Coacoa内存管理 - By 尤明

引用计数的概念

以办公室开灯关灯举例

内存管理的思考方式

自己`生成`的对象，自己`持有`

```
* alloc
* new
* copy
* mutableCopy
```

```
// 自己生成并持有对象
id obj = [[NSObject alloc] init];
```

非自己`生成`的对象，自己也能`持有`

```
// 取得非自己生成的对象但自己不持有对象
id obj = [NSMutableArray array];
```

```
// 取得非自己生成的对象但自己持有对象
id obj = [NSMutableArray array];

// 自己持有对象
[obj retain];
```

不再需要自己`持有`的对象时`释放`

```
// 自己生成并持有对象
id obj = [[NSObject alloc] init];

// 释放对象
[obj release];

// 一般会在release之后，将对象指针赋为nil，以避免指针悬空的情况
obj = nil;
```

```
// 取得非自己生成的对象但自己持有对象
id obj = [NSMutableArray array];

// 自己持有对象
[obj retain];

// 释放对象
[obj release];
obj = nil;

// 对象不可再被访问
```

非自己`持有`的对象无法`释放`

```
// 自己生成并持有对象
id obj = [[NSObject alloc] init];

// 释放对象
[obj release];

// 一般会在release之后，将对象指针赋为nil，以避免指针悬空的情况
// obj = nil;

// 释放之后再次释放已非自己持有的对象，应用程序会崩溃
[obj release];
```

```
// 取得非自己生成的对象且自己不持有该对象
id obj = [NSMutableArray array];

// 释放了非自己持有的对象，会导致应用程序崩溃
[obj release];
```

ARC规则

所有权修饰符

```
__strong 修饰符
__weak 修饰符
__unsafe_unretained修饰符
__autoreleasing修饰符
```

- **__strong** 修饰符

> *__strong修饰符表示对对象的“强引用”。持有强引用的变量在超出其作用域时被废弃，随着强引用的失效，引用的对象会随之释放。* `Demo`
> *__strong修饰符修饰的变量，不仅只在变量作用域中，在赋值上也能正确地管理其对象的所有者。* `Demo`
> *_strong修饰符同后面要讲的_weak修饰符和__autoreleasing修饰符一起，可以保证将附有这些修饰符的自动变量初始化为nil。* `Demo`

- **__weak** 修饰符

> *循环引用容易发生内存泄露。所谓内存泄露就是应当废弃的对象在超出其生存周期后继续存在。* `Demo`
> *__weak修饰符在持有某对象的弱引用时，若该对象被废弃，则此弱引用将自动失效且处于nil被赋值的状态（空弱引用）* `Demo`

- **__unsafe_unretained** 修饰符

> *附有__unsafe_unretained 修饰符的变量不属于编译器的内存管理对象* `Demo`
> *赋值给附有__unsafe_unretained修饰符变量的对象在通过该变量使用时，如果没有确保其确实存在，那么应用程序就会崩溃* `Demo`

- **__autoreleasing** 修饰符

> *在使用ARC时，不能使用autorelease方法，也不能使用NSAutoreleasePool类*
> *指定"@autoreleasepool块"来替代"NSAutoreleasePool"类对象生成、持有以及废弃这一范围*
> *另外，要通过将对象赋值给附加了__autoreleasing修饰符的变量来替代调用autorelease方法* `Demo`

显式的附加autoreleasing修饰符同显式地附加strong修饰符一样罕见，原因如下:

（`考虑两种使用autoreleasepool的场景`）

非自己生成并持有的对象:

> 编译器会检查方法名是否以alloc/new/copy/mytableCopy开始，如果不是就说明是非自己生成并持有的对象,编译器会自动将返回的对象注册到autoreleasepool中

`Demo`

自己生成并持有的对象，在函数中返回时

> 由于return使得对象变量超出其作用域，所以强引用对应的自己生成并持有的对象会被自动释放，但该对象如果作为函数的返回值，编译器会自动将其注册到autoreleasepool中

`Demo`