



作者 伯恩的遗产 (/users/5acac5529187) 2015.04.06 19:07\*

写了35249字，被707人关注，获得了648个喜欢

(/users/5acac5529187)

+ | 添加关注 (/sign\_in)

# 当你不爱我的时候，请告诉我(KVC/KVO/NSNotification)

字数1593 阅读1517 评论2 喜欢22

## KVC KVO NSNotification

### 一、KVC

- 基本概念

1. 它是一种可以直接通过字符串类型的属性名(key)来访问某个类属性的机制。而不是通过调用 Setter、Getter方法访问。
2. 所有NSObject对象都可以使用KVC。
3. KVC既支持带有对象值的属性，也支持基本数据类型和结构。基本数据类型会被自动封装和解装。

- 基本使用

1. 通过 `setValue: forKey:` 设置对象的值。
2. 通过 `valueForKey:` 获得对象的值。
3. 通过 `setValue: forKeyPath:` 设置指定路径的对象值。
4. 通过 `valueForKeyPath:` 获得指定路径的对象值。



示例代码如下：

```
//Person.h

@interface Person : NSObject

@property (nonatomic, copy) NSString *name;
@property (nonatomic, assign) NSInteger age;
@property (nonatomic, strong) Person *girlFriend;
@property (nonatomic, copy) NSArray *books;

@end
```

```
- (void)viewDidLoad {
    [super viewDidLoad];

    Person *person = [[Person alloc] init];
    NSLog(@"%@", person.name, person.age);

    [person setValue:@"Bourne" forKey:@"name"];
    [person setValue:@21 forKey:@"age"];

    NSLog(@"%@", [person valueForKey:@"name"], person.age);

    Person *girl = [[Person alloc] init];
    [person setValue:girl forKey:@"girlFriend"];

    NSLog(@"%@", [person valueForKeyPath:@"girlFriend.name"], person.girlFriend.age);

    [person setValue:@"willYou?" forKeyPath:@"girlFriend.name"];
    [person setValue:@20 forKeyPath:@"girlFriend.age"];

    NSLog(@"%@", person.girlFriend.name, person.girlFriend.age);
}
```

## 打印输出

```
2015-04-06 16:49:19.192 KVC-KV0-test[5685:7429728] 1: (null) - 0
2015-04-06 16:49:19.193 KVC-KV0-test[5685:7429728] 2: Bourne - 21
2015-04-06 16:49:19.194 KVC-KV0-test[5685:7429728] 3: (null) - 0
2015-04-06 16:49:19.194 KVC-KV0-test[5685:7429728] 4: willYou? - 20
```

## • 一对多关系

1. 如果向NSArray等集合类型数据所包含的对象发送请求消息，它会查询数组中每个对象来查找这个键值，并返回一个打包后的数组。
2. 如果想要向数组等集合类型的数据所包含的对象发送赋值消息，它也会查询数组中每个对象来查找这个键值，并赋同一个值。**注意：不要企图发送一个数组让她为找到的对象按顺序赋值，这样会报错！**

示例代码如下：

```
//Book.h

@interface Book : NS0bject

@property (nonatomic, assign) NSInteger price;

@end


- (void)setAndGetBooks {
    Person *person = [[Person alloc] init];

    Book *book1 = [[Book alloc] init];
    Book *book2 = [[Book alloc] init];
    Book *book3 = [[Book alloc] init];

    NSArray *myBooks = @[book1, book2, book3];

    [person setValue:myBooks forKey:@"books"];

    NSLog(@"%@", [person valueForKeyPath:@"books.price"]);

    [person setValue:@1 forKeyPath:@"books.price"];

    NSLog(@"%@", [person valueForKeyPath:@"books.price"]);
}
```

打印输出

```
2015-04-06 17:12:48.256 KVC-KV0-test[5886:7519923] 1: (
    0,
    0,
    0
)
2015-04-06 17:12:48.257 KVC-KV0-test[5886:7519923] 2: (
    1,
    1,
    1
)
```

## 二、KVO

- 基本概念

1. KVO: Key-Value Observing, 它提供一种机制，当指定的对象的属性被修改后，则对象就会接收到通知。就是每次指定的被观察的对象的属性被修改后，KVO就会自动通知相应的观察者。属于一种非正式的 Protocol。
2. 可以观察任意属性，包括基本数据类型，对一或对多关系。对多关系的观察者将会被告知发生变化的类型，也即使任意发生变化的对象。

- 使用方法

1. 注册观察者，指定被观察者的属性。
2. 实现回调方法，接收变更通知。
3. 移除观察者身份。

- 注册观察者

为了正确接收属性的变更通知，观察者必须首先发送一个 `addObserver:forKeyPath:options:context:` 消息给被观察者。用以传送需要观察的对象和需要观察的属性的关键路径，选型参数指定了发生变更时提供给观察者的信息，使用 `NSKeyValueObservingOptionNew` 可以在回调方法中获得旧值，使用 `NSKeyValueObservingOptionOld` 可以在回调方法中获得新值。

### 示例

```
@"books.price" options:NSKeyValueObservingOptionNew|NSKeyValueObservingOptionOld context:NULL];
```

- 回调方法

当监听属性发生变化时，坚持着将会收到一条 `observeValueForKeyPath: ofObject: change: context:` 消息。触发观察的对象、键路径、包含变化细节的字典都会传给观察者。

```
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object change:(NSDictionary *)  
if([keyPath isEqualToString:@"books.price"]) {  
    NSLog("通知我啦！");  
}  
}
```

- 移除观察者

你可以发送一条包含观察者，键路径的方法 `` 给被观察的对象，来移除这个观察。

**注意：注册了观察者就需要手动移除观察者！**

```
[person removeObserver:self forKeyPath:@"books.price"];
```

## 三、NSNotification

- 概述

与KVO不同的是，KVO在属性上通过 K-V 发生改变时，自动调用 `observeValueForKeyPath: ofObject: change: context:` 方法，而 `NSNotification` 在需要的时候自行发送通知才调用，且方法自定义。

- 使用方法

1. `NSNotificationCenter` 注册观察者时指定事件(以字符串命名)，及该事件触发时该执行的 Selector 或 Block
2. `NSNotificationCenter` 在某个时机激发事件(以字符串命名)

**注意：编译器根据这个事件的名字找到之前注册过这个名字的观察者，所以这个名字一般设置成静态属性，保证两次使用是同一个字符串！**

### 3. 观察者在收到感兴趣的事件时，执行相应的 Selector 或 Block

- **注册观察者**

使用默认的通知中心，指定观察者、需要接收的通知名称、接收通知时执行的方法，最后一个参数是表示会对哪个发送者对象发出的事件作出响应，`nil` 时表示接受所有发送者的事件。

```
[ [NSNotificationCenter defaultCenter] addObserver:self selector:@selector(action:) name:@"NOTIF
```

- **激发观察者**

需要指定通知的名称，编译器根据这个事件的名字找到之前注册过这个名字的观察者，还可以指定传给观察者的对象。

```
NSObject *test = [[NSObject alloc] init];
[[NSNotificationCenter defaultCenter] postNotificationName:@"NOTIFICATION_NAME" object:test];

//或者以下的方式：

NSNotification *notification = [NSNotification notificationWithName:@"NOTIFICATION_NAME" object:test];
[[NSNotificationCenter defaultCenter] postNotification:notification];
```

- **执行事件**

观察者接受到通知后就会执行注册时指定的方法，并获得被观察者传回的对象。

```
- (void)action:(NSNotification *)notification {
    NSLog("通知我啦！");
}
```

## **声明：**

1. 以上内容属于本人整理的笔记，如有错误的地方希望能告诉我，大家共同进步。
2. 以上内容有些段落或语句可能是本人从其他地方Copy而来，如有侵权，请及时告诉我。