

責任鏈模式 / 2019.12.29

DESIGN PATTERN 讀書會 / BOB CHANG

CHAIN-OF-RESPONSIBILITY PATTERN

自我介紹

- ▶ 張家銘 Bob Chang
- ▶ 國立宜蘭大學電子系學士畢業
- ▶ iOS app 開發自學者
- ▶ 2015 July 開始學習 `Swift`
- ▶ twitter @bob910078
- ▶ bob910078@gmail.com





<https://cdn2.ettoday.net/images/1408/1408485.jpg>

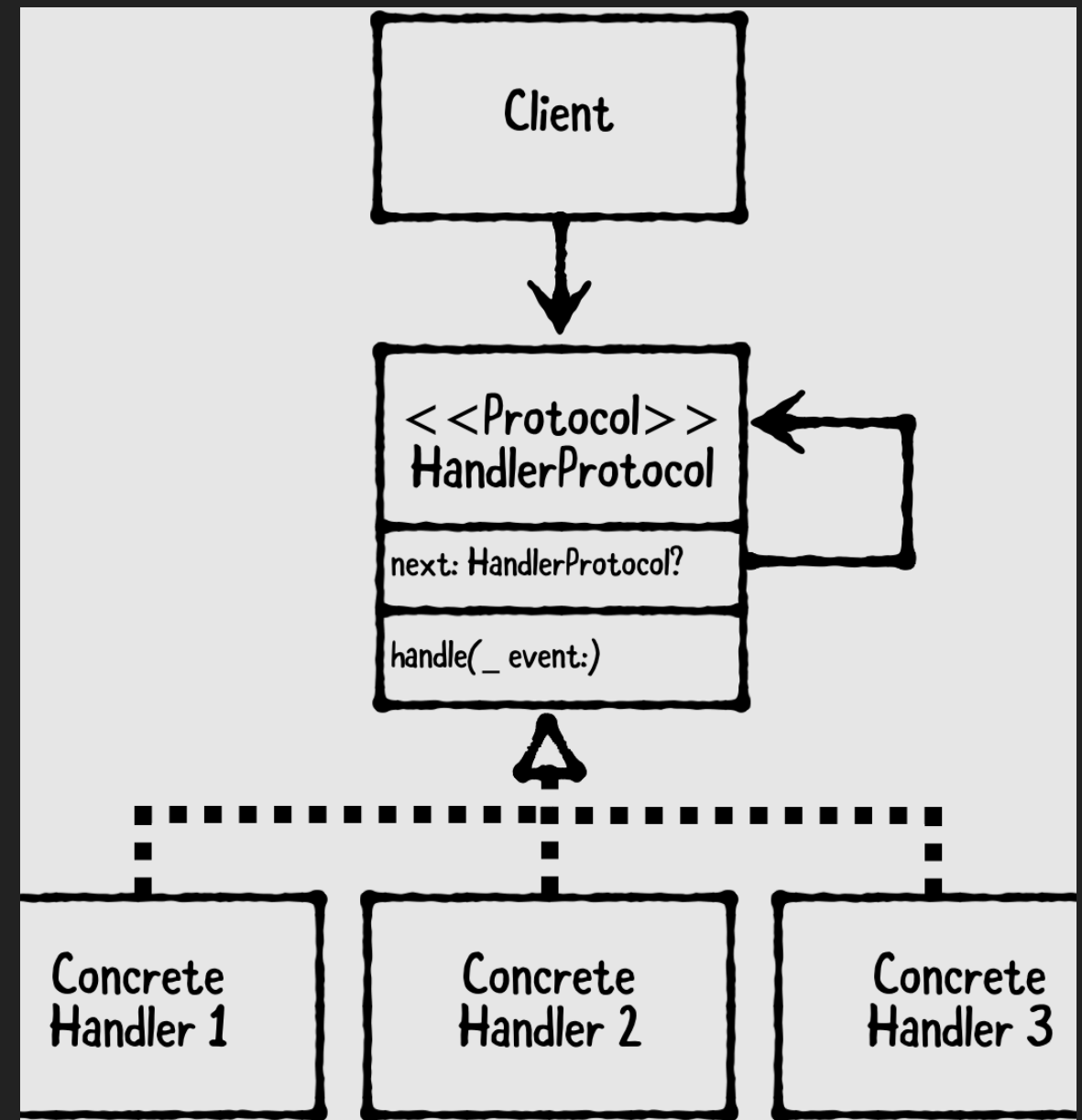
自己不做事，交給 下一位做事的 PATTERN

也像是在教室裡傳紙條給某人



吃到飽

```
Protocol HandlerProtocol {  
    var next: HandlerProtocol?  
    func handle(_ event: Int)  
}
```



WHEN USE IT? WHAT TO NOTICE? HOW TO USE IT?

▶ 什麼時候可以用這個 pattern ?

- ✿ 需要有一系列的物件去處理相似的事件或是資料
- ✿ (例如自動販賣機的硬幣偵測、觸摸螢幕的事件處理)

▶ 使用時可能需要注意什麼 ?

- ✿ 我們這一系列的物件都無法處理輸入事件的狀況
- ✿ (都沒有人要處理該怎麼辦)

▶ 如何使用這個 pattern ?

- ✿ 責任鏈中的每位響應者都必須要接受某一事件，
- ✿ 並自行決定「是否處理」或是「傳遞給下一位響應者」

典型應用：UIRESPONDER

The screenshot shows a web browser window with the title "UIResponder". The breadcrumb navigation shows "UIKit > Touches, Presses, and Gestures > UIResponder". The main content area is titled "Class UIResponder" with the description "An abstract interface for responding to and handling events." Below this is a "Declaration" section with a code block containing the line `class UIResponder : NSObject`. An "Overview" section follows, explaining that responder objects are instances of UIResponder and form the event-handling backbone of a UIKit app. On the right side, there are sections for "Language" (Swift and Objective-C), "SDKs" (iOS 2.0+, tvOS 9.0+, Mac Catalyst 13.0+), "Framework" (UIKit), and "On This Page" (Declaration).

UIKit > Touches, Presses, and Gestures > UIResponder

Class

UIResponder

An abstract interface for responding to and handling events.

Language

Swift | [Objective-C](#)

SDKs

iOS 2.0+

tvOS 9.0+

Mac Catalyst 13.0+

Framework

UIKit

On This Page

[Declaration](#)

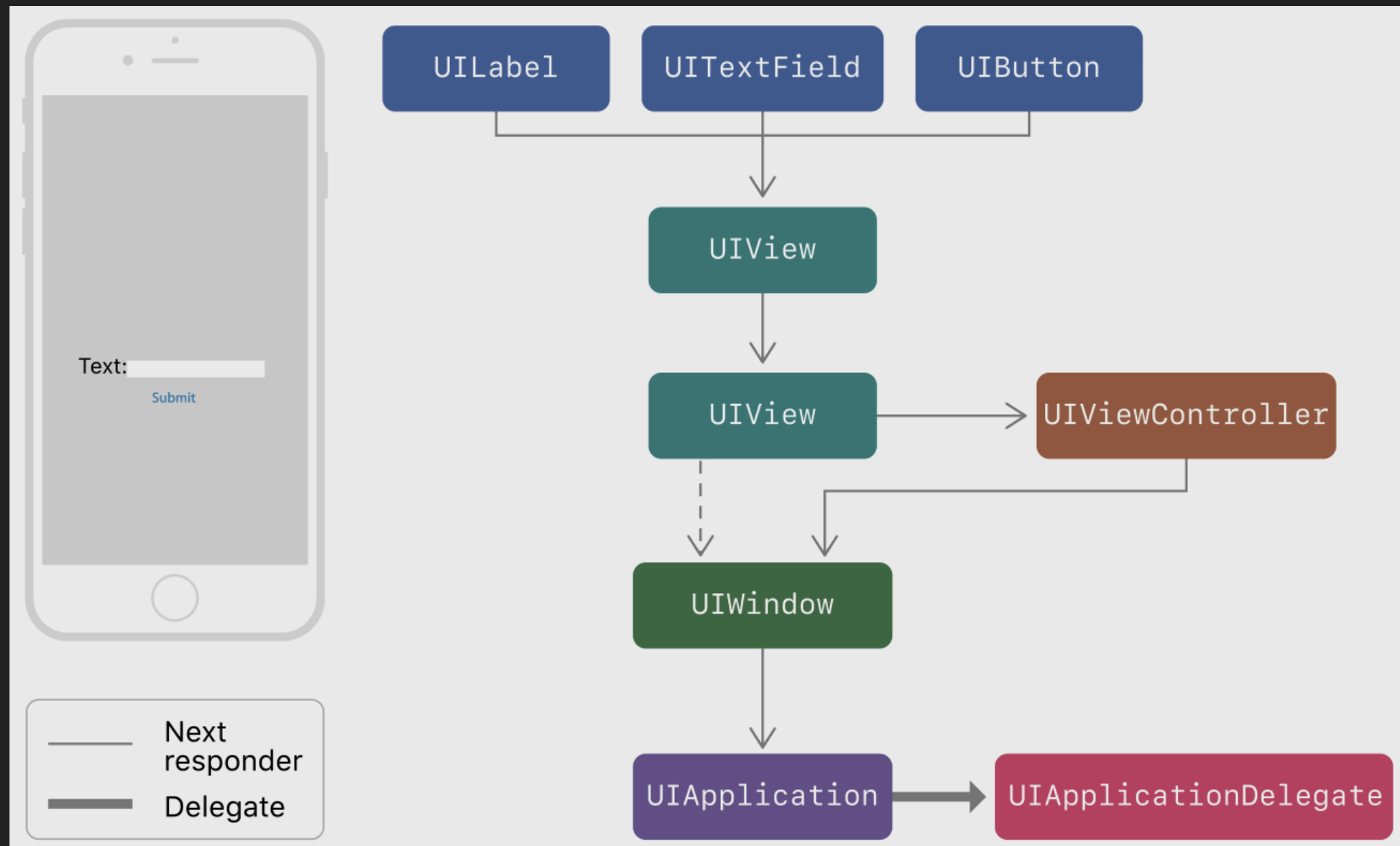
Declaration

```
class UIResponder : NSObject
```

Overview

Responder objects—that is, instances of [UIResponder](#)—constitute the event-handling backbone of a UIKit app. Many key objects are also responders, including the [UIApplication](#) object, [UIViewController](#) objects, and all [UIView](#) objects (which includes [UIWindow](#)). As events occur, UIKit dispatches them to your app's responder objects for handling.

RESPONDER CHAINS IN AN APP



典型應用：UIRESPONDER (續)

- ▶ UIResponder 雖然是一個類別，但是其精神不出 chain-of-responder pattern
- ▶ 重要的兩個概念：optional 繼任者 和 事務處理區塊
 - ▶ `var next: UIResponder?`
 - ▶ `func touch began(Set<UITouch>, with: UIEvent?)`

OVERRIDE UIRESPONDER.TOUCHESBEGAN(_ :WITH:)

- ▶ UIView
- ▶ UIViewController
- ▶ UIWindow
- ▶ UIApplication
- ▶ AppDelegate

```
extension UIView {  
    override func touchesBegan(  
        _ touches: Set<UITouch>,  
        with event: UIEvent?) {  
        print("UIView")  
        next?.touchesBegan(touches, with: event)  
    }  
}
```

結束