

# 10 iPhone Memory Management Tips

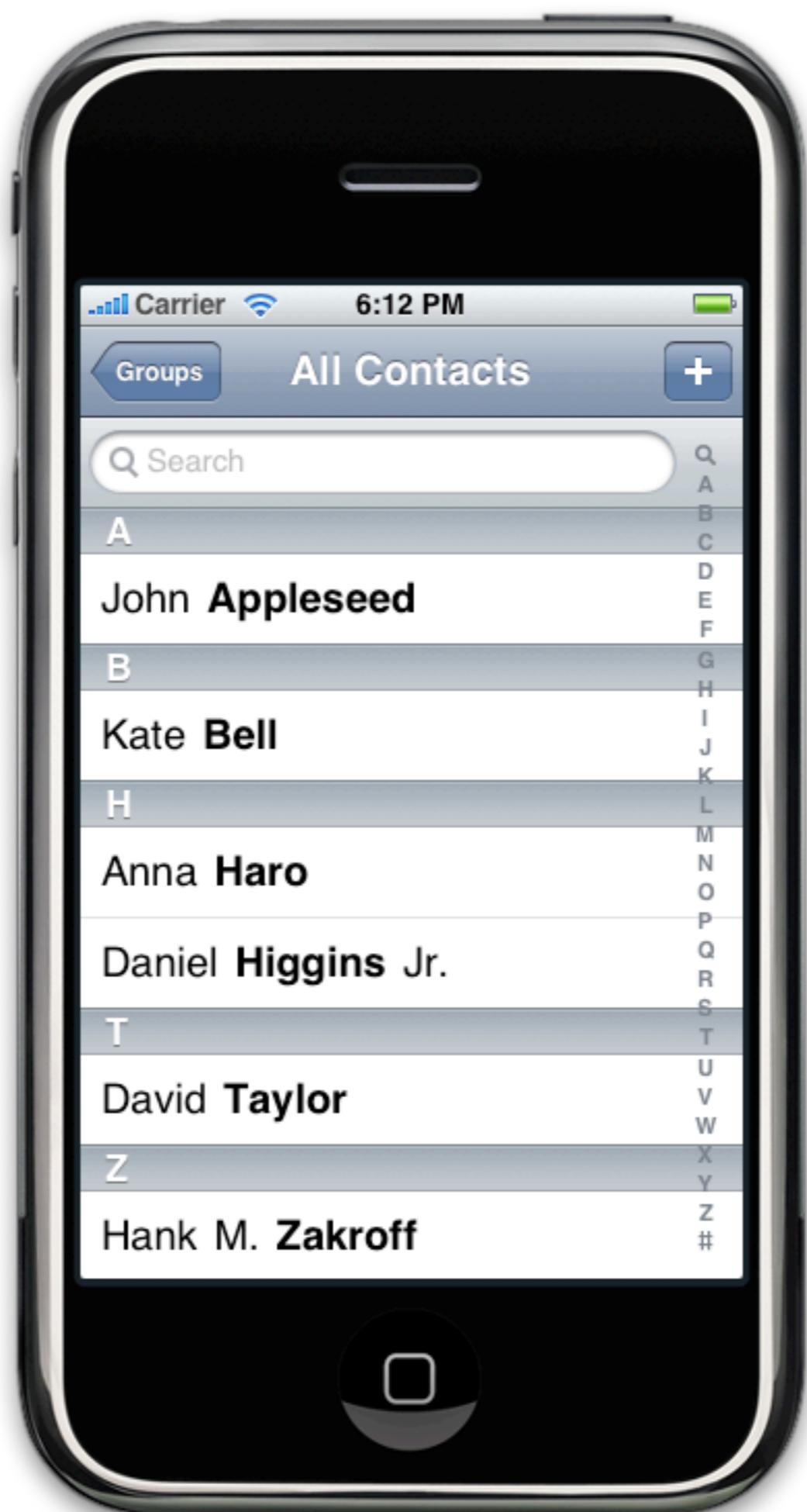
Adrian Kosmaczewski

January 28th, 2009

[http://kosmaczewski.net/  
2009/01/28/  
10-iphone-memory-management-tips/](http://kosmaczewski.net/10-iphone-memory-management-tips/)



# Facts



**128 MB RAM**





**I 23 MB RAM**

**~ 40 MB for you!**

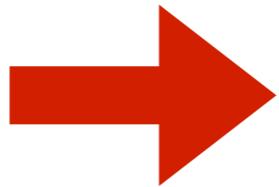
# Garbage Collection



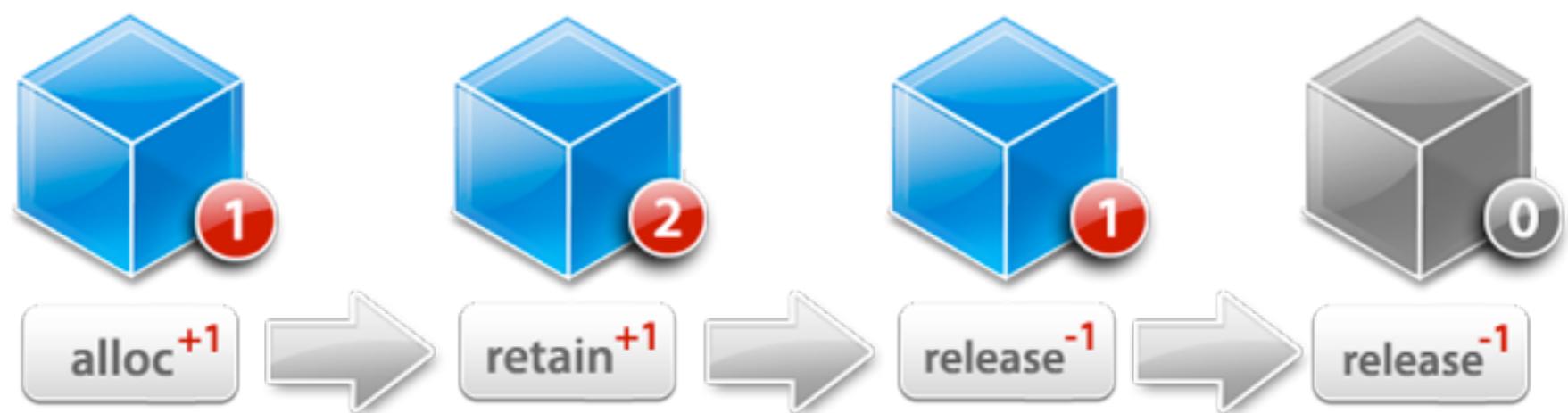
[ alloc | copy | retain ]

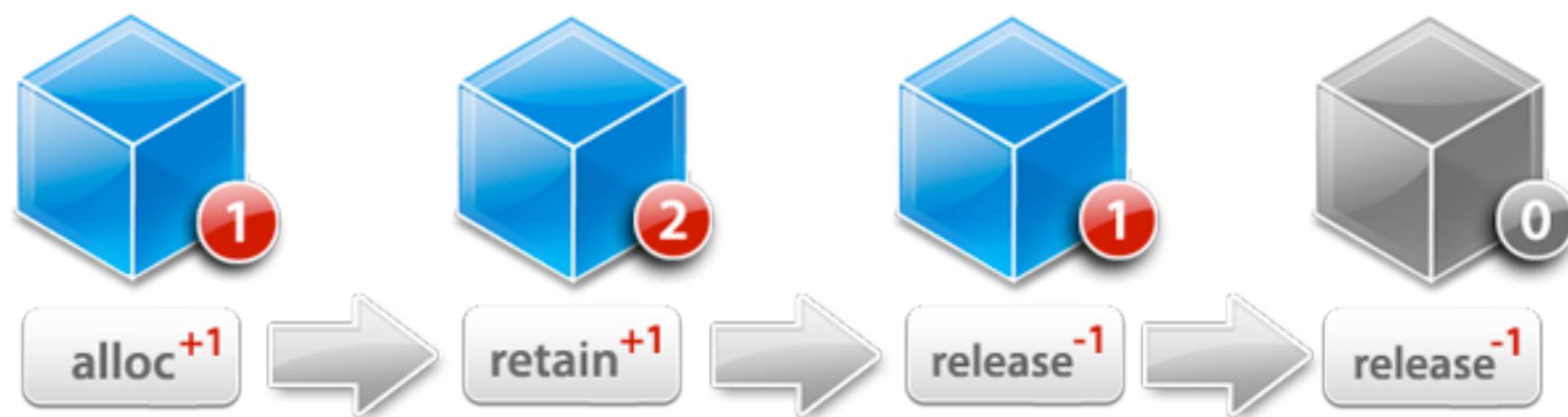
[ release ]

[ alloc | copy | retain ]



[ release ]





[http://cocoadevcentral.com/d/learn\\_objectivec/](http://cocoadevcentral.com/d/learn_objectivec/)

# Objects on the heap

```
// C++ automatic object on the stack  
// Memory freed when out of scope  
  
std::string name("Adrian");  
  
// C++ object on the heap  
  
std::string *name = NULL;  
  
name = new std::string("Adrian");  
  
delete name;
```

... autorelease];

# Virtual Memory





# Tips



# I) Respond to Memory Warnings

```
- (void)didReceiveMemoryWarning  
{  
    [super didReceiveMemoryWarning];  
}
```

```
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)app
{
    [[ImageCache sharedImageCache] removeAllImagesInMemory];
}
```

```
NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
[center addObserver:self
    selector:@selector(whatever:)
    name:UIApplicationDidReceiveMemoryWarningNotification
    object:nil];
```



## 2) Avoid Autoreleased Objects

(when possible)

```
// Instead of  
NSString *string = [NSString  
    stringWithFormat:@"value = %d", var];  
  
// use  
NSString *string = [[NSString alloc]  
    initWithFormat:@"value = %d", var];  
  
...  
[string release];
```

```
NSAutoreleasePool *pool =
[[NSAutoreleasePool alloc] init];
for (id item in array)
{
    id anotherItem = [item create];
    [anotherItem doSomethingWithIt];
}
[pool release];
```



# 3) Load Lazily and Reuse

```
- (void)tableView:(UITableView *)tableView
    didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    Item *item = [items objectAtIndex:indexPath.row];
    if (detailController == nil)
    {
        detailController = [[DetailController alloc] init];
    }
detailController.item = item;
    [self.navigationController
        pushViewController:detailController
            animated:YES];
}
```

`viewWillAppear:`

`viewDidDisappear:`

**tabBarController:didSelectViewController:**



4) Avoid UIImage's  
imageNamed:

[http://www.alexcurylo.com/blog/2009/01/13/  
imagenamed-is-evil/](http://www.alexcurylo.com/blog/2009/01/13/imagenamed-is-evil/)

```
@implementation UIImage (AKLoadingExtension)
+ (UIImage *)newImageFromResource:(NSString *)filename
{
    NSString *imageFile = [[NSString alloc] initWithFormat:@"%@/%@",
                           [[NSBundle mainBundle] resourcePath], filename];
    UIImage *image = nil;
    image = [[UIImage alloc] initWithContentsOfFile:imageFile];
    [imageFile release];
    return image;
}
@end
```

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

@interface ImageCache : NSObject
{
@private
    NSMutableArray *keyArray;
    NSMutableDictionary *memoryCache;
    NSFileManager *fileManager;
}

+ (ImageCache *)sharedImageCache;

- (UIImage *)imageForKey:(NSString *)key;
- (BOOL)hasImageWithKey:(NSString *)key;
- (void)storeImage:(UIImage *)image withKey:(NSString *)key;
- (BOOL)imageExistsInMemory:(NSString *)key;
- (BOOL)imageExistsInDisk:(NSString *)key;
- (NSUInteger)countImagesInMemory;
- (NSUInteger)countImagesInDisk;
- (void)removeImageWithKey:(NSString *)key;
- (void)removeAllImages;
- (void)removeAllImagesInMemory;
- (void)removeOldImages;

@end
```

```
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)app
{
    [[ImageCache sharedImageCache] removeAllImagesInMemory];
}
```

<http://kosmaczewski.net/projects/iphone-image-cache/>



# 5) Build Custom Table Cells and Reuse Them Properly

```
- (UITableViewCell *)tableView:(UITableView *)tableView
    cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    Item *item = [items objectAtIndex:indexPath.row];
    static NSString *identifier = @"ItemCell";

    ItemCell *cell = (ItemCell *)[tableView
        dequeueReusableCellWithIdentifier:identifier];
    if (cell == nil)
    {
        cell = [[[ItemCell alloc] initWithIdentifier:identifier]
                autorelease];
    }
    cell.item = item;
    return cell;
}
```



# 6) Override Setters Properly

```
@interface SomeClass
{
@private
    NSArray *items;
    NSString *name;
    id<SomeProtocol> delegate;
}

@property (nonatomic, retain) NSArray *items;
@property (nonatomic, copy) NSString *name;
@property (nonatomic, assign) id<SomeProtocol> delegate;

@end
```

```
- (void)setItems:(NSArray *)obj
{
    if (obj == items)
    {
        return;      // (merci Marco! :)
    }
    [items release];
    items = nil;
    items = [obj retain];

    if (items != nil)
    {
        // create the internal
        // structure of the cell
        // if not present,
        // and change the
        // widget values
    }
}
```

```
- (void)setName:(NSString *)obj
{
    [name release];
    name = nil;

    // I always copy NSStrings!
    name = [obj copy];

    if (name != nil)
    {
        // create the internal
        // structure of the cell
        // if not present,
        // and change the
        // widget values
    }
}
```

```
- (void)setDelegate:(id<SomeProtocol>)obj
{
    // do not retain!
    // This is an "assign" property
    delegate = obj;

    if (delegate != nil)
    {
        // create the internal
        // structure of the cell
        // if not present,
        // and change the
        // widget values
    }
}
```



# 7) Beware of Delegation

```
@interface SomeClass <WidgetDelegate>
{
    @private
        Widget *widget;
}
@end
```

```
@implementation SomeClass

- (id)init
{
    if (id = [super init])
    {
        widget = [[Widget alloc] init];
        widget.delegate = self;
    }
    return self;
}
```

```
- (void)dealloc
{
    // widget might be retained by someone else!
    widget.delegate = nil;
    [widget release];
    widget = nil;
}

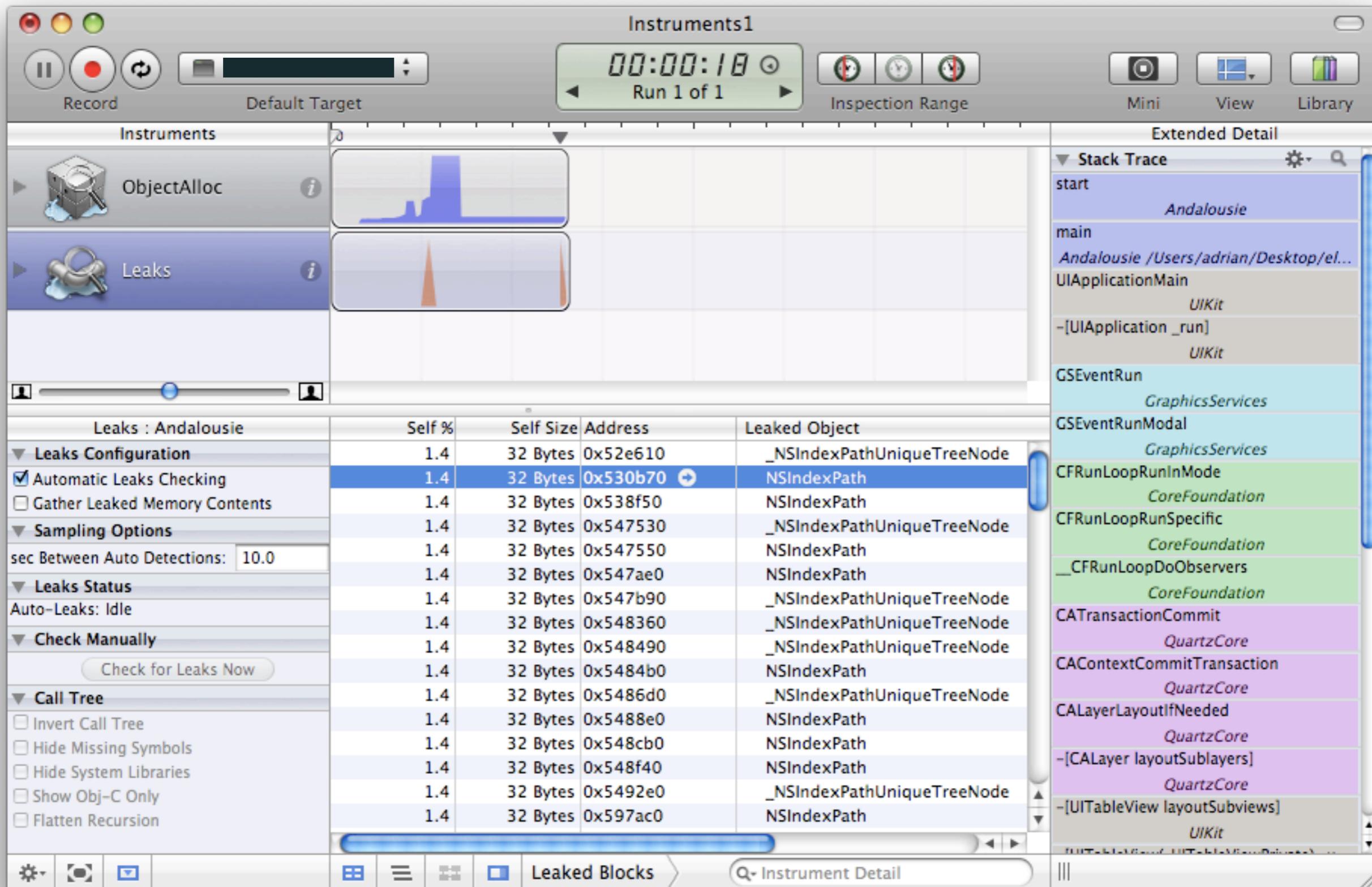
# pragma mark -
# pragma mark WidgetDelegate methods

- (void)widget:(Widget *)obj method:(BOOL)value
{
    // and here something happens...
}

@end
```



# **8) Use Instruments**





# 9) Use a Static Analysis Tool

<http://clang.llvm.org/StaticAnalysis.html>

```
scan-build -k -V xcodebuild -configuration Debug
```

client - scan-build results

http://127.0.0.1:8181/

client - scan-build results

User:	adrian@Socrates.local
Working Directory:	/Users/adrian/Desktop/trunk/client
Command Line:	xcodebuild -configuration Debug
Date:	Wed Jan 28 18:14:43 2009
Version:	checker-0.139 (2009-01-12 17:38:46)

### Bug Summary

Results in this analysis run are based on analyzer build checker-0.139.

Bug Type	Quantity	Display?
All Bugs	9	<input checked="" type="checkbox"/>
Dead Store		
dead nested assignment	1	<input checked="" type="checkbox"/>
Memory (Core Foundation/Objective-C)		
[naming convention] leak of returned object	1	<input checked="" type="checkbox"/>
leak	6	<input checked="" type="checkbox"/>
Other		
missing -dealloc	1	<input checked="" type="checkbox"/>

### Reports

Bug Group	Bug Type	File	Line	Path Length	View	Report	Open
...	...	...	...	...	...	...	...

client - scan-build results

http://127.0.0.1:8181/ Google

Memory (Core Foundation/Objective-C)	
[naming convention] leak of returned object	1 <input checked="" type="checkbox"/>
leak	6 <input checked="" type="checkbox"/>
Other	
missing -dealloc	1 <input checked="" type="checkbox"/>

## Reports

Bug Group	Bug Type	File	Line	Path Length			
Memory (Core Foundation/Objective-C)	[naming convention] leak of returned object	[REDACTED]	125	3	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Dead Store	dead nested assignment	[REDACTED]	273	1	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Memory (Core Foundation/Objective-C)	leak	[REDACTED]	38	8	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Memory (Core Foundation/Objective-C)	leak	[REDACTED]	339	2	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Memory (Core Foundation/Objective-C)	leak	[REDACTED]	306	5	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Memory (Core Foundation/Objective-C)	leak	[REDACTED]	34	7	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Memory (Core Foundation/Objective-C)	leak	[REDACTED]	204	6	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Memory (Core Foundation/Objective-C)	leak	[REDACTED]	339	2	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>
Other	missing -dealloc	[REDACTED]	65	1	<a href="#">View Report</a>	<a href="#">Report Bug</a>	<a href="#">Open File</a>

Controller.m

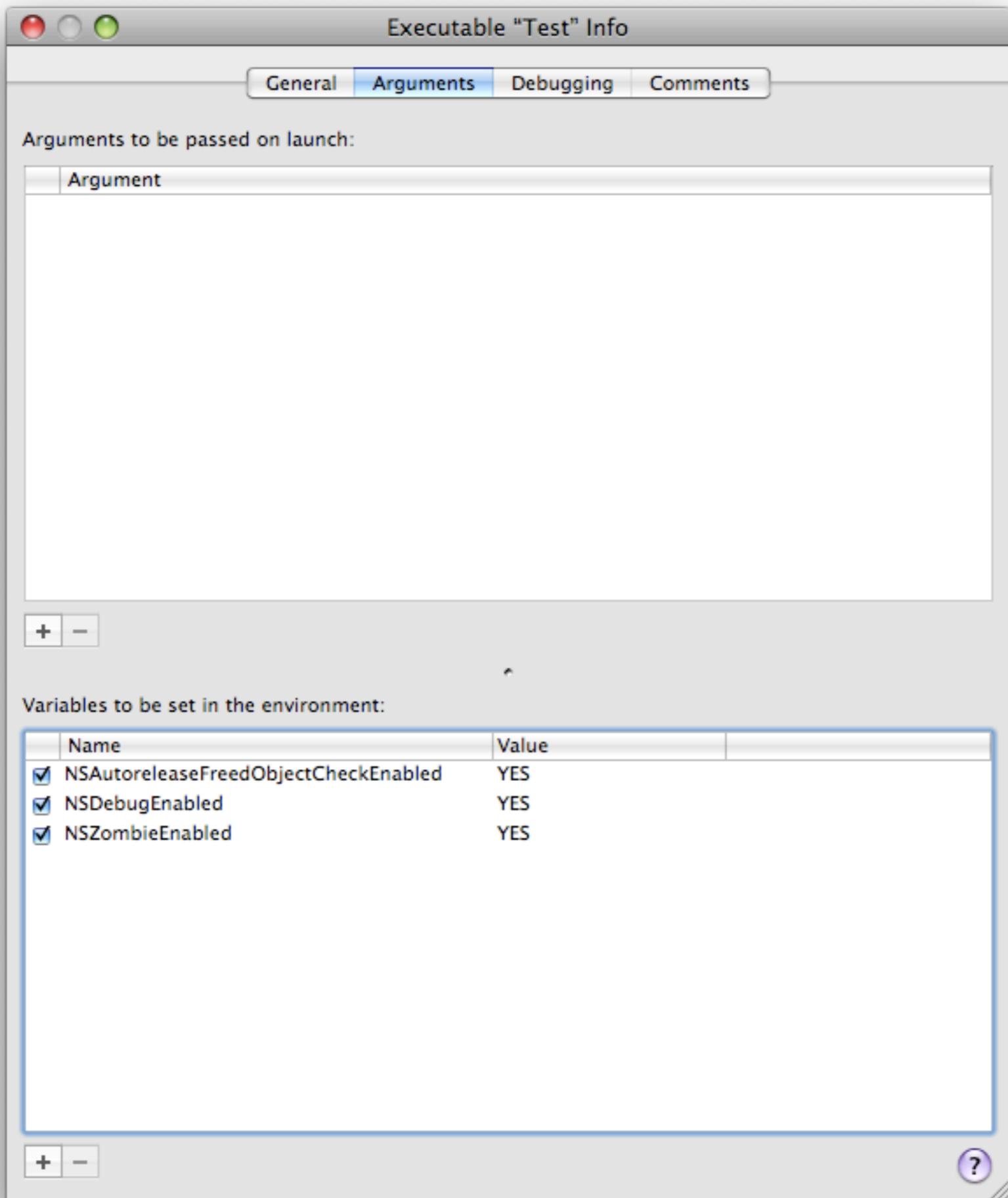
http://127.0.0.1:8181/report-4slmBA.html#EndPath

```
331 {  
332     [████████ removeAllObjects];  
333     [self.tableView reloadData];  
334  
335     NSInteger selectedSegmentIndex = [segmentedControl selectedSegmentIndex];  
336     NSNumber *newId = [NSNumber numberWithInt:selectedSegmentIndex];  
  
            [1] Method returns an object with a +1 retain count (owning reference).  
337     NSArray *keys = [[NSArray alloc] initWithObjects:@"segmentIndex", nil];  
338     NSArray *values = [[NSArray alloc] initWithObjects:newId, nil];  
  
            [2] Object allocated on line 337 and stored into 'keys' is no longer  
            referenced after this point and has a retain count of +1 (object leaked).  
339     NSDictionary *info = [NSDictionary dictionaryWithObjects:values forKeys:keys];  
340  
341     // This notification is caught by the ██████████ class  
342     NSNotificationCenter *center = [NSNotificationCenter defaultCenter];  
343     [center postNotificationName████████  
            object:nil  
            userInfo:info];  
344  
345     ██████████ get];  
346 }  
347  
348 - (void)viewWillAppear:(BOOL)animated  
349 {  
350     if (!loaded)  
351     {  
352         // Perform lazy-loading of the visual structure of this class  
353         loaded = YES;  
354     }  
355 }
```



10) Use  
**NSZombieEnabled**







**Thanks!**

# Questions?