一个iOS菜菜的白话文记录

不停的写博客不是为了炫耀什么，仅仅只是为了个人的一些学习总结，没有过多的什么意思，因为很多东西都能够在网络上找到。如Blog标题我只是一个iOS入门级菜鸟。只有当你的基础足够的扎实时候，才能像YYKit作者那样对iOS平台技术有如此深厚的理解。

Navigate…

- Blog
- Archives

后台执行一段代码

Oct 25th, 2015 1:22 am

记录一些使用后台的代码

方法一、让App每隔一段时间，都执行一段代码

```
#pragma mark UIApplicationDelegate

- (void)applicationDidEnterBackground:(UIApplication *)application
{
  if ([application respondsToSelector:@selector(setKeepAliveTimeout:handler:)])
  {
      [application setKeepAliveTimeout:600 handler:^{

          DDLogVerbose(@"KeepAliveHandler");

          // 这里写在后台执行的代码.
      }];
  }
}
```

注意:

1. 必须在Info.plist里设UIBackgroundModes键的array值之一voip字符串
2. timeout必须>=600
3. 唤醒app的时间间隔是不精准
4. 唤醒后只有10秒执行时间，即handler里的代码要在10秒类执行完，10秒后app再次被阻塞
5. 使用backgroundTimeRemaining属性，来返回剩余时间
6. 该函数的效果在回到前台运行时，依然会继续执行
7. clearKeepAliveTimeout函数用来清除handler

方法二、后台执行一次性的任务，好像最长是10分钟

```
// AppDelegate.h文件
@property (assign, nonatomic) UIBackgroundTaskIdentifier backgroundUpdateTask;

// AppDelegate.m文件

- (void)applicationDidEnterBackground:(UIApplication *)application {
        [self beingBackgroundUpdateTask];
     // 在这里加上你需要长久运行的代码
  [self endBackgroundUpdateTask];
}

- (void)beingBackgroundUpdateTask {
  self.backgroundUpdateTask = [[UIApplication sharedApplication]       beginBackgroundTaskWithExpirationHandler:^{
             [self endBackgroundUpdateTask];
  }];
}

- (void)endBackgroundUpdateTask {
     [[UIApplication sharedApplication] endBackgroundTask: self.backgroundUpdateTask];   self.backgroundUpdateTask = UIBackgroundTaskInvalid;
}
```

此种方法提交的后台任务优先级比较低，当系统内存紧张时，首先会关闭这种类似的后台任务。

方法三、当App进入后台之前，后台播放一个0 KB的mp3音频文件，来提高方法二申请的后台任务的权限

- 在plish文件中加入背景播放的支持

```
key: Required background modes
value: App plays audio
```

- 在AppDelegate的如下函数申请后台任务执行

```
- (void)applicationDidEnterBackground:(UIApplication *)application {

    //1. 开启一个后台任务
    myTask = [[UIApplication sharedApplication] beginBackgroundTaskWithExpirationHandler:^{
        [application endBackgroundTask:myTask];
        myTask = UIBackgroundTaskInvalid;
    }];

    //2. 后台完成的代码
    //开启一个NSTimer，不断的执行读取服务器数据
    //3. 完成后提交任务
    //如果是无限制重复执行的任务，可以不写如下两句
    //[application endBackgroundTask:myTask];
    //myTask = UIBackgroundTaskInvalid;
}
```

- 在`AppDelegate`如下方法，播放一个无声音的MP3文件，提高后台任务的权限

```
- (void)applicationWillResignActive:(UIApplication *)application {

    //在App即将失去焦点时，在后台播放一个无声的MP3，来提高后台任务的权限
    [self backgroudTaskViaMp3];
}


- (void)backgroudTaskViaMp3 {

    //1. 使用指定的MP3文件的url，
    NSString *string = [[NSBundle mainBundle] pathForResource:@"轻音乐 - 萨克斯回家" ofType:@"mp3"];

    //2. 把音频文件转换成url格式
    NSURL *url = [NSURL fileURLWithPath:string];

    //3. 使用音频文件的url，创建一个音频播放器
    _player = [[AVAudioPlayer alloc] initWithContentsOfURL:url error:nil];

    //4. 设置代理
//    _player.delegate = self;

    //5. 设置音乐播放次数为一直循环
    _player.numberOfLoops = -1;

    //6. 预播放
    [_player prepareToPlay];

    //7.
}
```

使用 UIBackgroundModes 后台完成获取数据

- 首先在Xcode工程中配置如下

- 在App启动完毕回调函数中，告诉系统App在后台，多长时间进行一次数据获取
  - 一定要设置application这个间隔时间
  - 否则，App程序 永远不能在后台被唤醒，执行任务
  - UIApplicationBackgroundFetchIntervalMinimum这个系统值，意思是告诉系统尽可能频繁的执行后台任务
  - 也应该指定一个你想要的的时间间隔
  - 例如，一个天气的应用程序，可能只需要几个小时才更新一次，iOS 将会在后台获取之间至少等待你指定的时间间隔

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{

    [application setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum];

    return YES;
}
```

- 当某个时刻不需要再执行后台数据获取任务时，设置时间间隔为 `never`

```
[application setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalNever];
```

- 最后在 AppDelegate.m 实现 `UIApplicationDelegate`的如下方法，完后后台数据获取的代码，当系统唤醒App时会回调执行如下函数
  - 注意: 只有 `30秒` 的时间来进行获取数据的操作
  - 最后一定要执行 `completionHandler`这个Block
    - 告诉系统任务操作结束
    - 系统会将更新UI之后的界面重新进行截图，并作为App切换时的缩略图

```
- (void)application:(UIApplication *)application performFetchWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler
{
    //如下模拟完成一个后台网络数据获取的操作

    NSURLSessionConfiguration *sessionConfiguration = [NSURLSessionConfiguration defaultSessionConfiguration];
    NSURLSession *session = [NSURLSession sessionWithConfiguration:sessionConfiguration];

    NSURL *url = [[NSURL alloc] initWithString:@"http://yourserver.com/data.json"];
    NSURLSessionDataTask *task = [session dataTaskWithURL:url
                                        completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

                                            if (error) {
                                                completionHandler(UIBackgroundFetchResultFailed);
                                                return;
                                            }

                                            // 解析响应/数据以决定新内容是否可用
                                            BOOL hasNewData = ...;

                                            //根据数据更新UI
                                            //根据数据更新UI
                                            if (hasNewData)
                                            {
                                                //1.
                                                dispatch_async(dispatch_get_main_queue(), {
                                                    //更新UI操作
                                                });

                                                //2. 告诉系统后台任务执行完毕
                                                completionHandler(UIBackgroundFetchResultNewData);

                                            } else {

                                                completionHandler(UIBackgroundFetchResultNoData);
                                            }
                                        }];

    // 开始任务
    [task resume];
}
```

测试后台数据获取

注意，下次需要改回来。 或者重新建立一个scheme专门用来测试后台任务。

Authored by Zain Oct 25th, 2015 1:22 am

« 最近面试都是问你时怎么做项目架构设计的 JavaScript与objc »

社交帐号登录:

- 微博
- QQ
- 人人
- 豆瓣
- 更多 »

最新最早最热

- 0 条评论
- 还没有评论，沙发等你来抢

说点什么吧...

发布

熊曾辉的技术博客正在使用多说