


个人资料



Crayondeng

访问：395647次

积分：5485

等级：BLOG 5

排名：第2029名

原创：130篇 转载：36篇

译文：6篇 评论：233条

文章搜索

文章分类

Learning iOS (41)

Sina Weibo Demo iOS (8)

iOS 网络编程 (8)

Cocos2d-iphone (16)

课程作业 (2)

Cocos2d-x (37)

Learning C++ (12)

Programming in Lua (4)

Cocos2d-x 设计模式 (9)

Cocos2d-x 游戏实践 (2)

Cocos2d-x 内存管理与优化 (4)

游戏算法 (1)

Algorithm&DataStructure (9)

Learning OpenGL (1)

iOS GameKit (0)

Cocos2d-x 3.0 (4)

objc.io 文章 (5)

Algorithm& (1)

DataStructure (1)

ReactiveCocoa (2)

Algorithm&DataStructure (5)

文章存档

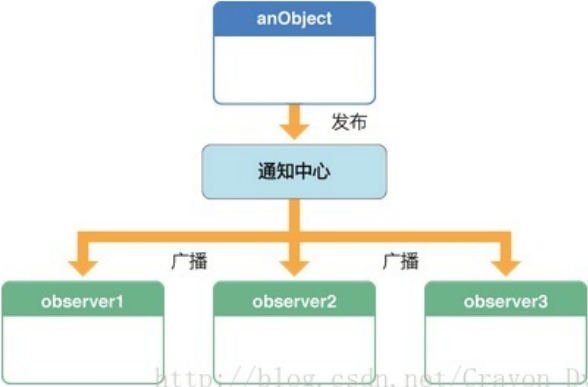
CSDN Android客户端发布 扒一扒最NB的开发项目 CSDN博主维权信息收集 最流行的语言都在这，想学就学！

iOS 通知中心 NSNotificationCenter & NSNotification

分类： Learning iOS 2013-07-18 19:37 10894人阅读 评论(3) 收藏 举报

通知中心 NSNotificationCenter NSNotification

通知中心是 Foundation 框架的一个子系统，它向应用程序中注册为某个事件观察者的所有对象广播消息（即通知）。（从编程角度而言，它是 NSNotificationCenter 类的实例）。该事件可以是发生在应用程序中的任何事情，例如进入后台状态，或者用户开始在文本栏中键入。通知是告诉观察者，事件已经发生或即将发生，因此让观察者有机会以合适的方式响应。通过通知中心来传播通知，是增加应用程序对象间合作和内聚力的一种途径。



```
graph TD; anObject[anObject] -- 发布 --> NotificationCenter[通知中心]; NotificationCenter -- 广播 --> observer1[observer1]; NotificationCenter -- 广播 --> observer2[observer2]; NotificationCenter -- 广播 --> observer3[observer3];
```

任何对象都可以观察通知，但要做到这一点，该对象必须注册，以接收通知。在注册时，它必须指定选择器，以确定由通知传送给调用的方法；方法签名必须只有一个参数：通知对象。注册后，观察者也可以指定发布对象。

（以上是官方文档中的解释）

-----华丽的分割线-----

通知中心包括两个重要的类：

（1）NSNotificationCenter: 实现NSNotificationCenter的原理是一个观察者模式，获得NSNotificationCenter的方法只有一种，那就是[NSNotificationCenter defaultCenter]，通过调用静态方法defaultCenter就可以获取这个通知中心的对象了，而NSNotificationCenter是一个单例模式，而这个通知中心的对象会一直存在于一个应用的生命周期。

（2）NSNotification: 这是消息携带的载体，通过它，可以把消息内容传递给观察者。

（3）一个NSNotificationCenter可以有许多的通知消息NSNotification，对于每一个NSNotification可以有很多的观察者Observer来接收通知。

通过上面的介绍可以知道，通过通知中心也可以实现不同类之间的参数传递。

注意当接受到消息后，不想再收到消息了，要把observer删除remove。

下面介绍如何使用（具体解释看文档）。

（1）NSNotification：用于创建传递的消息

[cpp]

2014年10月 (1)
2014年09月 (2)
2014年05月 (4)
2014年04月 (2)
2014年03月 (6)
展开

阅读排行	
iOS 通知中心 NSNotifica	(10881)
iOS 两种方法实现左右滑	(10381)
iOS UILocalNotification	(8580)
Cocos2d-x 3.0 新特性体	(8281)
A星(A*, A Star)算法详解	(8281)
iOS UIBezierPath类 介绍	(8265)
Cocos2d-x CCEditBox &	(8179)
Cocos2d-x CCNotificatio	(7170)
iOS 正则表达式 NSRegu	(6944)
iOS KVO & KVC	(6865)

评论排行	
iOS 两种方法实现左右滑	(67)
iOS 新浪微博客户端Dem	(25)
iOS 新浪微博客户端Dem	(17)
Cocos2d-x CCTableView	(10)
数据结构 之 并查集	(10)
iOS KVO & KVC	(8)
iOS 新浪微博客户端Dem	(8)
Cocos2d-x CCEditBox &	(7)
Cocos2d-x 游戏实践 -- 游	(5)
详细介绍在windows系统	(5)

推荐文章	
* 2015博文大赛精彩文章	
* 为什么我说Rust是靠谱的编程语言	
* Android UI常用实例 如何实现欢迎界面 (Splash Screen)	
* Android应用层View绘制流程与源码分析	
* Android屏幕适配全攻略	
* ios_swift开发资源整理	

最新评论	
Cocos2d-x 3.0 新特性体验 (2)	
没有梦想__何必远方: 学习了	
iOS UIBezierPath类 介绍	
lu_zy: -(void)drawArc: (CGContextRef)context{ /*添加弧形对象 ...	
一次买卖问题 (single sell profit)	
任焱: 谢谢分享 学习了`(*^__^*)'	
探究 C++ Singleton (单例模式)	
任焱: 谢谢分享 学习了`(*^__^*)'	
Dangling pointer (悬垂指针、迷	
任焱: 谢谢分享 学习了`(*^__^*)'	
C语言 内存分配 地址 指针 数组	
任焱: 谢谢分享 学习了`(*^__^*)'	
数据结构 之 并查集	
任焱: 谢谢分享 学习了`(*^__^*)'	
数据结构 之 并查集	
任焱: 谢谢分享学习了	

```
01. Creating Notifications
02. + notificationWithName:object:
03. + notificationWithName:object:userInfo:
04. Getting Notification Information
05. - name
06. - object
07. - userInfo
```

(2) NSNotificationCenter : 用于发送消息

```
[cpp]
01. Getting the Notification Center
02. + defaultCenter
03. Managing Notification Observers
04. - addObserverForName:object:queue:usingBlock:
05. - addObserver:selector:name:object:
06. - removeObserver:
07. - removeObserver:name:object:
08. Posting Notifications
09. - postNotification:
10. - postNotificationName:object:
11. - postNotific          ct:userInfo:
```

demo(例子中基本上涉及了两种方法了):
定义了两个类: Poster 和 Observer (接受消息)

Poster.h

```
[cpp]
01. #import <Foundation/Foundation.h>
02.
03. @interface Poster : NSObject
04.
05. -(void)postMessage;
06.
07. @end
```

Poster.m

```
[cpp]
01. #import "Poster.h"
02.
03. @implementation Poster
04.
05. -(void)postMessage{
06.
07. //1.下面两条语句等价
08. //二者的区别是第一条语句是直接发送消息内容, 第二条语句先创建一个消息, 然后再发送消息
09. [[NSNotificationCenter defaultCenter] postNotificationName:@"PosterOne" object:@"This is posterone"];
10.
11. // [[NSNotificationCenter defaultCenter] postNotification:
12. [[NSNotification notificationWithName:@"PosterOne" object:@"This is posterone"]];
13.
14. //2.下面两条语句等价
15. //参数: userInfo --- Information about the the notification.
16. [[NSNotificationCenter defaultCenter] postNotificationName:@"PosterTwo" object:@"This is posterone" userInfo:@{
17. [NSDictionary dictionaryWithObject:@"value" forKey:@"key"]];
18.
19. // [[NSNotificationCenter defaultCenter] postNotification:
20. [[NSNotification notificationWithName:@"PosterTwo" object:@"This is postertwo" userInfo:
21. [NSDictionary dictionaryWithObject:@"value" forKey:@"key"]];
22. }
23. @end
```

Observer.h

```
[cpp]
01. #import <Foundation/Foundation.h>
02.
```

iOS 两种方法实现左右滑动出现侧边栏
qq_27211167: 能不能把第二种不用第三方的demo发给我，谢谢啊 291044486@qq.com

数据结构之并查集

丁国华: 谢谢分享 学习了`(*^__^*)`

```
03. @interface Observer : NSObject
04.
05. -(void)observer;
06.
07. @end
```

Observer.m

```
[cpp]
01. #import "Observer.h"
02.
03. @implementation Observer
04.
05. -(void)observer {
06.     [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(callBack1:) name:
07.
08.     [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(callBack2:) name:
09.
10.     //删除所有的observer
11.     // [[NSNotificationCenter defaultCenter] removeObserver:self];
12.     //删除名字为name的observer
13.     // [[NSNotificationCenter defaultCenter] removeObserver:self name:@"PosterOne" object:nil];
14.
15. }
16.
17. -(void)callBack1:(NSNotification*)notification{
18.     NSString *nameString = [notification name];
19.     NSString *objectString = [notification object];
20.     NSLog(@"name = %@,object = %@",nameString,objectString);
21. }
22.
23. -(void)callBack2:(NSNotification*)notification{
24.     NSString *nameString = [notification name];
25.     NSString *objectString = [notification object];
26.     NSDictionary *dictionary = [notification userInfo];
27.     NSLog(@"name = %@,object = %@,userInfo = %@",nameString,objectString,
28.     [dictionary objectForKey:@"key"]);
29. }
30. @end
```

main.m

```
[cpp]
01. #import <Foundation/Foundation.h>
02. #import "Poster.h"
03. #import "Observer.h"
04.
05. int main(int argc, const char * argv[])
06. {
07.
08.     @autoreleasepool {
09.
10.         //注意这里的顺序,要先observer,然后再poster
11.         Observer *myObserver = [[Observer alloc] init];
12.         [myObserver observer];
13.
14.         Poster *poster = [[Poster alloc] init];
15.         [poster postMessage];
16.     }
17.     return 0;
18. }
```

好了，大概有关的内容都差不多了吧 😊

附：

不过有个方法

addObserverForName:object:queue:usingBlock:

还不太懂如何使用，暂时放一下，有知道了麻烦评论告诉了。

上一篇 iOS 委托模式

下一篇 iOS KVO & KVC