iOS (http://lib.csdn.net/base/ios)

iOS (http://lib.csdn.net/base/ios) - Objective-C (http://lib.csdn.net/ios/node/671) - ARC和MRC (http://lib.csdn.net/ios/knowledge/1458)

Objective-C 内存管理机制

作者：lover0920 (http://my.csdn.net/lover0920)

一、Objective-C内存管理的对象

1、IOS开发中，内存中的对象主要有两类

一类是值类型，比如int、float、struct等基本数据类型。

一类是引用类型，也就是继承自NSObject类的所有的OC对象。前一种值类型不需要我们管理，后一种引用类型是需要我们管理内存的，一旦管理不好，就会产生非常糟糕的后果。

2、为什么值类型不需要管理，而引用类型需要管理呢?

1）那是因为他们分配内存方式不一样。

A、值类型会被放入栈中，他们依次紧密排列，在内存中占有一块连续的内存空间，遵循先进后出的原则。

B、引用类型会被放到堆中，当给对象分配内存空间时，会随机的从内存当中开辟空间，对象与对象之间可能会留有不确定大小的空白空间，因此会产生很多内存碎片，需要我们管理。

2）栈内存与堆内存从性能上比较：栈内存要优于堆内存，这是因为栈遵循先进后出的原则，但是当数据量过大时，存入栈会明显的降低性能。因此，我们会把大量的数据存入堆中，然后栈中存放堆的地址，当需要调用数据时，就可以快速的通过栈内的地址找到堆中的数据。

3）值类型和引用类型之间是可以相互转化的：

A、把值类型转化为引用类型的过程叫做装箱，比如把int包装为NSNumber，这个过程会增加程序的运行时间，降低性能。

B、把引用类型转为值类型的过程叫做拆箱，比如把NSNumer转为float，在拆箱的过程中，我们一定要注意数据原有的类型，如果类型错误，可能导致拆箱失败，因此会存在安全性的问题。

C、手动的拆箱和装箱，都会增加程序的运行时间，降低代码可读性，影响性能。

二、为什么要使用内存管理

因为Objc中所有的对象都是动态内存分配的，也就是都创建在堆控件中，整个过程由程序员来控制。其实objc对象的所有权，谁申请谁释放等都强调的是一件事，那就是责任制。整个对象的生命周期我们都要负责，所以我们需要一个稳定而合理的内存管理，出现了内存泄露的问题，因为程序员懒或疏忽了或是由程序结构造成的。

三、Objective-C管理内存的方式

1、引用计数

这是一种古老但有效的内存管理方式。每个对象(特指：类的实例)内部都有一个retainCount的引用计数，对象刚被创建时，引用计数为1；可以手动调用retain方法使引用计数加1；同样也可以手动调用release方法使引用计数减1。调用release方法时，如果retainCount值减到0，系统将自动调用对象的dealloc方法(类似于c#中的dispose方法)，开发人员可以在dealloc中释放或清理资源。

2、OC中提供了两种内存管理机制MRC和ARC来满足不同的需求。

1）MRC与ARC区别如下图所示。

2）MRC手动管理内存（人工引用计数Mannul Reference Counting）

MRC模式下，所有的对象都需要手动的添加retain、release代码来管理内存。使用MRC，需要遵守谁创建，谁回收的原则。也就是谁alloc，谁release；谁retain，谁release。

当引用计数为0的时候，必须回收，引用计数不为0，不能回收，如果引用计数为0，但是没有回收，会造成内存泄露。如果引用计数为0，继续释放，会造成野指针。为了避免出现野指针，我们在释放的时候，会先让指针=nil。

3）ARC自动管理内存(自动引用计数Automatic Reference Counting)。

ARC是IOS5推出的新功能，通过ARC，可以自动的管理内存。在ARC模式下，只要没有强指针（强引用）指向对象，对象就会被释放。在ARC模式下，不允许使用retain、release、retainCount等方法。并且，如果使用dealloc方法时，不允许调用[super dealloc]方法。

ARC模式下的property变量修饰词为strong、weak，相当于MRC模式下的retain、assign。strong :代替retain，缺省关键词，代表强引用。weak：代替assign，声明了一个可以自动设置nil的弱引用，但是比assign多一个功能，指针指向的地址被释放之后，指针本身也会自动被释放。

四、与内存有关的修饰符

1、读写性修饰符：readwrite | readonly

readwrite:表明这个属性是可读可写的，系统为我们创建这个属性的setter和getter方法。

readonly:表明这个属性只能读不能写，系统只为我们创建一个getter方法，不会创建setter方法

系统默认是readwrite

2、strong | weak | assign | retain | copy

strong就表示是强引用类型，就相当于MRC中的retain。针对对象类型进行内存管理。如果对基本数据类型使用，则Xcode会直接报错。当给对象类型使用此修饰符时，setter方法会先将旧的对象属性release掉，再对新的对象进行一次赋值并进行一次retain操作。

weak就表示的时弱引用类型，就相当于MRC中的assign。但是它比assign安全；assign是直接赋值，可能会出现野指针的情况。但是weak不会出现野指针，weak会自动置成 nil。

assign:表示直接赋值，用于基本数据类型(NSInteger和CGFloat)和C数据类型(如int, float, double, char等)另外还有id类型，这个修饰符不会牵涉到内存管理。但是如果是对象类型，使用此修饰符则可能会导致内存泄漏或EXC_BAD_ACCESS错误

copy:主要用在NSString类型，表示复制内容。

对于普通的OC对象系统默认属性是strong，对于基本数据类型系统默认属性是assign。

3、原子性修饰符:atomic | nonatomic

atomic:表示是线程安全的。

nonatomic:表示是非线程安全的，使用此属性性能会提高一些。

系统默认是atomic

4、getter和setter修饰符

@property(getter = getMethodName, setter = setMethodName) Object *obj;

这两个属性修饰符用于设置自定义生成的getter和setter方法名，使用之后将不再使用系统默认的setter和getter方法名。

五、MRC与ARC混编

MRC与ARC理论上是不能兼容的，也就是你如果创建的项目是ARC模式的，在你的代码中是不能使用release，否则会出现内存问题。现在大部分程序都会选择ARC的方式，但是很多第三方的框架是MRC模式，如果想把这些第三方的文件加到自己项目中，需要进行标识，否则编译的时候会出现错误。

在ARC的项目中，对MRC的文件可以添加编译选项-fno-objc-arc的标识；在MRC的项目中，对ARC的文件可以添加编译选项 -fobjc-arc的标识。 步骤如下图所示。

把MRC文件转为ARC，实际上是去掉文件中的retain、release，因此也通过下图中方式完成。

查看原文>> (http://blog.csdn.net/lover0920/article/details/50460080)

2

看过本文的人也看了：

- iOS知识结构图 (http://lib.csdn.net/base/ios/structure)
- Objective-C入门解读与内存管理方式 (http://lib.csdn.net/article/ios/42751)

- Objective-C 【内存管理&手动内存管理... (http://lib.csdn.net/article/ios/43324)
- Objective-C之成魔之路【17-内存管理】... (http://lib.csdn.net/article/ios/41507)
- 从Java到Objective-C：从内存管理说起 (http://lib.csdn.net/article/ios/43059)
- objective-c中的内存管理——引用计数、... (http://lib.csdn.net/article/ios/43314)

发表评论

输入评论内容

发表

0个评论