iOS内存管理

暴风体育 分享 iOS

引用计数

- 查看引用计数：
  MRC: [object retainCount]
  ARC: CFGetRetainCount((__bridge CFTypeRef)object)
- 强引用/弱引用
  普通指针： void *
  强引用：**类似Boost中shared_ptr**；引用计数+1；会引起循环引用
  弱引用：类似Boost中weak_ptr；和普通指针一样，引用计数不变；用来解决循环引用

new/alloc/retain/copy/release/dealloc

扩展：mutableCopy / autorelease

循环引用

eg. A -> B -> ... -> X -> B (->表示强引用)

- delegate

  > ARC下用weak，MRC下用assign
  > weak修饰的变量在释放时自动指向nil。(扩展思考：如何实现？)

- Block

  > 在Block中使用self容易出现循环引用，因此很多人在使用block时，都会申明一个__weak修饰的self。
  > **不是所有block中使用self都需要用__weak来修饰。**

> 思考：
> 1.为什么苹果编译器在block中引用会默认__strong的？在技术上很容易做到。
> 2.在什么情况下使用__weak会有问题？有什么问题？
> **扩展阅读**：
> @weakify / @strongify

- NSTimer

> 当创建NSTimer对象时，会默认对self有个强引用，所以在self打算释放之前，需要调用NSTimer的invalidate来释放对self的引用。

引用计数加1

内存释放法则

> **谁创建，谁释放**

深拷贝/浅拷贝

源对象类型	拷贝方法	副本对象类型	是否产生了新对象	拷贝类型
NSString	copy	NString	NO	浅拷贝（指针拷贝）
	mutableCopy	NSMutableString	YES	深拷贝（内容拷贝）
NSMutableString	copy	NString	YES	深拷贝（内容拷贝）
	mutableCopy	NSMutableString	YES	深拷贝（内容拷贝）

源对象类型	拷贝方法	副本对象类型	是否产生了新对象	拷贝类型
NSString	copy	NString	NO	浅拷贝（指针拷贝）
	mutableCopy	NSMutableString	YES	深拷贝（内容拷贝）
NSMutableString	copy	NString	YES	深拷贝（内容拷贝）
	mutableCopy	NSMutableString	YES	深拷贝（内容拷贝）
NSArray	copy	NSArray	NO	浅拷贝（指针拷贝）
	mutableCopy	NSMutableArray	YES	深拷贝（内容拷贝）
NSMutableArray	copy	NSArray	YES	深拷贝（内容拷贝）
	mutableCopy	NSMutableArray	YES	深拷贝（内容拷贝）
NS*	copy	NS*	NO	浅拷贝（指针拷贝）
	mutableCopy	NSMutable*	YES	深拷贝（内容拷贝）
NSMutable*	copy	NS*	YES	深拷贝（内容拷贝）
	mutableCopy	NSMutable*	YES	深拷贝（内容拷贝）

工具

- instruments