GeekBand 极客班

互联网人才 + 油站!

GeekBand 极客班

www.geekband.com

设计模式四

孔祥波

回顾

- 委托模式delegate
- 观察者
- 消息通知
- KVC/KVO

归档和解档(Serialization)

需求/动机

- MVC 中的M 担负数据持久化任务
- Data 在多进程中传递，跨操作系统，网络
- 存储磁盘

字符串存储basic

```
NSString *String = @"写入文件";
NSString *path = @"/Users/apple/Desktop/abc/test.txt";
NSError *error;
[String writeToFile:path atomically:YES encoding:NSUTF8StringEncoding error:&error];

if(error){
    NSLog(@"写入失败：%@",[error localizedDescription]);        //[error
        localizedDescription]打印出主要错误信息  这个在工程中没有提示  可以再打印外面写就有提示
}else{
    NSLog(@"写入成功");
}
```

NSKeyedArchiver

```
-(void)save:(id)sender{
    NSString *homePath = NSHomeDirectory();
    NSString *path = [homePath stringByAppendingPathComponent:@"save.txt"]; //将文件保存到
        path这个路径下，并把文件名设置为save.txt
    [NSKeyedArchiver archiveRootObject:_text.text toFile:path];
    NSLog(@"保存成功");
}
-(void)getit:(id)sender{
    NSString *homePath = NSHomeDirectory();
    NSString *path = [homePath stringByAppendingPathComponent:@"save.txt"];
    NSString *string = [NSKeyedUnarchiver unarchiveObjectWithFile:path];
    _text.text = string;
}
```

NSData

- 归档[NSKeyedArchiver archivedDataWithRootObject:someObject];

- 解档[NSKeyedUnarchiver unarchiveObjectWithData:someData];

```
@implementation NSUserDefaults (ColorHandling)

- (void)setColor:(NSColor *)theColor forKey:(NSString *)key
{
  NSData *data = [NSKeyedArchiver archivedDataWithRootObject:
      theColor];
  [self setObject:data forKey:key];
}

- (NSColor *)colorForKey:(NSString *)key
{
  NSData *data = [self dataForKey:key];
  return [NSKeyedUnarchiver unarchiveObjectWithData:data];
}

@end
```

实现NSCoding protocol

```objc
@protocol NSCoding

- (void)encodeWithCoder:(NSCoder *)aCoder;
- (nullable instancetype)initWithCoder:(NSCoder *)aDecoder;

@end
```

```
@interface WordInformation : NSObject <NSCoding>
{
  NSString              *word;
  NSString              *clue;
  NSMutableDictionary   *puzzleSpecificAttributes;
}

@end

// Coding keys
static NSString   *CodingKeyWord = @"word";
static NSString   *CodingKeyClue = @"clue";
static NSString   *CodingKeyPuzzleSpecificAttributes =
    @"puzzleSpecificAttributes";
```

```
- (id)initWithCoder:(NSCoder *)coder
{
  if (nil != (self = [super init]))
  {
    [self setWord:[coder decodeObjectForKey:CodingKeyWord]];
    [self setClue:[coder decodeObjectForKey:CodingKeyClue]];
    [self setPuzzleSpecificAttributes:[coder decodeObjectForKey:
        CodingKeyPuzzleSpecificAttributes]];
  }
  return self;
}

- (void)encodeWithCoder:(NSCoder *)coder
{
  [coder encodeObject:[self word] forKey:CodingKeyWord];
  [coder encodeObject:[self clue] forKey:CodingKeyClue];
  [coder encodeObject:[self puzzleSpecificAttributes] forKey:
      CodingKeyPuzzleSpecificAttributes];
}
```

非Object 类型处理

归档NSKeyedArchiver

```
- (void)encodeObject:(nullable id)objv forKey:(NSString *)key;
- (void)encodeConditionalObject:(nullable id)objv forKey:(NSString *)key;
- (void)encodeBool:(BOOL)boolv forKey:(NSString *)key;
- (void)encodeInt:(int)intv forKey:(NSString *)key; // native int
- (void)encodeInt32:(int32_t)intv forKey:(NSString *)key;
- (void)encodeInt64:(int64_t)intv forKey:(NSString *)key;
- (void)encodeFloat:(float)realv forKey:(NSString *)key;
- (void)encodeDouble:(double)realv forKey:(NSString *)key;
- (void)encodeBytes:(nullable const uint8_t *)bytesp length:(NSUInteger)lenv forKey:
    (NSString *)key;
```

解档NSUnKeyedArchiver

```
- (BOOL)decodeBoolForKey:(NSString *)key;
- (int)decodeIntForKey:(NSString *)key;     // may raise a range exception
- (int32_t)decodeInt32ForKey:(NSString *)key;
- (int64_t)decodeInt64ForKey:(NSString *)key;
- (float)decodeFloatForKey:(NSString *)key;
- (double)decodeDoubleForKey:(NSString *)key;
- (nullable const uint8_t *)decodeBytesForKey:(NSString *)key returnedLength:(nullable
    NSUInteger *)lengthp NS_RETURNS_INNER_POINTER;  // returned bytes immutable, and they go
    away with the unarchiver, not the containing autorelease pool
```

复制模式

复制

- 创建一个对象的新副本
- 复制一个复杂对象时，保护一个一样的对象，还是包含原来对象的副本
- 用户界面上的复制/粘贴
- 有些对象封装了独一无二的资源，复制没有意义
- 浅复制和深复制。顾名思义，浅复制，并不拷贝对象本身，仅仅是拷贝指向对象的指针；深复制是直接拷贝整个对象内存到另一块内存中。

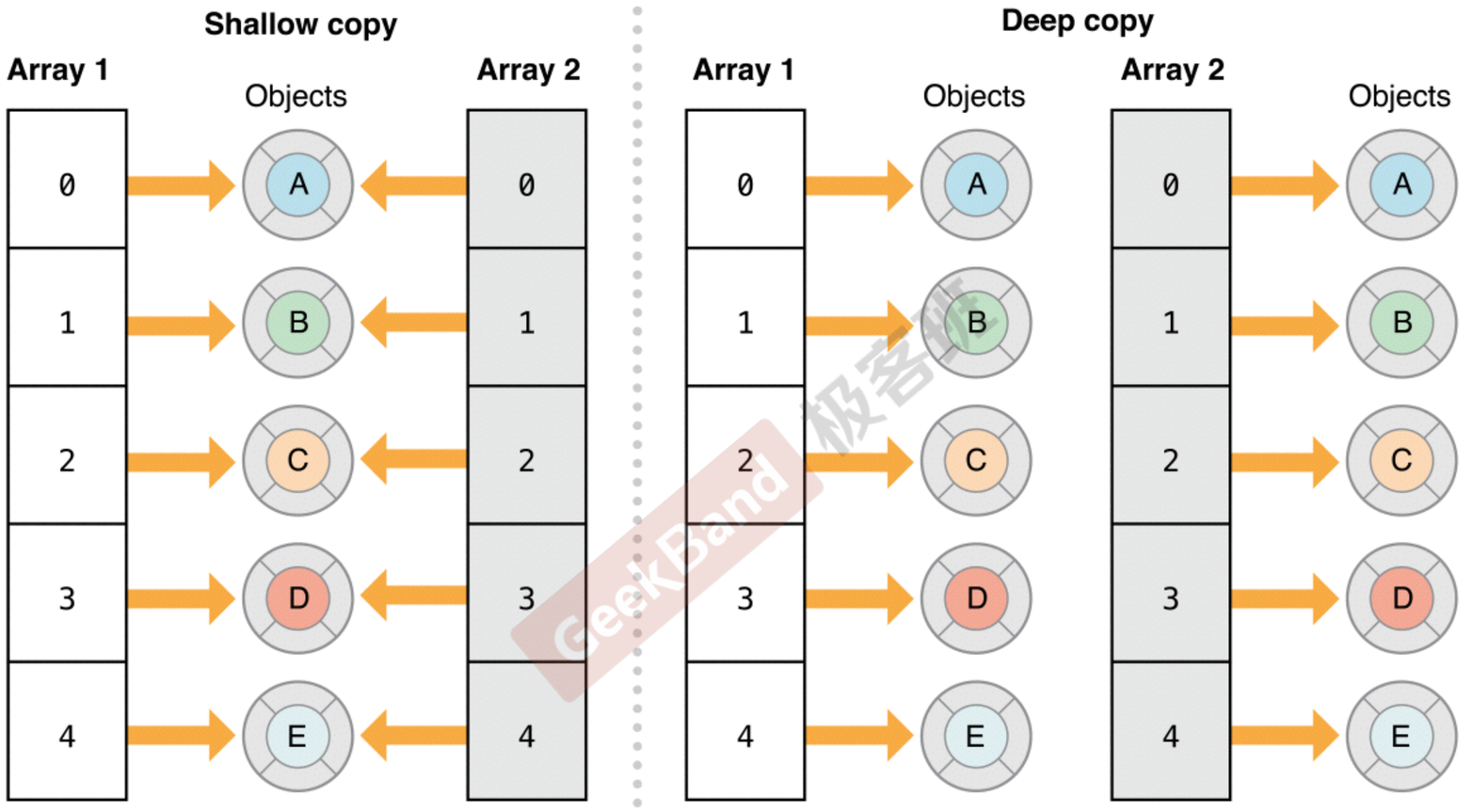
Cocoa系统

NSObject

```
- (id)copy;
- (id)mutableCopy;

+ (id)copyWithZone:(struct _NSZone *)zone OBJC_ARC_UNAVAILABLE;
+ (id)mutableCopyWithZone:(struct _NSZone *)zone OBJC_ARC_UNAVAILABLE;
```

Protocol

```
@protocol NSCopying

- (id)copyWithZone:(nullable NSZone *)zone;

@end

@protocol NSMutableCopying

- (id)mutableCopyWithZone:(nullable NSZone *)zone;

@end
```



再简单些说：**浅复制就是指针拷贝；深复制就是内容拷贝。**

集合的浅复制 (shallow copy)

集合的浅复制有非常多种方法。当你进行浅复制时，会向原始的集合发送retain消息，引用计数加1，同时指针被拷贝到新的集合。

现在让我们看一些浅复制的例子:

```
1.  NSArray *shallowCopyArray = [someArray copyWithZone:nil];
2.
3.  NSSet *shallowCopySet = [NSSet mutableCopyWithZone:nil];
4.
5.  NSDictionary *shallowCopyDict = [[NSDictionary alloc] initWithDictionary:someDictionary copy
    Items:NO];
```

集合的深复制 (deep copy)

集合的深复制有两种方法。可以用 initWithArray:copyItems: 将第二个参数设置为YES即可深复制，如

```
1.  NSDictionary shallowCopyDict = [[NSDictionary alloc] initWithDictionary:someDictionary copyI
    tems:YES];
```

如果你用这种方法深复制，集合里的每个对象都会收到 copyWithZone: 消息。如果集合里的对象遵循 NSCopying 协议，那么对象就会被深复制到新的集合。如果对象没有遵循 NSCopying 协议，而尝试用这种方法进行深复制，会在运行时出错。copyWithZone: 这种拷贝方式只能够提供一层内存拷贝(one-level-deep copy)，而非真正的深复制。

第二个方法是将集合进行归档(archive)，然后解档(unarchive)，如:

```
1.  NSArray *trueDeepCopyArray = [NSKeyedUnarchiver unarchiveObjectWithData:[NSKeyedArchiver
    archivedDataWithRootObject:oldArray]];
```

为何copy?

- 修改数据（增，删，排序）
- 复制的是个指针，还是指针指向的对象(新对象和原对象在内存中的地址是否一样)

在非集合类对象中：对immutable对象进行copy操作，是指针复制，mutableCopy操作时内容复制；对mutable对象进行copy和mutableCopy都是内容复制。用代码简单表示如下：

- [immutableObject copy] // 浅复制
- [immutableObject mutableCopy] //深复制
- [mutableObject copy] //深复制
- [mutableObject mutableCopy] //深复制

在集合类对象中，对immutable对象进行copy，是指针复制，mutableCopy是内容复制；对mutable对象进行copy和mutableCopy都是内容复制。但是：集合对象的内容复制仅限于对象本身，对象元素仍然是指针复制。用代码简单表示如下：

- [immutableObject copy] // 浅复制
- [immutableObject mutableCopy] //单层深复制
- [mutableObject copy] //单层深复制
- [mutableObject mutableCopy] //单层深复制

小结

- 归档和解档(Serialization)
- 复制模式