【iOS】Category VS Extension 原理详解

（一）Category

1、什么是 Category?

category 是 Objective-C 2.0 之后添加的语言特性，别人口中的分类、类别其实都是指的 category。category 的主要作用是为已经存在的类添加方法。除此之外，apple 还推荐了 category 的另外两个使用场景。

可以把类的实现分开在几个不同的文件里面。这样做有几个显而易见的好处。

- 可以减少单个文件的体积
- 可以把不同的功能组织到不同的 category 里
- 可以由多个开发者共同完成一个类
- 可以按需加载想要的 category
- 声明私有方法

apple 的 SDK 中就大面积的使用了 category 这一特性。比如 UIKit 中的 UIView。apple 把不同的功能 API 进行了分类，这些分类包括 UIViewGeometry、UIViewHierarchy、UIViewRendering 等。

不过除了 apple 推荐的使用场景，广大开发者脑洞大开，还衍生出了 category 的其他几个使用场景：

- 模拟多继承（另外可以模拟多继承的还有 protocol）
- 把 framework 的私有方法公开

2、category 特点

- category 只能给某个已有的类扩充方法，不能扩充成员变量。
- category 中也可以添加属性，只不过 @property 只会生成 setter 和 getter 的声明，不会生成 setter 和 getter 的实现以及成员变量。
- 如果 category 中的方法和类中原有方法同名，运行时会优先调用 category 中的方法。也就是，category 中的方法会覆盖掉类中原有的方法。所以开发中尽量保证不要让分类中的方法和原有类中的方法名相同。

避免出现这种情况的解决方案是给分类的方法名统一添加前缀。比如 category_。

- 如果多个 category 中存在同名的方法，运行时到底调用哪个方法由编译器决定，最后一个参与编译的方法会被调用。

如下图，给 UIView 添加了两个 category（one 和 two），并且给这两个分类都添加了名为 log 的方法：

```
#import "UIView+one.h"

@implementation UIView (one)
- (void)log {
    NSLog(@"调用分类1的方法");
}
@end
```

UIView+one

```
#import "UIView+two.h"

@implementation UIView (two)
- (void)log {
    NSLog(@"调用分类2的方法");
}

@end
```

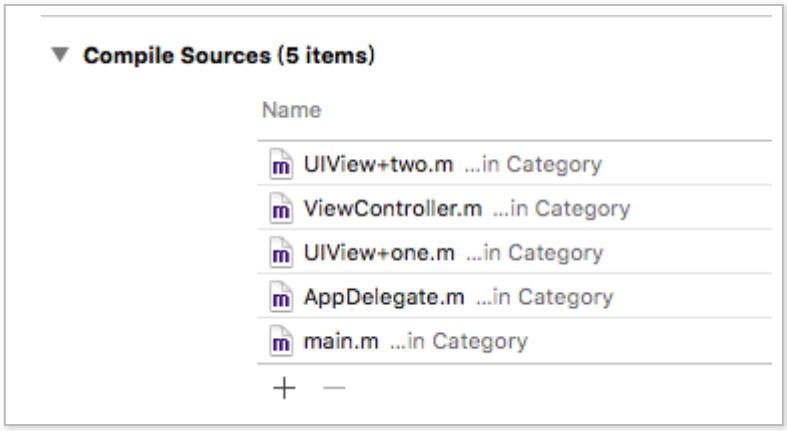
UIView+two

在 viewController 中引入这两个 category 的. h 文件。调用 log 方法：

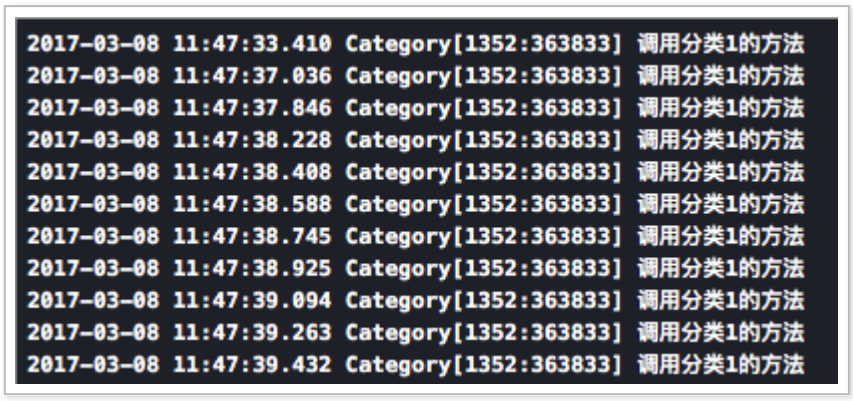
```
UIView *view = [UIView new];
[view log];
```

调用 category 方法

当编译顺序如下图所示时，调用 UIView + one.m 的 log 方法，如下图：

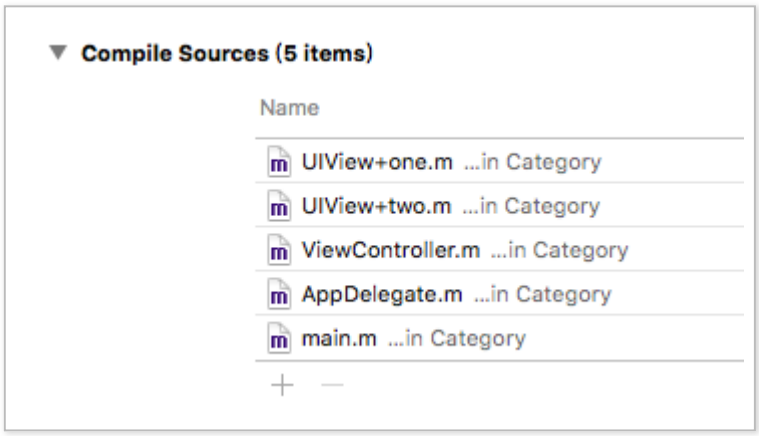


编译顺序

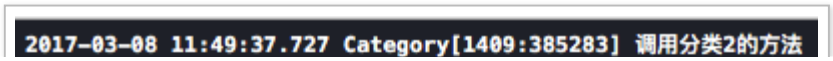


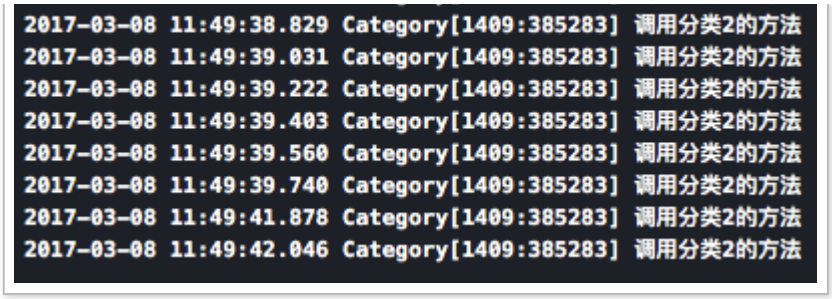
调用结果

将 UIView + one.m 移动到 UIView + two.m 上面，调用 UIView + two.m 的 log 方法，如下图：



编译顺序





调用结果

3、调用优先级

分类 (category) > 本类 > 父类。即，优先调用 cateory 中的方法，然后调用本类方法，最后调用父类方法。

注意：category 是在运行时加载的，不是在编译时。

4、为什么 category 不能添加成员变量?

Objective-C 类是由 Class 类型来表示的，它实际上是一个指向 objc_class 结构体的指针。它的定义如下：

```
typedef struct objc_class *Class;
```

objc_class 结构体的定义如下：

```
struct objc_class {
    Class isa  OBJC_ISA_AVAILABILITY;
#if !__OBJC2__
    Class super_class                                       OBJC2_UNAV
    const char *name                                        OBJC2_UNAV
    long version                                            OBJC2_UNAV
    long info                                               OBJC2_UNAV
    long instance_size                                      OBJC2_UNAV
    struct objc_ivar_list *ivars                            OBJC2_UNAV
    struct objc_method_list **methodLists                   OBJC2_UNAV
    struct objc_cache *cache                                OBJC2_UNAV
    struct objc_protocol_list *protocols                    OBJC2_UNAV
#endif
} OBJC2_UNAVAILABLE;
```

在上面的 objc_class 结构体中，ivars 是 objc_ivar_list（成员变量列表）指针；methodLists 是指向 objc_method_list 指针的指针。在 Runtime 中，

objc_class 结构体大小是固定的，不可能往这个结构体中添加数据，只能修

改。所以 ivars 指向的是一个固定区域，只能修改成员变量值，不能增加成员变量个数。methodList 是一个二维数组，所以可以修改 * methodLists 的值来增加成员方法，虽没办法扩展 methodLists 指向的内存区域，却可以改变这个内存区域的值（存储的是指针）。因此，可以动态添加方法，不能添加成员变量。

5、category 中能添加属性吗?

Category 不能添加成员变量（instance variables），那到底能不能添加属性（property）呢?

这个我们要从 Category 的结构体开始分析:

```
typedef struct category_t {
    const char *name;  //类的名字
    classref_t cls;  //类
    struct method_list_t *instanceMethods;  //category
    struct method_list_t *classMethods;  //category中所
    struct protocol_list_t *protocols;  //category实现
    struct property_list_t *instanceProperties;  //cat
} category_t;
```

从 Category 的定义也可以看出 Category 的可为（可以添加实例方法，类方法，甚至可以实现协议，添加属性）和不可为（无法添加实例变量）。

但是为什么网上很多人都说 Category 不能添加属性呢?

实际上，Category 实际上允许添加属性的，同样可以使用 @property，但是不会生成_变量（带下划线的成员变量），也不会生成添加属性的 getter 和 setter 方法的实现，所以，尽管添加了属性，也无法使用点语法调用 getter 和 setter 方法（实际上，点语法是可以写的，只不过在运行时调用到这个方法时候会报方法找不到的错误，如下图）。但实际上可以使用 runtime 去实现 Category 为已有的类添加新的属性并生成 getter 和 setter 方法。详细内容可以看峰哥之前的文章：《iOS Runtime 之四：关联对象》

```
#import "EOCPerson.h"

@interface EOCPerson (BaseInfo)
@property(nonatomic,copy) NSString *
@end
```

给 category 添加 property

```
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     EOCPerson *person = [[EOCPerson alloc] init];
22     person.name = @"vvs";                    Thread 1: breakpoint 1.2
23
24 }
25
26 @end
```

5 -[ViewCon...viewDidLoad]

```
2017-03-09 23:57:10.482 associatedObject[733:25454] -[EOCPerson setName:]:
unrecognized selector sent to instance 0x600000008790
(lldb)
```

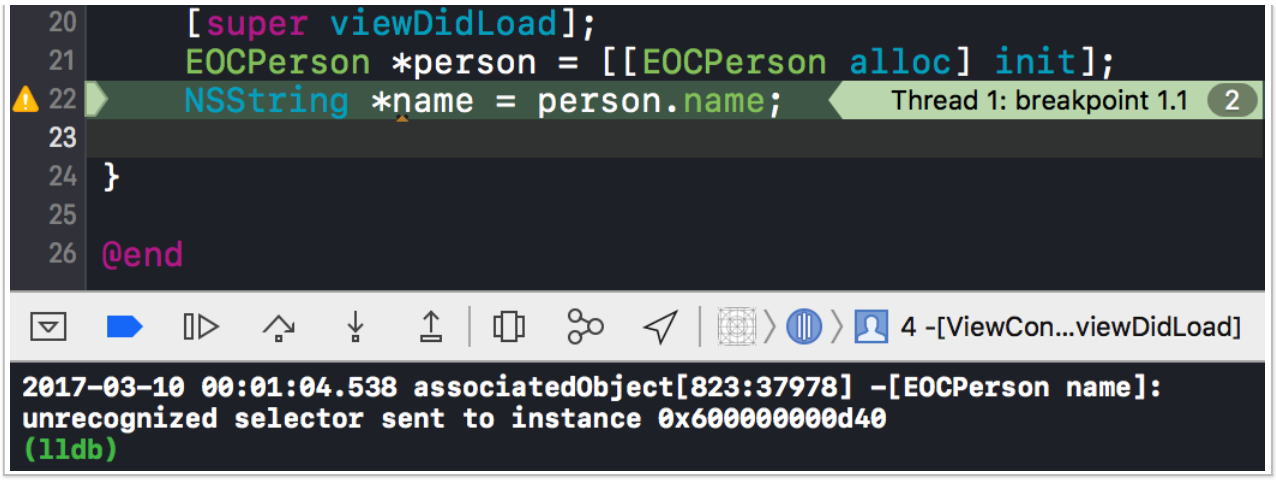
调用 category 中 property 的 setter（报方法找不到的错误）

```
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     EOCPerson *person = [[EOCPerson alloc] init];
22     NSString *name = person.name;    Thread 1: breakpoint 1.1  2
23
24 }
25
26 @end
```

4 -[ViewCon...viewDidLoad]

```
2017-03-10 00:01:04.538 associatedObject[823:37978] -[EOCPerson name]:
unrecognized selector sent to instance 0x600000000d40
(lldb)
```

```
19 - (void)viewDidLoad {
```



调用 category 中 property 的 getter（报方法找不到的错误）

需要注意的有两点：

- 1)、category 的方法没有“完全替换掉”原来类已经有的方法，也就是说如果 category 和原来类都有 methodA，那么 category 附加完成之后，类的方法列表里会有两个 methodA。
- 2)、category 的方法被放到了新方法列表的前面，而原来类的方法被放到了新方法列表的后面，这也就是我们平常所说的 category 的方法会“覆盖”掉原来类的同名方法，这是因为运行时在查找方法的时候是顺着方法列表的顺序查找的，它只要一找到对应名字的方法，就会罢休，殊不知后面可能还有一样名字的方法。

（二）Extension

1、 什么是 extension

extension 被开发者称之为扩展、延展、匿名分类。extension 看起来很像一个匿名的 category，但是 extension 和 category 几乎完全是两个东西。和 category 不同的是 extension 不但可以声明方法，还可以声明属性、成员变量。extension 一般用于声明私有方法，私有属性，私有成员变量。

2、 extension 的存在形式

category 是拥有. h 文件和. m 文件的东西。但是 extension 不然。
extension 只存在于一个. h 文件中，或者 extension 只能寄生于一个类的.

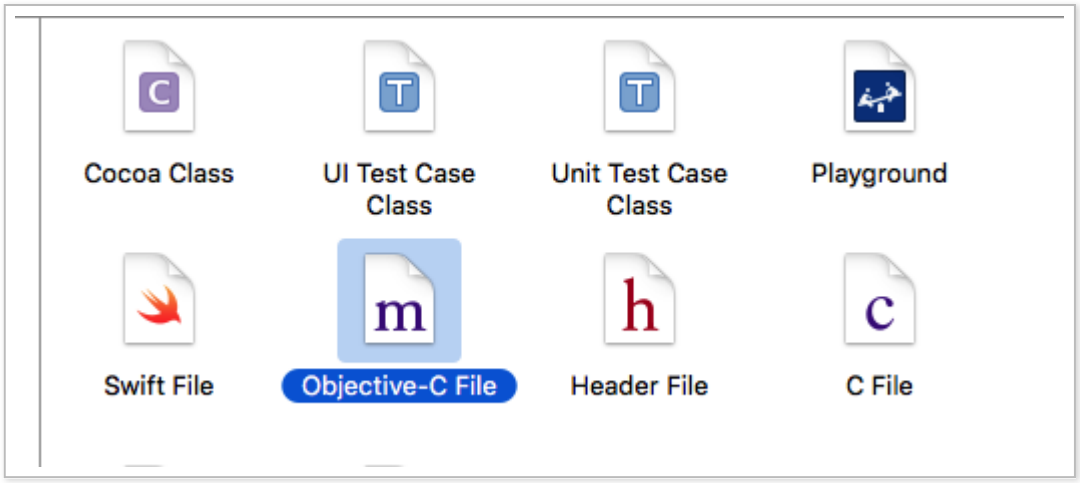
m 文件中。比如，viewController.m 文件中通常寄生这么个东西，其实这就

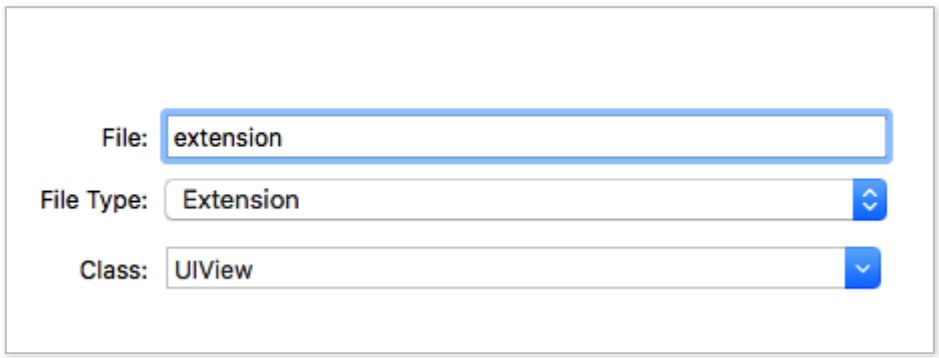
是一个 extension：

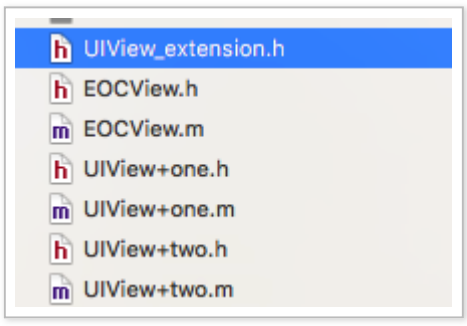
```
@interface ViewController ()

@end
```

当然我们也可以创建一个单独的 extension 文件，如下图：







UIView_extension.h 中声明方法：

```
#import <UIKit/UIKit.h>

@interface UIView ()
- (void)testExtension;
@end
```

导入 UIView_extension.h 文件进行使用：

```
#import "EOCView.h"
#import "UIView_extension.h"

@implementation EOCView
- (void)testExtension {
    NSLog(@"test");
}
```

注意：extension 常用的形式并不是以一个单独的. h 文件存在，而是寄生在类的. m 文件中。

（三）category 和 extension 的区别

就 category 和 extension 的区别来看，我们可以推导出一个明显的事实，extension 可以添加实例变量，而 category 是无法添加实例变量的（因为在运行期，对象的内存布局已经确定，如果添加实例变量就会破坏类的内部布局，这对编译型语言来说是灾难性的）。

- extension 在编译期决议，它就是类的一部分，但是 category 则完全不一样，它是在运行期决议的。extension 在编译期和头文件里的 @interface 以及实现文件里的 @implement 一起形成一个完整的类，它、extension 伴随类的产生而产生，亦随之一起消亡。
- extension 一般用来隐藏类的私有信息，你必须有一个类的源码才能为一个类添加 extension，所以你无法为系统的类比如 NSString 添加 extension，除非创建子类再添加 extension。而 category 不需要有类的源码，我们可以给系统提供的类添加 category。
- extension 可以添加实例变量，而 category 不可以。

- extension 和 category 都可以添加属性，但是 category 的属性不能生成

成员变量和 getter、setter 方法的实现。

文 / VV 木公子（简书作者）

如果您是 iOS 开发者，或者对本篇文章感兴趣，请关注本人，后续会更新更多相关文章！敬请期待！

参考文章

iOS Category 详解

Objective-C 的 Category 与关联对象实现原理

全文完