessment

Personal project to demonstrate individual skills

Final App Showcase

Present your group's MVP to demonstrate applied learning

"From interactive learning to collaborative app development - build your iOS skills through practical application!"

Day 1 Agenda

✓ Apple Ecosystem

✓ Playground Overview

✓ Swift Basics

✓ Enums

✓ Optionals and Guard

✓ Closures

Day 2 Agenda

✓ Class

✓ Struct

✓ Protocol

✓ Extension

✓ Generics

Day 3 Agenda

✓ Memory Management

✓ Type Casting

✓ Error Handling

✓ Access Control

✓ Introduction to iOS

✓ UIKit

Memory Management

weak

- Creates a reference that doesn't increase the reference count
- Automatically set to `nil` when the referenced object is deallocated
- Used to avoid retain cycles, especially in parent-child relationships
- Example: `weak var artifact: CulturalArtifact?`

unown

- Similar to weak, but doesn't change to `nil`
- Use when you're sure the reference will never be nil during its lifetime
- Can cause crashes if used incorrectly (accessing after deallocation)
- Example: `unowned let camera: Photographer`

[weak self]

- Prevents retain cycles in closures that capture `self`
- Allows `self` to be deallocated even if the closure is still alive

Swift Type Casting

Operator	Description	Usage
is	<ul style="list-style-type: none">Used to check if an instance is of a certain type.	If item is Monument {}
as	<ul style="list-style-type: none">Upcasting (casting to a superclass)	let culturalItem = monument as CulturalItem
as?	<ul style="list-style-type: none">Downcast to a subclassReturns an optional of the target type	If let monument = item as? Monument
as!	<ul style="list-style-type: none">Forces a downcastif not possible, it triggers a runtime error.	let knownMonument = culturalItems[0] as! Monument

Error Handling

- Defining Errors using enums
- Throwing Errors
- Handling errors with do-catch blocks
- Different variations of the try keyword (try, try?, try!)
- Rethrowing errors
- Using the Result type
- Asynchronous error handling

Error Handling

try and try?

```
// Using do-catch to handle errors
do {
    let result = try recognizeMonument(image: "taj_mahal")
    print(result)
} catch SwiftLensError.unrecognizedMonument {
    print("Oops! We couldn't recognize this monument.")
} catch SwiftLensError.networkError {
    print("Oops! Network Error.")
} catch {
    print("An unexpected error occurred: \(error)")
}

// Using try? to handle errors (returns an optional)
print("\nNaina: We can also use try? for a simpler approach.")
if let result = try? recognizeMonument(image: "eiffel_tower") {
    print(result)
} else {
    print("Failed to recognize the monument.")
}
```

Error Handling

Result Type

```
// API call function
func fetchIPInfo(completion: @escaping (Result<IPInfo, Error>) -> Void) {
    let apiUrl = URL(string: "http://ip-api.com/json/")!
    let session = URLSession.shared
```

API Error Handling

```
fetchIPInfo { result in
    switch result {
    case .success(let ipInfo):
        print("API Response:")
        print("Status: \(ipInfo.status)")
        print("Country: \(ipInfo.country)")
        print("City: \(ipInfo.city)")
        print("ISP: \(ipInfo.isp)")
        print("Latitude: \(ipInfo.lat)")
        print("Longitude: \(ipInfo.lon)")
        // ... print other fields as needed
    case .failure(let error):
        print("Error: \(error.localizedDescription)")
    }
}
```

Swift Access Control

Access Level	Description	Scope
open	Least restrictive; allows access from any source file in any module. Can be subclassed and overridden outside its defining module.	Class only
public	Allows access from any source file in any module. Cannot be subclassed or overridden outside its defining module.	Any declaration
internal	Allows access from any source file within the same module. This is the default level.	Any declaration
fileprivate	Restricts access to the current source file.	Any declaration
private	Most restrictive; limits access to the enclosing declaration and extensions in the same file.	Any declaration

UIKit Storyboard and Xcode Basics

UI Storyboard Basics

Label: Used to display text (e.g., welcome message)

Button: Triggers actions when tapped

@IBOutlet: Connects UI elements to code @IBOutlet weak var welcomeLabel: UILabel!

@IBAction: Connects button taps to code @IBAction func buttonTapped(_ sender: UIButton) { }

Xcode Interface Elements

Project Navigator: Shows project files and structure

Inspector Pane:

Attributes Inspector: Modify properties of UI elements

Connection Inspector: Manage outlets and actions

Debug/Console Window: Displays output and errors

Interface Builder: Visual editor for storyboards

Assistant Editor: Split view to see code and storyboard simultaneously

UIKit Storyboard and Xcode Basics

Creating a Basic UI

- Open storyboard in Interface Builder
- Drag and drop a Label and a Button onto the view
- Use Assistant Editor to create outlets and actions
- Implement button action to update label text:

```
@IBAction func buttonTapped(_ sender: UIButton) {  
    welcomeLabel.text = "Welcome to UIKit!"  
}
```

Running and Testing

- Use the play button or Cmd+R to build and run the app
- Test the UI in the simulator
- Check the console for any errors or debug information

Common Errors

IBOutlet not set

Error: "Key-value coding-compliant" error

Cause: IBOutlet connection is missing or incorrect in Interface Builder

Missing ViewController in Assistant Editor

Issue: When the Assistant Editor is open, the ViewController is not shown

Cause: Incorrect file linking or naming conventions

Crash when accessing an unconnected IBOutlet

Error: "Unexpectedly found nil while unwrapping an Optional value"

Cause: Attempting to use an IBOutlet that hasn't been properly connected

Debugging in iOS: Expectations vs. Reality



Expectation

Methodical detective work



Reality

Why is this optional nil?!

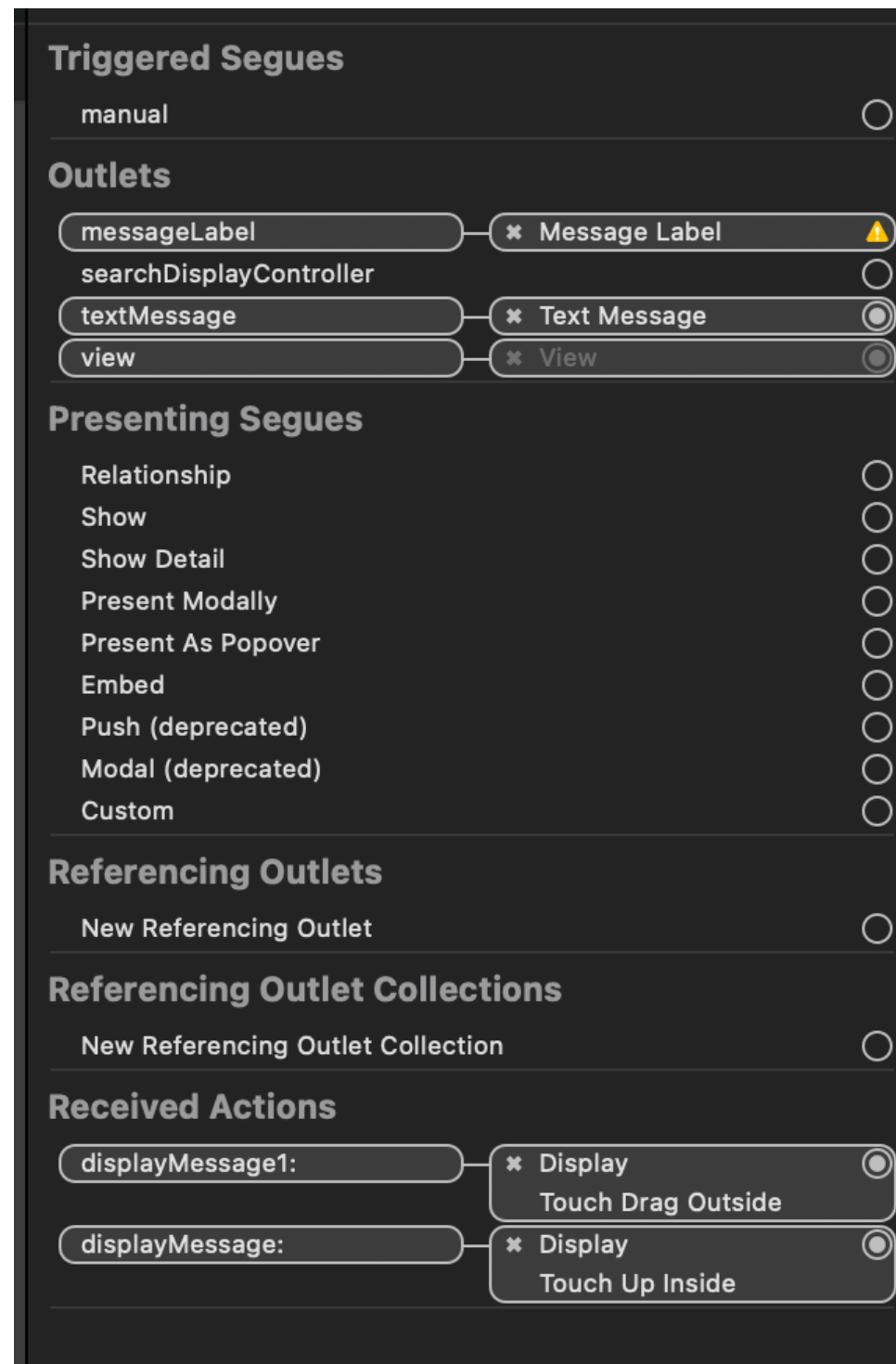
When you spend hours debugging only to realize you forgot to connect an IBOutlet...

Common Errors

IBOutlet not set

Error: "Key-value coding-compliant" error

Cause: IBOutlet connection is missing or incorrect in Interface Builder

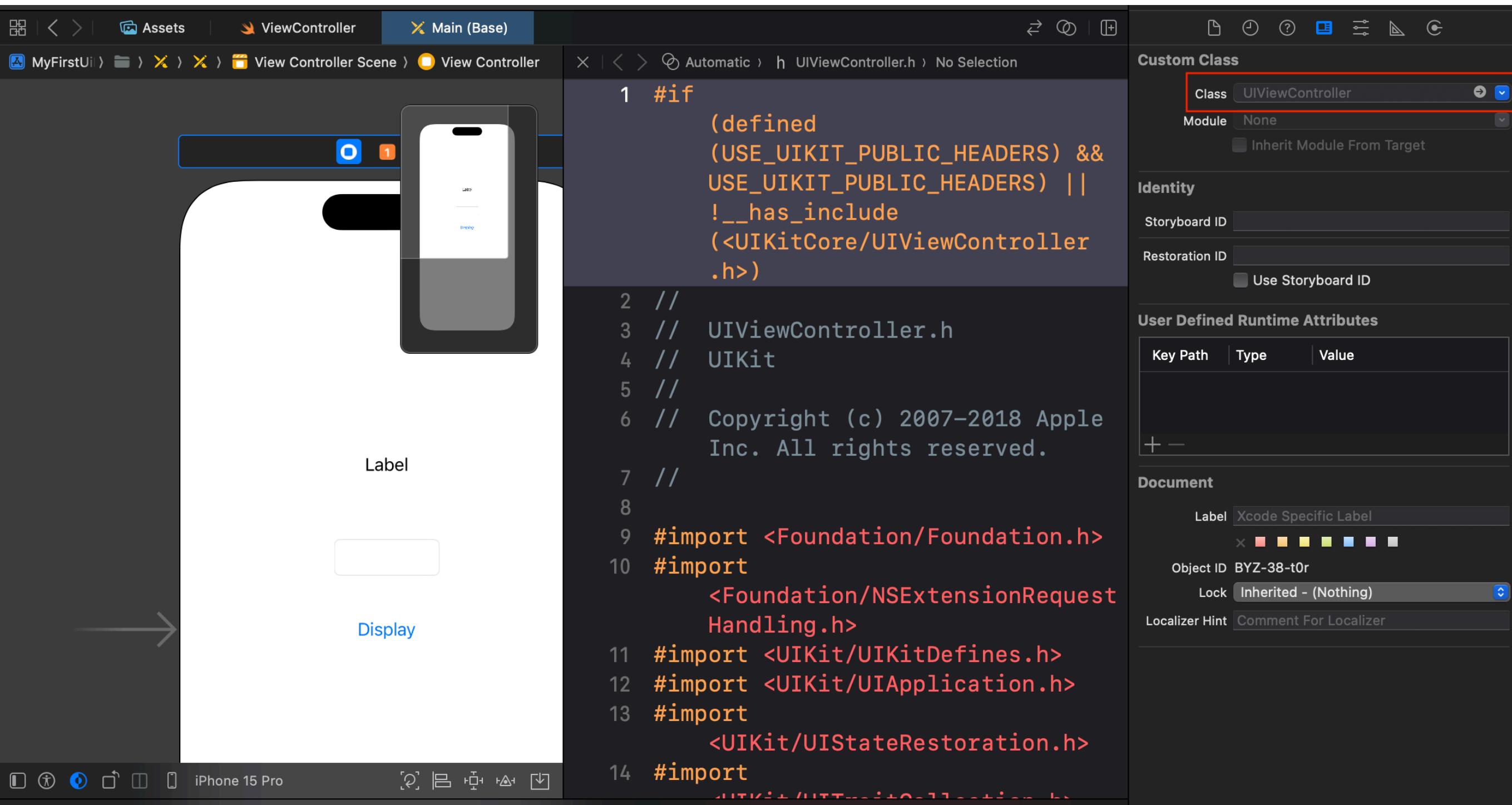


Common Errors

Missing ViewController in Assistant Editor

Issue: When the Assistant Editor is open, the ViewController is not shown

Cause: Incorrect file linking or naming conventions

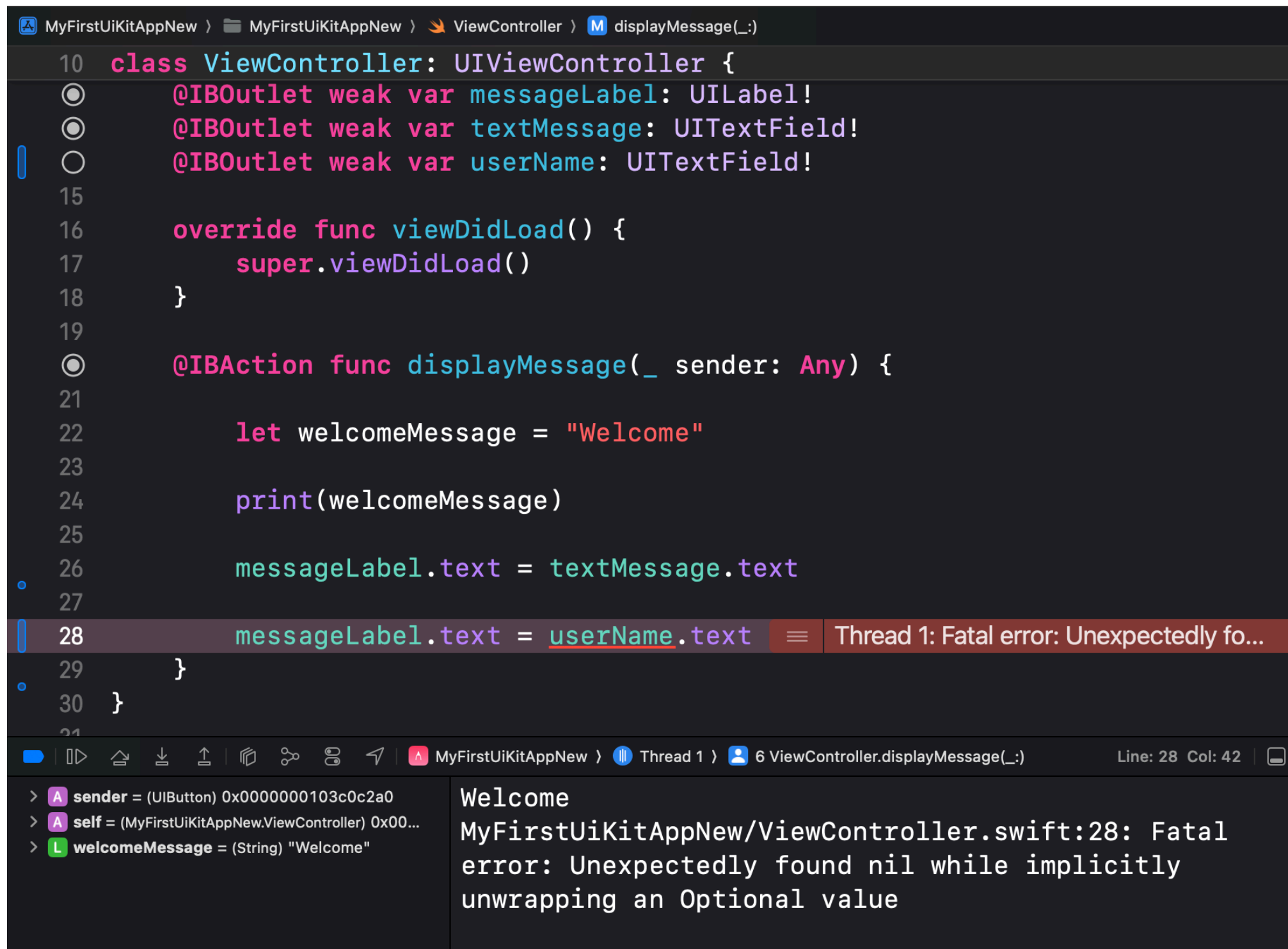


Common Errors

Crash when accessing an unconnected IBOutlet

Error: "Unexpectedly found nil while unwrapping an Optional value"

Cause: Attempting to use an IBOutlet that hasn't been properly connected



```
10 class ViewController: UIViewController {
11     @IBOutlet weak var messageLabel: UILabel!
12     @IBOutlet weak var textMessage: UITextField!
13     @IBOutlet weak var userName: UITextField!
14
15
16     override func viewDidLoad() {
17         super.viewDidLoad()
18     }
19
20     @IBAction func displayMessage(_ sender: Any) {
21
22         let welcomeMessage = "Welcome"
23
24         print(welcomeMessage)
25
26         messageLabel.text = textMessage.text
27
28         messageLabel.text = userName.text
29     }
30 }
```

Thread 1: Fatal error: Unexpectedly fo...

MyFirstUIKitAppNew > Thread 1 > 6 ViewController.displayMessage(_:) Line: 28 Col: 42

> [A] sender = (UIButton) 0x0000000103c0c2a0
> [A] self = (MyFirstUIKitAppNew.ViewController) 0x00...
> [L] welcomeMessage = (String) "Welcome"

Welcome
MyFirstUIKitAppNew/ViewController.swift:28: Fatal
error: Unexpectedly found nil while implicitly
unwrapping an Optional value