iOS学习笔记总结整理

**一.内存管理情况**

**1- autorelease**，当用户的代码在持续运行时，自动释放池是不会被销毁的，这段时间内用户可以安全地使用自动释放的对象。当用户的代码运行告一段 落，开始等待用户的操作，自动释放池就会被释放掉（调用dealloc），池中的对象都会收到一个release，有可能会因此被销毁。  
**2-成员属性：**  
    readonly：不指定readonly，默认合成getter和setter方法。外界毫不关心的成员，则不要设置任何属性，这样封装能增加代码的独立性和安全性。相当于私有成员？  
    assign：表示只进行简单的赋值，不会发送retain消息。  
    retain：会向旧值发送release消息，向新值发送retain消息。  
    copy：向旧值发release消息，向新值发送copy消息。  
**3-引用计数特殊情况**：  
    NSArray：当把一个对象添加到NSArray时，对象的引用计数会加1，移除时会减1。  
    UIView：addGestureRecognizer会对手势识别器的引用计数加1，removeGestureRecognizer会减1。  
**二 .图形API**  
这个讲的好，清晰明确：[http://www.cocoachina.com/bbs/read.php?tid=77731&page=1](http://www.douban.com/link2?url=http%3A//www.cocoachina.com/bbs/read.php%3Ftid%3D77731%26page%3D1" \t "_blank)  
**1-Quartz2D**，的绘图总是发生在图形环境（Graphics Context）中。视图会在调用drawRect:方法进行绘图之前，创建好图形环境，  
    通过UIGraphicsGetCurrentContext函数来获得这个图形环境。  
    如果直接在图像或者PDF上绘图，需要调用CGBitmapContextCreate或者CGPDFContextCreate函数来创建图形环境。  
    Quartz2D绘图的基础元素是路径。路径可以是一些基础几何形状，也可以是这些几何形状的组合。  
    当需要创建一条路径时，应当调用CGContextBeginPath函数；  
    当需要将路径绘制的起点移动到一个位置时，应当调用CGContextMoveToPoint函数；  
    当想绘制一条线段时，应当调用CGContextAddLineToPoint。 CTM（当前变换矩阵）将绘图从用户空间映射到设备控件。  
    当图形环境刚刚创建时，CTM初始化为一个单位矩阵。  
    对CTM进行平移变换应当调用CGContextTranslateCTM函数，进行旋转变换应当调用CGContextRotateCTM函数，进行缩放变换应当调用CGContextScaleCTM函数。  
**2-UIBezierPath**  
使用UIBezierPath绘图的好处：  
    -采用系统坐标系  
    -在Core Graphics的基础上封装，具有Core Graphics的主要功能，无需考虑图形环境。  
    UIBezierPath\* pathLines = [UIBezierPath bezierPath];  
    [pathLines moveToPoint:point1]; // 移动到point1位置  
    [pathLines addLineToPoint:point2]; // 画一条从point1到point2的线  
    pathLines.lineWidth = 5.0; // 线宽  
    ［UIColor redColor] set]; // 颜色设置  
    [pathLines stroke]; // 开始描绘  
    在同一UIBazierPath中，只能采用相同的颜色和线宽。  
    可以调用closePath，从当前点画一条直线到当前子路径的初始点。如果要填充路径，应当使用fill方法，此方法会填充路径中得所有封闭子路径。  
**三.容器**  
    -NSArray和NSMutableArray  
        +array：创建一个空数组  
        +arrayWithArray：从另一个数组创建新的数组  
        +arrayWithContentsOfFile：读文件创建数组  
        +arrayWithObject：创建一个数组，其中包含一个给定对象  
        +arrayWithObjects  
        +arrayWithObjects:count: 从C数组创建  
        -containsObject：是否包含一个元素  
        -count：数量  
        -lastObject：返回最后一个  
        -objectAtIndex：返回某一个  
        -objectsAtIndexes：返回一组，类型为NSArray  
        -indexOfObject：返回对象索引  
        -arrayByAddingObject：原数组最后加一个对象，产生一个新的数组  
        -arrayByAddingObjectsFromArray：在原数组的最后添加另一个数组中的所有对象，产生一个新数组  
        -subarrayWithRange：抽取原数组中得一部分，产生一个新的数组  
        -isEqualToArray：比较两个数组是否相同  
        -writeToFile:atomically：保存数组至一个文件  
        -writeToURL:atomically：保存数组至一个URL  
        -addObject：在数组最后添加一个对象  
        -addObjectsFromArray：在原数组最后添加另一个数组的全部对象  
        -insertObject:atIndex：向原数组制定位置添加一个对象  
        -insertObjects:atIndexes：向原数组中一系列位置添加一系列对象  
        -removeAllObjects：移除数组中的全部对象  
        -removeLastObject：移除数组中最后一个对象  
        -removeObject：移除一个对象  
        -removeObjectAtIndex：移除位于指定位置的对象  
        -removeObjectsAtIndexes：移除位于一系列位置的对象  
        -replaceObjectAtIndex:withObject用给定对象替换位于指定位置的对象  
        -replaceObjectsAtIndexes:withObjects：多对象，多位置版本  
        -setArray：用另一个数组中的所有对象来替换当前数组中的所有对象  
    -NSDictionary与NSMutableDictionary  
        +dictionary  
        +dictionaryWithContentsOfFile  
        +dictionaryWithContentsOfURL  
        +dictionaryWithDictionary  
        +dictionaryWithObject:forKey  
        +dictionaryWithObjects:forKeys  
        +dictionaryWithObjectsAndKeys  
        -count：数量  
        -allKeys：返回一个数组，包含字典中的所有关键字。  
        -allKeysForObject：返回一个数组，包含所有对应到给定对象的关键字。  
        -valueForKey：通过字符串查找数值。  
        -writeToFile:atomically  
        -writeToURL:atomically  
        -setValue:forKey： 加键-值对  
        -addEntriesFromDictionary： 添加另一个字典中得所有条目  
        -setDictionary：将原字典中条目设置为另一个字典中得所有条目  
        -removeObjectForKey：移除一个关键字的对应条目  
        -removeAllObjects：移除所有条目  
        -removeObjectsForKeys：移除一系列关键字的对应条目  
  
**四.文件系统**  
**1- 概述**  
     bundle（沙盒）：一个应用只能在自己的沙盒中读取或存储文件，而不能操作其他沙盒中的文件。好处：保护应用的隐私不受侵犯，保护系统文件的安排，应用删除后相关文件也能被全部清除。  
    每个应用都有自己的沙盒，沙盒的根目录的获取方法是NSHomeDirectory()。  
    根目录下的内容：  
         -.app文件，实际上是一个文件夹，包含了可执行文件、Nib文件、图片资源、plist等。XCode中能看到的资源，及编译后的可执行文件，都封装在app中。  
        -Documents文件夹，用于存放你的应用所产生的数据，该文件夹可通过iTunes备份，可以存储游戏进度等。  
        -Library文件夹，用于存放用户偏好和临时文件。  
        -tmp文件夹是系统的中转站。  
**2-文件管理器**  
    NSFileManager，defaultManager()返回一个文件管理器的单例（多线程下不安全）。init()，在多线程编程中应尽量使用init()。  
    代理方法：-fileManager:shouldRemoveItemAtPath和-fileManager:shouldRemoveItemAtURL在移除操作之前被调用。  
    -removeItemAtPath:error：删除位于指定路径的文件、连接、目录（及其所有子目录、文件）。  
    -removeItemAtURL:error：同上。  
    -contentOfDirectoryAtPath：查找所有位于给定路径的子路径和文件。返回值为一个数组，其中包含了NSString对象。查找只在当前目录进行，不会进入下一层目录。  
    -subpathsAtPath：查找给定路径下的所有子路径。深度查找，不限于当前层，也会查找package的内容。  
    -fileExistsAtPath：判断文件是否位于一个路径下面。  
    -isReadableFileAtPath：查询文件的可读性  
    -isWritableFileAtPath：可写性  
    -isExecutableFileAtPath：查询文件的可执行性  
    -isDeletableFileAtPath：可删除性  
**3-NSString的路径功能**  
    -pathWithComponent：参数是一堆components构成的数组，返回的路径是由这些components连接而成的路径字符串，相邻components之间用/隔开。  
    -pathComponents：返回一个数组，包含路径中的components。  
    -fileSystemRepresentation：返回C字符串  
    -isAbsolutePath：判断是否为绝对路径  
    -pathExtension：返回文件的扩展名，没有的就返回空字符串  
    -stringByAppendingPathComponents ：向现有路径添加一个component。斜杠/会被自动加上  
    -stringByAppendingPathExtension：向现有路径加上文件的扩展名  
    -stringByDeletingLastPathComponent：移除最后一个路径component  
    -stringByDeletingPathExtension：删除路径扩展名  
    -stringByAppendingPaths：参数为一个数组，此方法将数组中的字符串对象作为路径一次添加到源字符串后面。  
例子：

1. NSString \*homePath = NSHomeDirectory();
2. NSString \*docPath = [homePath stringByAppendingFormat:@"/Documents"];

**五.UIActionSheet 按钮列表**  
**六.旋转**  
    **1- interfaceOrientation**属性查询当前的界面方向。  
    **2- rotatingHeaderView**方法来查询当前导航视图。  
    **3- rotatingFooterView**来查询当前的标签条。  
**七.快速拨打电话接口**  
openURL能帮助你运行Maps，SMS，Browser,Phone甚至其他的应用程序。这是Iphone开发中我经常需要用到的一段代码，它仅仅只有一行而已。

1. ［UIApplication sharedApplication] openURL:[NSURL URLWithString:@"tel://8004664411"］;

这个程序通过基础的协议支持拨打电话的功能。  
摘自：[http://blog.csdn.net/smilelance/article/details/6213125](http://www.douban.com/link2?url=http%3A//blog.csdn.net/smilelance/article/details/6213125" \t "_blank)  
**八.区分设备类型的方法**  
打电话功能只有iPhone支持，对于其他设备对应按钮应该禁用。  
**1.** 用[UIDevice currentDevice].model，这个返回的是一个NSString，你可以做如下判断就能知道设备是iPad还是iPhone.

1. **if** (［UIDevice currentDevice].model rangeOfString:@"iPad"].location != NSNotFound) {
2. NSLog(@"This is an iPad!");
3. }

**2.** 用UI\_USER\_INTERFACE\_IDIOM()方法，这是系统定义的一条宏。使用方法也很简单。

1. **if** (UI\_USER\_INTERFACE\_IDIOM() ＝＝ UIUserInterfaceIdiomPad) {
2. NSLog(@"This is an iPad!");
3. }

摘自：[http://blog.sina.com.cn/s/blog\_3f88614f0100sdcm.html](http://www.douban.com/link2?url=http%3A//blog.sina.com.cn/s/blog_3f88614f0100sdcm.html" \t "_blank)  
  
**九.并发编程**  
   **1-GCD-Grand Central Dispatch**，GCD使用队列来管理多个任务，而每个任务是以块（其定义类似于函数）来描述的。GCD从队列中读取这些要完成的任务，然后其多核执行 引擎负责将这些任务分配给它所管理的线程。和手动线程管理相比，GCD能极大地优化线程管理，并极大地减少创建和销毁线程所带来的额外开销。除了在多核系 统上得性能改进之外，GCD还能帮助提高应用的响应速度，并让代码更为干净，易于维护。  
    **2-NSOperationQueue-操作队列**，可以极大地减少开发者在并发编程中的负担，并且操作队列有一套Objective-C的API，使用起来方便。  
    基本操作单位-NSOperation，操作。  
    操作是封装了一个与任务有关的代码与数据的抽象类，需要继承NSOperation，并且描述相关的任务。操作对象是一次性的，它只能执行一次，下一次，需要创建一个新的实例。  
    可以通过addDependency:方法为操作添加相互依赖性。假设操作A依赖于操作B，如果操作B没有结束，操作A就不会开始执行。完成是指 执行完，或者取消了操作。有时需要对操作是否成功作出判断。通过removeDependency:方法来移除操作的相互依赖性。  
    操作的一个属性是isConcurrent。当使用操作队列来管理这些操作时，isConcurrent应当设置为NO，这时应当重载 NSOperation的main函数（在类中保存controller对象-weak，并在main中调用controller的方法）。 isReady属性用于描述操作是否已经可以开始执行，isExecuting属性表示操作是否正在执行当中，isFinished属性表示操作是否顺利 完成，isCancelled属性则表示操作已经取消。  
    可以在操作运行中取消它的运行，只需要给它发送cancel消息即可。在继承NSOperation类时，也需要支持cancel方法。  
    如果只需要简单的操作，不想继承NSOperation类，可以使用NSInvocationOperation类，这个类中的-initWithTarget:selector:object:方法能帮助你方便地选择人物的对象和相应的功能。  
    操作的执行顺序取决于各自的优先级和相互之间的依赖性。  
    操作队列有两种方式来执行其中的操作。第一种是开辟新线程来运行这些操作，第二种则是通过GCD来运行这些操作。但是无论如何，都不需要手动管理 线程。尽量使用操作队列，只有在操作队列的效率明显不足时才转向GCD。因为操作队列是封装在GCD的基础之上的，虽然增加了一些额外开销，但是也具有更 加完善的功能。例子是TestNSOperation。

**十.忙碌指示器**  
    UIActivityIndicator。

1. UIActivityIndicatorView \*activityIndicatior = ［UIActivityIndicatorView alloc] initWithActivityIndicatorStyle: UIActivityIndicatorViewStyleWhiteLarge];
2. activityIndicator.center = CGPointMake(512, 384);
3. [self.view addSubview: activityIndicator];
4. [activityIndicator startAnimating];
5. [activityIndicator stopAnimating];
6. [activityIndicator removeFromSuperView];

**十一.媒体播放**  
  
**1- 音频**  
   **[1]** 音乐往往是存储在iPod曲库（注意位置）中的，可以通过媒体选择器（media picker）或者媒体查询（media query）读取，然后用音乐播放器MPMusicPlayerController播放。

1. MPMusicPlayerController \*musicPlayer = [MPMusicPlayerController applicationMusicPlayer];
2. [musicPlayer setShufleMode: MPMusicShuffleModeSongs];
3. [musicPlayer setRepeatMode: MPMusicRepeatModeAll];
4. [musicPlayer setQueueWithQuery: [MPMediaQuery songsQuery］;
5. [musicPlayer play];

    applicationMusicPlayer返回的播放器，在你的应用中播放音乐。它不会影响到iPod播放器，也不能从iPod播放器重获取信息。  
    iPodMusicPlayer返回的是iPod播放器，在你推出应用后，所有的设置都会影响到之后设备上的iPod播放器。  
    获得音乐播放器后，需要为它设置一个播放队列。可以用setQueueWithQuery:放方法，通过媒体查询MPMediaQuery来设置 播放队列，也可以用setQueueWithItemCollection:方法，通过MPMdiaItemCollection来设置播放队列。  
    重复模式repeatMode可以设置为不重复、重复当前曲目、或整个播放列表；乱序播放shuffleMode可以设置为不乱序、乱序播放曲目或乱序播放专辑；音量volume的设置与音频播放器一样。  
    skipToNextItem跳到下一首，skipToPreviousItem跳到上一首，skipToBegin跳到第一首。  
    对应的宏都是以MPMusic开头。  
    **[2]** 利用系统声音服务来播放短暂音效（时长30秒以内），并震动：

1. AudioServicesPlaySystemSound(kSystemSoundID\_Vibrate);

    播放指定音效：

1. NSURL \*fileURL = [NSURL fileURLWithPath: path isDirectory: NO];
2. // 创建音效ID
3. SystemSoundID soundID;
4. AudioServiceCreateSystemSoundID((CFURLRef) fileURL, &soundID);
5. // 播放声音
6. AudioServicesPlaySystemSound(soundID);

**[3]** 音频播放器  
    没有时长限制

1. NSURL \*fileURL = [NSURL fileURLWithPath: path isDirectory: NO];
2. // 用URL来初始化音频播放器-播放的不是iPod曲库中的音乐
3. AVAudioPlayer\* player = ［AVAudioPlayer alloc] initWithContentsOfURL: fileURL error: NO];
4. // 准备播放
5. [player prepareToPlay];
6. // 设置代理
7. [player setDelegate: self];

    方法：play、pause、stop。可以通过playing属性查询播放器是否正在播放当中，可以通过volume属性来修改和查询播放器的播放增益（从0.0到1.0），可通过setting属性查询播放器其他设置。  
  
    duration表示音频的时间长度， currentTime表示当前播放到的时间。播放结束后可以通过代理方法audioPlayerDidFinishPlaying:来处理播放后设置。   
**2- 视频**  
    视频播放可以采用网页视图（UIWebView）进行嵌入式播放（能播放YouTube视频），或者采用电影播放器（MPMoviePlayerController）进行播放。  
   **[1]** 电影播放器

1. MPMoviePlayerController \*player = ［MPMoviePlayerController alloc]initWithContentURL: url];
2. // 设置播放器的大小，并将其加入视图中
3. [player.view setFrame: rectFrame];
4. [self.view addSubView: player.view];
5. 播放器的背景视图backgroundView。
6. 全屏[player setFullscreen: YES animated: YES];
7. 播放另一个影片[player setContentURL: newURL];
8. [player requestThumbnailImagesAtTimes:arrayTimes timeOption:MPMovieTimeOptionNearestKeyFrame]; // 表示播放器不会在你所指定的时间去截取预览，而是在绝对时间的附近几帧中寻找效果最好的帧做为预览。

scalingMode规定了影片的缩放模式。

initialPlaybackTime用来控制视频开始播放的时间，单位是秒。

如果视频源在网络上，那么需要正确设置服务器端的mimeType。  
**十二.运行环-runloop**  
    运行环作用于一个iOS应用的整个生命周期。它负责监视各种输入事件，并且在合适的时候对这些输入进行分配。应用的每一个线程都有且仅有一个运行环。你自己不需要创建也不需要销毁运行环，但是可以通过currentRunLoop方法来获取当前的运行环。  
  
**十三.定时器**  
    由于运行环机制，定时器的精度不高，只能用于一般性延时。  
    例子：拼图游戏，DeskViewController.m。  
    NSObject类的定时方法。  
    performSelector: withObject: afterDelay: 运行方法，参数，时间（秒）。  
    performSelectorOnMainThread: withObject: waitUntilDone: 在主线程中，运行参数selector所指定的方法，如果waitUntilDone参数为YES，那么当前线程会被阻拦，直到selector运行完。  
    performSelector: onThread: withObject: waitUntilDone:同上，但不一定在主线程中运行。  
    performSelectorInBackground: withObject: 开启一个新线程，用于运行selector方法，selector方法应负责线程的初始化。  
    cancelPreviousPerformRequestsWithTarget:取消与一个目标相关的所有计划好的动作。  
    cancelPreviousPerformRequestsWithTraget: selector: object只取消特定的计划动作。  
**十四.随机数**  
0~1之间随机数

1. CG\_INLINE **float** genRandomNum()
2. {
3. **return** (**float**)arc4random/ARCRANDOM\_MAX;
4. }

**十五.加速度传感器**  
    获得加速度传感器读数：

1. UIAccelerometer \*accelerometer = [UIAccelerometer shareAccelerometer]; // 单例
2. accelerometer.delegate = self;
3. accelerometer.updateInterval = 1.0f / 10.0f; // 传感器更新频率，普通游戏10~30HZ，虚拟现实类游戏，30~60HZ
4. accelerometer.x, accelerometer.y, accelerometer.z // 传感器读数

    若不想再接受传感器读数，可将代理设置为nil

1. [UIAccelerometer shareAccelerometer].delegate = nil;
3. #define alpha 0.05
4. // 获得重力引起的加速度读数：
5. - (**void**)accelerometer: (UIAccelerometer \*)accelerometer didAccelerate: (UIAcceleration \*)acceleration
6. {
7. // 简单低通滤波器
8. accelX = (acceleration.x \* alpha) + (accelX) \* (1.0 - alpha);
9. accelY = (acceleration.y \* alpha) + (accelY) \* (1.0 - alpha);
10. accelZ = (acceleration.z \* alpha) + (accelZ) \* (1.0 - alpha);
11. }
13. // 获得动作引起的加速度读数：
14. - (**void**)accelerometer: (UIAccelerometer \*)accelerometer didAccelerate: (UIAcceleration \*)acceleration
15. {
16. // 减去低通滤波输出，得到高通滤波输出
17. accelX = acceleration.x - ((acceleration.x \* alpha) + (accelX) \* (1.0 - alpha));
18. accelY = acceleration.y - ((acceleration.y \* alpha) + (accelY) \* (1.0 - alpha));
19. accelZ = acceleration.z - ((acceleration.z \* alpha) + (accelZ) \* (1.0 - alpha));
20. }

**十六.地图与连接服务器**  
**[1]** 添加框架MapKit.framework。使用MKMapView来呈现地图。注意应当直接使用此类，而不是继承之。如果希望在MKMapView类之上添加功能，可以使用MKMapViewDelegate协议。  
    初始化：

1. MKMapView \*mapView = ［MKMapView alloc] initWithFrame: rect];

    初始化之后并不直接显示，还需要指定显示的地图区域：

1. CLLocationCoordinate2D coordinate;
2. coordinate.latitude = latitudeValue; // 纬度
3. coordinate.longtitude = longtitudeValue; // 精度
4. mapView.region = MKCoordinateRegionMakeWithDistance(coordinate, width, height); // 指定显示区域，width和height单位都是米

    之后可以通过addSubview添加地图。  
    属性：showsUserLocation-为YES，系统会持续跟踪用户的位置  
          userLocationVisible-为YES，将显示用户所在位置  
    显示地图之后，常常希望在地图上添加标注，这需要创建一个类，并实现MKAnnotation协议，这个类叫做标注对象。标注对象往往实现 setCoordinate:方法来设置其坐标。在地图视图上，可以设置标注对象的坐标，然后添加进去，这样地图上就会出现一个标注。代理方法title 和subtitle能够在标注上显示标题和副标题。

1. // 初始化
2. mapView = ［MKMapView alloc] initWithFrame: CGRectMake(100, 100, 550, 700)];
3. mapView.showsUserLocation = TRUE;
4. mapView.mapType = MKMapTypeStandard;
5. mapView.delegate = self;
6. // 设置坐标
7. CLLocationCoordinate2D coordinate;
8. coordinate.latitude = 37.31;
9. coordinate.longtitude = -122.03;
10. mapView.region = MKCoordinateRegionMakeWithDistance(coordinate, 4000, 6000); // 4000米宽，6000米高的区域
11. [self.view insertSubview: mapView atIndex: 0];

**十七.通过storyboard来初始化一个controller**

1. CBigDesignImageViewController \*imageViewController = [self.storyboard instantiateViewControllerWithIdentifier:@"BigImageController"];

   BigImageController是要在storyboard中设置的Identifier属性。  
  
**十八.给UIView子类设置阴影，包括各种控件**  
   **1** 导入QuartzCore framework  
  **2** #import <QuartzCore/QuartzCore.h>  
    **3** 编码：

1. ［imgView layer] setShadowOffset:CGSizeMake(5, 5)]; // 阴影的范围
2. ［imgView layer] setShadowRadius:2]; // 阴影扩散的范围控制
3. ［imgView layer] setShadowOpacity:1]; // 阴影透明度
4. ［imgView layer] setShadowColor:[UIColor brownColor].CGColor]; // 阴影的颜色

**十九.设置UIScrollView滚动速度**

1. // 自动滚动太快，效果不好，这里把动画设置慢点，注意下面要直接赋值contentOffset，不要用带animated参数的函数，否则动画会出问题，因为两处都是动画效果。
2. [UIScrollView animateWithDuration:1.0f
3. delay:0
4. options:UIViewAnimationCurveLinear
5. animations:^{
6. scrollView.contentOffset = CGPointMake(0, 0);
7. }
8. completion:^(**BOOL** finished){}
9. ];
10. // 如果在减速滚动过程中，按了刷新按钮，执行上面的动画，会出现重置的位置，y不是0的情况，这里再调用一次，滚动到0。
11. [scrollView setContentOffset:CGPointMake(0, 0) animated:YES];

**二十.EXC\_BAD\_ACCESS**  
Here’s the advice I generally give to developers when you hit an EXC\_BAD\_ACCESS error:  
-Set the NSZombieEnabled argument in your executable options, which sometimes helps narrow down the cause  
-Run with Apple Instruments such as Leaks to look for memory issues  
-Set a breakpoint in your code and step through until you narrow down where it’s crashing  
Tried and true “comment out code till it works” then backtrack from there :]  
  
Xcode4 下设置 NSZombieEnabled 的方法：   
你可以点击 Xcode4 菜单 Product -> Edit Scheme -> Arguments, 然后将点击”加号”, 将 NSZombieEnabled 参数加到 Environment Variables 窗口中, 后面的数值写上 ”YES”.   
或者在 Xcode4 菜单 Product -> Edit Scheme -> Diagnostics 设置窗口中直接勾上 Enable Zombie Objects 即可，Xcode 可用 cmd＋shift＋< 进到这个窗口。   
Xcode4 已经考虑到了现在的要求，所以提供了更便捷的设置的方式，你也可以在这个窗口中设置其他一些参数，你肯定能由此获得更多的帮助信息。

好了，看完这篇iOS学习笔记整理，不知道对你是不是有所启发