Automated testing with Appium on iOS

Appium

- Appium is an open source test automation framework for use with native, hybrid and mobile web applications.

- He runs iOS, Android and Windows applications using the WebDriver protocol.

CodeceptJS

- CodeceptJS is a modern end-to-end testing framework with special BDD-style syntax.

Required software

- Node
- Appium
- CodeceptsJS
- Allure - Simple and flexible multilingual test reporting tool
- HomeBrew - Package manager

Installation process

- **HomeBrew**
    - > /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

- **Node**
    - > brew install node

- **Allure**
    - > brew install allure

- **Appium**
    - > npm install -g appium
- **Appium Client**
    - > npm install wd

- **iOS-deploy**
  - iOS-deploy is a small utility for installing and debugging iPhone apps from the command line, without using Xcode.
  - > brew install ios-deploy

- **iDeviceInstaller**

  - iDeviceInstaller is a tool to interact with the installation_proxy of an iOS device that allows you to install, update, uninstall, archive, restore and enumerate installed or archived apps. It is also required to run tests on real devices.

  - > brew install ideviceinstaller

- **iOS WebKit Debug Proxy**
- Appium uses this tool to access WebViews on real devices
  - > brew install ios-webkit-debug-proxy

- **Appium Doctor**

- Appium doctor is used to verify that all Appium prerequisites are installed correctly

  - > npm install -g appium-doctor

- **Run Appium Doctor to verify that the environment is ok**

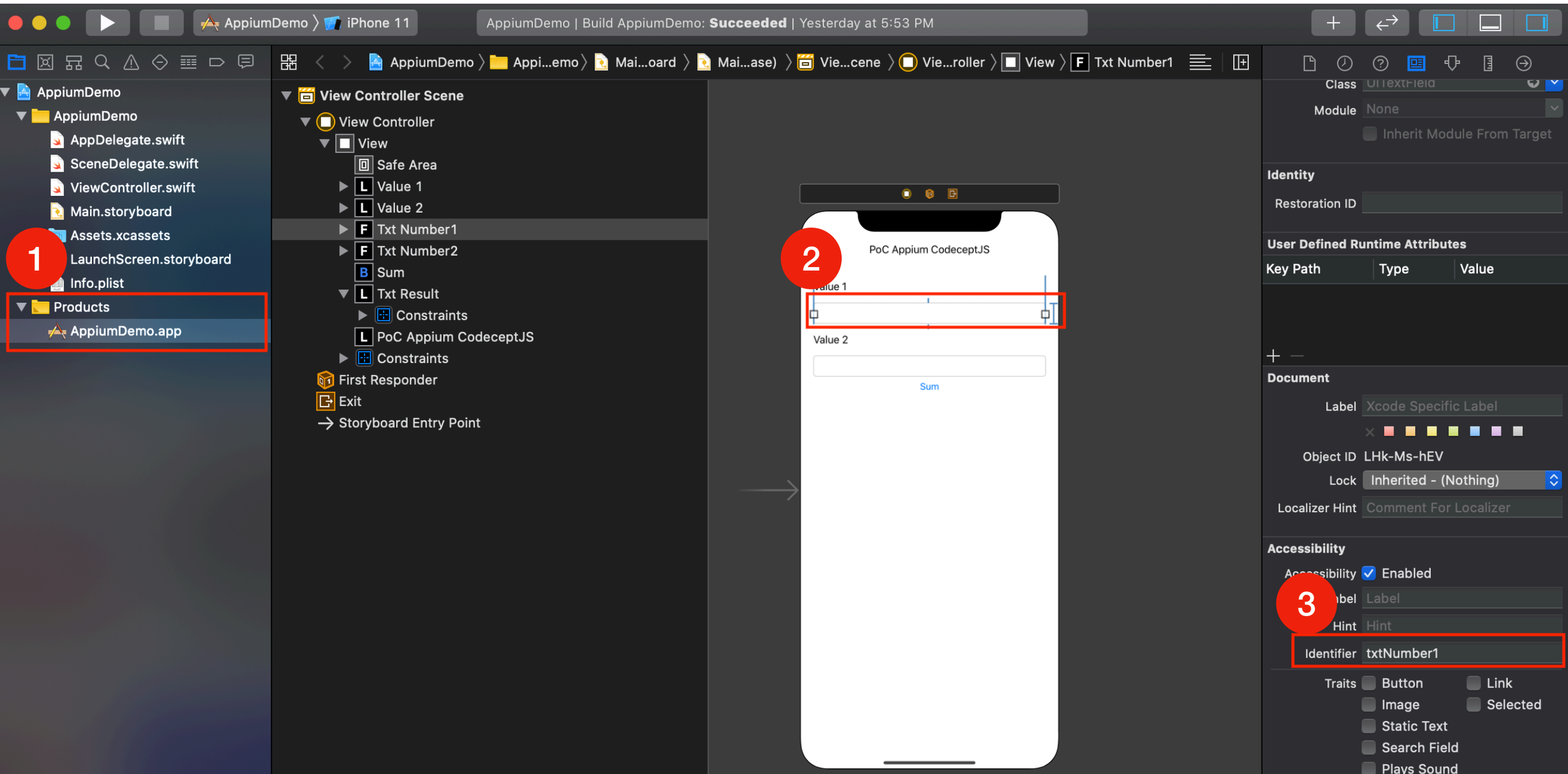
  - > appium-doctor

- **iOS WebKit Debug Proxy**
- Appium uses this tool to access WebView on real devices
  - > brew install ios-webkit-debug-proxy

- **CodeceptJS**
  - > npm install codeceptjs webdriverio --save

AppiumDemo Project

- **AppiumDemo Project**



1. App generated after project build

2. TextField in storyboard

3. Identifier of the selected TextField

Project setup

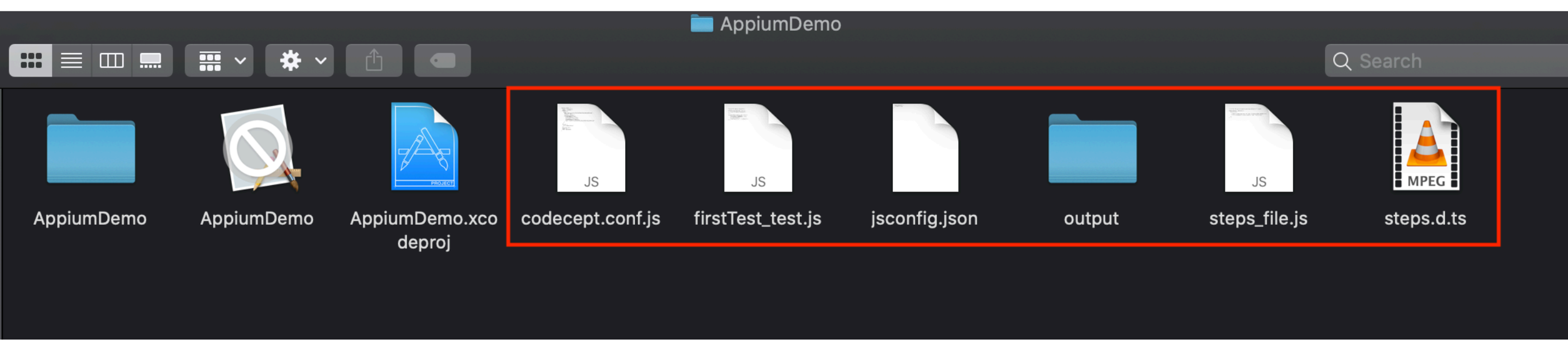
- **Create the CodeceptJS project**
  - > npm init -y

- **Start the CodeceptJS project**
  - > npx codeceptjs init

```
➜  ios git:(appium) ✗ npx codeceptjs init
npx: installed 197 in 6.405s

[  Welcome to CodeceptJS initialization tool
   It will prepare and configure a test environment for you

 Installing to /Users/rrovaron/raizen/ios
 ? Where are your tests located? ./*_test.js
 ? What helpers do you want to use? Appium
 ? Where should logs, screenshots, and reports to be stored? ./output
[? Would you like to extend the "I" object with custom steps? Yes
 ? Where would you like to place custom steps? ./steps_file.js
[? Do you want to choose localization for tests? English (no localization)
[Configure helpers...
[? [Appium] Application package. Path to file or url ./input
 ? [Appium] Mobile Platform iOS
 ? [Appium] Device to run tests on emulator
[
 Steps file created at /Users/rrovaron/raizen/ios/steps_file.js
[Config created at /Users/rrovaron/raizen/ios/codecept.conf.js
 Directory for temporary output files created at './output'
 Intellisense enabled in /Users/rrovaron/raizen/ios/jsconfig.json
 TypeScript Definitions provide autocompletion in Visual Studio Code and other IDEs
 Definitions were generated in steps.d.ts

 Almost done! Next step:
 Create your first test by executing `npx codeceptjs gt` command
```

- **Create the first test**
  - > npx codecept gt

- **Items created in the project**



- **Editing codecept.conf.js**

codecept.conf.js

codecept.conf.js > config

```
exports.config = {
  tests: './*_test.js',
  output: './output',
  helpers: {
    Appium: {
      app: '/Users/rrovaron/Desktop/AppiumDemo/AppiumDemo.app',
      platform: 'iOS',
      desiredCapabilities: {
        "platformName": "iOS",
        "platformVersion": "13.2",
        "deviceName": "iPhone 11",
        "automationName": "XCUITest",
        "app": "/Users/rrovaron/Desktop/AppiumDemo/AppiumDemo.app"
      }
    }
  },
  include: {
    I: './steps_file.js'
  },
  bootstrap: null,
  mocha: {},
  name: 'AppiumDemo'
}
```

1

2

1. The .app generated in the project must be added (Xcode), with its absolute path

2. The KEY and its respective properties must be added, mentioning the platform (iOS), the iOS version that will be tested in the simulator/device, the name of the simulator/ device, the testing framework (XCUITest), and the absolute path to the app that will be tested

Creation of the first test

- **Editing codecept.conf.js**

JS *firstTest_test.js* ×

JS firstTest_test.js > ...

```
1
2   Feature('PoC Appium CodeceptJS');
3
4   Scenario('Test Open App', (I) => {
5     I.see('PoC Appium CodeceptJS');
6   });
7
8   Scenario('Test create a sum', (I) => {
9     I.fillField('~txtNumber1', '1');
10      I.fillField('~txtNumber2', '2');
11    I.click('~btnSum');
12    I.see('Result is 3', '~txtResult');
13  });
14
```

1 2 3 4 5 6

1. **Feature** name

2. **Scenerio** name

3. Command to search for certain text somewhere on the screen

4. Command to fill text in a given field (with the identifier)

5. Command to click on a certain element on the screen (with the identifier)

6. Command to search for certain text in a certain element on the screen (with the identifier)

Test run

- **Run Appium Client**
  - > appium

- **Run tests**
  - > npx codeceptjs run

```
/Users/rrovaron/Desktop/AppiumDemo
[➜ AppiumDemo npx codeceptjs run
CodeceptJS v2.3.6
Using test root "/Users/rrovaron/Desktop/AppiumDemo"

featureTest --
  ✔ test something in 3ms

 OK  | 1 passed   // 5m
```