项目知识点

好用的网址：

<http://www.gaituba.com/> 改图吧

[https://icon.wuruihong.com/#/ios](https://icon.wuruihong.com/" \l "/ios) 图标工程

<http://www.qidianlife.com/Singular/index.php?m=App&c=Article&a=index&id=829>

ios的99个绝佳资源

<https://blog.csdn.net/app_ios/article/details/52622309> 常用的第三方框架

<https://www.sioe.cn/yingyong/yanse-rgb-16/> RGB颜色转换

二、更新指定第三方库

终端执行：

pod update 库名

只安装新增库，不更新已安装的库

pod install  
把Podfile内全部的库更新重新安装

pod install --verbose --no-repo-update  
该命令只安装新添加的库，已更新的库忽略

pod update 库名 --verbose --no-repo-update  
该命令只更新指定的库，其它库忽略

XHInputView 好用得三方 评论，留言框

pod 'SJVideoPlayer'

pod 'SJBaseVideoPlayer’  非常好用的视频播放第三方，界面美观，视频流畅

//   一句代码实现  进入后台后使定时器持续运行

    [[UIApplication sharedApplication] beginBackgroundTaskWithExpirationHandler:^{

        [[UIApplication sharedApplication]endBackgroundTask:UIBackgroundTaskInvalid];

    }];

@property (nonatomic, assign) int trans\_id;    //业务id,避免关键字转换成非关键字

- (void)setValue:(id)value forUndefinedKey:(NSString \*)key

{

    if([key isEqualToString:@"id"]){

        self.trans\_id = [value intValue];

    }

}

//微信最新版本1.86SDK  AppDelegate要加上这两个方法

- (BOOL)application:(UIApplication \*)application handleOpenURL:(NSURL \*)url {

    return  [WXApi handleOpenURL:url delegate:self];

}

- (BOOL)application:(UIApplication \*)application openURL:(NSURL \*)url sourceApplication:(NSString \*)sourceApplication annotation:(id)annotation {

    return [WXApi handleOpenURL:url delegate:self];

}

- (BOOL)application:(UIApplication \*)application continueUserActivity:(NSUserActivity \*)userActivity restorationHandler:(void(^)(NSArray<id<UIUserActivityRestoring>> \* \_\_nullable restorableObjects))restorationHandler {

    return [WXApi handleOpenUniversalLink:userActivity delegate:self];

}

 知识点

自旋锁的优点在于，因为自旋锁不会引起调用者睡眠，所以不会进行线程调度，CPU 时间片轮转等耗时操 作。所有如果能在很短的时间内获得锁，自旋锁的效率远高于互斥锁。

缺点在于，自旋锁一直占用 CPU，他在未获得锁的情况下，一直运行－－自旋，所以占用着 CPU，如果 不能在很短的时 间内获得锁，这无疑会使 CPU 效率降低。

自旋锁不能实现递归调用。

自旋锁：atomic、OSSpinLock、dispatch\_semaphore\_t

互 斥 锁 ： pthread\_mutex 、 @ synchronized 、 NSLock 、 NSConditionLock 、 NSCondition 、 NSRecursiveLock

Block知识点：

Block主要分布在栈区\_ \_NSStackBlock\_ \_、堆区\_ \_NSMallocBlock\_ \_、全局区\_ \_NSGlobalBlock\_ \_

项目中遇到的问题

1 使用静态库的好处

1，模块化，分工合作

2，避免少量改动经常导致大量的重复编译连接

3，也可以重用，注意不是共享使用

动态库使用有如下好处：

1使用动态库，可以将最终可执行文件体积缩小

2使用动态库，多个应用程序共享内存中得同一份库文件，节省资源

3使用动态库，可以不重新编译连接可执行程序的前提下，更新动态库文件达到更新应用程序的目的。

//设置渲染色 是iOS7界面中的一个设置UIImage的渲染模式，你可以设置一个UIImage在渲染时是否使用当前视图的Tint Color。

vc1.tabBarItem.image = [[UIImage imageNamed:@"iconTab\_shouyehui"] imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];

如何防止反编译

  0.  本地数据加密。

  •  iOS应用防反编译加密技术之一：对NSUserDefaults，sqlite存储文件数据加密，保护帐号和关键信息

  0.  URL编码加密。

  •  iOS应用防反编译加密技术之二：对程序中出现的URL进行编码加密，防止URL被静态分析

  0.  网络传输数据加密。

  •  iOS应用防反编译加密技术之三：对客户端传输数据提供加密方案，有效防止通过网络接口的拦截获取数据

  0.  方法体，方法名高级混淆。

  •  iOS应用防反编译加密技术之四：对应用程序的方法名和方法体进行混淆，保证源码被逆向后无法解析代码

  0.  程序结构混排加密。

  •  iOS应用防反编译加密技术之五：对应用程序逻辑结构进行打乱混排，保证源码可读性降到最低

url中含有中文或特殊字符导致加载失败，需要进行一次转码。

- (NSString \*)getUTF8EncodeStringWithURLString:(NSString \*)urlString

{

    if (urlString && urlString.length > 0)

    {

        NSString \*encodedString = (NSString \*)

        CFBridgingRelease(CFURLCreateStringByAddingPercentEscapes(kCFAllocatorDefault,

                                                                  (CFStringRef)urlString,

                                                                  (CFStringRef)@"!$&'()\*+,-./:;=?@\_~%#[]",

                                                                  NULL,

                                                                  kCFStringEncodingUTF8));

        return encodedString;

    }

    else

    {

        return @"";

    }

}

dispatch queue分为下面三种：

 private dispatch queues，同时只执行一个任务，通常用于同步访问特定的资源或数据。

 global dispatch queue，可以并发地执行多个任务，但是执行完成的顺序是随机的。

 Main dispatch queue 它是在应用程序主线程上执行任务的

   影响App启动慢的因素

  0.  main()函数之前耗时的影响因素

  •  动态库加载越多，启动越慢。

  •  ObjC类越多，启动越慢

  •  C的constructor函数越多，启动越慢

  •  C++静态对象越多，启动越慢

  •  ObjC的+load越多，启动越慢

  0.  main()函数之后耗时的影响因素

  •  执行main()函数的耗时

  •  执行applicationWillFinishLaunching的耗时

  •  rootViewController及其childViewController的加载、view及其subviews的加载

  •  TCP为传输控制层协议，为面向连接、可靠的、点到点的通信；

  •  UDP为用户数据报协议，非连接的不可靠的点到多点的通信；

  •  TCP侧重可靠传输，UDP侧重快速传输。

 1.SDWebImage 加载图片的流程

1.入口 setImageWithURL:placeholderImage:options: 会先把 placeholderImage 显示，然后 SDWebImageManager 根据 URL 开始处理图片。

2.进入 SDWebImageManager-downloadWithURL:delegate:options:userInfo:，交给 SDImageCache 从缓存查找图片是否已经下载 queryDiskCacheForKey:delegate:userInfo:.

3.先从内存图片缓存查找是否有图片，如果内存中已经有图片缓存，SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo: 到 SDWebImageManager。

4.SDWebImageManagerDelegate 回调 webImageManager:didFinishWithImage: 到 UIImageView+WebCache 等前端展示图片。

5.如果内存缓存中没有，生成 NSInvocationOperation 添加到队列开始从硬盘查找图片是否已经缓存。

6.根据 URLKey 在硬盘缓存目录下尝试读取图片文件。这一步是在 NSOperation 进行的操作，所以回主线程进行结果回调 notifyDelegate:。

7.如果上一操作从硬盘读取到了图片，将图片添加到内存缓存中（如果空闲内存过小，会先清空内存缓存）。SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo:。进而回调展示图片。

8.如果从硬盘缓存目录读取不到图片，说明所有缓存都不存在该图片，需要下载图片，回调 imageCache:didNotFindImageForKey:userInfo:。

9.共享或重新生成一个下载器 SDWebImageDownloader 开始下载图片。

10.图片下载由 NSURLConnection 来做，实现相关 delegate 来判断图片下载中、下载完成和下载失败。

11.connection:didReceiveData: 中利用 ImageIO 做了按图片下载进度加载效果。

12.connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder 做图片解码处理。

13.图片解码处理在一个 NSOperationQueue 完成，不会拖慢主线程 UI。如果有需要对下载的图片进行二次处理，最好也在这里完成，效率会好很多。

14.在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成，imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader。

15.imageDownloader:didFinishWithImage: 回调给 SDWebImageManager 告知图片下载完成。

16.通知所有的 downloadDelegates 下载完成，回调给需要的地方展示图片。

17.将图片保存到 SDImageCache 中，内存缓存和硬盘缓存同时保存。写文件到硬盘也在以单独 NSInvocationOperation 完成，避免拖慢主线程。

18.SDImageCache 在初始化的时候会注册一些消息通知，在内存警告或退到后台的时候清理内存图片缓存，应用结束的时候清理过期图片。

19.SDWI 也提供了 UIButton+WebCache 和 MKAnnotationView+WebCache，方便使用。

20.SDWebImagePrefetcher 可以预先下载图片，方便后续使用。

7、如何异步下载多张小图最后合成一张大图？

使用Dispatch Group追加block到Global Group Queue,这些block如果全部执行完毕，就会执行Main Dispatch Queue中的结束处理的block。

dispatch\_queue\_t queue = dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0);

dispatch\_group\_t group = dispatch\_group\_create();

dispatch\_group\_async(group, queue, ^{ /\*加载图片1 \*/ });

dispatch\_group\_async(group, queue, ^{ /\*加载图片2 \*/ });

dispatch\_group\_async(group, queue, ^{ /\*加载图片3 \*/ });

dispatch\_group\_notify(group, dispatch\_get\_main\_queue(), ^{

        // 合并图片

});

1、入口 setImageWithURL:placeholderImage:options: 会先把 placeholderImage 显示，然后 SDWebImageManager 根据 URL 开始处理图片。

2、进入 SDWebImageManager-downloadWithURL:delegate:options:userInfo:，交给 SDImageCache 从缓存查找图片是否已经下载 queryDiskCacheForKey:delegate:userInfo:.

3、先从内存图片缓存查找是否有图片，如果内存中已经有图片缓存，SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo: 到 SDWebImageManager。

4、SDWebImageManagerDelegate 回调 webImageManager:didFinishWithImage: 到 UIImageView+WebCache 等前端展示图片。

5、如果内存缓存中没有，生成 NSInvocationOperation 添加到队列开始从硬盘查找图片是否已经缓存。

6、根据 URLKey 在硬盘缓存目录下尝试读取图片文件。这一步是在 NSOperation 进行的操作，所以回主线程进行结果回调 notifyDelegate:。

7、如果上一操作从硬盘读取到了图片，将图片添加到内存缓存中（如果空闲内存过小，会先清空内存缓存）。SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo:。进而回调展示图片。

8、如果从硬盘缓存目录读取不到图片，说明所有缓存都不存在该图片，需要下载图片，回调 imageCache:didNotFindImageForKey:userInfo:。

9、共享或重新生成一个下载器 SDWebImageDownloader 开始下载图片。

10、图片下载由 NSURLConnection 来做，实现相关 delegate 来判断图片下载中、下载完成和下载失败。

11、connection:didReceiveData: 中利用 ImageIO 做了按图片下载进度加载效果。connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder 做图片解码处理。

12、图片解码处理在一个 NSOperationQueue 完成，不会拖慢主线程 UI。如果有需要对下载的图片进行二次处理，最好也在这里完成，效率会好很多。

13、在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成，imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader。imageDownloader:didFinishWithImage: 回调给 SDWebImageManager 告知图片下载完成。

14、通知所有的 downloadDelegates 下载完成，回调给需要的地方展示图片。将图片保存到 SDImageCache 中，内存缓存和硬盘缓存同时保存。写文件到硬盘也在以单独 NSInvocationOperation 完成，避免拖慢主线程。

15、SDImageCache 在初始化的时候会注册一些消息通知，在内存警告或退到后台的时候清理内存图片缓存，应用结束的时候清理过期图片。

16、SDWI 也提供了 UIButton+WebCache 和 MKAnnotationView+WebCache，方便使用。

17、SDWebImagePrefetcher 可以预先下载图片，方便后续使用。

<https://github.com/lb2281075105/LBXMLYFM-Swift.git>

<https://github.com/lb2281075105/LBU25-Swift.git>   都是swift5.0的

//分享微信纯图片

 //1.创建多媒体消息结构体

    WXMediaMessage \*mediaMsg = [WXMediaMessage message];

    //2.创建多媒体消息中包含的图片数据对象

    WXImageObject \*imgObj = [WXImageObject object];

    //图片真实数据

    imgObj.imageData = [NSData dataWithContentsOfURL:photo.url];

    //多媒体数据对象

    mediaMsg.mediaObject = imgObj;

    //3.创建发送消息至微信终端程序的消息结构体

    SendMessageToWXReq \*req = [[SendMessageToWXReq alloc] init];

    //多媒体消息的内容

    req.message = mediaMsg;

    //指定为发送多媒体消息（不能同时发送文本和多媒体消息，两者只能选其一）

    req.bText = NO;

    //指定发送到会话(聊天界面)

    req.scene = WXSceneSession;

    //发送请求到微信,等待微信返回onResp

[WXApi sendReq:req];

项目资料：

idcp@365hy.com    Hykj0202  这是我们公司自己的开发者账号

**APP版本管理-后台管理**

**账号：admin    密码：xmkj2012**

芽儿阅读苹果发布账号： 3460269766@qq.com   Love8899

易高家居苹果发布账号：508463239@qq.com  w7825103WYJ

喂呗苹果开发者账号  ：

app@sh-alibaby.com   asdfghjB1                      com.alibaby.bear

这个是我们内测的企业打包的苹果账号：

apple@wind4app.com    Hykj0202wind4

直通车开发者账号

jjkjyxgs@163.com   Wxj401662503

友利开发账号

2696263813@qq.com    Zw520520

同城配送账号

[qiuqiuzhang85@qq.com](mailto:qiuqiuzhang85@qq.com)  Baidai85

快驴账号

[3469679937@qq.com](mailto:3469679937@qq.com)   Jcw129129

安誉科技账号

账号：[hr@bioanyu.com](mailto:hr@bioanyu.com)

密码：Agsbio2019

版路数字城苹果开发者账号：[329643118@qq.com](mailto:329643118@qq.com)

密码：Yubowangmei1210