**Key-Value Observing (KVO)**

Điều khiến cho Objective-C (và Obj-C bridged Swift) đặc biệt đó là tất cả các variable instance mà ta add vào class đều trở thành observable thông qua KVO.

Swift kế thừa KVO từ Objective-C

**KVO trong Objective-C:**

@interface Person: NSObject

@property (nonatomic, strong) NSString \*name;

@property (nonatomic, assign) NSInteger age;

@end

Bắt đầu observe

@implementation SomeOtherClass

- (void)observeChanges:(Person \*)person {

[person addObserver:self

forKeyPath:@"age"

options:NSKeyValueObservingOptionNew

context:nil];

}

- (void)observeValueForKeyPath:(NSString \*)keyPath

ofObject:(id)object

change:(NSDictionary<NSKeyValueChangeKey,id> \*)change

context:(void \*)context {

if ([keyPath isEqualToString:@"age"]) {

NSNumber \*ageNumber = change[NSKeyValueChangeNewKey];

NSInteger age = [ageNumber integerValue];

NSLog(@"New age is: %@", age);

}

}

@end

Bây giờ mỗi khi age thay đổi giá trị trong Person thì sẽ print “New age is:…”

Có 2 menthod liên quan đến KVO ở trên:

1. **addObserve(:forKeyPath:ofObject:change:context:)** menthod này attach observer đến object.
2. **observeValueForKeyPath(:ofObject:change:context)** method mà ta cần override trong observer class, nó được khọi khi có observation notifications.
3. Bên cạnh đó, **removeObserver:forKeyPath:context:** cho phép ta ngừng việc observe lại khi observer gọi làm **dealloc** menthod

Các parameters được sử dụng trong KVO menthod:

- (void)addObserver:(NSObject \*)observer

forKeyPath:(NSString \*)keyPath

options:(NSKeyValueObservingOptions)options

context:(nullable void \*)context;

* **Observer**: là object sẽ nhận change notifications.
* **keyPath**: là string paramether, là tên của property ta muốn observe. Trong trường hợp cấu trúc object phức tạp ta có thể sử dụng “person.father.age”
* **option**: là 1 enum cho phép custom thông tin, và thời điểm muốn bắn notification.
* **NSKeyValueObservingObtionNew** và **NSKeyValueObservingOptionOld** kiểm soát dữ liệu mới và cũ
* **NSKeyValueObservingOptionInitial**: kích hoạt notification ngay sau khi subcription
* **NSKeyValueObservingOptionPrior**: nhận diện sự thay đổi của collection, vd: insert, delete trong NSArray
* Context: is a reference to object of an arbitrary class, which can be helpful for identifying the subscription in certain complex use cases, such as when working with CoreData. In most other cases you simply provide nil here.

Menthod xử lý update notifications

- (void)observeValueForKeyPath:(NSString \*)keyPath

ofObject:(id)object

change:(NSDictionary<NSKeyValueChangeKey,id> \*)change

context:(void \*)context

* keyPath: keyPath của Observe Value thay đổi dùng để xác định property thay đổi
* object: object chứa properties vừa thay đổi
* change: dictionary với thông tin về changed value, dựa trên NSKeyValueObservingOptions nó có thể gồm các giá trị:
* NSKeyValueChangeNewKey: giá trị mới nhất của properties
* NSKeyValueChangeOldKey: giá trị cũ của properties
* NSKeyValueChangeIndexesKey và NSKeyValueChangeKindKey: thông tin về sự thay đổi trong collection.
* Context: is the reference provided upon subscription. Again, used for proper observation identification and in most cases can be ignored.

**KVO trong Swift**

Mặc định KVO bị disable trong swift. Ta phải tạo Objective-C class để enable KVO bằng cách set base class với NSObject và add @objc dynamic đến các variables:

class Person: NSObject {

@objc dynamic var age: Int

@objc dynamic var name: String

}

class PersonObserver: NSObject {

func observe(person: Person) {

person.addObserver(self, forKeyPath: "age",

options: .new, context: nil)

}

override func observeValue(forKeyPath keyPath: String?,

of object: Any?,

change: [NSKeyValueChangeKey : Any]?,

context: UnsafeMutableRawPointer?) {

if keyPath == "age",

let age = change?[.newKey] {

print("New age is: \(age)")

}

}

}

1 cách khác mới hơn để sử dụng KVO với ưu điểm là more flexible, safe, swift-friendly:

class PersonObserver {

var kvoToken: NSKeyValueObservation?

func observe(person: Person) {

kvoToken = person.observe(\.age, options: .new) { (person, change) in

guard let age = change.new else { return }

print("New age is: \(age)")

}

}

deinit {

kvoToken?.invalidate()

}

}

NSKeyValueObservation đang tồn tại bug: (khi dùng để update thanh tiến trình công việc)

NSKeyValueObservation error: <https://drewag.me/posts/2019/09/02/careful-with-fancy-kvo-callback>

**Key value coding (KVC)**

* Bản thân **NSObject** cũng tuân theo **KVC**. Cho phép set, get giá trị của các properties thông qua **key** và **keyPatch** bởi method **setValue** và **getValue**.
* Mục đích chính là để **KVO** xác định khi giá trị của 1 **properties** nào đó thay đổi
* Khi gọi **setValue** nó sẽ set giá trị mới cho property và gọi 2 methods **willChangeValue** và **didChangeValue** (ta có thể tự gọi 2 methods này)
* Property được khai báo **dynamic** khi được set giá trị sẽ tự động gọi các hàm **willChangeValue** và **didChangeValue**. 1 property không khai báo **dynamic** mà muốn **observe** ta có thể tự gọi hàm **willChangeValue** và **didChangeValue** khi thay đổi giá trị của property hoặc dùng hàm **setValue** để set giá trị mới cho property
* Khi khai báo dynamic thì nó sẽ được gán message dispatch

**Ưu điểm:**

* Multable observer
* Class bị observe không cần biết đến class đi observe
* Có thể biết được giá trị trước đó của properties được observe

**Nhược điểm:**

* Tất cả các noti bắn về đều qua 1 method: **observeValue(forKeyPath:, of:, change:, context:)**
* Nếu key và keyPath sử dụng string thì sẽ dễ crash khi object bị observe thay đổi tên property (nên sử dụng #keypath khi dùng swift)
* Phải unsubcribe tại deinit
* So với gọi 1 function thì thời gian nó thực hiện noti hơn 200 lần (đối với swift và 30 lần đối với Objective C)

Link:

* <https://medium.com/hackernoon/kvo-kvc-in-swift-12f77300c387>
* https://nalexn.github.io/kvo-guide-for-key-value-observing/