Lamps
GUI Testing View Controllers in Xcode

Jerome Chan

June 14, 2023

1 Introduction

The main purpose of this project is to demonstrate a simple way of setting up individual View Controllers for user interface testing purposes.

2 Application

The application consists of a main screen with three tabs that only display different background colors and a PassCodeViewController.

2.1 PassCodeViewController

The PassCodeViewController has three functions: create a passcode, validate a passcode, and edit a passcode. We will test each of these functions individually.

3 Setting up tests

Data needs to pass from the XCTestCase to the running application so that the running application knows how to set up the view controller for testing. This is done by storing the relevant data in the launch environment by the XCTestCase and retrieving by the app's scene delegate.

3.1 Setting up

```
let app = XCUIApplication()
app.launchEnvironment[TestLauncher.TestPhrase] = "ON"
app.launchEnvironment[TestLauncher.VCPhrase] =
    TestLauncherViewControllers.PassCode.rawValue
app.launchEnvironment[TestLauncher.ModelPhrase] =
    TestLauncherModels.PassCode1.rawValue
```

3.2 Retrieving

```
func sceneWillEnterForeground(_ scene: UIScene) {
    // Called as the scene transitions from the background to the foreground.
    // Use this method to undo the changes made on entering the background.

    if let vc = TestLauncher.viewController() {
        window?.rootViewController?.show(vc, sender: nil)
    }
}
```

4 TestLauncher

The TestLauncher is used to store the keywords for the launch environment as well as the code to retrieve

```
enum TestLauncher {
    static let TestPhrase = "TestLauncher-testing"
    static let VCPhrase = "TestLauncher-VC"
    static let ModelPhrase = "TestLauncher-model"
}
```

```
extension TestLauncher {
    static func isTestModeOn() -> Bool {
        return ProcessInfo.processInfo.environment.keys.contains(TestLauncher.TestPhrase)
    }
    static func vcPhrase() -> String? {
        return ProcessInfo.processInfo.environment[TestLauncher.VCPhrase]
    }
    static func modelPhrase() -> String? {
        return ProcessInfo.processInfo.environment[TestLauncher.ModelPhrase]
    }
    static func storyboard() -> UIStoryboard? {
        if let sb = TestLauncherViewControllers(rawValue:  vcPhrase() ?? "") {
            return sb.storyboard
        } else {
            return nil
        }
    }
    static func viewController() -> UIViewController? {
        if
            let vcPhrase = TestLauncher.vcPhrase(),
            let modelPhrase = TestLauncher.modelPhrase(),
            let vcEnum = TestLauncherViewControllers(rawValue: vcPhrase),
            let modelEnum = TestLauncherModels(rawValue: modelPhrase),
            let vc = TestLauncher.storyboard()?.instantiateInitialViewController() {
            switch vcEnum {
            case .PassCode:
                if
                    let vc = vc as? PassCodeViewController {
                    switch modelEnum {
                    case .PassCode1:
                        vc.model = PassCodeModel(passCodeString: "yishuntoapayoh")
                        vc.mode = .createPassCode
                    case .PassCode2:
                        vc.model = PassCodeModel(passCodeString: "parislondon")
                        vc.mode = .validatePassCode
                    case .PassCode3:
                        vc.model = PassCodeModel(passCodeString: "marsvenus")
                        vc.mode = .editPassCode
                    }
                }
            }
            return vc
        } else {
            return nil
        }
    }
}
```

5 TestLauncherViewControllers

This stores the different view controllers that are to be tested.

```swift
enum TestLauncherViewControllers: String {
    case PassCode = "PassCodeViewController"
}
```

6 TestLauncherModels

This stores the different modesl the view controllers need for testing.

```swift
enum TestLauncherModels: String {
    case PassCode1 = "PassCodeModel1"
    case PassCode2 = "PassCodeModel2"
    case PassCode3 = "PassCodeModel3"
}
```