iOS黑魔法－Method Swizzling

2016-01-21 11:36 编辑：suiling 分类：iOS开发 来源：刘小壮 投稿 0 1226

Method Swizzling 黑魔法

招聘信息：iOS软件工程师

本文是投稿文章，作者：刘小壮

公司年底要在新年前发一个版本，最近一直很忙，好久没有更新博客了。正好现在新版本开发的差不多了，抽空总结一下。

由于最近开发新版本，就避免不了在开发和调试过程中引起崩溃，以及诱发一些之前的bug导致的崩溃。而且项目比较大也很不好排查，正好想起之前研究过的Method Swizzling，考虑是否能用这个苹果的“黑魔法”解决问题，当然用好这个黑魔法并不局限于解决这些问题.....

需求

就拿我们公司项目来说吧，我们公司是做导航的，而且项目规模比较大，各个控制器功能都已经实现。突然有一天老大过来，说我们要在所有页面添加统计功能，也就是用户进入这个页面就统计一次。我们会想到下面的一些方法：

手动添加

直接简单粗暴的在每个控制器中加入统计，复制、粘贴、复制、粘贴...

上面这种方法太Low了，消耗时间而且以后非常难以维护，会让后面的开发人员骂死的。

继承

我们可以使用OOP的特性之一，继承的方式来解决这个问题。创建一个基类，在这个基类中添加统计方法，其他类都继承自这个基类。

热门资讯


宅男福音！Xcode 程序员鼓励师插件 Miku
点击量 20971


看了一天简历的感悟~心真累
点击量 16762


iOS 架构模式--解密MVC，MVP，MVVM
点击量 11874


最轻巧的自动布局--ZXPAutoLayout框架
点击量 11789


实践干货！猿题库 iOS 客户端架构设计
点击量 11408


如何做优化，UITabelView才能
点击量 9842


我从55个Swift标准库协议中学到了什么？
点击量 9615


iOS面试一般性问题
点击量 8432


由App的启动说起
点击量 7012


基于彻底解耦合的实验性iOS架构
点击量 6515

综合评论

强调一点，在苹果官方文档中是不建议这么做的，UITabBarController不建议
348162291 评论了 控制器跳转之偷梁换柱...

有内存泄露吧
taontech 评论了 UITableview代理方法与Viewcont...

吃惊...
黎明破晓残夜尽 评论了 UITableview代理方法与Viewcont...

//很简单啊，毫无违和感，不用改变工程的结构
//引入头文件
#import "AppDelegate.h"
//override
- (void)viewDidAppear:

然而，这种方式修改还是很大，而且定制性很差。以后有新人加入之后，都要嘱咐其继承自这个基类，所以这种方式并不可取。

Category

我们可以为UIViewController建一个Category，然后在所有控制器中引入这个Category。当然我们也可以添加一个PCH文件，然后将这个Category添加到PCH文件中。

我们创建一个Category来覆盖系统方法，系统会优先调用Category中的代码，然后在调用原类中的代码。

我们可以通过下面的这段伪代码来看一下：

```
#import "UIViewController+EventGather.h"
@implementation UIViewController (EventGather)
- (void)viewDidLoad {
   NSLog(@"页面统计:%@", self);
}
@end
```

Method Swizzling

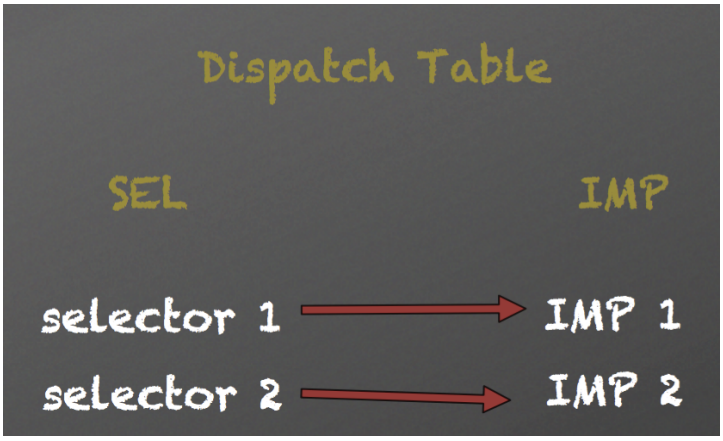
我们可以使用苹果的“黑魔法”Method Swizzling，Method Swizzling本质上就是对IMP和SEL进行交换。

Method Swizzling原理

Method Swizzing是发生在运行时的，主要用于在运行时将两个Method进行交换，我们可以将Method Swizzling代码写到任何地方，但是只有在这段Method Swilzzling代码执行完毕之后互换才起作用。

而且Method Swizzling也是iOS中AOP(面相切面编程)的一种实现方式，我们可以利用苹果这一特性来实现AOP编程。

首先，让我们通过两张图片来了解一下Method Swizzling的实现原理



图一

(BOOL)animated
{
暗の天使 评论了 控制器跳转之偷梁换柱...

绝对不能让我加设计看到这篇文章，不然我就没有好日子过了
Counting 评论了 【设计干货】UI动效的必备原则总结...

河南籍中枪，深圳研发主管的干活。我说句公道话：
| Tig | 评论了 看了一天简历的感悟~心真累...

mark
北冥_至尊 评论了 超全！整理常用的iOS第三方资源...

不错
逗魂 评论了 超全！整理常用的iOS第三方资源...

tab外最好再套一个navigation，基本都会用得到
beancafe 评论了 控制器跳转之偷梁换柱...

怒马
shen0713 评论了 超全！整理常用的iOS第三方资源...

相关帖子

葛洲坝集团电力有限责任公司打造民族品牌企业

［未知风险，友情提示］：有关Sending API Usage to iTunes Connect… 无法上传App的问题处理

关于json数据解析的...

群聊天 XMPP

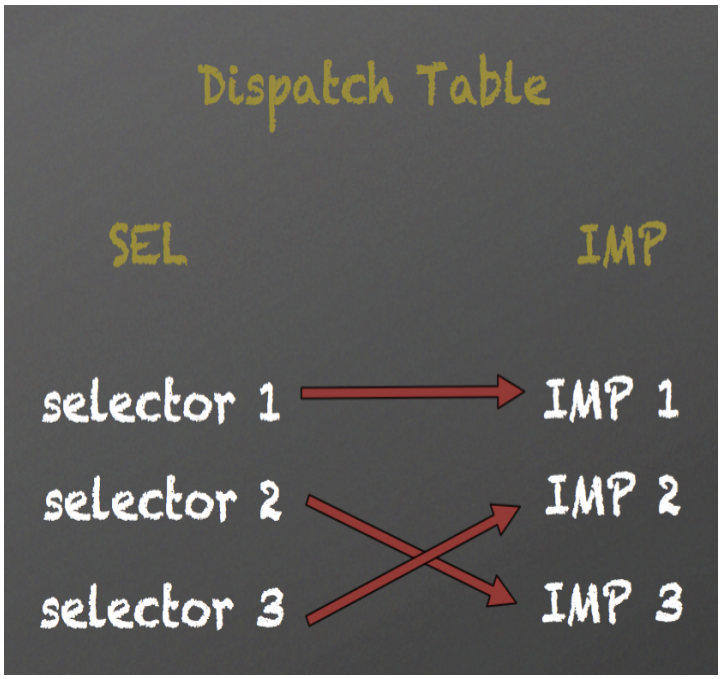
集成支付宝时报这错误是咋回事

iOS面试题答案

APP 转让问题!!!

Xcode7 开发证书

怎么将一个数字存在一个字节的类型里面在转为NSData



图二

上面图一中selector2原本对应着IMP2，但是为了更方便的实现特定业务需求，我们在图二中添加了selector3和IMP3，并且让selector2指向了IMP3，而selector3则指向了IMP2，这样就实现了“方法互换”。

在OC语言的runtime特性中，调用一个对象的方法就是给这个对象发送消息。是通过查找接收消息对象的方法列表，从方法列表中查找对应的SEL，这个SEL对应着一个IMP(一个IMP可以对应多个SEL)，通过这个IMP找到对应的方法调用。

在每个类中都有一个Dispatch Table，这个Dispatch Table本质是将类中的SEL和IMP(可以理解为函数指针)进行对应。而我们的Method Swizzling就是对这个table进行了操作，让SEL对应另一个IMP。

Method Swizzling使用

在实现Method Swizzling时，核心代码主要就是一个runtime的C语言API：

```
OBJC_EXPORT void method_exchangeImplementations(Method m1, Method m2)
  __OSX_AVAILABLE_STARTING(__MAC_10_5, __IPHONE_2_0);
```

实现思路

就拿上面我们说的页面统计的需求来说吧，这个需求在很多公司都很常见，我们下面的Demo就通过Method Swizzling简单的实现这个需求。

我们先给UIViewController添加一个Category，然后在Category中的+(void)load方法中添加Method Swizzling方法，我们用来替换的方法也写在这个Category中。由于load类方法是程序运行时这个类被加载到内存中就调用的一个方法，执行比较早，并且不需要我们手动调用。而且这个方法具有唯一性，也就是只会被调用一次，不用担心资源抢夺的问题。

定义Method Swizzling中我们自定义的方法时，需要注意尽量加前缀，以防止和其他地方命名冲突，Method Swizzling的替换方法命名一定要是唯一的，至少在被替换的类中必须是唯一的。

```
#import "UIViewController+swizzling.h"
#import @implementation UIViewController (swizzling)

+ (void)load {
    [super load];
    // 通过class_getInstanceMethod()函数从当前对象中的method list获取method结构体，如果是类
    Method fromMethod = class_getInstanceMethod([self class], @selector(viewDidLoad));
    Method toMethod = class_getInstanceMethod([self class], @selector(swizzlingViewDidL
```

```
    /**
     *  我们在这里使用class_addMethod()函数对Method Swizzling做了一层验证，如果self没有实现
     *  而且self没有交换的方法实现，但是父类有这个方法，这样就会调用父类的方法，结果就不是我们
     *  所以我们在这里通过class_addMethod()的验证，如果self实现了这个方法，class_addMethod(
     */
    if (!class_addMethod([self class], @selector(viewDidLoad), method_getImplementatior
        method_exchangeImplementations(fromMethod, toMethod);
    }
}

// 我们自己实现的方法，也就是和self的viewDidLoad方法进行交换的方法。
- (void)swizzlingViewDidLoad {
    NSString *str = [NSString stringWithFormat:@"%@", self.class];
    // 我们在这里加一个判断，将系统的UIViewController的对象剔除掉
    if(![str containsString:@"UI"]){
        NSLog(@"统计打点 : %@", self.class);
    }
    [self swizzlingViewDidLoad];
}
@end
```

看到上面的代码，肯定有人会问：楼主，你太粗心了，你在swizzlingViewDidLoad方法中又调用了[self swizzlingViewDidLoad];，这难道不会产生递归调用吗？

答：然而....并不会????。

还记得我们上面的图一和图二吗？Method Swizzling的实现原理可以理解为”方法互换“。假设我们将A和B两个方法进行互换，向A方法发送消息时执行的却是B方法，向B方法发送消息时执行的是A方法。

例如我们上面的代码，系统调用UIViewController的viewDidLoad方法时，实际上执行的是我们实现的swizzlingViewDidLoad方法。而我们在swizzlingViewDidLoad方法内部调用[self swizzlingViewDidLoad];时，执行的是UIViewController的viewDidLoad方法。

Method Swizzling类簇

之前我也说到，在我们项目开发过程中，经常因为NSArray数组越界或者NSDictionary的key或者value值为nil等问题导致的崩溃，对于这些问题苹果并不会报一个警告，而是直接崩溃，感觉苹果这样确实有点“太狠了”。

由此，我们可以根据上面所学，对NSArray、NSMutableArray、NSDictionary、NSMutableDictionary等类进行Method Swizzling，实现方式还是按照上面的例子来做。但是....你发现Method Swizzling根本就不起作用，代码也没写错啊，到底是什么鬼？

这是因为Method Swizzling对NSArray这些的类簇是不起作用的。因为这些类簇类，其实是一种抽象工厂的设计模式。抽象工厂内部有很多其它继承自当前类的子类，抽象工厂类会根据不同情况，创建不同的抽象对象来进行使用。例如我们调用NSArray的objectAtIndex:方法，这个类会在方法内部判断，内部创建不同抽象类进行操作。

所以也就是我们对NSArray类进行操作其实只是对父类进行了操作，在NSArray内部会创建其他子类来执行操作，真正执行操作的并不是NSArray自身，所以我们应该对其“真身”进行操作。

下面我们实现了防止NSArray因为调用objectAtIndex:方法，取下标时数组越界导致的崩溃：

```
#import "NSArray+LXZArray.h"
#import "objc/runtime.h"
@implementation NSArray (LXZArray)
+ (void)load {
    [super load];
    Method fromMethod = class_getInstanceMethod(objc_getClass("__NSArrayI"), @selector(
    Method toMethod = class_getInstanceMethod(objc_getClass("__NSArrayI"), @selector(lx
    method_exchangeImplementations(fromMethod, toMethod);
}

- (id)lxz_objectAtIndex:(NSUInteger)index {
    if (self.count > index) {
        // 这里做一下异常处理，不然都不知道出错了。
        @try {
            return [self lxz_objectAtIndex:index];
        }
        @catch (NSException *exception) {
            // 在崩溃后会打印崩溃信息，方便我们调试。
            NSLog(@"---------- %s Crash Because Method %s  ----------\n", class_getName
            NSLog(@"%@", [exception callStackSymbols]);
    }
        @finally {}
    } else {
        return [self lxz_objectAtIndex:index];
    }
```

```
26 }
27 @end
```

大家发现了吗，__NSArrayI才是NSArray真正的类，而NSMutableArray又不一样????。我们可以通过runtime函数获取真正的类：

```
1 objc_getClass("__NSArrayI")
```

下面我们列举一些常用的类簇的“真身”：

类	“真身”
NSArray	__NSArrayI
NSMutableArray	__NSArrayM
NSDictionary	__NSDictionaryI
NSMutableDictionary	__NSDictionaryM

Method Swizzling封装

在项目中我们肯定会在很多地方用到Method Swizzling，而且在使用这个特性时有很多需要注意的地方。我们可以将Method Swizzling封装起来，也可以使用一些比较成熟的第三方。
在这里我推荐Github上星最多的一个第三方 – jrswizzle

里面核心就两个类，代码看起来非常清爽。

```
1 #import @interface NSObject (JRSwizzle)
2 + (BOOL)jr_swizzleMethod:(SEL)origSel_ withMethod:(SEL)altSel_ error:(NSError**)error_;
3 + (BOOL)jr_swizzleClassMethod:(SEL)origSel_ withClassMethod:(SEL)altSel_ error:(NSError*
4 @end
5
6 // MethodSwizzle类
7 #import BOOL ClassMethodSwizzle(Class klass, SEL origSel, SEL altSel);
8 BOOL MethodSwizzle(Class klass, SEL origSel, SEL altSel);
```

Method Swizzling危险吗?

既然Method Swizzling可以对这个类的Dispatch Table进行操作，操作后的结果对所有当前类及子类都会产生影响，所以有人认为Method Swizzling是一种危险的技术，用不好很容易导致一些不可预见的bug，这些bug一般都是非常难发现和调试的。
这个问题可以引用念茜大神的一句话：使用 Method Swizzling 编程就好比切菜时使用锋利的刀，一些人因为担心切到自己所以害怕锋利的刀具，可是事实上，使用钝刀往往更容易出事，而利刀更为安全。

上一篇：源码推荐(01.20B)：JS与OC中Webview交互， 两行代码搞定TouchID验证

相关资讯

详解苹果的黑魔法 - KVO 的奥秘	Method Swizzling和分类的妙用–从AppDelegate轻量化处
Method Swizzling 和 AOP 实践	Method Swizzling和AOP(面向切面编程)实践
Objective-C Method Swizzling	iOS app调试的黑魔法--第三方库
Method Swizzling	宏定义的黑魔法 - 宏菜鸟起飞手册

我来说两句

你怎么看？快来评论一下吧！

发表评论

所有评论（0）