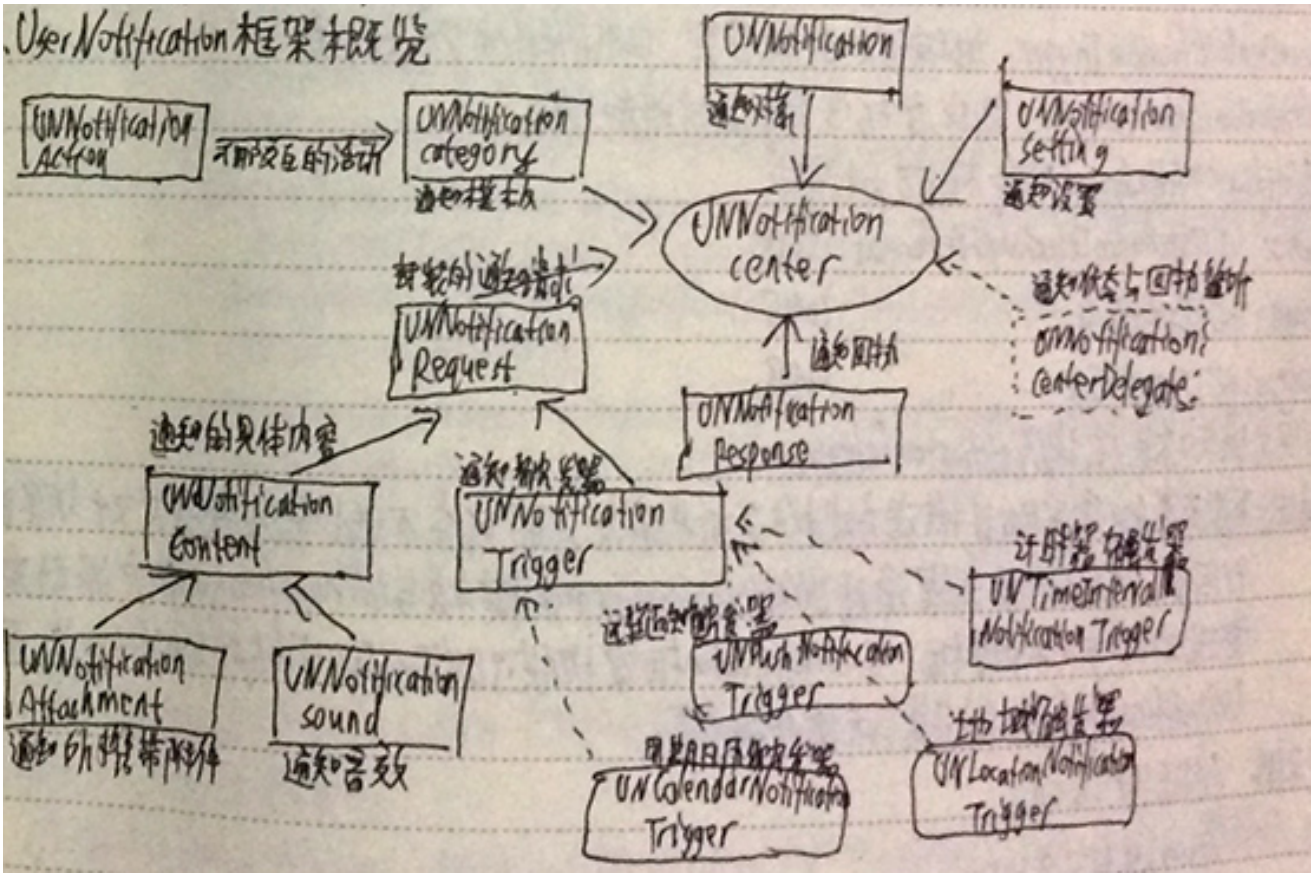
IOS10-通知框架 UserNotification

1. 简述：在 iOS10 系统中，通知被整合进了 UserNotification 框架，除了使通知的处理脱离了 UIApplication，通知功能的相关开发更加结构化和模块化，还新增了许多更加灵活的开发接口。现在，开发者可以为通知自定义 UI 模版，添加媒体附件。
2. UserNotification 特点：
   (1) 通知处理代码可以从 AppDelegate 中剥离
   (2) 通知的注册、设置、处理更加结构化，更易于模块开发
   (3) 支持自定义通知音效和启动图
   (4) 支持向通知内容添加媒体附件，例如音频，视频
   (5) 支持开发者自定义多套通知模版
   (6) 支持完全自定义的通知界面
   (7) 支持自定义通知中的用户交互按钮
   (8) 通知的触发更容易管理
3. UserNotification 框架概览



4. 核心类介绍

(1) UNNotificationCenter:通知管理中心，单例、通知的注册，接收通知后的回调处理等，是 UserNotification 框架的核心

(2) UNNotification:通知对象，其中封装了通知请求

(3) UNNotificationSettings:通知相关设置

(4) UNNotificationCategory:通知模板

(5) UNNotificationAction:用于定义通知模板中的用户交互行为

(6) UNNotificationRequest:注册通知请求，其中定义了通知的内容和触发方式

(7) UNNotificationContent:通知的具体内容

(8) UNNotificationTrigger:通知的触发器，由子类具体定义

(9) UNNotificationAttchment:通知的附件类，为通知内容添加媒体附件

(10) UNNotificationSound:定义通知音效

(11) UNPushNotificationTrigger:远程通知的触发器，UNNotificationTrigger 子类

(12) UNTimeIntervalNotificationTrigger:计时通知的触发器，UNNotificationTrigger 子类

(13) UNCalendarNotificationTrigger:周期通知的触发器，UNNotificationTrigger 的子类

(14) UNNotificationCenterDelegate:协议，其中方法用于监听通知状态

5. NSNotificationAttachment 附件格式及大小要求

Audio:kUTTypeAudioInterChangeFileFormat 5MB

Image:kUTTypeJPEG 10MB

Movie:kUTTypeMPEG 50MB

6. 自定义通知模板 UNNotificationCategory

(1) 简述：聊天类 App 常才用后台推送方式推送新消息，用户在不进入应用情况下对消息进行回复，这种功能就是通过 UNNotificationCategory 模版与 UNNotification 用户活动来实现的，关于文本回复框，UNNotification 框架提供了 UNTextInputNotificationAction 类，是 UNNotificationAction 的子类，示例代码如下：

```
#import "ViewController.h"
#import <UserNotifications/UserNotifications.h>

@interface ViewController ()<UNUserNotificationCenterDelegate>

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    [self registNotification];

    //创建用户活动
    //options可选参数如下:
    //UNNotificationActionOptionAuthenticationRequired//需要解锁下使用
    // UNNotificationActionOptionDestructive//是否显示破坏性指示
    //UNNotificationActionOptionForeground//是否允许活动在后台启动app
    //UNNotificationActionOptionNone//无设置
    UNTextInputNotificationAction *action = [UNTextInputNotificationAction actionWithIdentifier:@"action" title:@"活动" options:
        UNNotificationActionOptionAuthenticationRequired textInputButtonTitle:@"回复" textInputPlaceholder:@"请输入回复"];

    //创建通知模版
    UNNotificationCategory *category = [UNNotificationCategory categoryWithIdentifier:@"myNotificationCategoryText" actions:
        @[action] intentIdentifiers:@[] options:UNNotificationCategoryOptionCustomDismissAction];
    [[UNUserNotificationCenter currentNotificationCenter] setNotificationCategories:[NSSet setWithObjects:category, nil]];
```

```
    //创建通知内容
    UNMutableNotificationContent *content = [UNMutableNotificationContent new];
    content.badge = @1;
    content.body = @"这是ios10的新通知";
    content.sound = [UNNotificationSound defaultSound];
    content.title = @"这是主标题";
    content.subtitle = @"这是副标题";
    content.categoryIdentifier = @"myNotificationCategoryText";

    //创建通知触发器
    UNTimeIntervalNotificationTrigger *trigger = [UNTimeIntervalNotificationTrigger triggerWithTimeInterval:5 repeats:NO];

    //创建通知请求
    UNNotificationRequest *request = [UNNotificationRequest requestWithIdentifier:@"notificationDefaultText" content:content trigger:
        trigger];

    //发送通知
    [[UNUserNotificationCenter currentNotificationCenter] addNotificationRequest:request withCompletionHandler:^(NSError *
        _Nullable error) {
        NSLog(@"通知成功");
    }];

    //设置代理接收通知消息
    [[UNUserNotificationCenter currentNotificationCenter] setDelegate:self];

    //注意：系统模板最多支持4个用户交互按钮UNNotificationAction
}
```

```
- (void)registNotification{
    [[UNUserNotificationCenter currentNotificationCenter] requestAuthorizationWithOptions:UNAuthorizationOptionBadge|
        UNAuthorizationOptionSound|UNAuthorizationOptionAlert completionHandler:^(BOOL granted, NSError * _Nullable error) {
        NSLog(@"通知注册成功");
    }];
}

#pragma mark UNUserNotificationCenterDelegate

- (void)userNotificationCenter:(UNUserNotificationCenter *)center willPresentNotification:(UNNotification *)notification
    withCompletionHandler:(void (^)(UNNotificationPresentationOptions))completionHandler{
    NSLog(@"前台接收到通知");
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center didReceiveNotificationResponse:(UNNotificationResponse *)response
    withCompletionHandler:(void (^)())completionHandler{
    NSLog(@"后台接收到通知，并打开了应用");
}
```

7. 自定义通知模板 UI

(1) 打开 New->Targe，创建一个 NotificationContent 拓展，自带一个 storyboard 文件，和一个 NotificationViewController 类，这个类自动遵守了 UNNotificationContentExtension 协议，这个协议专门用来处理自定义 UI 的内容显示

(2) UNNotificationContentExtension 协议如下：

//接收到通知时调用

//开发者可以从 notification 对象中拿到附件等内容进行 UI 判断

-(void)didReceiveNotification:(UNNotification *)notification;

//当用户点击了通知中的用户交互按钮时会调用

//response 对象中有通知内容等相关信息，在回调中开发者可以传入一个 UNNotificationContentExtensitonResponseOption 参数告诉系统如何处理这次活动，如：将用户活动交由宿主 app 处理

-(void)didReceiveNotificationResponse:(UNNotificationResponse *)response completionHandler:(void(^)(completion:))

//另外还有一些关于媒体按钮的设置和播放暂停回调

注：自定义的通知界面上虽然可以放按钮，任何 UI 控件，但是不能和用户交互，唯一可以交互方式是通过协议中媒体按钮和其回调方法

(3) 使用定义好的 UI 模板

1) 拓展的 info.plist 里面 NSExtensionAttributes 字典进行一些配置，开发者可配置如下：

UNNotificationExtensionCategory:设置模版的 categoryId，用于和 UNNotificationContent 绑定

UNNotificationExtensitonInitialContentSizeRatio:宽高比，宽为固定

UNNotificationExtensitonDefaultContentHidden:是否隐藏系统默认通知界面

8. 通知回调的处理

见 6 例

9. 注册通知

见 6 例