轻松学习之 IMP指针的作用

可能大家一直看到有许多朋友在Runtime相关文章中介绍IMP指针的概念，那么IMP究竟有什么实际作用呢？让我们先从一个函数看起来。

**Method Swizzling**

如果对Runtime有一定了解的话，一定听说过或者用过这个函数：

|  |  |
| --- | --- |
| 1 | void method\_exchangeImplementations(Method m1, Method m2) |

它通常叫做method swizzling，算是ObjC的"黑魔法"了，作用就是在程序运行期间动态的给两个方法互换实现，比如有这样一种使用场景：

我们的程序中有许多个ViewController，我想在对项目改动最小的情况下，在当每个Controller执行完ViewDidLoad以后就在控制台把自己的名字打印出来，方便我去做调试或者了解项目结构。

有许多朋友会这样说，让所有控制器都继承一个BaseController不就可以了吗？我在这里要解释一下这样做的缺点：假如你的项目里有许多Controller的话，你就需要把项目里凡是没有继承自BaseController的每个Controller都做一次修改了，而且随意更改层级结构会发生意想不到的错误。

其实我们的目的就是重写ViewDidLoad的方法，并在他的方法最后加上几句Log，所以我们需要给UIViewController建立一个category，因为我们知道，如果在Catagory中重写一个方法，就会覆盖它的原有方法实现，但是，这样做以后就没有办法调用系统原有的方法，因为在一个方法里调用自己的方法会是一个死循环。所以我们的解决办法就是，另外写一个方法来和viewDidLoad“交换”，这样外部调用viewDidLoad就会调到新建的这个方法中，同样，我们调用新建的方法就会调用到系统的viewDidLoad中了。



**IMP指针**

其实，还有一种更加简单的方法可以让我们办到相同的目的，运用IMP指针，IMP就是Implementation的缩写，顾名思义，它是指向一个方法实现的指针，每一个方法都有一个对应的IMP，所以，我们可以直接调用方法的IMP指针，来避免方法调用死循环的问题。

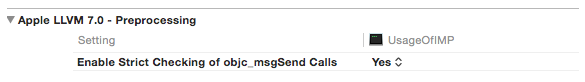
调用一个IMP的方式和调用普通C函数相同，比如：

|  |  |
| --- | --- |
| 1 | id returnObjc = someIMP(objc,SEL,params...); |

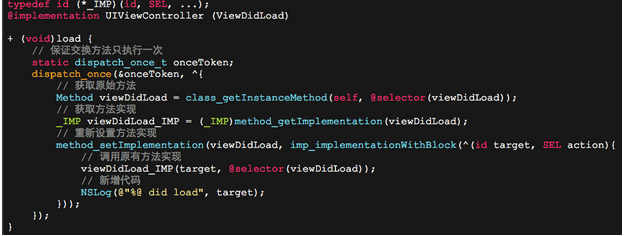
不过如果你的项目没有做其他配置的话这样调用编译器是不会通过的，我们来看一下先它的定义：

|  |  |
| --- | --- |
| 1  2  3  4  5 | if !OBJC\_OLD\_DISPATCH\_PROTOTYPES  typedef void (\*IMP)(void /\* id, SEL, ... \*/ );  else  typedef id (\*IMP)(id, SEL, ...);  endif |

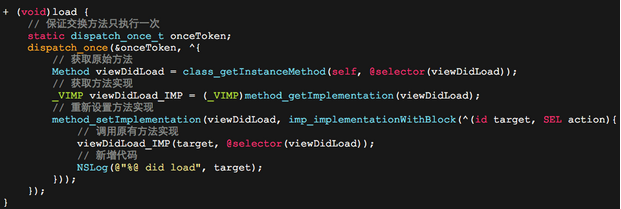
在默认情况下你的工程是打开这个配置的



这种情况下IMP被定义为无参数无返回值的函数。所以你需要到工程里搜索到这个选项并把它关闭。这样的麻烦就是，每次使用，你都需要修改工程配置，所以这里我再介绍另外一种办法：重新定义一个和有参数的IMP指针相同的指针类型，在获取IMP时把它强转为此类型。这样运用IMP指针后，就不需要额外的给ViewController写新的方法：



还有一个地方我们需要注意，如果这样直接调用IMP的话就会发生经典的EXC\_BAD\_ACCESS错误，我们定义的IMP指针是一个有返回值的类型，而其实我们获取的viewDidLoad这个方法是没有返回值的，所以我们需要新定义一个和IMP相同类型的函数指针比如VIMP，把他的返回值定位Void，这样如果你修改的方法有返回值就用IMP，没有返回值就用VIMP。



blob.png

值得注意的是，如果你重写的方法有返回值，不要忘记在最后做return。

**总结**

实际上直接调用一个方法的IMP指针的效率是高于调用方法本身的，所以，如果你有一个合适的时机获取到方法的IMP的话，你可以试着调用它。

这是只是IMP使用的场景之一，它还有许多作用，希望大家多多发现。