iOS 获取隐私权限大全

Wynter_Wang (/u/8b9cef3b9fa8) + 关注
2018.12.10 11:06 字数 274 阅读 270 评论 0 喜欢 0
(/u/8b9cef3b9fa8)

从iOS 10开始获取用户隐私数据都需要在info.plist文件中配置对应的权限。在没有配置权限下调系统接口直接闪退，也是很苹果。

目前涉及16种隐私权限，分别是蓝牙、日历、相机、通讯录、Face ID、健康分享、住宅配件、位置、麦克风、运动与健身、媒体与Apple Music、NFC、相册、提醒事项、Siri、 语音识别。

配置权限的XML格式

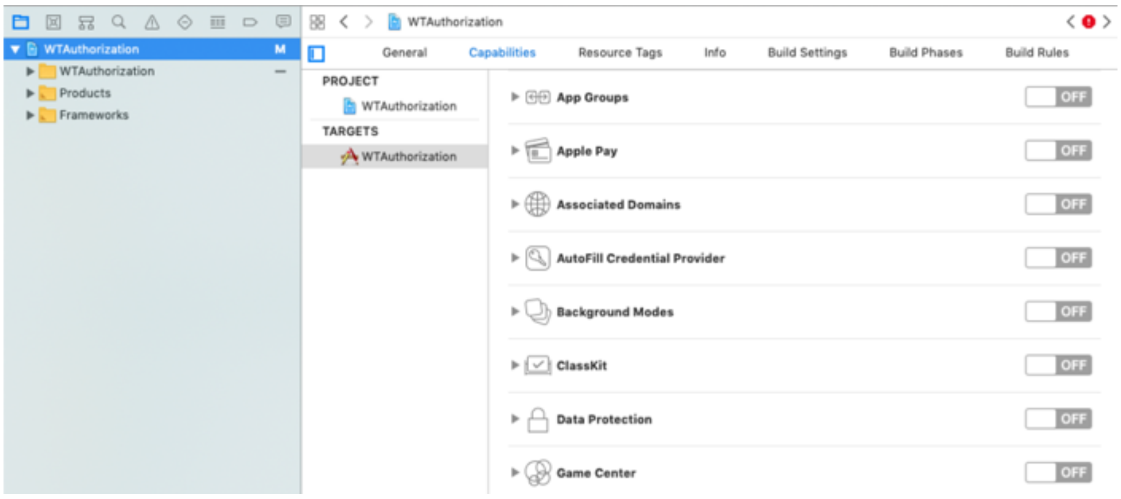
```
<!-- 蓝牙 -->
<key>NSBluetoothPeripheralUsageDescription</key>
<string>App需要您的同意,才能访问蓝牙</string>

<!-- 日历 -->
<key>NSCalendarsUsageDescription</key>
<string>App需要您的同意,才能访问日历</string>

<!-- 相机 -->
<key>NSCameraUsageDescription</key>
<string>App需要您的同意,才能访问相机</string>

<!-- 通讯录 -->
<key>NSContactsUsageDescription</key>
<string>App需要您的同意,才能访问通讯录</string>

<!-- Face ID -->
<key>NSFaceIDUsageDescription</key>
<string>App需要您的同意,才能访问Face ID</string>

<!-- 健康分享 -->
<key>NSHealthShareUsageDescription</key>
<string>App需要您的同意,才能访问健康分享</string>

<!-- 健康更新 -->
<key>NSHealthUpdateUsageDescription</key>
<string>App需要您的同意,才能访问健康更新 </string>

<!-- 住宅配件 -->
<key>NSHomeKitUsageDescription</key>
<string>App需要您的同意,才能访问住宅配件 </string>

<!-- 位置 -->
<key>NSLocationUsageDescription</key>
<string>App需要您的同意,才能访问位置</string>

<!-- 始终访问位置 -->
<key>NSLocationAlwaysUsageDescription</key>
<string>App需要您的同意,才能始终访问位置</string>
```

```
<!-- 在使用期间访问位置 -->
<key>NSLocationWhenInUseUsageDescription</key>
<string>App需要您的同意,才能在使用期间访问位置</string>

<!-- 麦克风 -->
<key>NSMicrophoneUsageDescription</key>
<string>App需要您的同意,才能访问麦克风</string>

<!-- 运动与健身 -->
<key>NSMotionUsageDescription</key>
<string>App需要您的同意,才能访问运动与健身</string>

<!-- 媒体资料库 -->
<key>NSAppleMusicUsageDescription</key>
<string>App需要您的同意,才能访问媒体资料库</string>

<!-- NFC -->
<key>NFCReaderUsageDescription</key>
<string>App需要您的同意,才能访问NFC</string>

<!-- 相册 -->
<key>NSPhotoLibraryUsageDescription</key>
<string>App需要您的同意,才能访问相册</string>

<!-- 提醒事项 -->
<key>NSRemindersUsageDescription</key>
<string>App需要您的同意,才能访问提醒事项</string>

<!-- Siri -->
<key>NSSiriUsageDescription</key>
<string>App需要您的同意,才能使用Siri功能</string>

<!-- 语音识别 -->
<key>NSSpeechRecognitionUsageDescription</key>
<string>App需要您的同意,才能使用语音识别功能</string>

<!-- 电视提供商 -->
<key>NSVideoSubscriberAccountUsageDescription</key>
<string>App需要您的同意,才能访问电视提供商</string>
```

需要引入的库

```
#import <CoreBluetooth/CoreBluetooth.h>
#import <EventKit/EventKit.h>
#import <AVFoundation/AVFoundation.h>
#import <Contacts/Contacts.h>
#import <LocalAuthentication/LocalAuthentication.h>
#import <HealthKit/HealthKit.h>
#import <HomeKit/HomeKit.h>
#import <CoreLocation/CoreLocation.h>
#import <CoreMotion/CoreMotion.h>
#import <StoreKit/StoreKit.h>
#import <CoreNFC/CoreNFC.h>
#import <Photos/Photos.h>
#import <Intents/Intents.h>
#import <Speech/Speech.h>
```

Capabilities中开启相应开关



Capabilities.png

请求获取隐私权限

```
#pragma mark - 请求访问蓝牙权限
- (void)requestAccessBluetoothWithCompletionBlock:(AuthorizationStateComple
tionBlock)completionBlock  {
    self.bluetoothManager = [[CBCentralManager alloc]initWithDelegate:self
queue:nil];
    self.bluetoothStateCompletionBlock = completionBlock;
}

#pragma mark - CBCentralManagerDelegate
- (void)centralManagerDidUpdateState:(CBCentralManager *)central {
    CBManagerState state = central.state;

    WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    if (state == CBManagerStateResetting) { // 重置或重新连接
        wtState = WTAuthorizationStateUnkonw;
    } else if (state == CBManagerStateUnsupported) {
        wtState = WTAuthorizationStateUnsupported;
    } else if (state == CBManagerStateUnauthorized) {
        wtState = WTAuthorizationStateUnauthorized;
    } else if (state == CBManagerStatePoweredOff) {
        wtState = WTAuthorizationStateDenied;
    } else if (state == CBManagerStatePoweredOn) {
        wtState = WTAuthorizationStateAuthorized;
    }

    [self respondWithState:wtState CompletionBlock:self.bluetoothStateCompl
etionBlock];
}
```

```
#pragma mark - 请求日历访问权限
- (void)requestAccessCalendarWithCompletionBlock:(AuthorizationStateCompletionBlock)completionBlock {
    EKAuthorizationStatus status = [EKEventStore authorizationStatusForEntityType:EKEntityTypeEvent];

    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    if (status == EKAuthorizationStatusNotDetermined) {
        EKEventStore *store = [[EKEventStore alloc] init];
        __weak __typeof(self)weakSelf = self;
        [store requestAccessToEntityType:EKEntityTypeEvent completion:^(BOOL granted, NSError *error) {
            if (error) {} else {
                if (granted) {
                    wtState = WTAuthorizationStateAuthorized;
                } else {
                    wtState = WTAuthorizationStateDenied;
                }
            }

            [weakSelf respondWithState:wtState CompletionBlock:completionBlock];
        }];
        return;
    } else if (status == EKAuthorizationStatusRestricted) {
        wtState = WTAuthorizationStateUnkonw;
    } else if (status == EKAuthorizationStatusDenied) {
        wtState = WTAuthorizationStateDenied;
    } else {
        wtState = WTAuthorizationStateAuthorized;
    }

    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
 #pragma mark - 请求相机访问权限
- (void)requestAccessCameraWithCompletionBlock:(AuthorizationStateCompletio
nBlock)completionBlock {
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    if ([UIImagePickerController isSourceTypeAvailable:UIImagePickerControl
lerSourceTypeCamera]) {

        AVAuthorizationStatus status = [AVCaptureDevice authorizationStatus
ForMediaType:AVMediaTypeVideo];

        if (status == AVAuthorizationStatusNotDetermined) {

            [AVCaptureDevice requestAccessForMediaType:AVMediaTypeVideo com
pletionHandler:^(BOOL granted) {
                if (granted) {
                    wtState = WTAuthorizationStateAuthorized;
                }else{
                    wtState = WTAuthorizationStateDenied;
                }
                [weakSelf respondWithState:wtState CompletionBlock:completi
onBlock];
            }];
            return;
        } else if (status == AVAuthorizationStatusRestricted) {
        } else if (status == AVAuthorizationStatusDenied) {
            wtState = WTAuthorizationStateDenied;
        } else {
            wtState = WTAuthorizationStateAuthorized;
        }

    } else {
        wtState = WTAuthorizationStateUnsupported;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - 请求通讯录访问权限
- (void)requestAccessContactsWithCompletionBlock:(AuthorizationStateComplet
ionBlock)completionBlock {
    CNAuthorizationStatus status = [CNContactStore authorizationStatusForEn
tityType:CNEntityTypeContacts];
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    if (status == CNAuthorizationStatusNotDetermined) {
        CNContactStore *contactStore = [[CNContactStore alloc] init];
        [contactStore requestAccessForEntityType:CNEntityTypeContacts compl
etionHandler:^(BOOL granted, NSError *error) {
            if (error) {} else {
                if (granted) {
                    wtState = WTAuthorizationStateAuthorized;
                } else {
                    wtState = WTAuthorizationStateDenied;
                }
            }
            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == CNAuthorizationStatusRestricted) {
    } else if (status == CNAuthorizationStatusDenied) {
        wtState = WTAuthorizationStateDenied;
    } else {
        wtState = WTAuthorizationStateAuthorized;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - Face ID访问权限
- (void)requestAccessFaceIDWithCompletionBlock:(AuthorizationStateCompletio
nBlock)completionBlock {
    if ([UIDevice currentDevice].systemVersion.floatValue < 11.0f) {
        [self respondWithState:WTAuthorizationStateUnsupported CompletionBl
ock:completionBlock];
    }
    LAContext *authenticationContext = [[LAContext alloc]init];
    NSError *error = nil;
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    BOOL canEvaluatePolicy = [authenticationContext canEvaluatePolicy:LAPol
icyDeviceOwnerAuthenticationWithBiometrics error:&error];
    if (canEvaluatePolicy) {
        if (authenticationContext.biometryType == LABiometryTypeFaceID) {
            [authenticationContext evaluatePolicy:LAPolicyDeviceOwnerAuthen
ticationWithBiometrics localizedReason:@"" reply:^(BOOL success, NSError *
_Nullable error) {
                if (error) {} else {
                    if (success) {
                        wtState = WTAuthorizationStateAuthorized;
                    } else {
                        wtState = WTAuthorizationStateDenied;
                    }
                }
                [weakSelf respondWithState:wtState CompletionBlock:completi
onBlock];
            }];
            return;
        } else {
            wtState = WTAuthorizationStateUnsupported;
        }
    }

    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - 获取健康心率 需要具体权限可以修改 HKQuantityTypeIdentifier
- (void)requestAccessHealthWithCompletionBlock:(AuthorizationStateCompletio
nBlock)completionBlock {
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    if ([HKHealthStore isHealthDataAvailable]) {
        HKHealthStore *healthStore = [[HKHealthStore alloc] init];
        __weak __typeof(self)weakSelf = self;
        HKQuantityType *heartRateType = [HKQuantityType quantityTypeForIden
tifier:HKQuantityTypeIdentifierHeartRate];
        HKAuthorizationStatus status = [healthStore authorizationStatusForT
ype:heartRateType];

        if (status == HKAuthorizationStatusNotDetermined) {
            NSSet *typeSet = [NSSet setWithObject:heartRateType];
            [healthStore requestAuthorizationToShareTypes:typeSet readTypes
:typeSet completion:^(BOOL success, NSError * _Nullable error) {
                if (success) {
                    HKAuthorizationStatus status = [healthStore authorizati
onStatusForType:heartRateType];
                    if (status == HKAuthorizationStatusNotDetermined) {
                        wtState = WTAuthorizationStateUnauthorized;
                    } else if (status == HKAuthorizationStatusSharingAuthor
ized) {
                        wtState = WTAuthorizationStateAuthorized;
                    } else {
                        wtState = WTAuthorizationStateDenied;
                    }
                }
                [weakSelf respondWithState:wtState CompletionBlock:completi
onBlock];
            }];
            return;
        } else if (status == HKAuthorizationStatusSharingAuthorized) {
            wtState = WTAuthorizationStateAuthorized;
        } else {
            wtState = WTAuthorizationStateDenied;
        }
    } else {
        wtState = WTAuthorizationStateUnsupported;
    }

    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - HomeKit
- (void)requestAccessHomeKitWithCompletionBlock:(AuthorizationStateCompleti
onBlock)completionBlock {
    self.homeManager = [[HMHomeManager alloc] init];
    self.homeManager.delegate = self;
    self.homeKitCompletionBlock = completionBlock;
}

#pragma mark - HMHomeManagerDelegate
- (void)homeManagerDidUpdateHomes:(HMHomeManager *)manager {
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    if (manager.homes.count > 0) {
        wtState = WTAuthorizationStateAuthorized;
       [self respondWithState:wtState CompletionBlock:self.homeKitCompletio
nBlock];
    } else {
        __weak __typeof(self)weakSelf = self;
        __weak HMHomeManager *weakHomeManager = manager;
        [manager addHomeWithName:@"Test Home" completionHandler:^(HMHome *
_Nullable home, NSError * _Nullable error) {
            if (error) {
                wtState = WTAuthorizationStateAuthorized;
            } else {
                if (error.code == HMErrorCodeHomeAccessNotAuthorized) {
                    wtState = WTAuthorizationStateDenied;
                }
            }

            [weakSelf respondWithState:wtState CompletionBlock:self.homeKit
CompletionBlock];

            if (home) {
                [weakHomeManager removeHome:home completionHandler:^(NSErro
r * _Nullable error) {
                }];
            }
        }];
    }
}
```

```
#pragma mark - 位置访问权限
- (void)requestAccessLocationWithCompletionBlock:(AuthorizationStateComplet
ionBlock)completionBlock {
    self.locationCompletionBlock = completionBlock;
    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate = self;
    [self.locationManager requestWhenInUseAuthorization];
    [self.locationManager startUpdatingLocation];
}

#pragma mark - CLLocationManagerDelegate
- (void)locationManager:(CLLocationManager *)manager didChangeAuthorization
Status:(CLAuthorizationStatus)status {
    WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    switch (status) {
        case kCLAuthorizationStatusNotDetermined:{
            wtState = WTAuthorizationStateUnauthorized;
            break;
        }
        case kCLAuthorizationStatusRestricted:{
            break;
        }
        case kCLAuthorizationStatusDenied:{
            wtState = WTAuthorizationStateDenied;
            break;
        }
        case kCLAuthorizationStatusAuthorizedAlways:{
            wtState = WTAuthorizationStateAuthorized;
            break;
        }
        case kCLAuthorizationStatusAuthorizedWhenInUse:{
            wtState = WTAuthorizationStateAuthorized;
            break;
        }
        default:
            break;
    }
    [self respondWithState:wtState CompletionBlock:self.locationCompletionB
lock];
}
```

```
#pragma mark - 麦克风
- (void)requestAccessMicrophoneWithCompletionBlock:(AuthorizationStateCompl
etionBlock)completionBlock {
    __weak __typeof(self)weakSelf = self;
   __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    AVAuthorizationStatus status = [AVCaptureDevice authorizationStatusForM
ediaType:AVMediaTypeAudio];

    if (status == AVAuthorizationStatusNotDetermined) {
        [[AVAudioSession sharedInstance] requestRecordPermission:^(BOOL gra
nted) {
            if (granted) {
                wtState = WTAuthorizationStateAuthorized;
            } else {
                wtState = WTAuthorizationStateDenied;
            }
            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == AVAuthorizationStatusRestricted) {
    } else if (status == AVAuthorizationStatusDenied) {
       wtState = WTAuthorizationStateDenied;
    } else {
        wtState = WTAuthorizationStateAuthorized;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - 运动与健身 不需要回调
- (void)requestAccessMotionWithCompletionBlock:(AuthorizationStateCompletio
nBlock)completionBlock {
    __weak __typeof(self)weakSelf = self;
    CMMotionActivityManager *manager = [[CMMotionActivityManager alloc] ini
t];
    NSOperationQueue *queue = [[NSOperationQueue alloc] init];
    [manager startActivityUpdatesToQueue:queue withHandler:^(CMMotionActivi
ty * _Nullable activity) {
        [manager stopActivityUpdates];
        [weakSelf respondWithState:WTAuthorizationStateAuthorized Completio
nBlock:completionBlock];
    }];
}
```

(/apps/redir
utm_source
banner-clicl

```
#pragma mark - 媒体与Apple Music
- (void)requestAccessMediaWithCompletionBlock:(AuthorizationStateCompletion
Block)completionBlock {
    if ([UIDevice currentDevice].systemVersion.floatValue < 9.3f) {
        [self respondWithState:WTAuthorizationStateUnsupported CompletionBl
ock:completionBlock];
    }
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;

    SKCloudServiceAuthorizationStatus status = [SKCloudServiceController au
thorizationStatus];
    if (status == SKCloudServiceAuthorizationStatusNotDetermined) {

        [SKCloudServiceController requestAuthorization:^(SKCloudServiceAuth
orizationStatus status) {
            switch (status) {
                case SKCloudServiceAuthorizationStatusNotDetermined: {
                    wtState = WTAuthorizationStateUnauthorized;
                }
                    break;
                case SKCloudServiceAuthorizationStatusRestricted: {
                }
                    break;
                case SKCloudServiceAuthorizationStatusDenied: {
                    wtState = WTAuthorizationStateDenied;
                }
                    break;
                case SKCloudServiceAuthorizationStatusAuthorized: {
                    wtState = WTAuthorizationStateAuthorized;
                }
                    break;
                default:
                    break;
            }
            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == SKCloudServiceAuthorizationStatusRestricted) {
    } else if (status == SKCloudServiceAuthorizationStatusDenied) {
        wtState = WTAuthorizationStateDenied;
    } else{
        wtState = WTAuthorizationStateAuthorized;
    }

    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

青藤之
喜欢你的

```
#pragma mark - NFC
- (void)requestAccessNFCWithCompletionBlock:(AuthorizationStateCompletionBl
ock)completionBlock {
    if ([UIDevice currentDevice].systemVersion.floatValue < 11.0f) {
        [self respondWithState:WTAuthorizationStateUnsupported CompletionBl
ock:completionBlock];
    }
    NFCNDEFReaderSession *session = [[NFCNDEFReaderSession alloc]initWithDe
legate:self queue:nil invalidateAfterFirstRead:YES];
    [session beginSession];
    self.NFCCompletionBlock = completionBlock;
}

#pragma mark - NFCNDEFReaderSessionDelegate
- (void)readerSession:(NFCNDEFReaderSession *)session didInvalidateWithErro
r:(NSError *)error {
    [self respondWithState:WTAuthorizationStateUnsupported CompletionBlock:
self.NFCCompletionBlock];
}

- (void)readerSession:(NFCNDEFReaderSession *)session didDetectNDEFs:(NSArr
ay<NFCNDEFMessage *> *)messages {
        [self respondWithState:WTAuthorizationStateAuthorized CompletionBlo
ck:self.NFCCompletionBlock];
}
```

```
#pragma mark - 相册权限
- (void)requestAccessPhotosWithCompletionBlock:(AuthorizationStateCompletio
nBlock)completionBlock {
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    PHAuthorizationStatus status = [PHPhotoLibrary authorizationStatus];
    if (status == PHAuthorizationStatusNotDetermined) {
        [PHPhotoLibrary requestAuthorization:^(PHAuthorizationStatus status
) {
            if (status == PHAuthorizationStatusNotDetermined) {
                wtState = WTAuthorizationStateUnauthorized;
            } else if (status == PHAuthorizationStatusRestricted) {

            } else if (status == PHAuthorizationStatusDenied) {
                wtState = WTAuthorizationStateDenied;
            } else {
                wtState = WTAuthorizationStateAuthorized;
            }
            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == PHAuthorizationStatusRestricted) {
    } else if (status == PHAuthorizationStatusDenied) {
        wtState = WTAuthorizationStateDenied;
    } else {
        wtState = WTAuthorizationStateAuthorized;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - 提醒事项
- (void)requestAccessRemindersWithCompletionBlock:(AuthorizationStateComple
tionBlock)completionBlock {
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    EKAuthorizationStatus status = [EKEventStore authorizationStatusForEnti
tyType:EKEntityTypeReminder];

    if (status == EKAuthorizationStatusNotDetermined) {
        EKEventStore *store = [[EKEventStore alloc] init];

        [store requestAccessToEntityType:EKEntityTypeReminder completion:^(
BOOL granted, NSError * _Nullable error) {
            if (error) {} else {
                if (granted) {
                    wtState = WTAuthorizationStateAuthorized;
                } else {
                    wtState = WTAuthorizationStateDenied;
                }
            }

            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == EKAuthorizationStatusRestricted) {

    } else if (status == EKAuthorizationStatusDenied) {
       wtState = WTAuthorizationStateDenied;

    } else {
        wtState = WTAuthorizationStateAuthorized;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - Siri
- (void)requestAccessSiriWithCompletionBlock:(AuthorizationStateCompletionB
lock)completionBlock {
    if ([UIDevice currentDevice].systemVersion.floatValue < 10.0f) {
        [self respondWithState:WTAuthorizationStateUnsupported CompletionBl
ock:completionBlock];
    }
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    INSiriAuthorizationStatus status = [INPreferences siriAuthorizationStat
us];
    if (status == INSiriAuthorizationStatusNotDetermined) {
        [INPreferences requestSiriAuthorization:^(INSiriAuthorizationStatus
 status) {
            if (status == INSiriAuthorizationStatusNotDetermined) {
                wtState = WTAuthorizationStateUnauthorized;
            } else if (status == INSiriAuthorizationStatusDenied) {
                wtState = WTAuthorizationStateDenied;
            } else if (status == INSiriAuthorizationStatusAuthorized) {
                wtState = WTAuthorizationStateAuthorized;
            }
            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == EKAuthorizationStatusRestricted) {

    } else if (status == EKAuthorizationStatusDenied) {
        wtState = WTAuthorizationStateDenied;
    } else {
        wtState = WTAuthorizationStateAuthorized;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

```
#pragma mark - 语音识别
- (void)requestAccessSpeechWithCompletionBlock:(AuthorizationStateCompletio
nBlock)completionBlock {
    if ([UIDevice currentDevice].systemVersion.floatValue < 10.0f) {
         [self respondWithState:WTAuthorizationStateUnsupported CompletionB
lock:completionBlock];
    }
    __weak __typeof(self)weakSelf = self;
    __block WTAuthorizationState wtState = WTAuthorizationStateUnkonw;
    SFSpeechRecognizerAuthorizationStatus status = [SFSpeechRecognizer auth
orizationStatus];

    if (status == SFSpeechRecognizerAuthorizationStatusNotDetermined) {
        [SFSpeechRecognizer requestAuthorization:^(SFSpeechRecognizerAuthor
izationStatus status) {

            if (status == SFSpeechRecognizerAuthorizationStatusNotDetermine
d) {
                wtState = WTAuthorizationStateUnauthorized;

            } else if (status == SFSpeechRecognizerAuthorizationStatusDenie
d) {
                wtState = WTAuthorizationStateDenied;

            } else if (status == SFSpeechRecognizerAuthorizationStatusAutho
rized) {
                wtState = WTAuthorizationStateAuthorized;
            }
            [weakSelf respondWithState:wtState CompletionBlock:completionBl
ock];
        }];
        return;
    } else if (status == SFSpeechRecognizerAuthorizationStatusDenied) {
        wtState = WTAuthorizationStateDenied;
    } else if (status == SFSpeechRecognizerAuthorizationStatusRestricted) {

    } else {
        wtState = WTAuthorizationStateAuthorized;
    }
    [self respondWithState:wtState CompletionBlock:completionBlock];
}
```

隐私权限请求完成应在主线程中完成回调

```
#pragma mark - 在主线程中完成回调
- (void)respondWithState:(WTAuthorizationState)state CompletionBlock:(Autho
rizationStateCompletionBlock)completionBlock {
    dispatch_async(dispatch_get_main_queue(), ^{
        if (completionBlock) {
            completionBlock(state);
        }
    });
}
```

总结

获取隐私权限需要分为四步：

- 在Info.plist文件中配置应用所需权限；
- 在项目的Targets->Capabilities中开启相应开关，目前Siri、Health、NFC、HomeKit

需要开启；

- 引入相关库；
- 使用代码获取对应的隐私权限。

参考

Protecting the User’s Privacy (https://developer.apple.com/documentation/uikit/core_app/protecting_the_user_s_privacy?language=objc)
Checking and Requesting Access to Data Classes in Privacy Settings (https://developer.apple.com/library/archive/samplecode/PrivacyPrompts/Introduction/Intro.html)

小礼物走一走，来简书关注我

赞赏支持

iOS (/nb/1906453)

举报文章 © 著作权归作者所有

Wynter_Wang (/u/8b9cef3b9fa8)
写了 16952 字，被 31 人关注，获得了 46 个喜欢
(/u/8b9cef3b9fa8)

＋关注

喜欢

更多分享

(/apps/redirect?utm_source=note-bottom-click)

登录 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-com 后发表评论

评论

智慧如你，不想发表一点想法 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-nocomments-text) 咩~

(/p/f7172e24be4f?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio

[续]iOS开发中的这些权限，你搞懂了吗？ (/p/f7172e24be4f?utm_campa...

前言 上篇文章iOS开发中的这些权限，你搞懂了吗？介绍了一些常用权限的获取和请求方法，知道这些方法的使用基本上可以搞定大部分应用的权限访问的需求。但是，这些方法并不全面，不能涵盖住所有权限

Jack_lin (/u/ef991f6d241c?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

(/p/86b773ae2307?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio

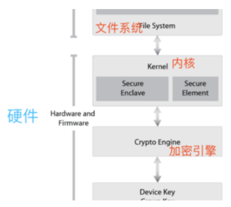
iOS开发中权限再度梳理 (/p/86b773ae2307?utm_campaign=maleskine...

前言 开源库JLAuthorizationManager :Objective-C版本 Swift版本 上篇文章iOS开发中的这些权限，你搞懂了吗？介绍了一些常用权限的获取和请求方法，知道这些方法的使用基本上可以搞定大部分应用的权限

Jack_lin (/u/ef991f6d241c?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

(/p/3a151735fa89?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendatio

iOS Security 苹果安全白皮书都讲了些啥？ (/p/3a151735fa89?utm_cam...

原文地址 使用Google 按段落翻译（上传文档的不太准），通读一遍后去掉设备或网络的超专业术语，也算是筛选掉对开发者意义不大的信息吧！ 苹果设计的 iOS 平台向来是以安全为核心，此次白皮书大概讲了

si1ence (/u/69aaea159515?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

App Programming Guide for iOS <2.Expected App B... (/p/7e05eb125...

App Programming Guide for iOS翻译 https://developer.apple.com/library/content/documentation/iPhone/Conceptual/iPhoneOSProgrammi...

鋼鉄侠 (/u/b92ff2c59c74?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

iOS权限配置 (/p/daa77194175e?utm_campaign=maleskine&utm_cont...

访问相应的功能时，要在plist配置相应的权限 摄像头(相机)权限 相册权限 蓝牙权限 日历权限 访问通讯录 访问使用Face ID 访问用户位置 访问及更改用户健康数据 访问设备麦克风 访问设备加速度计 访问用户提醒

枯流水 (/u/b6d791614c88?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

iOS简单实用的手写签名实现 (/p/532ed3bd2a04?utm_campaign=males...

文章来源：爱听音乐的狗博客 在iOS中实现手写签名主要运用到的知识点：UIBezierPath、手势、图片的截取、图片的压缩 签名封装 新建"HJSignatureView"继承自UIView，在分装类的.h文件中提供"清除签

爱听音乐的狗 (/u/df3254a31733?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

宋强 坚持分享第253天 (/p/6fe8d507f43f?utm_campaign=maleskine&...

今天是孩子四岁的生日，去蛋糕店订了一个很大的蛋糕，早早的送到了幼儿园，让他感受下在幼儿园和小朋友一起过生的感受，下午老师发来了照片，看他跟老师同学一起庆祝，跟老师同学一起分享蛋糕，很开

漫步奋斗路 (/u/dff80bca6979?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

我讲《孔乙己》(/p/62321a8ca768?utm_campaign=maleskine&utm_c...

今天和我十班学生共同学习了《孔乙己》，感慨颇多。《孔乙己》老文章了。对这篇文章的解读就如在一片花生地里拾花生，都有成百上千的人拾过了，还有落下的吗？ 于是这篇文章我又读了4节课...... 不是为了

寨子外的天空 (/u/1b4e57a7d1ea?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio

---

行香子两阕 (/p/8308ab545993?utm_campaign=maleskine&utm_cont...

2008-12-28 15:26 [一]行香子。勿忘我 雨后黄昏，寂寂春林。晚风潜，凉意三分。拈花独立，默默沉吟，念泪随风，风随雨，雨随云。 勿忘丝网，勿忘千结。凭君呵，醉里殷勤。心葩欲绽，往事谁询？问可知

高天洁雨18 (/u/c5a02f9e3689?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendatio