iOS Runtime常见问题分析

周辉

讲师介绍

- 06年毕业于武汉理工大学
- 08年进行iOS开发
- 10年加入大众点评
- 平台产品事业部移动架构团队

FBI Warning

本PPT中涉及敏感代码，请注意保密，维护公司代码安全

- iOS Runtime介绍
- 动态数据模型
- 美团性能监控
- Hot Patch技术
- 越狱玩法
- 常见Crash分析

Category

- 添加方法
- 添加变量
- 隐藏方法
- 提供私有方法原型

Category用法
添加方法

```
@interface UIColor (Util)

+ (UIColor *)colorWithHexString:(NSString *)hexString;

+ (UIColor *)colorWithIntRed:(NSInteger)r green:(NSInteger)g blue:(NSInteger)b;
+ (UIColor *)colorWithIntRed:(NSInteger)r green:(NSInteger)g blue:(NSInteger)b alpha:
(NSInteger)a;

@end
```

Category用法
添加变量

```
@interface UIViewController (urlAction)
@property (nonatomic, strong) NVURLAction *urlAction;
@end

@implementation UIViewController (urlAction)

- (void)setUrlAction:(NVURLAction *)urlAction {
    objc_setAssociatedObject(self, "UIViewControllerNVURLAction", urlAction,
OBJC_ASSOCIATION_RETAIN);
}

- (NVURLAction *)urlAction {
    return objc_getAssociatedObject(self, "UIViewControllerNVURLAction");
}

@end
```

Category用法
隐藏方法

NVMainViewController.h

```
@interface NVMainViewController : TSTabBarController <UITabBarControllerDelegate,
NVNavigatorDelegate, UINavigationControllerDelegate, QQApiInterfaceDelegate>

- (void)showSplash:(NSString *)message;

@end
```

NVMainViewController.m

```
@interface NVMainViewController ()

@property (nonatomic, strong) NVFindHomeViewController * findHomeViewController;
@property (nonatomic, strong) NVExploreViewController * findViewController;
@property (nonatomic, strong) NVUserCenterViewController * userCenterViewController;
@property (nonatomic, strong) TGMainViewController  *tuanHomepageViewController;

@property (nonatomic, assign) NSInteger cityId;
@end

@implementation NVMainViewController
{
    NVNotifyViewController  *_notifyController;
}
```

Category用法
提供私有方法原型

```
@interface NVBaseNavigator (Private)
- (NSMutableDictionary *)loadPattern;
@end

@interface NVNavigator (Private)
- (void)pushViewController:(UIViewController *)controller withURLAction:
(NVURLAction *)urlAction;
- (void)openViewController:(UIViewController *)controller withURLAction:
(NVURLAction *)urlAction;
- (BOOL)handleNeedsLoginAction:(NVURLAction *)urlAction withController:
(UIViewController *)controller;
@end
```

Category注意事项

名称冲突

覆盖顺序以编译顺序为准

静态库的调用

使用静态库中的Category时，需要在调用者项目的Other Link flags中增加:
-all_load (-ObjC -force_load)

-ObjC allow the static library to use objective-c specific stuffs like kvc or categories.
-all_load solve a bug in gcc/llvm, where -ObjC is not correctly used.

-force_load 可以对单个库进行控制

基本概念

类的定义

objc.h

```
typedef struct objc_class *Class;
typedef struct objc_object {
    Class isa;
} *id;
```

runtime.h

```
struct objc_class {
    Class isa;

#if !__OBJC2__
    Class super_class
    const char *name
    long version
    long info
    long instance_size
    struct objc_ivar_list *ivars
    struct objc_method_list **methodLists
    struct objc_cache *cache
    struct objc_protocol_list *protocols
#endif

} OBJC2_UNAVAILABLE;
```

基本概念

方法的定义

objc.h

```
typedef struct objc_selector    *SEL;

#if !OBJC_OLD_DISPATCH_PROTOTYPES
typedef void (*IMP)(void /* id, SEL, ... */ );
#else
typedef id (*IMP)(id, SEL, ...);
#endif
```

runtime.h

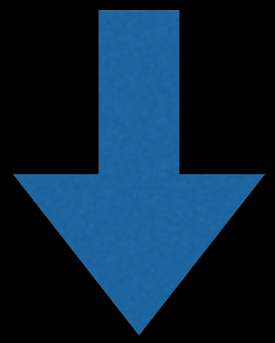
```
struct objc_method {
    SEL method_name
    char *method_types
    IMP method_imp
}

struct objc_method_list {
    struct objc_method_list *obsolete

    int method_count
#ifdef __LP64__
    int space
#endif
    /* variable length structure */
    struct objc_method method_list[1]
}
```

基本概念

- Method Signature
  - 形如“v@:”
- NSInvocation
  - target
  - selector
  - MethodSignature
  - returnValue
  - arguments
  - invoke

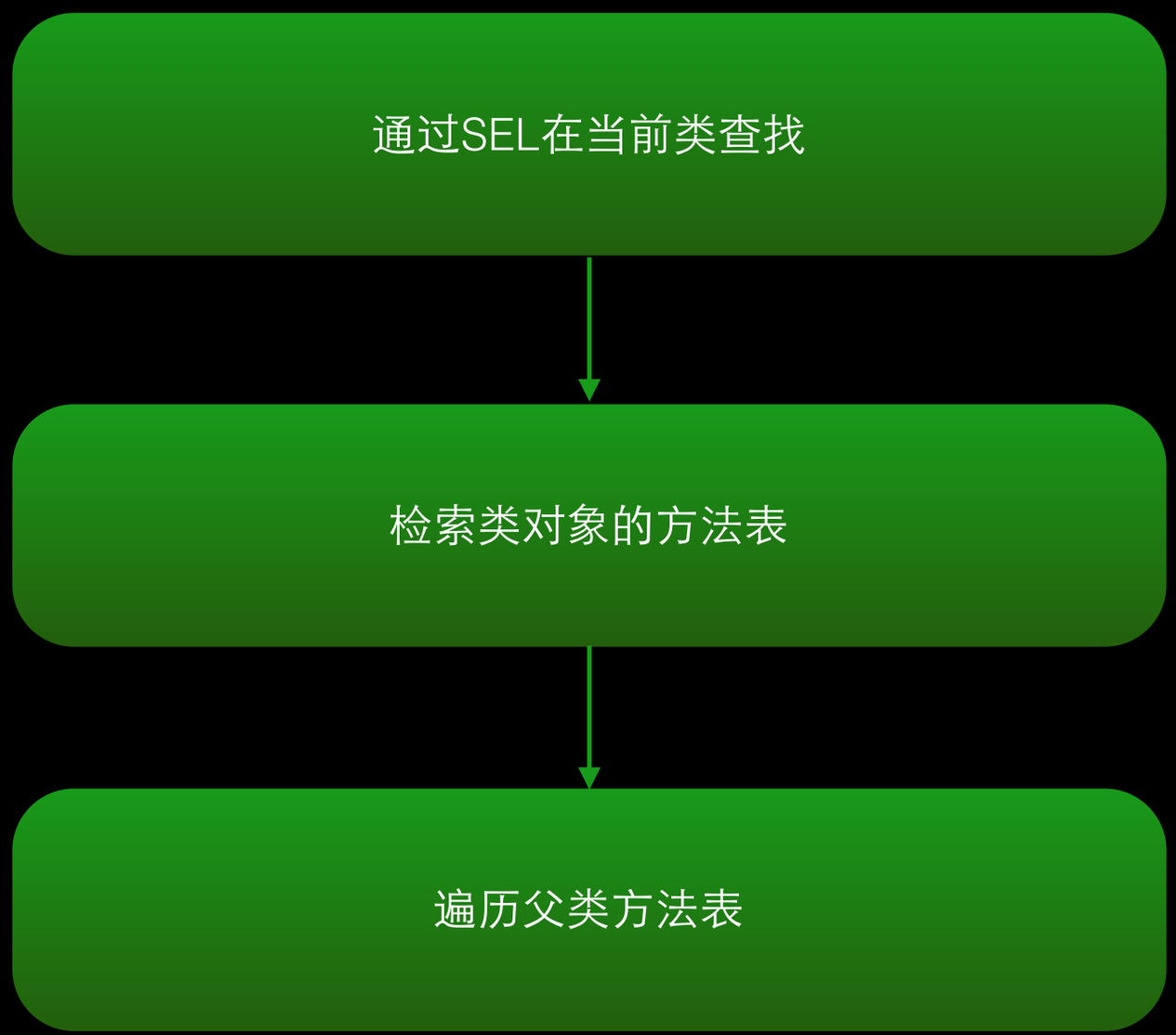
消息调用过程

[target doMethodWith:var];



```
objc_msgSend(target,@selector(doMethodWith:),var1);
```

查找IMP



缓存优化

找不到IMP的处理

动态决议

```
+ (BOOL)resolveInstanceMethod:(SEL)name;
+ (BOOL)resolveClassMethod:(SEL)name;
```

```
void dynamicMethodIMP(id self, SEL _cmd)
{
    // 动态增加的方法 ....
}

+ (BOOL) resolveInstanceMethod:(SEL)aSEL
{
    if (aSEL == @selector(resolveThisMethodDynamically))
    {
          class_addMethod([self class], aSEL, (IMP) dynamicMethodIMP, "v@:");
          return YES;
    }
    return [super resolveInstanceMethod:aSel];
}
```

找不到IMP的处理

消息转发

更换方法签名:

```
- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector;
```

Selector方式转发:

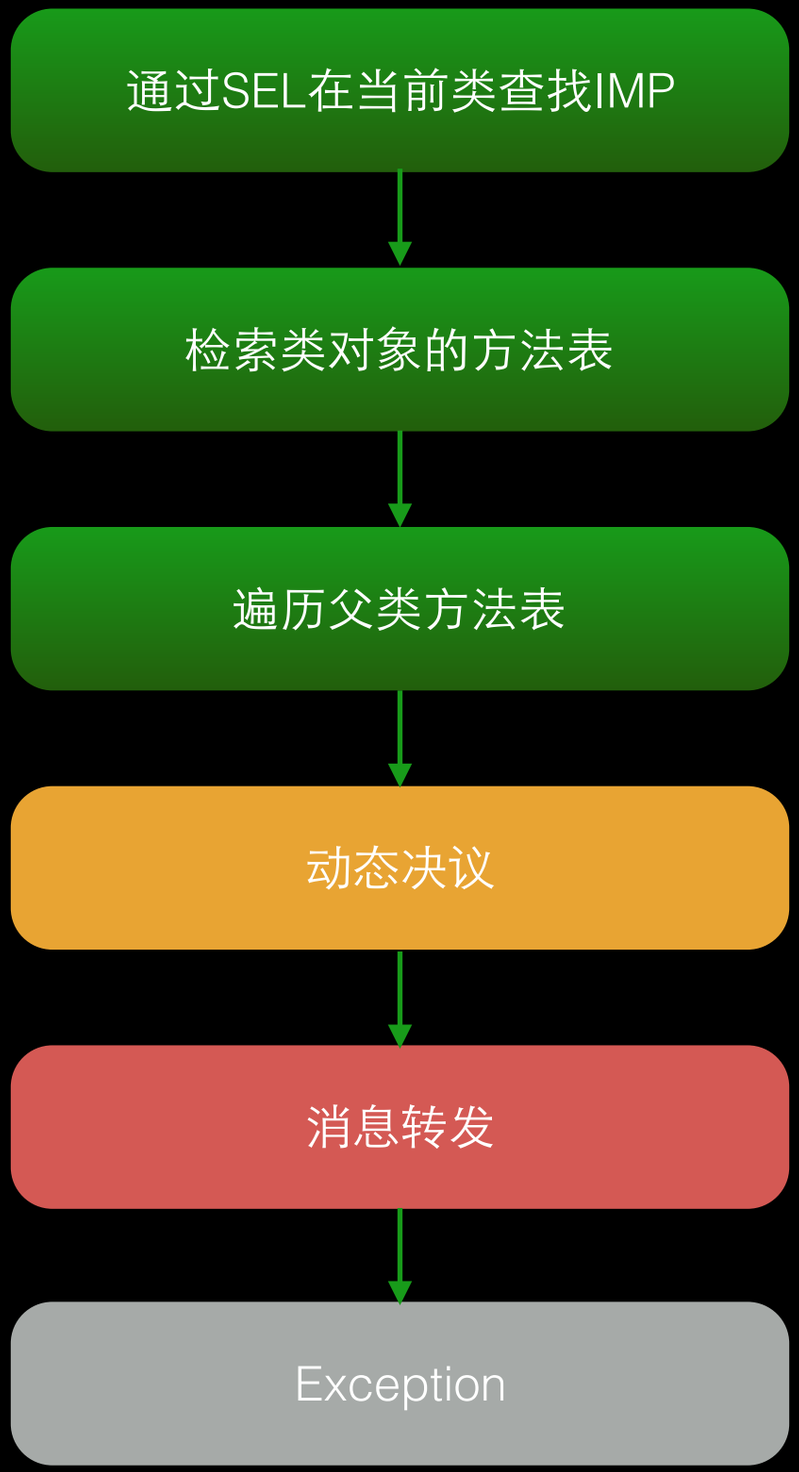
```
- (id)forwardingTargetForSelector:(SEL)aSelector;
```

NSInvocation方法转发:

```
- (void)forwardInvocation:(NSInvocation *)anInvocation;
```

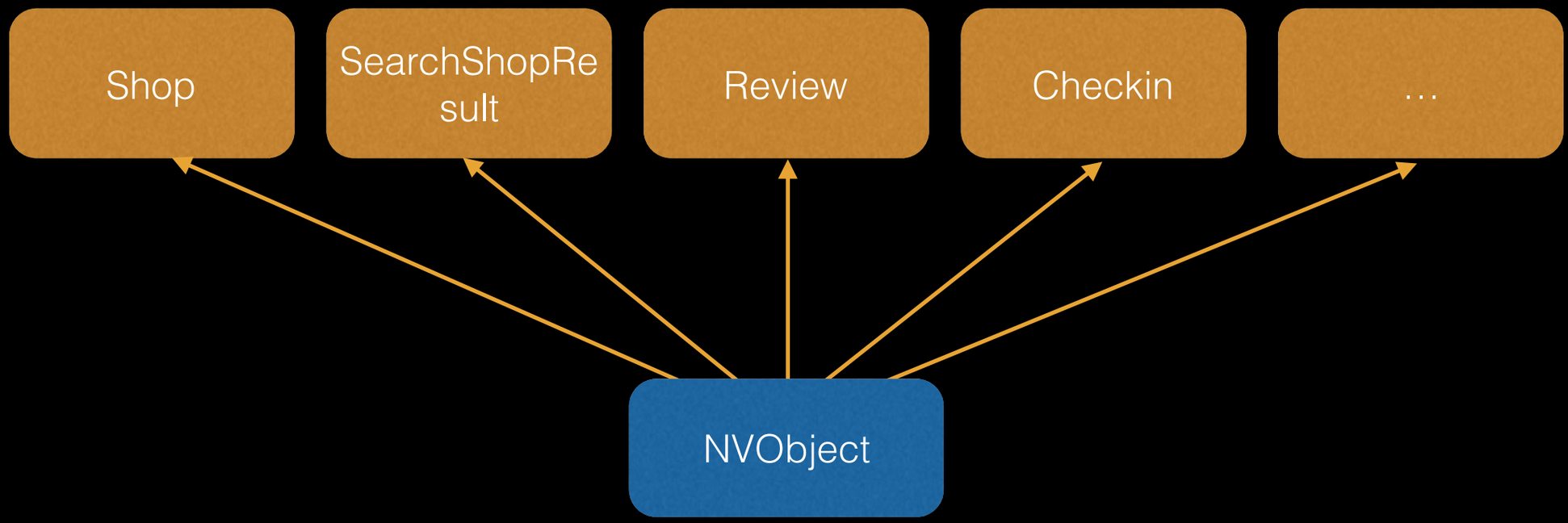
```
- (void)forwardInvocation:(NSInvocation *)anInvocation {
    SEL name = [anInvocation selector];
    NSLog(@" >> forwardInvocation for selector %@", NSStringFromSelector(name));

    Proxy * proxy = [[[Proxy alloc] init] autorelease];
    if ([proxy respondsToSelector:name]) {
        [anInvocation invokeWithTarget:proxy];
    } else {
        [super forwardInvocation:anInvocation];
    }
}

- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector {
    return [Proxy instanceMethodSignatureForSelector:aSelector];
}
```

消息调用全过程

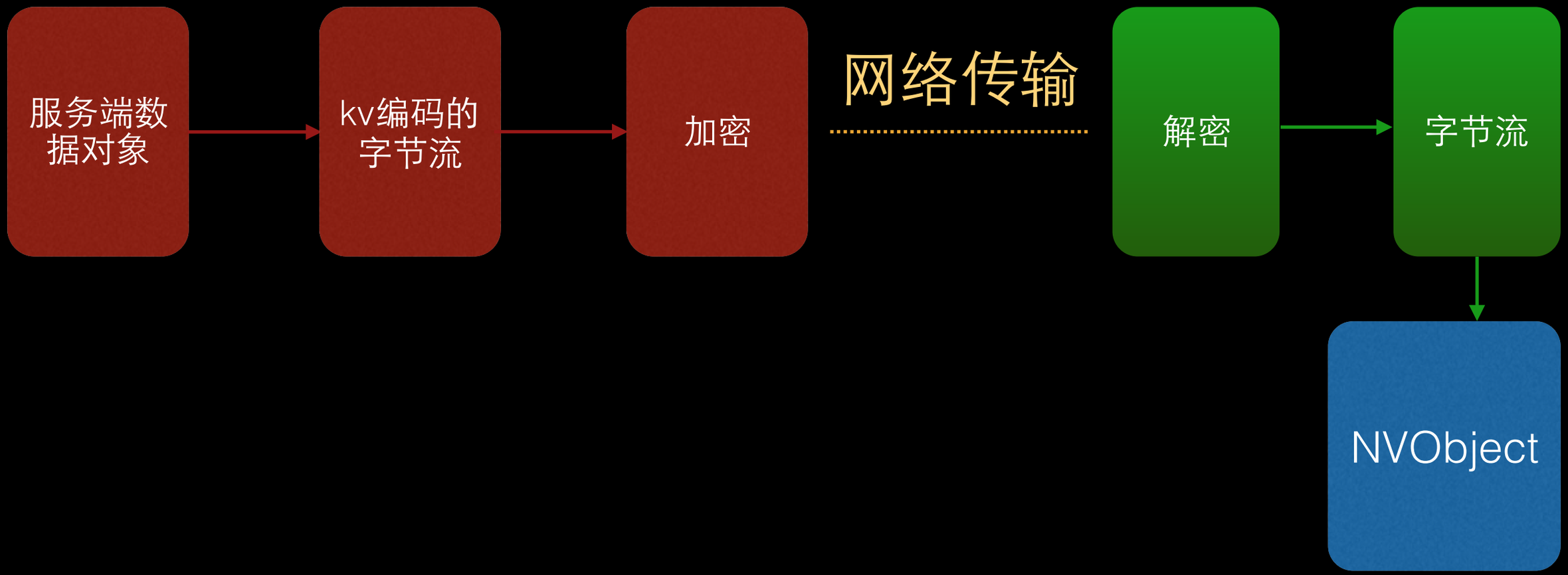


动态数据模型

万能的NVObject



网络数据传输过程



NVObject数据格式

数据结构:
ShopInfo {
    String name;
    Double time;
}

Hash:
==> 0x123
==> 0x456
==> 0x789

Value:
==> “俏江南”
==> “123123.1”

Models.h

```
@class ResultList;

@interface ResultList : NVObject
/** 表示结果总数 unknow = -1*/
@property (nonatomic, assign, readonly) NSInteger recordCount;
/** start from 0*/
@property (nonatomic, assign, readonly) NSInteger startIndex;
@property (nonatomic, assign, readonly) BOOL isEnd;
/** 表示下一页请求的start*/
@property (nonatomic, assign, readonly) NSInteger nextStartIndex;
/** 用于控制list数据为空时的客户端显示*/
@property (nonatomic, copy, readonly) NSString * emptyMsg;
/** 用于搜索关键字优化统计*/
@property (nonatomic, copy, readonly) NSString * queryID;
@property (nonatomic, strong, readonly) NSArray * list;
@end
```

NVObject对象的方法实现

参见NVObject.m

#pragma mark - Forwarding

```
- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector;
- (void)forwardInvocation:(NSInvocation *)anInvocation;
```

```
- (BOOL)booleanForHash:(NSInteger)hash;
- (BOOL)booleanForKey:(NSString *)name;
……
- (NSArray *)anyArrayForHash:(NSInteger)hash;
- (NSArray *)anyArrayForKey:(NSString *)name;
```

直接看代码

NVObject的修改

```
@interface NVObjectEditor : NSObject
- (id)initWithObject:(NVObject *)obj;
- (NVObjectEditor *)setBoolean:(BOOL)val forHash:(NSInteger)hash;
- (NVObjectEditor *)setBoolean:(BOOL)val forKey:(NSString *)name;

……

- (NVObjectEditor *)setStringArray:(NSArray *)val forHash:(NSInteger)hash;
- (NVObjectEditor *)setStringArray:(NSArray *)val forKey:(NSString *)name;

- (NVObject *)generate;
@end
```

```
NVObjectEditor *dataEditor = [[NVObjectEditor alloc] initWithObject:_photoInfo];
[dataEditor setObject:photoInfoEditorView.photoCategory
forKey:kDraftPhotoCategoryObjKey];
[dataEditor setString:photoInfoEditorView.photoName forKey:kDraftPhotoNameStrKey];
_photoInfo = [dataEditor generate];
```

NVObject对象的优缺点

优点:

- 提升端到端的时间
- 根据文档自动生成
- 减少对象和方法数量
- 可以提供语法提示
- 便于持久化

缺点:

- 不便于修改
- 不便于定位错误
- 无法自省（hash无法还原成key）

消息调用黑魔法

Method Swizzle

```
@implementation NSObject (SAKSwizzle)
+ (BOOL)sak_SwizzleMethod:(SEL)originalSEL withMethod:(SEL)alternateSEL error:(NSError**)error
{
    Method originalMethod = class_getInstanceMethod(self, originalSEL);
    Method alternateMethod = class_getInstanceMethod(self, alternateSEL);

    class_addMethod(self,
                originalSEL,
                class_getMethodImplementation(self, originalSEL),
                method_getTypeEncoding(originalMethod));
    class_addMethod(self,
                alternateSEL,
                class_getMethodImplementation(self, alternateSEL),
                method_getTypeEncoding(alternateMethod));

    method_exchangeImplementations(class_getInstanceMethod(self, originalSEL),
class_getInstanceMethod(self, alternateSEL));
    return YES;
}

+ (BOOL)sak_SwizzleClassMethod:(SEL)originalSEL withClassMethod:(SEL)alternateSEL error:
(NSError**)error{
    return [GetClass((id)self) sak_SwizzleMethod:originalSEL withMethod:alternateSEL
error:error];
}
@end
```

美团性能监控

```
@implementation UIViewController (Performance)

+ (void)load
{
    NSError *error = nil;

    [[UIViewController class] sak_SwizzleMethod:@selector(viewWillDisappear:)
                                     withMethod:@selector(performanceViewWillDisappear:)
                                          error:&error];
}

- (void)performanceViewWillDisappear:(BOOL)animated
{
    if ([[SAKPerformanceMonitoringDataProcessor sharedDataProcessor] isReporting] &&
[[SAKPerformanceTracingViewControllerCollector sharedViewControllerCollector] isReporting] ||
[SAKPerformanceMonitoringDataProcessor sharedDataProcessor].networkEnable) {
        // 性能测试上报
    }
    [self performanceViewWillDisappear:animated];
}

@end
```

Hot Patch技术

- Wax
- NU
- JSPatch

Wax

屠毅敏大师 2012年在开源Wax基础上构建WaxPatch
https://github.com/mmin18/WaxPatch
广泛运用与淘宝、支付宝等大型App中

用Lua给你的程序打patch（续）

2013-03-11 iOS开发 收藏，稍后阅读

今日分享：用Lua给你的程序打patch（续）

昨天仔细阅读了一下大众点评的屠毅敏开源的WaxPatch项目，他是通过修改wax的实现达到动态打Patch的功能。主要是修改了wax中的wax_instance.m文件。具体做法是：在wax中加入了class_replaceMethod来替换原始实现。

Wax

- Wax开源地址 https://github.com/probablycorey/wax
- 可以访问原生Cocoa API
- 支持Lua类型和OC类型的自动转换
- 自动内存管理
- 代码更加简洁
- 阿里接手并维护 https://github.com/alibaba/Wax

Wax打补丁

Wax Patch

class_replaceMethod替换原有方法，适合于打补丁
在原有方法前加上“ORIG”

```
waxClass{"MainViewController", UITableViewController}

function tableView_cellForRowAtIndexPath(self, tableView,
indexPath)
    local cell =
self:ORIGtableView_cellForRowAtIndexPath(tableView, indexPath)
-- 被覆盖的方法会在名字前加上'ORIG'继续保留
    cell:textLabel():setText("" .. (10 - indexPath:row()))
    cell:detailTextLabel():setText("http://github.com/mmin18")
    cell:textLabel():setTextColor(UIColor:redColor())
    return cell
end
```

NU

点评目前主打patch工具
http://lib.mobile.dp/#!library/ios/tech/nu_tutorial.md

JSPatch

使用JS语言的极小引擎

优势：
- 使用JS语言
- 符合Apple规则
- 小巧
- 支持Block

原理：
JavaScriptCore.framework，利用OC runtime进行方法调用

越狱玩法

- 越狱
  - 9.0.2及其以下版本
- 安装Cydia
- 官方源
- 威锋源apt.so
- 自定义源

静态分析工具

- class-dump-z: 用于简单分析出工程中的类名和函数名
- IDA：强大的反编译工具
- Hopper Disassembler：类似IDA
- Reveal：UI层解析工具
- dumpdecrypted：砸壳工具

Cycript

强大的运行时命令行工具
（Cydia中搜索安装[cycript][ssh connect]）

例子：快捷开关VPN

```
~ root# cycript -p SpringBoard

cy# dlopen("/System/Library/PreferenceBundles/VPNPreferences.bundle/
VPNPreferences", RTLD_LAZY);
cy# var c = [[objc_getClass("VPNBundleController") alloc] initWithParentListControll
er:nil]
#"<VPNBundleController: 0x19168b50>"
cy# [c _setVPNActive:YES]
cy# [c _setVPNActive:NO]
```

做成程序

mac安装Theos

```
export THEOS=/opt/theos
git clone git://github.com/DHowett/theos.git $THEOS
```

```
$ $THEOS/bin/nic.pl
NIC 1.0 - New Instance Creator
------------------------------
  [1.] iphone/application
  [2.] iphone/library
  [3.] iphone/preference_bundle
  [4.] iphone/tool
  [5.] iphone/tweak
Choose a Template (required): 1
Project Name (required): iPhoneDevWiki
Package Name [com.yourcompany.iphonedevwiki]: net.howett.iphonedevwiki
Authour/Maintainer Name [Dustin L. Howett]:
Instantiating iphone/application in iphonedevwiki/...
Done.
$
```

私有库的获取

- Framework Headers
- SpringBoard Headers

https://github.com/MP0w/iOS-Headers

```
@interface SpringBoard : NSObject
// 点击Home键
-(_Bool)clickedMenuButton;
// 锁屏
-(void)unlockUIFromSource:(int)arg1 withOptions:(id)arg2;
// 注销
-(void)_relaunchSpringBoardNow;
// 打开指定程序
-(void)launchIcon:(id)arg1 fromLocation:(int)arg2;
// 重启
- (void)reboot;
@end
```

方法注入

制作Tweak

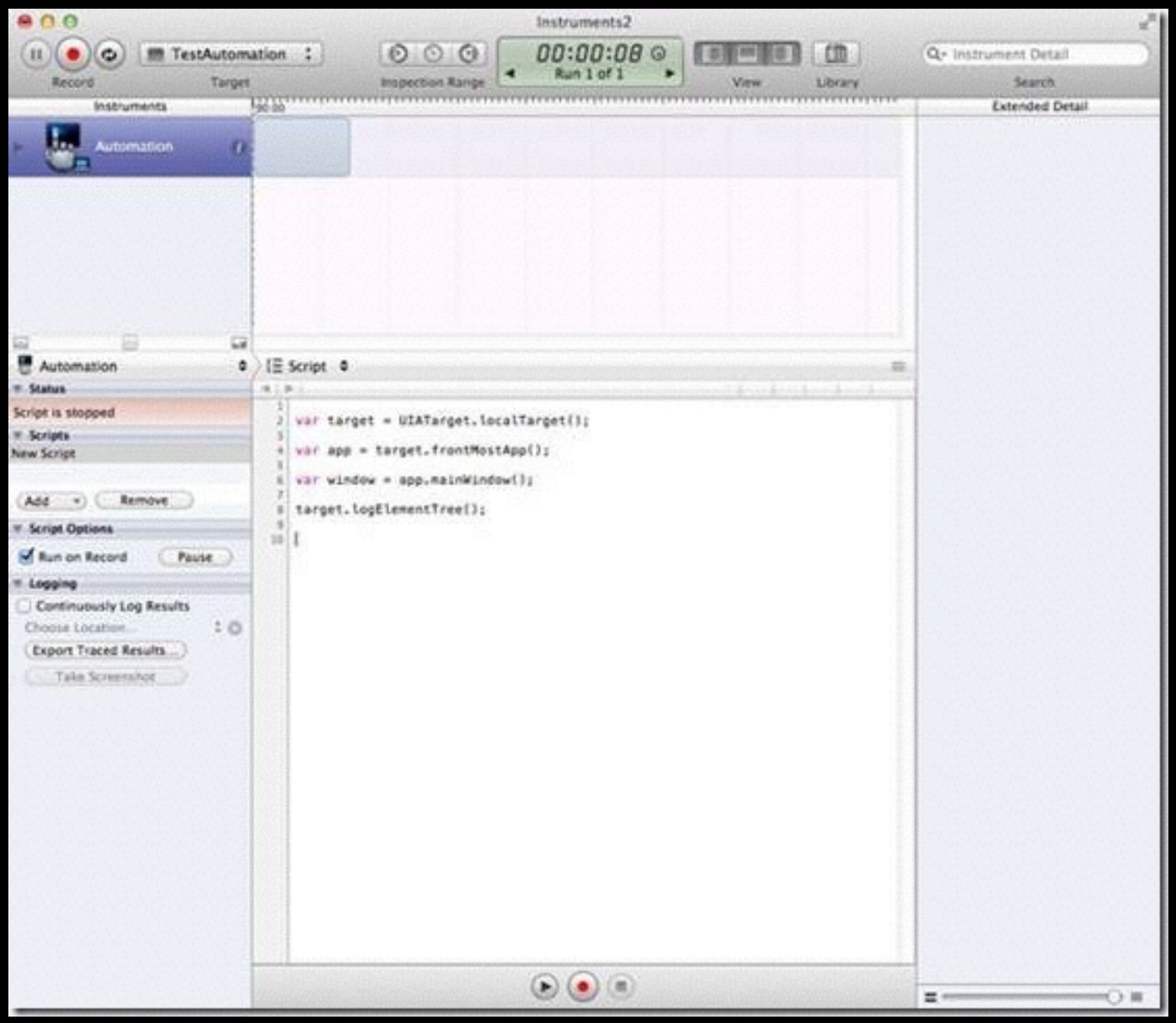
```
#import <SpringBoard/SpringBoard.h>
%hook SpringBoard
-(void)applicationDidFinishLaunching:(id)application {
    %orig;
    UIAlertView *alert = [[UIAlertView alloc]
    initWithTitle:@"Welcome" message:@"Welcome to your iOS
    Device Ted!" delegate:nil cancelButtonTitle:@"security.ios-
    wiki.com" otherButtonTitles:nil];
    [alert show];
    [alert release];
}
%end
```

系统事件的监听

```
for (NSString *notificationName in [Common allSystemNotificationNames]) {
    CFNotificationCenterAddObserver(CFNotificationCenterGetDarwinNotifyCenter(),
                                    NULL,
                                    __notification_callback,
(CFStringRef)notificationName,
                                    NULL,
                                    CFNotificationSuspensionBehaviorHold);
}

static void __notification_callback(CFNotificationCenterRef center, void *observer,
CFStringRef name, const void *object, CFDictionaryRef userInfo) {
    NSString *nameString = (NSString *)CFBridgingRelease(name);
}
```

```
@"com.apple.springboard.showingAlertItem",
@"com.apple.springboard.silent-vibrate.changed",
@"com.apple.springboard.syncingUnblocked",
@"com.apple.springboard.unambiguousOrientation",
@"com.apple.springboard.wallpaperuniquenesschanged",
@"com.apple.system.IOAudio2.oink.Codec",
@"com.apple.system.SCNetworkConnectionOnDemand",
@"com.apple.system.SystemConfiguration.dns_configuration",
@"com.apple.system.config.network_change",
@"com.apple.system.hostname",
@"com.apple.system.memorystatus",
@"com.apple.system.periodic",
@"com.apple.system.periodicthermalupdate",
```

UIAutomation




```
var target = UIATarget.localTarget();
var app = target.frontMostApp();
var window = app.mainWindow();
target.logElementTree();
```

可用私有库进行程序开发

需要连接XCode，打开开发者模式(可破解)

程序升级

apt-get方式

```
sudo apt-get update
&& sudo apt-get upgrade -y --force-yes
&& sudo apt-get dist-upgrade -y -force-yes
&& sudo apt-get install “com.test.app” -y --force-yes
```

dpkg方式

```
sudo dpkg -i /tmp/app.deb
```

常见Crash分析

Crash文件获得

- Xcode-Organizer
- Itunes Connect
- 代码拦截获取
- Cat上传

symbolicatecrash xxx.crash xxx.app.dSYM > test.txt

常见Crash分析

Cat中统计的Crash

Status	Count	SampleLinks
Signal (null) was raised	177	Looooooooooooooooooooooooo oooooooooooooooooog
Signal 11 was raised	25	Loooooooooooooooooooooooog
Signal 10 was raised	23	Loooooooooooooooooooooog
Signal 5 was raised	17	Looooooooooooooog
[__NSArrayM objectAtIndex:]: index 2 beyond bounds [0 .. 1	12	Loooooooooog
[__NSCFConstantString objectForKey:]: unrecognized selector sent to instan ce	3	Log
[__NSCFNumber realReviewRect]: unrecognized selector sent to instance	2	Lg
com.mysql.jdbc.MysqlDataTruncation	2	Lg

符号表不可少

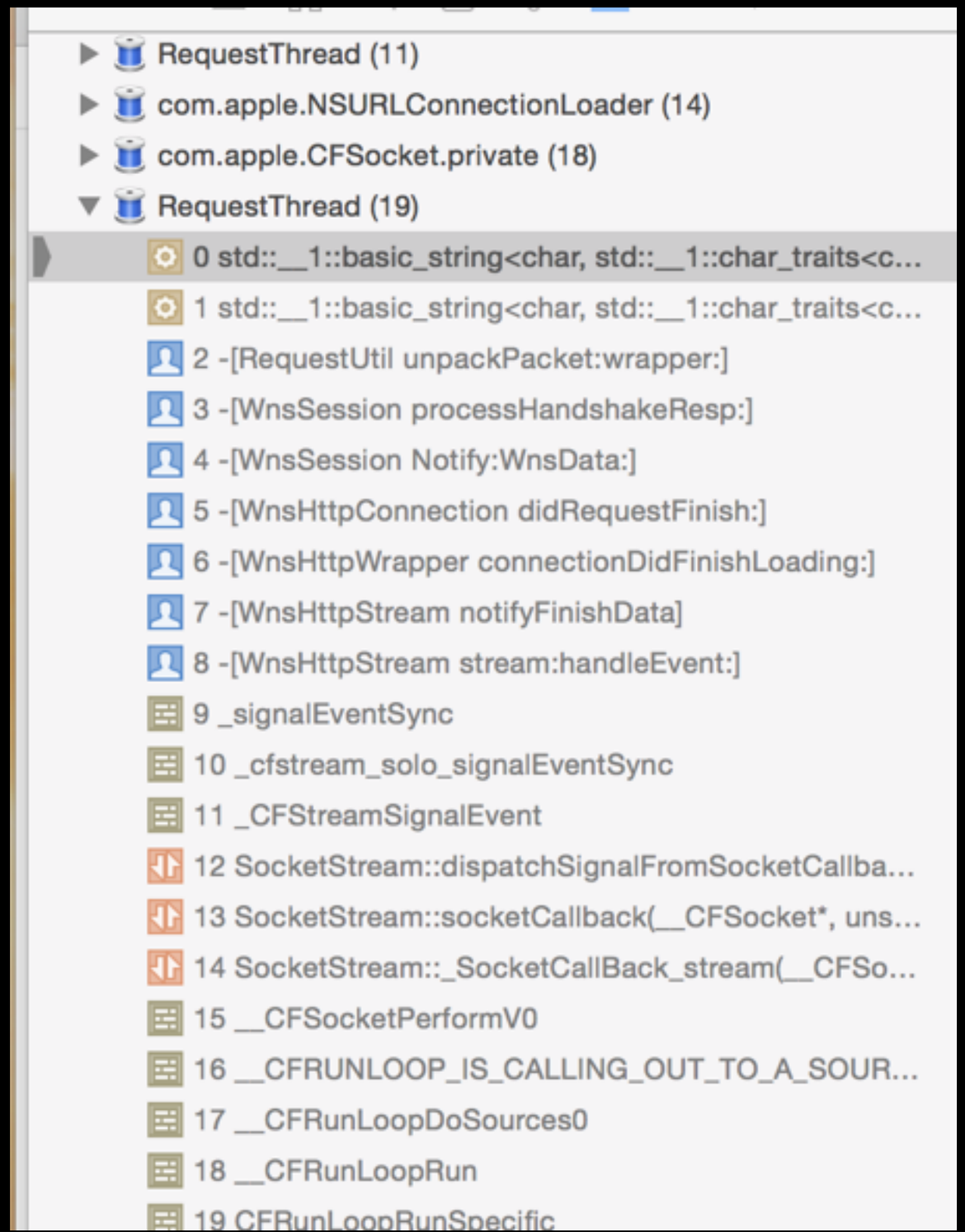
Thread 8 Crashed:
0 libc++.1.dylib 0x3653632a std::__1::basic_string, std::__1::allocator >::compare(unsigned long, unsigned long, char const*, unsigned long) const + 2
1 libc++.1.dylib 0x36536433 std::__1::basic_string, std::__1::allocator >::compare(char const*) const + 103
2 DPScope 0x01460175 CTXAppidConvert::IsConnectionAppId(char const*) + 797613
3 DPScope 0x0144640d CTXAppidConvert::IsConnectionAppId(char const*) + 691781
4 DPScope 0x0144721d CTXAppidConvert::IsConnectionAppId(char const*) + 695381

……..

Notable Addresses:
{
"stack@0x65d469c": {
"address": 22992441,
"type": "string",
"value": "wnscloud.handshake"
}
}

符号表不可少

实际情况是
WNS SDK
在Handshake时
引起的Crash：



Crash拦截

1，未捕获的异常Exception：
NSSetUncaughtExceptionHandler(&HandleException);

2，EXC_BAD_ACCESS引起的异常信号：
signal(SIGABRT, SignalHandler);
signal(SIGILL, SignalHandler);
signal(SIGSEGV, SignalHandler);
signal(SIGFPE, SignalHandler);
signal(SIGBUS, SignalHandler);
signal(SIGPIPE, SignalHandler);

常见Crash

- 数组越界
- 多线程引起的问题
- OC方法缺陷
- 数组遍历引起的Crash

真实案例

```
- (BOOL)sendSession:(TNSession *)s {
    BOOL sent = NO;
    for (TNConnection *conn in _connections) {
        if ([conn isConnected]){
            if ([conn send:s.request]) {
                // 发送成功
                s.connection = conn;
                sent = YES;
                break;
            } else {
                // 发送失败，关闭通道
                [conn close];
            }
        }
    }
    return sent;
}
```

Thanks