

Automate iOS tests with XCUITest framework



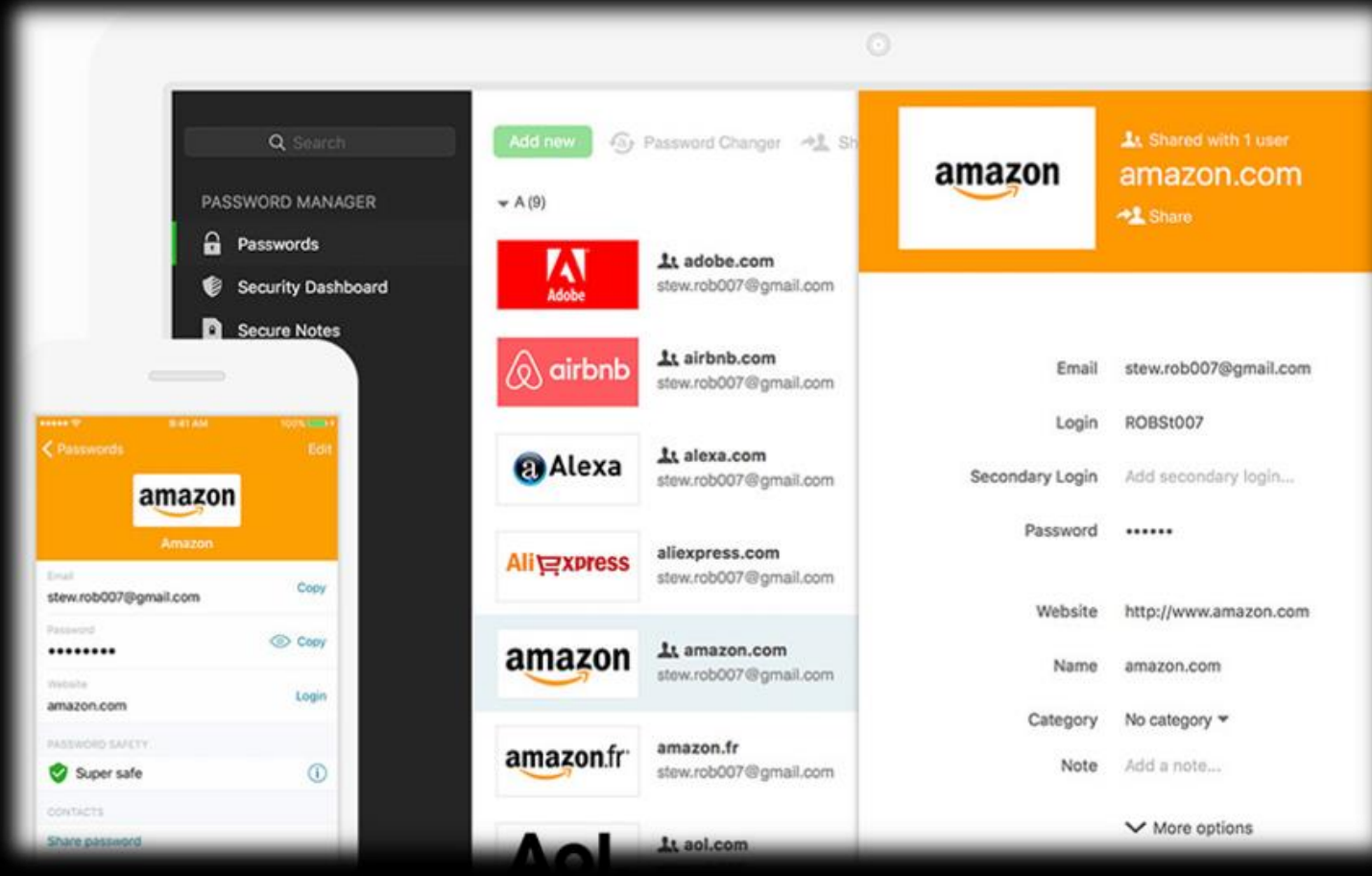
Ibrahim AZZAM

Senior Automation Engineer

@Dashlane

What is Dashlane ?

Dashlane is a Password Manager, that helps you manage your identity and your payments in a simple and secure way everywhere.





Agenda

- What are the tools we need ?
- The 2 ways to write a test ?
- How to parallelize test runs ?
- How to run tests and integrate them in CI ?

What is XCTest? 🤔

- A test framework that comes with Xcode by Apple
- It is used to test native apps (Swift, Objective-C)
- It allows interacting with an app's UI elements



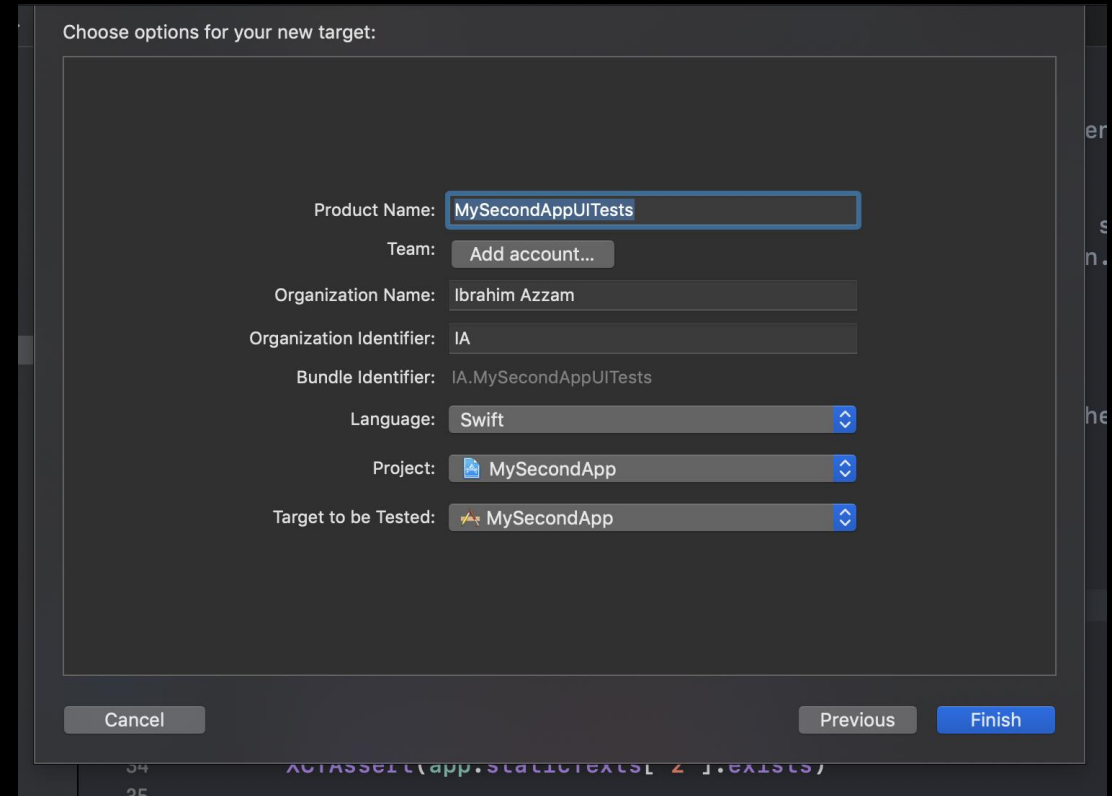
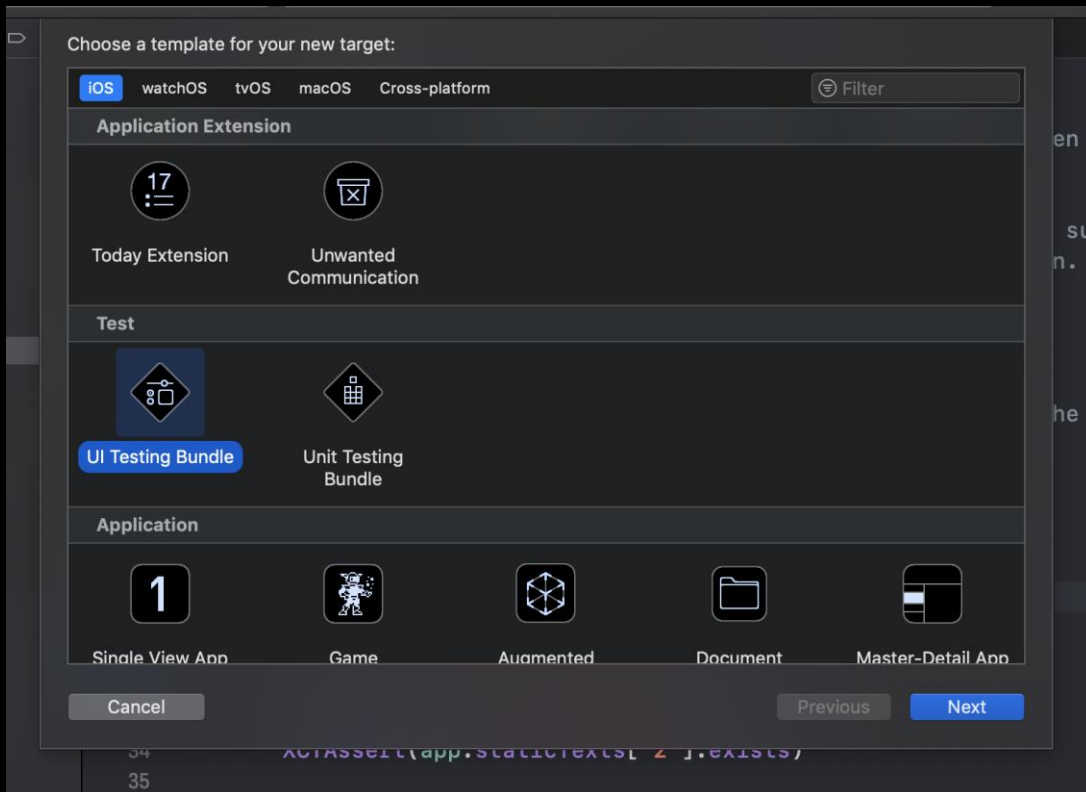
What is Swift? 🤔



- Swift is a programming language created by Apple in 2014
- It is modern, fast and easy to learn
- It is used to write tests in XCTest for Apple apps

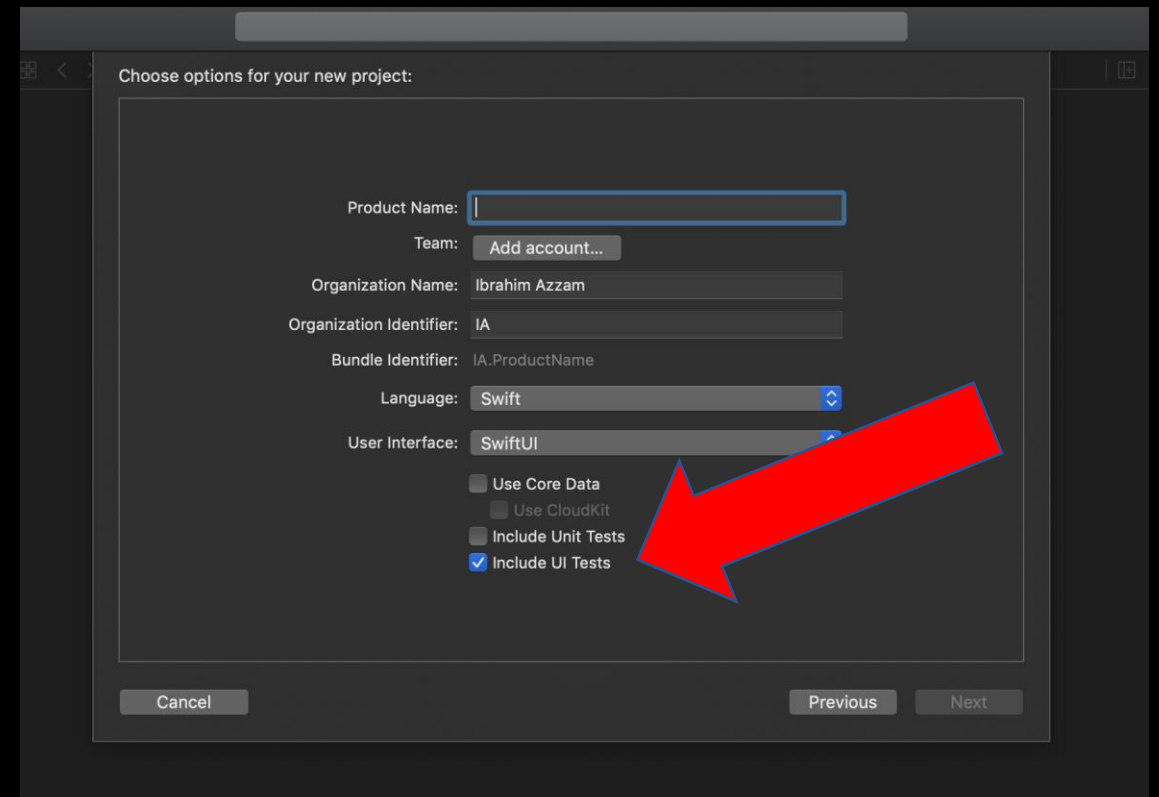
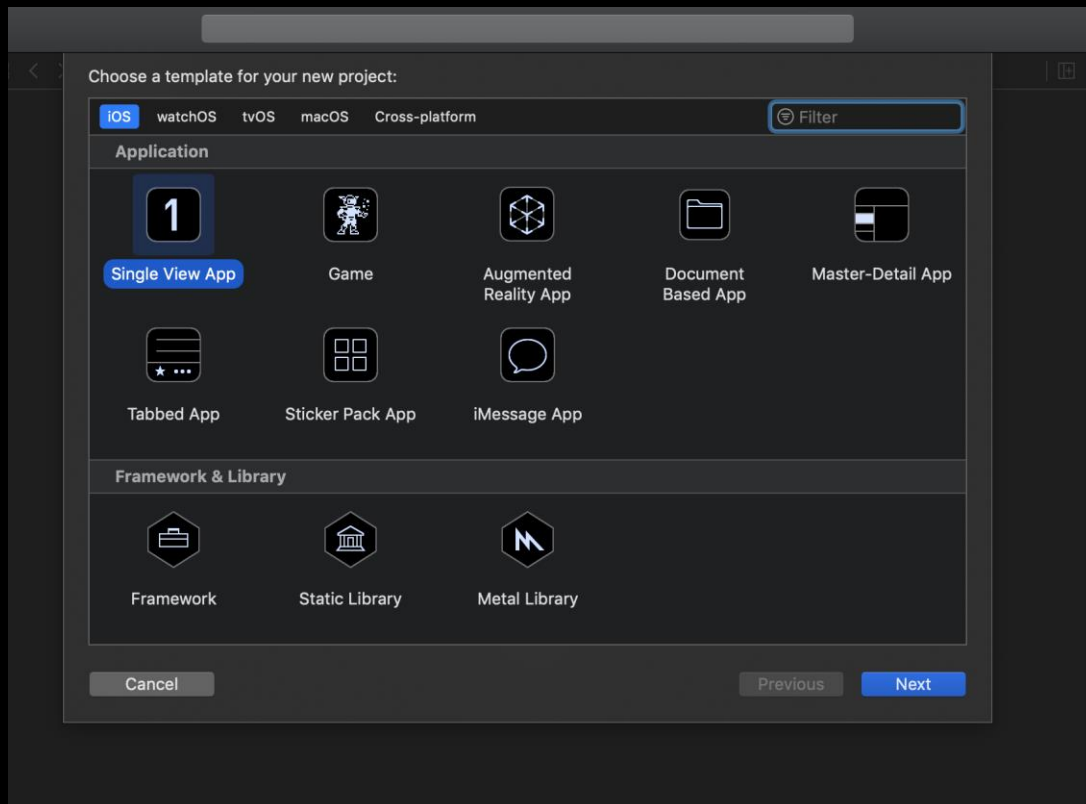
Add to existing project

1. Open your Xcode project
2. Go to: **File -> New -> Target**
3. Select **iOS UI Testing Bundle** and hit **Next**:



Create new project

1. Open Xcode
2. Go to: **File -> New -> Project**
3. Select **Single View App** and hit **Next**:

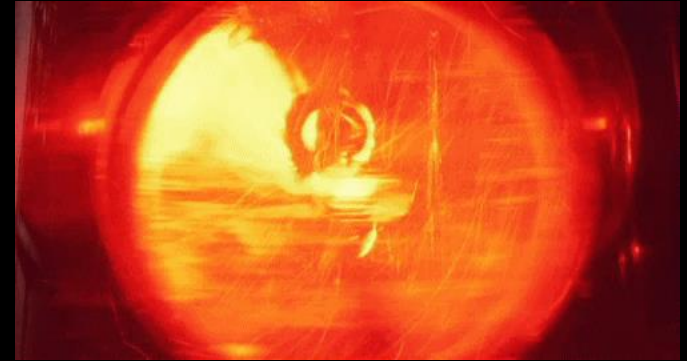


'NameOfTheApp'UITests.swift

- It contains the template code with sample test
- It also contains a `setUp()` and `tearDown()` methods
- It will contain the tests that are grouped into classes

WARNING

WARNING



Your test methods always have to start with « **test** », otherwise you won't be able to run them

The 2 ways of writing a Test Case?



Launch App

```
class UITests: XCTestCase {  
    let app = XCUIApplication()  
  
    override func setUp() {  
        super.setUp()  
  
        app.launch()  
    }  
}
```

XCUIELEMENT Class

func tap()

func doubleTap()

func twoFingerTap()

func tap(withNumberOfTaps: UInt, numberOfTouches: UInt)

func press(forDuration: TimeInterval)

func press(forDuration: TimeInterval, thenDragTo: XCUIElement)

func swipeLeft()

func swipeRight()

func swipeUp()

func swipeDown()

func pinch(withScale: CGFloat, velocity: CGFloat)

func rotate(CGFloat, withVelocity: CGFloat)

...

How to Tap a Button ?

```
app.buttons["Click here"].tap()
```

How to Type Text into a Text Field ?

```
let textField = app.textFields["First Name"]  
textField.tap()  
textField.typeText("John")
```

How to Assert an Element Exists ?

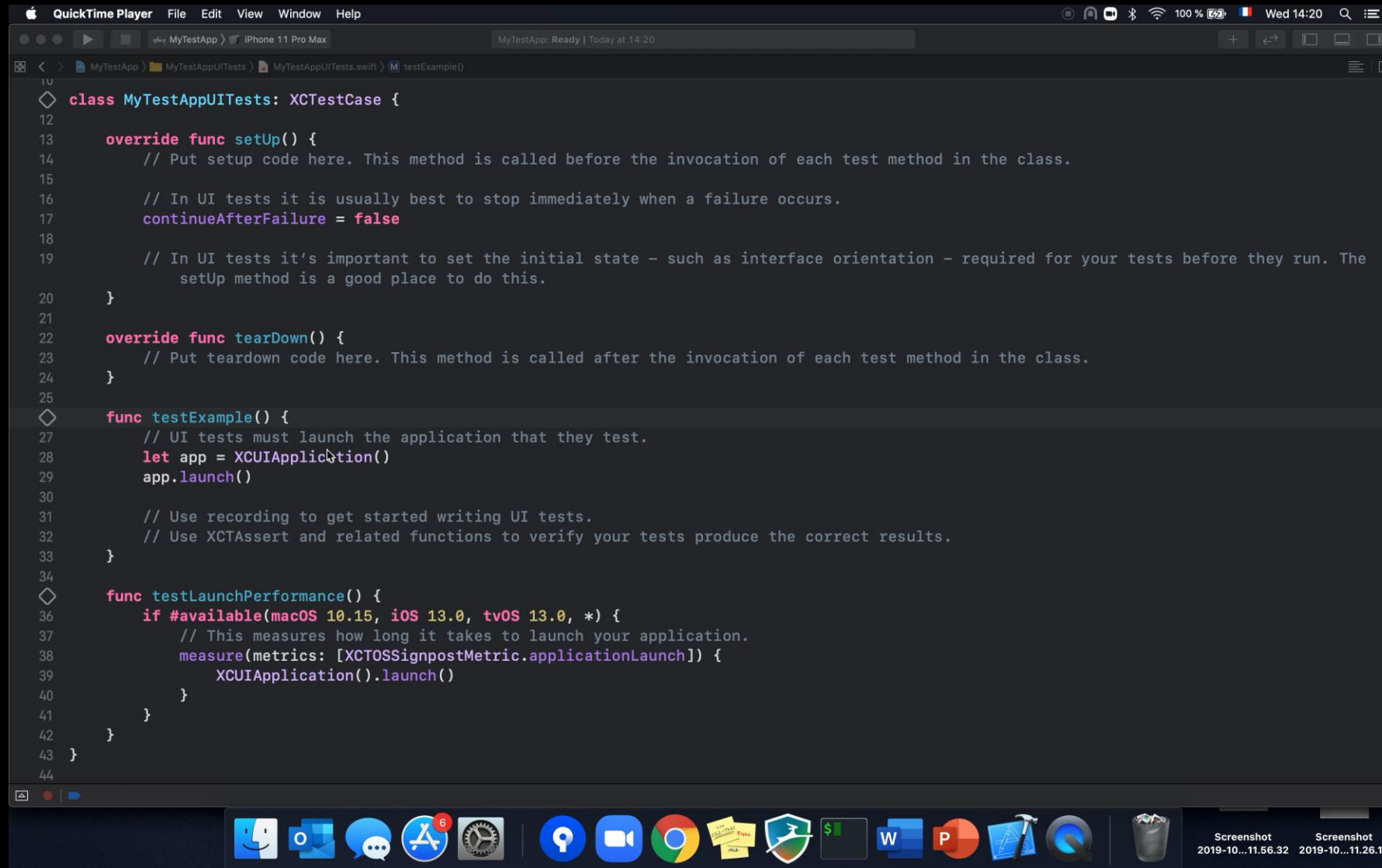
```
XCTAssert(app.staticTexts["Some Text"].exists)
```

We're Ready to Create Our First Test Case



First Test Case

```
QuickTime Player File Edit View Window Help
MyTestApp iPhone 11 Pro Max MyTestApp: Ready | Today at 14:20
MyTestApp MyTestAppUITests MyTestAppUITests.swift testExample()
10
11 class MyTestAppUITests: XCTestCase {
12
13     override func setUp() {
14         // Put setup code here. This method is called before the invocation of each test method in the class.
15
16         // In UI tests it is usually best to stop immediately when a failure occurs.
17         continueAfterFailure = false
18
19         // In UI tests it's important to set the initial state - such as interface orientation - required for your tests before they run. The
           setUp method is a good place to do this.
20     }
21
22     override func tearDown() {
23         // Put teardown code here. This method is called after the invocation of each test method in the class.
24     }
25
26     func testExample() {
27         // UI tests must launch the application that they test.
28         let app = XCUIApplication()
29         app.launch()
30
31         // Use recording to get started writing UI tests.
32         // Use XCTAssert and related functions to verify your tests produce the correct results.
33     }
34
35     func testLaunchPerformance() {
36         if #available(macOS 10.15, iOS 13.0, tvOS 13.0, *) {
37             // This measures how long it takes to launch your application.
38             measure(metrics: [XCTOSSignpostMetric.applicationLaunch]) {
39                 XCUIApplication().launch()
40             }
41         }
42     }
43 }
44
```



What about recording ?

The image shows a screenshot of the Xcode IDE running on a Mac. The main window displays Swift code for UI tests. The code includes a `testBlue()` function and a `testRecorded()` function. The `testBlue()` function performs two taps on the "Add 1 to Blue" button and asserts that the text "1" and "2" exist. The `testRecorded()` function is currently empty. To the right of the code editor, a simulated iPhone 11 Pro interface is shown. The interface displays three counters: a blue counter with the value 0, a red counter with the value 0, and a green counter with the value 0. Below the counters are three buttons: "Add 1 to Blue" (blue), "Add 3 to Red" (red), and "Add 5 to Green" (green). At the bottom of the screen is a "Reset" button. The Xcode interface includes a sidebar on the left with a project navigator, a top menu bar, and a bottom toolbar. The system tray at the bottom of the screen shows various application icons and a system clock indicating the time is 6:32 on 2019-10-11.

```
25 }
26
27 ✓
28 func testBlue() {
29     app.buttons["Add 1 to Blue"].tap()
30     XCTAssert(app.staticTexts["1"].exists)
31     app.buttons["Add 1 to Blue"].tap()
32     XCTAssert(app.staticTexts["2"].exists)
33 }
34
35 ◇
36 func testRecorded(){
37 }
38
39 }
40
```

9:57

0
0
0

Add 1 to Blue
Add 3 to Red
Add 5 to Green

Reset

6:32 2019-10-11.26.19

Run tests

The image shows a screenshot of the Xcode IDE. On the left, the Swift code for `MySecondAppUITests` is displayed. The code includes a `setUp` method for launching the app and a `testBlue` method that performs UI assertions. On the right, a simulated iPhone 11 Pro device is shown with a home screen featuring icons for Files, Shortcuts, Contacts, Watch, and two instances of MySecondApp.

```
10 class MySecondAppUITests: XCTestCase {
11     let app = XCUIApplication()
12     override func setUp() {
13         // Put setup code here. This method is called before the invocation of each test method
14         // in the class.
15         app.launch()
16         super.setUp()
17         // In UI tests it is usually best to stop immediately when a failure occurs.
18         continueAfterFailure = false
19
20         // In UI tests it's important to set the initial state - such as interface orientation -
21         // required for your tests before they run. The setUp method is a good place to do this.
22     }
23     override func tearDown() {
24         // Put teardown code here. This method is called after the invocation of each test method
25         // in the class.
26     }
27
28     func testBlue() {
29         app.buttons["Add 1 to Blue"].tap()
30         XCTAssert(app.staticTexts["1"].exists)
31         app.buttons["Add 1 to Blue"].tap()
32         XCTAssert(app.staticTexts["2"].exists)
33     }
34 }
35 }
36 }
```

The device screen shows the time 2:38, 100% battery, and the date Wed 14:38. The home screen has a red and orange background with icons for Files, Shortcuts, Contacts, Watch, and two instances of MySecondApp. The dock at the bottom contains icons for Safari, Messages, and another app.

At the bottom of the screenshot, the Windows taskbar is visible with various application icons. A status bar at the bottom right indicates "Screenshot 0.32 2019-10-11.26.19".

What happens when you have more than 1 test ?



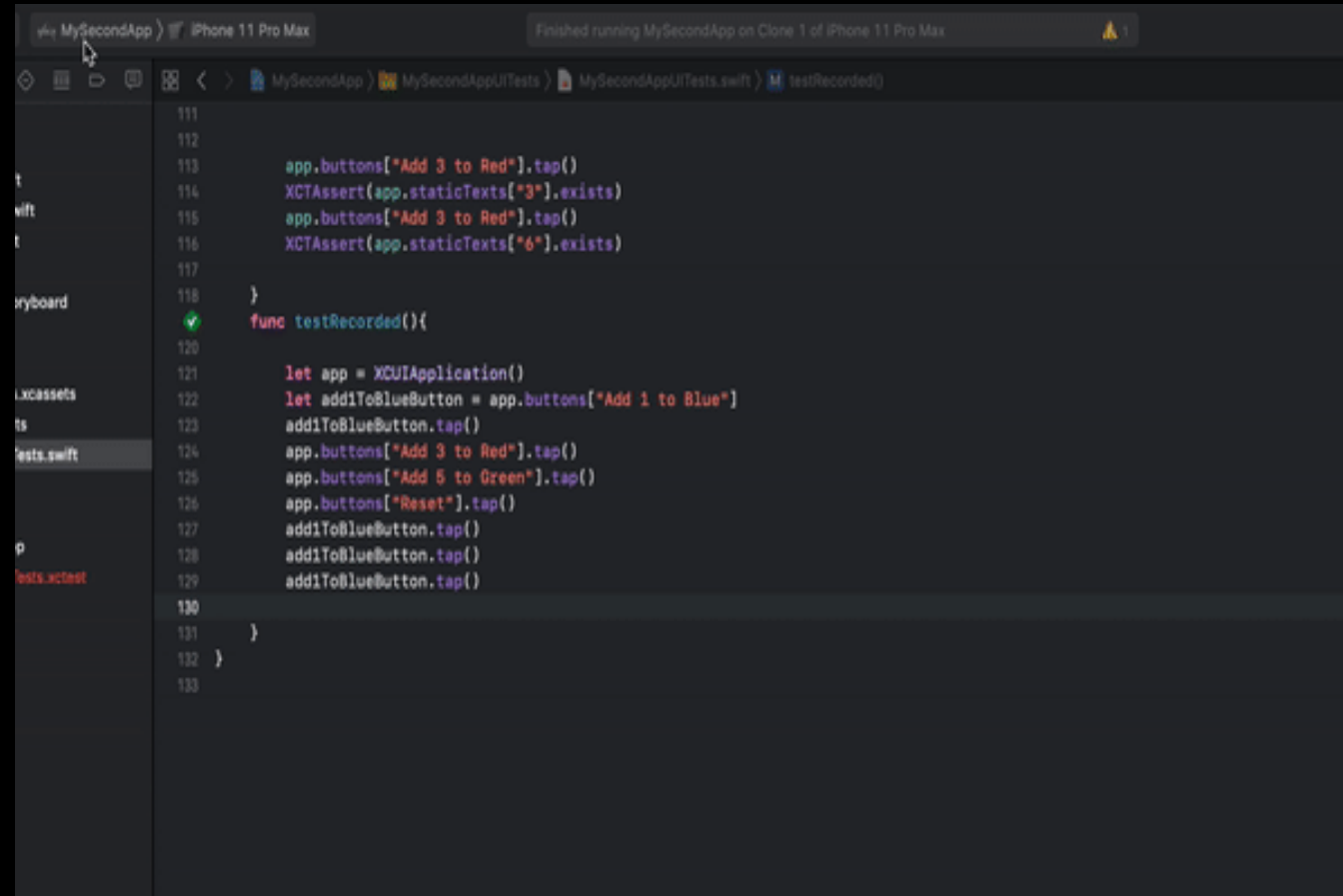
Parallel Testing



Parallel testing allows multiple tests to be run simultaneously, with the main advantage being dramatically shortened XCTest and XCUITest times.

How to enable parallel testing

1. Select your target scheme in Xcode, and "Edit Scheme..."
2. Find the settings for "Test", and press on the "Info" tab
3. You'll see a list of your Unit and UI tests, press on the associated "Options..." button
4. Select "Execute in parallel on Simulator"
5. Optionally select "Randomize execution order"



The screenshot shows the Xcode IDE with a Swift test file open. The file is named `testRecorded()` and is located within the `MySecondApp` project. The code defines a `testRecorded()` function that performs several UI actions and assertions. The actions include tapping buttons labeled "Add 3 to Red", "Add 1 to Blue", "Add 5 to Green", and "Reset". The assertions check for the existence of static text elements with labels "3" and "6". The code is as follows:

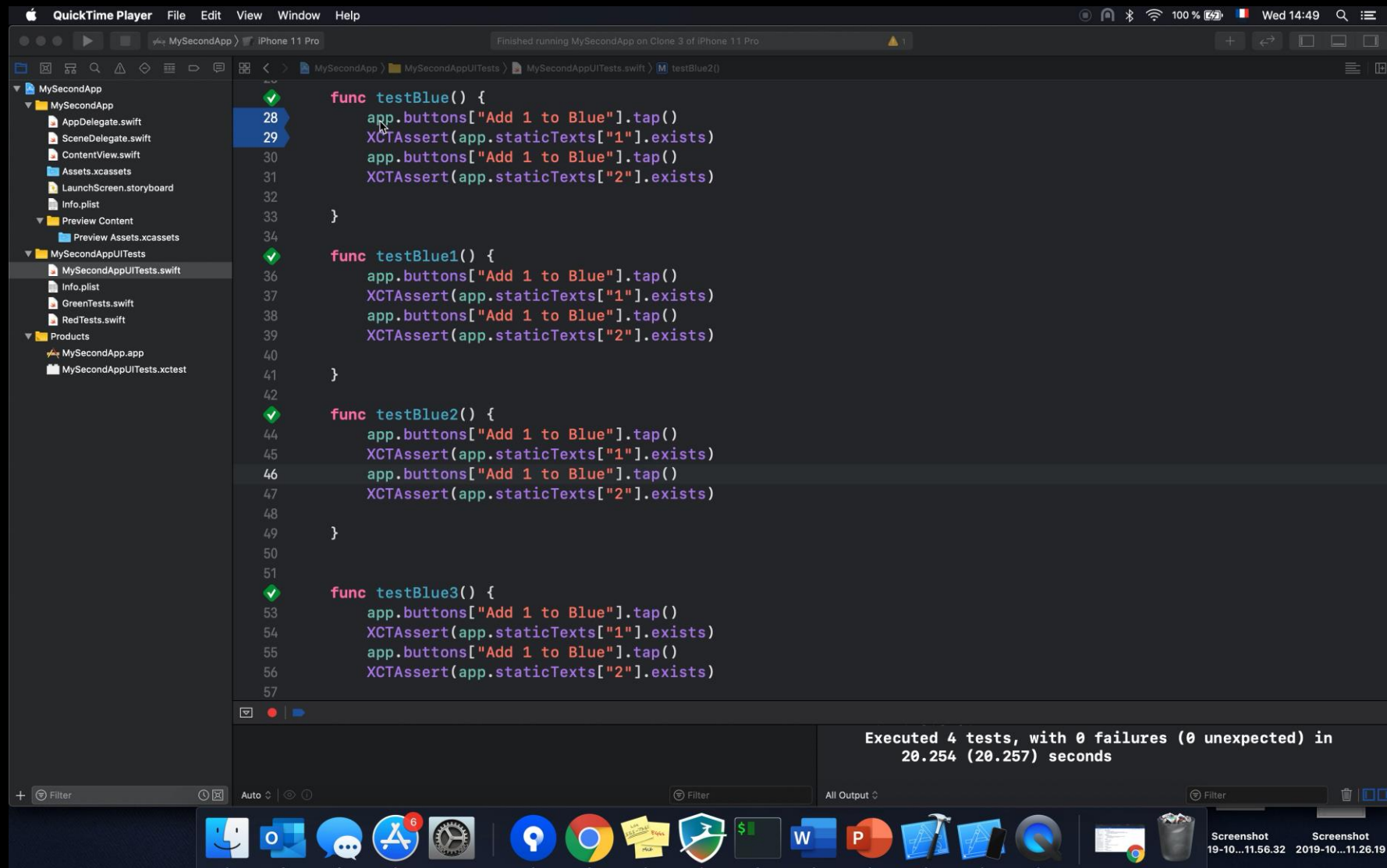
```
111
112
113     app.buttons["Add 3 to Red"].tap()
114     XCTAssert(app.staticTexts["3"].exists)
115     app.buttons["Add 3 to Red"].tap()
116     XCTAssert(app.staticTexts["6"].exists)
117
118 }
119
120
121 func testRecorded(){
122
123     let app = XCUIApplication()
124     let add1ToBlueButton = app.buttons["Add 1 to Blue"]
125     add1ToBlueButton.tap()
126     app.buttons["Add 3 to Red"].tap()
127     app.buttons["Add 5 to Green"].tap()
128     app.buttons["Reset"].tap()
129     add1ToBlueButton.tap()
130     add1ToBlueButton.tap()
131     add1ToBlueButton.tap()
132 }
133
```

Considerations



- Independent tests
- Not suitable for Performance tests
- Enough resources on the machine

Run parallelized tests from XCode



And from command line (used for CI)

The screenshot shows a macOS desktop environment with a QuickTime Player window in the foreground. The background window is a terminal window titled "MySecondApp" showing the execution of a test command. The terminal output indicates that 4 tests were executed successfully with 0 failures in 22.410 seconds. The terminal window also shows a file explorer on the left side of the screen, displaying the project structure for "MySecondApp".

```
MySecondApp
├── MySecondApp
│   ├── AppDelegate.swift
│   ├── Info.plist
│   └── MySecondAppApp.swift
├── MySecondAppUITests
│   ├── Info.plist
│   └── MySecondAppUITests.swift
└── MySecondAppTests
    ├── Info.plist
    └── MySecondAppTests.swift
```

```
iazzam@parml-iazzam: ~/repos/MySecondApp
└─$ xcodebuild test -project MySecondApp.xcodeproj -scheme MySecondApp \
    -destination 'platform=iOS Simulator,name=iPhone 11 Pro' \
    -parallel-testing-enabled YES \
    -parallel-testing-worker-count 3
```

Executed 4 tests, with 0 failures (0 unexpected) in 22.410 (22.413) seconds

How much

- On 1 device: 58%
- On 3 devices: 28%

• 48%



More things to explore

- Run tests under different configurations with Test plans
- Use more of xcodebuild, xcresulttool, and xccov in CI
- Visual regression testing

Try Dashlane and get 6 months premium using the code 'SQA19'



@azzamibrahim