Unit Test 簡單經驗分享

iOS @ Taipei 2017-03-28

1.為什麼要寫 Unit Test

1. 因為我們寫Method的時候，不一定可以完整的考慮所有狀況
2. 因為需求有可能增加或變更，增加或變更的時候，你不可能一個一個全部考慮完，並且確認其正確性及穩定性
3. God will make mistakes , Because we are human , So , We will make mistakes

2.Unit Test 能夠解決所有的問題嗎

1. NO , 因為這是 Unit Test , 就是單元測試
   只能跟你說, 每一個經過測試的 Method 在絕大多數的狀況下是正常運作的
   (取決於, 你 tesing 的範圍涵蓋)
2. NO , 因為還有會其他會影響執行時的一堆問題(網路, 電力, 記憶體, 使用者操作非你想像)

3.撰寫 Unit Test 的時機點

1. Start Coding Before

    Write Unit Test -> Write Code -> Tesing -> Code refactoring

這個好處是，你會根據明確的需求只開發出完成需求的功能，不會有多餘的部分
(只需要加法的Method，你不會因為不知道明確的需求，寫成包含加減乘除的Method)

3.撰寫 Unit Test 的時機點

2. 有一堆已經寫好的Code

Write Code -> Write Unit Test -> Tesing -> Code refactoring

你已經有一堆已經寫好但是沒有任何TestCase的Method

4.Unit Test 的撰寫原則

1.一個 test case 只測一個 Method

你應該為計算機上面的每個計算功能都寫一個 test case
而不是為整個計算機寫一個 test case

2.一個 Method 要測好幾個部分

臨界值 跟 超過臨界值 跟 臨界邊緣值 跟 正常值 會多於七個

4.Unit Test 的撰寫原則

func findYourMate(hope age:Int)->String

我們有一個幫你找到伴侶的 Method
我們不希望違法，所以我們讓你找的年齡不會小於 18
以人類的壽命來說，我們讓你找的年齡不會大於 80
這時候我們要測得就會是 17 , 18 , 19 , 79 , 80 , 81 , 21, 31, 41....
是不是符合你想要得到的狀況

5.Unit Test 的應用型態

TDD (Test-Driven Development) 測試驅動開發

https://goo.gl/4pcktq

TDD三定律:

(1)沒有測試之前不要寫任何功能代碼

(2)只編寫恰好能夠體現一個失敗情況的測試代碼

(3)只編寫恰好能通過測試的功能代碼

5.Unit Test 的應用型態

BDD (Behavior-Driven Development) 行為驅動開發

https://goo.gl/QAmbdI

6.Unit Test AAA原則

1.Arrange
    初始化目標物件
    初始化方法參數
    建立模擬物件行為
    設定環境變數,期望結果
2.Act
    實際呼叫目標測試物件的方法
3.Assert
    驗證目標物件是否如同預期運作

7.感言

看了上面那麼多, 我知道要寫, 我也想寫
但是, 我看到我自己的 Code 後, 我放棄了...
因為, 不論您是看書自學, 不論是花錢上課學的
從來都是教您怎樣寫, 沒有教您要怎樣為了可以被 test 寫
所以, 寫不出來, 會沮喪挫折是正常的,
~~跟過去的 Code 說 ByeBye~~

8.參考資料

KKBOX iOS/Mac OS X 基本開發教材 - 單元測試
https://zonble.gitbooks.io/kkbox-ios-dev/unit_test/index.html

30天快速上手TDD
https://goo.gl/U1XiAF

練習寫可以被 Test 的 Code

實作時間