

iOS UITest (Build in Xcode)

技术变化 (UIAutomation---->XCUITest)

UITest是Xcode7.0之后苹果推出的，主要功能是UI测试。

相比之前而言，可以通过OC或者Swift更熟悉的语言编写UI测试代码。

UIAutomation 已经被苹果弃用,UITest最大的亮点是支持**屏幕录制**——通过对App的操作自动生成相应的测试脚本代码.即使我们不懂开发也可以很快写出**基本UI测试代码**.

UITest

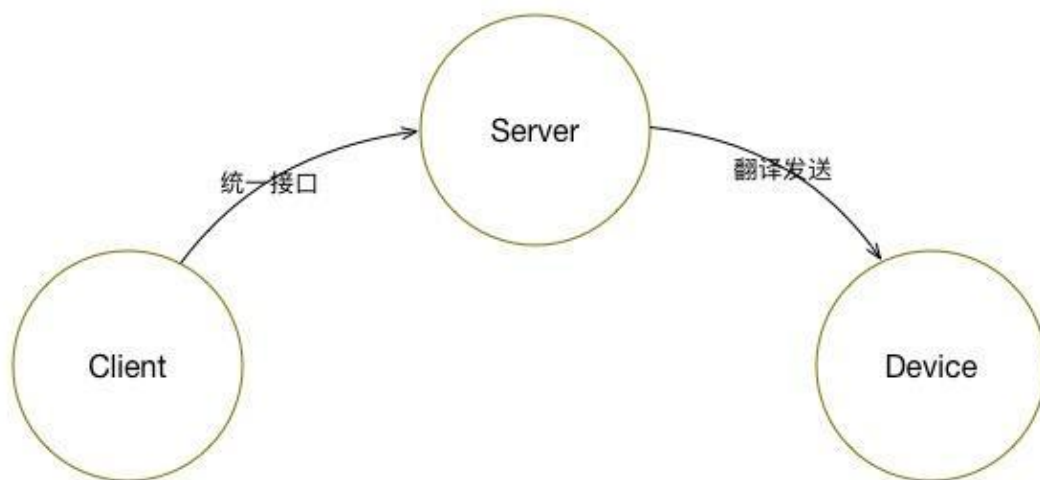
优点：1.效率很高，查找元素不需要服务器通讯。

2.相对稳定。对环境要求低。

缺点：需要源码，需要稍微了解OC/Swift语言(但是Swift很容易看懂)。

Appium

Client/Server架构是一把双刃剑：



优点：跨平台（iOS这边是UIAutomation和UITest，Android则是UiAutomator和Instrumentation），多语言支持（Java、Python、js、Ruby等）。

缺点：过多的通讯时间，环境配置复杂，兼容不同版本工作量大。

Appium's primary support for automating iOS apps is via the `XCUITest` driver. This driver leverages Apple's `XCUITest` libraries under the hood in order to facilitate automation of your app. This access to `XCUITest` is mediated by the `WebDriverAgent` server. `WebDriverAgent` (also referred to as "WDA") is a project managed by Facebook, to which the Appium core team contributes heavily. WDA is a `WebDriver`-compatible server that runs in the context of an iOS simulator or device and exposes the `XCUITest` API. Appium's `XCUITest` driver manages WDA as a subprocess opaque to the Appium user, proxies commands to/from WDA, and provides a host of additional functionality (like simulator management and other methods, for example).

UITest 能够完成的动作：

元素：单击，双击，拖拽，截图。

手势：捏合，旋转，滑动屏幕（上下左右）

根据文字查找元素。

```
var debugDescription: String
    Provides debugging information about the element.
```

App的UI结构不是特别稳定时，两者都需要进行用例更新和代码修复。

App构成内容复杂时（或者不太原生的UI用法），查找元素过程都容易失败。

iOS-XCUITest使用教程

前提：测试的App基于iOS 9.3 以及之后的系统版本，Xcode软件版本不低于7，才可以使用该技术，UI Testing 在易用性上比 KIF 这样的框架要有所进步，随着 UI Testing 的推出，Apple 也将原来的 UI Automation 一系列内容标记为弃用。这意味着 UI Testing 至少在今后一段时间内将会是 iOS 开发中的首选工具。但是我们也应该看到，基于 Accessibility 的测试方式有时候并不是很直接。在这

个限制下，我们只能得到 UI 的代理对象，而不是 UI 元素本身，这让我们无法得到关于 UI 元素更多的信息 (比如直接获取 UI 元素中的内容，或者与 ViewController 中的相关的值)，现在的 UI Testing 在很大程度上还停留在比较简易的阶段，适合保证App的主流程的顺畅运行、用于回归测试。在技术探索过程中，发现测试脚本在iOS 11系统表现很好，但是在iOS 9等系统上会出现找不到按钮的情况等问题。随着iOS系统版本的推进，相信XCUITest技术的兼容性、性能会不断得到改进。

iOS-UITest项目地址：https://git.ms.netease.com/netease-precious-metals-client/ios-client/tree/lipeng_UITest_4.11

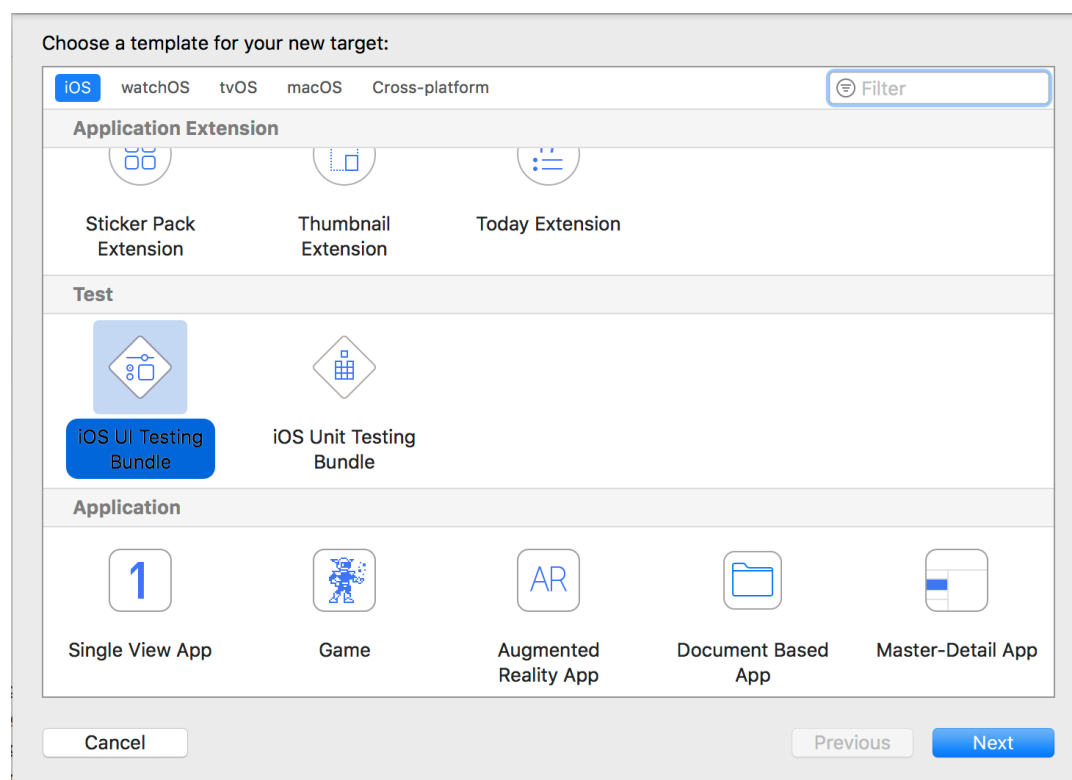
一、给已有的Xcode源码工程添加UITest功能：

1.在项目中新建UITest模块

(1) 如果是新建空白项目，可以勾选Include UITest相关选项。

(2) 如果是已有项目（比如pull下来的完整的iOS客户端代码），在Xcode中点击File->New->Target

如图选择：



点击Next，进入选项配置：

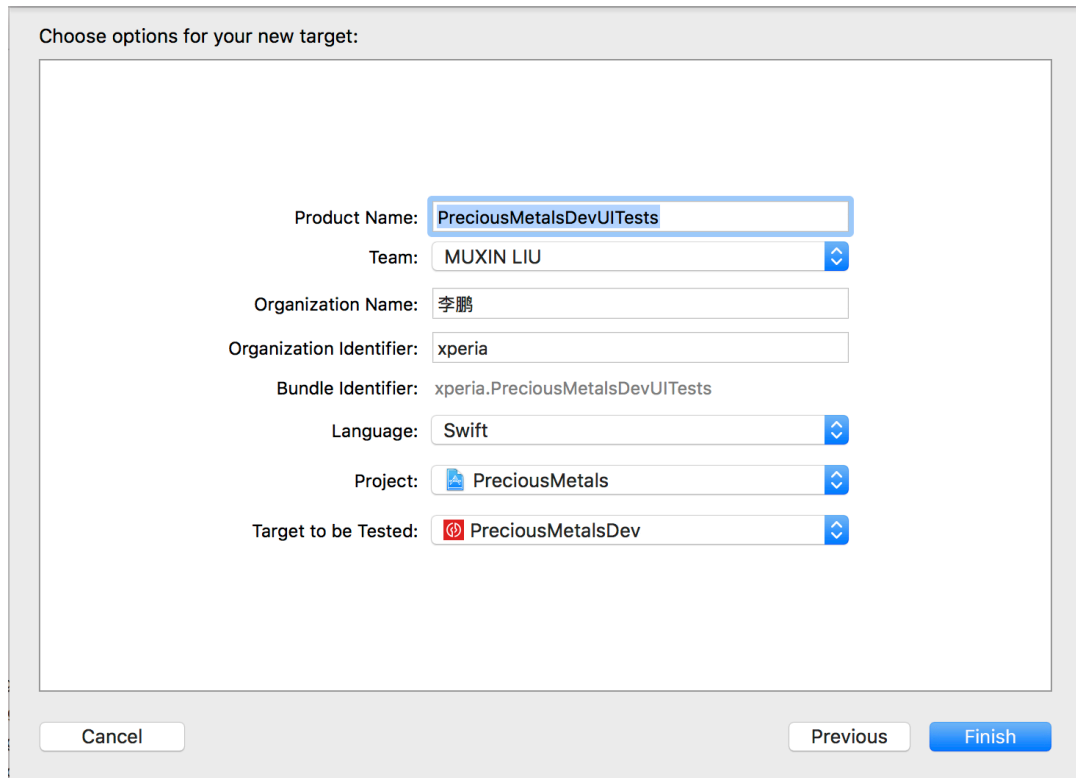
第一步：选择Project。

第二步：选择最底部的Target to be Tested，这里选择Dev结尾的包，因为该包

在模拟器上测试不需要签名。

此时Product Name会根据选项自动生成。切换自己需要的语言（Objective-C、Swift（推荐））

完成。



The image shows a screenshot of the 'Choose options for your new target' dialog box in Xcode. The dialog has a title bar and a main content area with several input fields. The fields are: 'Product Name' with the value 'PreciousMetalsDevUITests', 'Team' with the value 'MUXIN LIU', 'Organization Name' with the value '李鹏', 'Organization Identifier' with the value 'xperia', 'Bundle Identifier' with the value 'xperia.PreciousMetalsDevUITests', 'Language' with the value 'Swift', 'Project' with the value 'PreciousMetals', and 'Target to be Tested' with the value 'PreciousMetalsDev'. At the bottom of the dialog, there are three buttons: 'Cancel', 'Previous', and 'Finish'.

Choose options for your new target:

Product Name: PreciousMetalsDevUITests

Team: MUXIN LIU

Organization Name: 李鹏

Organization Identifier: xperia

Bundle Identifier: xperia.PreciousMetalsDevUITests

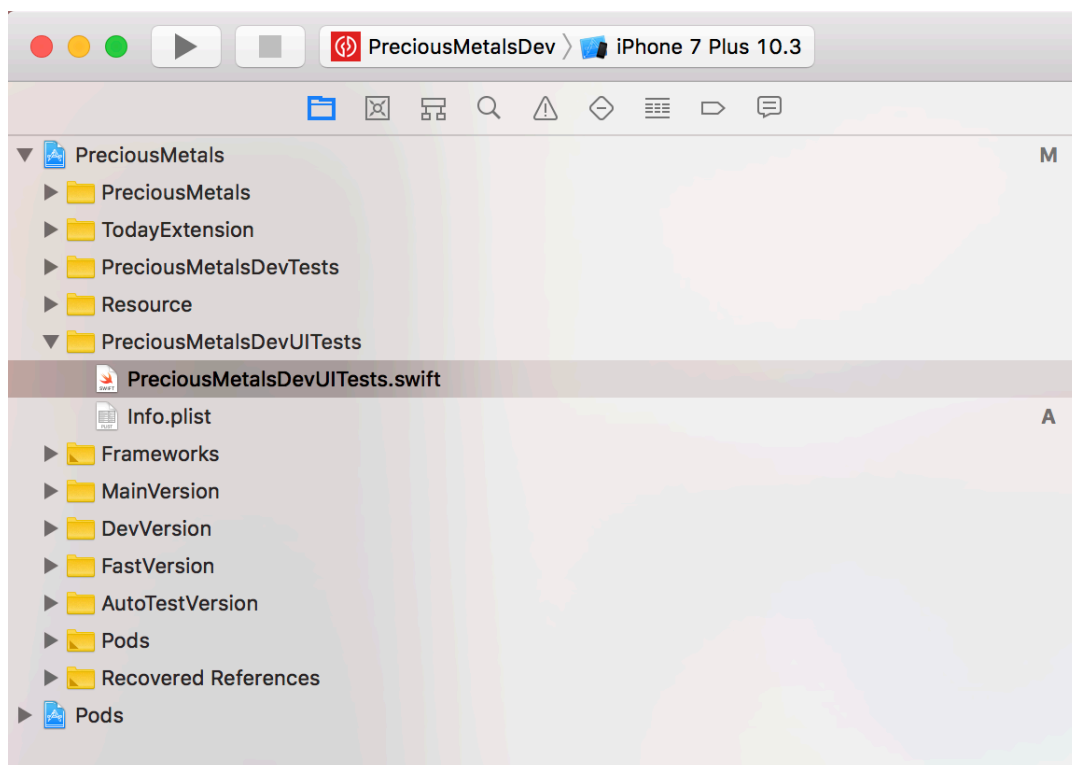
Language: Swift

Project: PreciousMetals

Target to be Tested: PreciousMetalsDev

Cancel Previous Finish

此时，项目中增加了新的以[Target+UITest]命名的文件夹，如图：



当前选中的与文件夹同名的（UITests结尾）文件就是我们可以编写测试代码的文件了。

2.测试代码文件介绍

在这里有初始化的几个方法：

setup：程序启动时会进入该方法，可以设置一些全局属性，比如测试如用例失败是继续执行还是停止，是否需要禁用App的过渡动画，启动的时候的屏幕方向等。

tearDown：程序结束时会进入该方法。

tearDownExample：测试方法，该方法系统生成，用于示例，我们可以自己写一些其他的方法，但是需要test打头，这样系统会识别出该方法为测试方法。

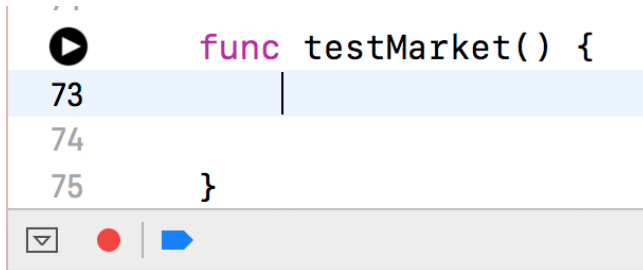
执行代码的方法：

当鼠标移动到tearDownExample函数时，会发现这一行前面出现一个播放形状(▶)的小图标，点击该图标程序会执行该测试方法。当遇到问题，或者没有达到预期值得时候，方法后面会出现红色，方法通过之后会出现绿色。

快捷键：control+option+command+U，当光标放在某个测试函数的代码中时，同样有效。

二、代码录制功能

在编写之前，了解下Xcode给提供的代码录制功能。



在上图中，我们找到控制台输出，哪里有一个红色的小圆点，先别着急点。先我们的光标移动到testMarket方法里之后，点击红色小圆点。进入录制阶段，等待程序的启动。当程序启动之后，点击程序的一个按钮，在光标相应的地方会生成一段代码。

三、常用基本操作

1.点击某个按钮，比如登录按钮

```
app.buttons["登录"].tap()
```

2.普通输入框输入文本

```
app.textFields["手机号"].tap() //要先聚焦文本框，才能继续输入
```

```
app.textFields["手机号"].typeText("13038865629")
```

注意密码输入框等输入文本会变成小圆点的地方使用app.secureTextFields["密码"]。

3.下拉刷新

```
app.swipeDown()
```

4.滚动查看屏幕下方的内容，上拉加载更多

```
app.swipeUp()
```

也可以左右滑动切换页面：app.swipeLeft/Right()

5.切换Tab

快讯

02月06日

17:06 【提示】LME铜库存减少2%，为两个月来最大跌幅。



17:06 欧洲央行管委魏德曼：欧元区个别地区或领域经济遭受的冲击仍可使欧元区经受“耐力测试”。

17:05 【提示】LME锌注销仓单增加19%，为去年11月以来最大。

17:05 【中国央行：2018年工作8项主要任务】
2018年，中国央行要保持货币政策稳健中性，切实防范化解金融风险，稳妥推进重要领域和关键环节金融改革，持续推动金融市场平稳健康发展，稳步推进人民币国际化，深度参与国际金融合作和全球经济金融治理，进一步推动外汇管理体制改
革，全面提高金融服务与管理水平。



```
app.tabBars.buttons["首页"].tap()
```

```
app.tabBars.buttons["行情"].tap()
```

6.让应用程序等待一会（比如数据载入的时候，或者延时操作）
`sleep(3)` //单位是秒

7.判断元素是否存在

```
if app.buttons["确定"].exists {  
    //点击确定按钮  
}
```

8.使用断言来验证App界面的状态

```
XCTAssert (app.buttons["个人中心"].count == 1) //count属性表示  
符合查询条件的结果个数
```

9.从导航栏返回上级页面

```
app.navigationBars["消息中心"].buttons["返回"].tap()
```



10.使用正则匹配，操作列表元素（进阶）

交易品及参考指数			
南交所	龙商华泰	参考指数	
[南]白银10g AG	33.75		+
[南]铜 CU	47.48		+
[南]钯 PD	217.98		+
[南]氧化钨 ND203	517.01		+
[南]螺纹钢湘 RBX01	45.16		+

```
let addButton = app.tables.cells.matching(NSPredicate(format: "label CONTAINS %@", "铜")).element(boundBy: 0)
addButton.tap()
//NSPredicate(format: "label CONTAINS %@", "铜") -- 筛选包含‘铜’字的文本
//element(boundBy: 0) -- 查询结果的第一个元素，该页面结果显然是唯一的，
//这种情况下可以使用firstMatch属性来代替，提升效率。
```


3.注意事项

(1) 模拟器点击textField和textView会出现问题，系统问题，暂未解决，不过在真机上未出现此问题。(等待验证)

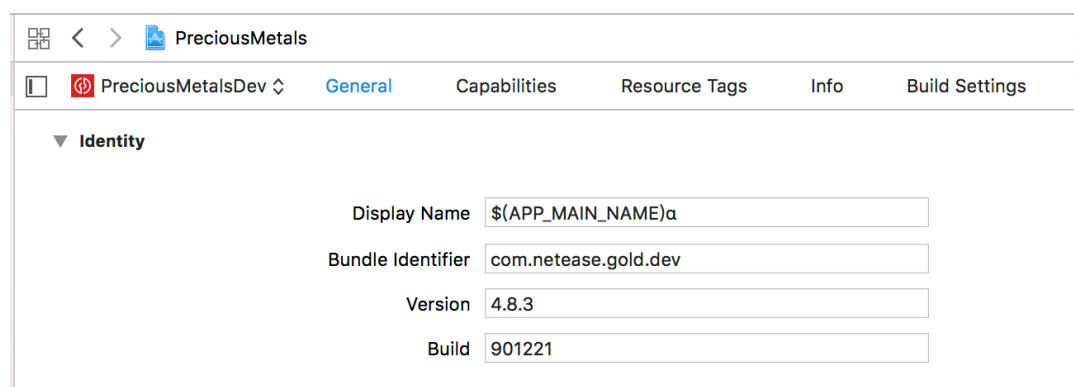
(2) 点击cell时需要先判断cell的个数，因为获取不到指针，所以只能通过数组元素获取。

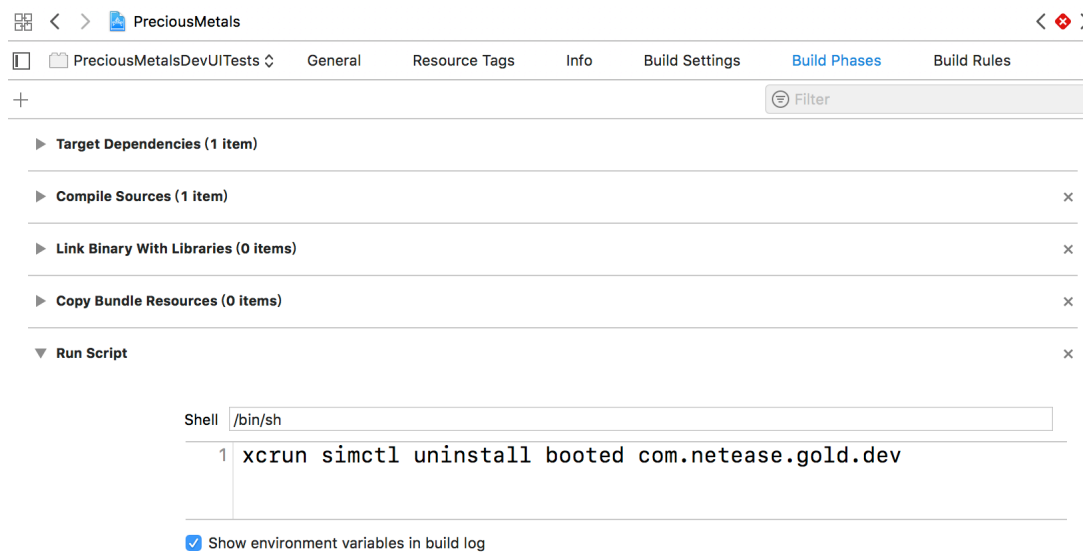
(3) 给输入框等输入值，应该先获取焦点，调用tap方法即可。

相关技巧

1.在测试之前执行卸载app，保证全新App来进入新用户引导界面。

点击“+”，选择Run Script，会出现一栏新的配置项，如图填入代码，最后一个参数是当前的Target（PreciousMetalsDev）的Bundle Identifier：





2.划过新用户引导页（示例代码）

```
let app = XCUIApplication()

//滑动新用户引导
let swipeView = app.scrollViews.element
swipeView.swipeLeft()
swipeView.swipeLeft()
app.buttons["开启易龙智投"].tap()

//进入我的Tab
app.tabBars.buttons["我"].tap()
```

3.尝试Jenkins自动化接入UITest工程

.gitlab-ci.yml文件，这个文件是Gitlab CI的配置文件，CI要做什么都可以在这个文件里面描述

```
stages:
```

```
- test
```

```
before_script:
```

```
- cd Example
```

```

- pod update

ui_test:
  stage: test
  script:
    # 这里先build一下, 防止log日志过多, 防止gitlab ci
    build log exceeded limit of 4194304 bytes.
    - xcodebuild -workspace
    IntegrationTesting.xcworkspace -scheme
    IntegrationTesting-Example -destination
    'platform=iOS Simulator,name=iPhone 7,OS=10.1' build
    >/dev/null 2>&1
    - xcodebuild -workspace
    IntegrationTesting.xcworkspace -scheme
    IntegrationTesting-Example -destination
    'platform=iOS Simulator,name=iPhone 7,OS=10.1' test
    | tee Log/`date +%Y%m%d_%H%M%S`.log | grep -A 5
    'error:'

```

4.易龙智投测试用例 Swift Version

1.获取App、环境变量

```

class PreciousMetalsDevUITests: XCTestCase {

    let app = XCUIApplication()

    override func setUp() {

```

2.一些测试用例

//测试我的Tab

```
func testMinetab() {
```

```
    //测试登录
```

```
    //testLogin()
```

```
    let tabBarsQuery = app.tabBars
```

```
    tabBarsQuery.buttons["行情"].tap()
```

```
    tabBarsQuery.buttons["交易"].tap()
```

```
    tabBarsQuery.buttons["直播"].tap()
```

```
    let button = tabBarsQuery.buttons["我"]
```

```
    button.tap()
```

```
    tabBarsQuery.buttons["首页"].tap()
```

```
    button.tap()
```

```
    let tablesQuery = app.tables
```

```
    //Section 1
```

```
    tablesQuery/*@START_MENU_TOKEN@*/.cells.staticTexts["市场与密码管理"]/*["cells.staticTexts["市场与密码管理"]", ".staticTexts["市场与密码管理\n"]", [[[-1,1],[-1,0]]], [1]]@END_MENU_TOKEN@*/.tap()
```

```
    app.navigationBar["市场与密码管理"].buttons["返回"].tap()
```

```
    tablesQuery/*@START_MENU_TOKEN@*/.cells.staticTexts["红包/免佣券"]/*["cells.staticTexts["红包\n免佣券"]", ".staticTexts["红包\n免佣券"]", [[[-1,1],[-1,0]]], [1]]@END_MENU_TOKEN@*/.tap()
```

```
    app.navigationBar["LDPMCouponsWebView"].buttons["返回"].tap()
```

```
    tablesQuery/*@START_MENU_TOKEN@*/.cells.staticTexts["我的活动"]/*["cells.staticTexts["我的活动"]", ".staticTexts["我的活动"]", [[[-1,1],[-1,0]]], [1]]@END_MENU_TOKEN@*/.tap()
```

```
    app.navigationBar["我的活动"].buttons["navigationbar back"].tap()
```

```
tablesQuery.cells.staticTexts["我的消息"].tap()
app.navigationBar["我的消息"].buttons["返回"].tap()

//Section 2
tablesQuery.cells.staticTexts["圈子"].tap()
app.navigationBar["圈子"].buttons["返回"].tap()
tablesQuery.cells.staticTexts["新人必读"].tap()
//terrible! ! ! !
sleep(2)
app.navigationBar["404 Not Found"].buttons["navigationbar
back"].tap()

tablesQuery.cells.staticTexts["帮助中心"].tap()
sleep(2)
app.navigationBar["帮助中心"].buttons["navigationbar back"].tap()

//swipe to bottom
app.swipeUp()

tablesQuery.cells.staticTexts["快捷登录设置"].tap()
app.navigationBar["快捷登录设置"].buttons["返回"].tap()
tablesQuery.cells.staticTexts["消息推送设置"].tap()
app.navigationBar["消息推送设置"].buttons["返回"].tap()
tablesQuery.cells.staticTexts["清除缓存"].tap()
app.buttons["清理易龙智投α缓存"].tap()
app.navigationBar["清理缓存"].buttons["返回"].tap()
tablesQuery.cells.staticTexts["关于"].tap()
app.navigationBar["关于"].buttons["返回"].tap()

assert(true, "OK TEST ")
}

//登录测试
```

```
func testLogin() {
    // Use recording to get started writing UI tests.
    // Use XCTAssert and related functions to verify your tests produce the
correct results.
    //滑动新用户引导
    let swipeView = app.scrollViews.element
    swipeView.swipeLeft()
    swipeView.swipeLeft()
    app.buttons["开启易龙智投"].tap()

    //进入我的Tab
    app.tabBars.buttons["我"].tap()

    //URS登录
    app.buttons["其他方式登录"].tap()
    let ursAcc = "ledelipeng5"
    let ursPwd = "050329@Boa"
    let mailTxt = app.textFields["网易邮箱 / 手机号"]
    let mailPwdTxt = app.textFields["密码"]

    mailTxt.tap()
    mailTxt.typeText(ursAcc)

    mailPwdTxt.typeText(ursPwd)
    let loginBtn = app.buttons["登录"]
    if self.canOperateElement(element:loginBtn) {
        loginBtn.tap()
    }

    sleep(1)
    //assert(tablesQuery.cells.staticTexts["圈子"], "登录完成")

    assert(true, "登录测试 PASS")
}
```

```
}
```

```
func canOperateElement(element:XCUIElement?) -> Bool {  
    if element != nil {  
        if element!.exists {  
            return true  
        }  
    }  
    return false  
}
```

5.UITest 可以帮我们模拟手动无法模拟的操作，如果用的比较好，整个App的质量会有很大的提高。

疯狂点击UI，测试稳定性。

```
for (NSInteger i=0; i < 50; i++)  
{  
    XCUIElement *button = app.buttons[@"Button"]; [button  
tap];  
}
```

常用操作

//下拉刷新(按住当前元素，向下拉动offset的距离来刷新)

```
func pullDownToRefresh(offset: Int) {  
    self.coordinate(withNormalizedOffset: CGVector(dx: 2, dy:  
2)).press(forDuration: 0.1, thenDragTo: self.coordinate(withNormalizedOffset:  
CGVector(dx: 2, dy: 2 + offset)))  
}
```

//强制点击UI元素

```

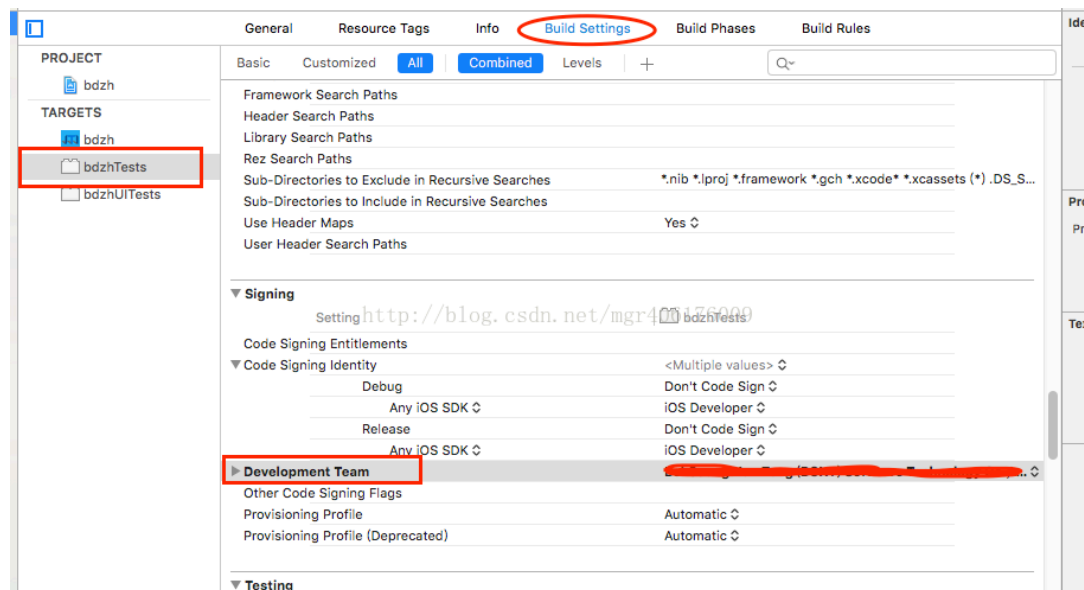
func forceTapElement() {
    if self.isHittable {
        self.tap()
    }
    else {
        let coordinate: XCUICoordinate =
self.coordinate(withNormalizedOffset: CGVector(dx:0.0, dy:0.0))
        coordinate.tap()
    }
}
}

```

真机调试

A: 错误 code signing is required for product type 'Unit Test Bundle' in SDK 'iOS 10.1'

针对当前Target选择Automatic manage signing



调研内容：

1.基本功能：

2.App是混合体，Web&Native

网易游戏的ATX?

<https://github.com/NetEaseGame/ATX>

Layer不可操作。那么要学会查看当前元素的类型，lldb的po命令，或者使用3D界面的UI Inspector。

禁用UIView动画来提高测试速度，主要是解决wait for app to idle:



认识核心的三个类

- XCUIApplication
- XCUIElement
- XCUIElementQuery

1.1 XCUIApplication类是Application的代理，就像我们项目工程中的AppDelegate，这个对象用来启动或者是终止UI测试程序，还可以在启动的时候设置一些启动参数，在获取程序中的UI元素的时候，就是通过这个类的实例。这个类继承自XCUIElement类

1.2 XCUIElement类是XCTest.framework对应用中的所有UI 控件的抽象，在UI测试中，没有UIKit中的UI类型，只是用这个类的实例表示所有的UI 控件，以及相应的交互方法，例如：执行手势（tap, press, swipe），滑动控件交互，拾取器交互，这个类采取了XCUIElementAttributes协议（描述UI元素的属性：Identity, Value, Interaction State, Size），XCUIElementTypeQueryProvider协议(为指定类型的子代元素提供ready-made查询，子代元素查询包含button，具体实现是:@property(readonly, copy) XCUIElementQuery

*buttons; 等一系列对UIKit中元素的映射)

1.3 XCUIElementQuery类是定位UI 元素的查询，这个类使用类似key-value的机制得到XCUIElement的实例，使用Type(XCUIElementType枚举), Predicate, Identifier创建query，使用elementAtIndex:, elementMatchingPredicate, elementMatchingType: identifier:方法访问匹配到的UI元素，此类采用XCUIElementTypeQueryProvider协议

辅助工具：Xcode---Open Developer Tools----Accessibility Inspector
查看UI元素的类型，属性值，层级结构，以及可以进行的操作

元素找不到，可能需要开发配合下：

(开启需要元素的Accessibility属性，设置AccessibilityValue，如图最后两行代码是针对可转出金额的文本输入框进行的设置。这并不会影响程序功能和逻辑。)



```
80  NSAssert(self.partnerId.length > 0, @"%s partnerId不能为空", NSStringFromClass(self.class));
81  if (self.partnerId.length <= 0) {
82      self.partnerId = self.exchangeAccountCarrier.defaultOpenedExchangeID;
83  }
84
85  self.enableBackGroundTapToResignFirstResponder = YES;
86  self.inputTranslator = [LDInputViewTranslator new];
87  self.inputTranslator.inputViewContainer = self.view;
88  [self.inputTranslator addTextFiled:self.transferMoneyTextField];
89  self.inputTranslator.translateOffset = -50;
90  self.transferMoneyTextField.delegate = self;
91
92  self.transferMoneyTextField.isAccessibilityElement = YES;
93  [self.transferMoneyTextField setAccessibilityValue:@"可转出金额"];
```

注意：密码TextField，类型是**secureTextFiled**，因为常常是输入的内容会变成小圆点，防止偷窥。（被坑过，在textField下面怎么也找不到元素，无法点击。。。)