系统定位

1. iOS8 以前使用 CLLocationManager：

   (1) 导入头文件<CoreLocation/CoreLocation>

   (2) 创建位置管理者 CLLocationManager，并添加到属性

   (3) 设置代理，遵守协议、实现协议方法，再代理方法中获取位置信息

   (4) 调用开始更新位置方法

   (5) 设置每隔多远定位一次和精确度，精确度越高越耗电，定位时间越长

   (6) 请求授权，iOS6 后，苹果开始加强保护用户隐私，在 info.plist 文件中定义 key 来提醒用户，提高用户允许定位的概率

   Privacy-Location Usage Description　　(String，Ios8 以前定位使用说明)

   (7) 如果要后台定位，需打开后台模式：Background Modes => Location updates

2. iOS8 之后使用 CLLocationManager

   (1) Ios8 以后，苹果进一步加强隐私保护，不会主动弹出对话框，需实现 request 方法，并在 plist 中设置相应的 key，才会弹出对话框

   (2) 当程序当前授权状态为未决定时，在前台时请求定位服务许可时使用 requestWhenInUseAuthorization。需在 plist 文件中设置一个 key：NSLocationWhenInUseUsageDescription，若不设置，系统忽略定位请求

   (3) 当用户授权 when-in-use 时，程序在前台可启用大部分定位服务，若想后台定位，需开启后台定位模式，但在状态栏会出现蓝条提示用户程序正在定位

   (4) 当程序当前授权状态为未决定时，请求前后台定位服务授权时使用 requestAlwaysAuthorization，必须在 plist 中设置一个 key:NSLocationAlwaysUsageDescription

   (5) 注意：

      1) iOS8 之后，想要定位，必须调用其中一个 request 授权方法。

      2) 如果两个授权方法都执行，会有如下情况：

         when-in-use 写前面，第一次打开程序请求授权，如果勾选了后台模式，进入后台会出现蓝条提示正在定位。第二次启动程序时 always 请求授权，之后进

入后台就不会出现蓝条了；always 写前面，只会请求一次授权，因为 when-in-use 不会执行，因为 always 授权已获得所有定位能力。

(6) 判断是否开启了定位服务

在定位前要先判断是否开启了定位服务，还有用户是否允许定位。

(7) 适配版本号的方法

when-in-use 和 always 是 Ios8 之后的方法，在 Ios7 会 crash，此时需作判断：

1)[[UIDevice currentDevice].systemVersion floadValue] >= 8.0

2)[_locationMa responseToSelector:@selector(requestWhenInUseAuthorization)]

(8) 代理方法的回调信息

当位置管理器获取到位置后,调用 locationManager:didUpdateLocations 方法，返回 location 信息

1) coordinate:当前位置坐标 latitude:纬度 longitude:精度

2) altitude:海拔，高度

3) horizontalAccuracy:纬度和经度的精度

4) verticalAccuracy:垂直精度(获取不到海拔时为负数)

5) course:行进方向，0 为真北

6) speed:以米/秒为单位的行进速度

7) description:位置描述信息

3. iOS9.0 之后使用 CLLocationManager

(1) iOS9 之后有一种新的请求定位方法 requestLocation，作用是：按照定位精确度从低到高排序，逐个进行定位，如果取到的位置不是精确度最高的那个，也会在定位超时后，通过代理告诉外界。

(2) requestLocation 必须实现 didUpdateLocation 和 didFailWithError 方法，但只会调用一次

(3) 使用 requestLocation 就不能同时使用 startUpdateLocation

(4) 实现 requestWhenInUseAuthorization 或 requestAlwaysAuthorization 方法，并设好相应的 key

(5) 默认在前台授权模式下不能后台定位，即使已勾选后台模式，还需设置 allowsBackgroundLocationUpdates 属性为 YES，需使用-responsedToSelector 判断，当定位完成时，设置为 NO，并且不再定位跟踪

4. 后台定位练习及 MYLocationManager 管理类封装

(1) 后台定位

1) 导入 CoreLocation 静态库，引入<CoreLocation/CoreLocation.h>

2) 工程文件=>Target=>Background Modes 开启 Location updates 服务

3) info.plist 添加铅后台定位授权声明：NSLocationAlwaysUsageDescription

4) 在 AppDelegate 代理方法里做相应定位处理

```
#pragma mark UIApplicationDelegates

- (void)applicationDidBecomeActive{
    _originalTime = 0;
    [[MYLocationManager defaultManager] applicationDidBecomeActive];
    [_timer setFireDate:[NSDate distantFuture]];
}

- (void)applicationDidEnterBackground{
    [_timer setFireDate:[NSDate dateWithTimeIntervalSinceNow:10]];
    _originalTime = CFAbsoluteTimeGetCurrent();
    [[MYLocationManager defaultManager] applicationDidEnterBackground];
}

- (void)applicationWillTerminate{
    [[MYLocationManager defaultManager] applicationWillTerminate];
}
```

5) 开启定时器 NSTimer，添加到运行循环

6) 实现定时器触发方法

```
#pragma mark private methods

- (void)onTimeFired{
    if(_originalTime != 0){
        [_logLabel setText:[NSString stringWithFormat:@"已经后台运行了%.1lf分钟", (CFAbsoluteTimeGetCurrent() - _originalTime)/60]];
    }
    [[MYLocationManager defaultManager] startUpdatingLocationWithReverseGeo:YES];
}
```

7）设置定位回调：

```
_timer = [[NSTimer alloc] initWithFireDate:[NSDate dateWithTimeIntervalSinceNow:10] interval:10 target:self selector:@selector(onTimeFired) userInfo:nil
   repeats:YES];
[[NSRunLoop currentRunLoop] addTimer:_timer forMode:NSRunLoopCommonModes];
[_timer setFireDate:[NSDate distantFuture]];
[[MYLocationManager defaultManager] startUpdatingLocationWithReverseGeo:YES];
[[MYLocationManager defaultManager] setLocationFinishBlock:^(CLLocationCoordinate2D location, CLPlacemark *addressDetail, NSError *error){
   [[MYLocationManager defaultManager] stopUpdatingLocation];
   if(!addressDetail){//仅定位
   }else{//获取了地址详情
      if(!error){
         [self.logLabel setAlpha:0];
         [_logLabel setText:[NSString stringWithFormat:@"当前定位城市:%@", addressDetail.locality]];
         [UIView animateWithDuration:0.33 animations:^{
            [self.logLabel setAlpha:1];
         }];
      }
   }
}];
```

(2) MYLocationManager 管理类封装

```
#import <Foundation/Foundation.h>
#import <CoreLocation/CoreLocation.h>
#import <UIKit/UIKit.h>

//----------------------------------------------------------CLLocationManagerOptions----------------------------------------------------------//

@interface CLLocationManagerOptions : NSObject

/**
 * 设置定位距离过滤参数 (当本次定位和上次定位之间的距离大于或等于这个值时，调用代理方法)
 */
@property(assign, nonatomic) CLLocationDistance distanceFilter;

/**
 * 设置定位精度(精度越高越耗电)
 */
@property(assign, nonatomic) CLLocationAccuracy desiredAccuracy;

@end
```

```
//----------------------------------------------------------MYLocationManager----------------------------------------------------------//

/**
* @ note 若addressDetail = nil即为定位回调，否者为地址反编译结果的回调
*/
typedef void(^MYLocationFinshBlock)(CLLocationCoordinate2D location, CLPlacemark *addressDetail, NSError *error);

@interface MYLocationManager : NSObject

/**
* 定位服务管理对象
*/
@property (strong, nonatomic) CLLocationManager *locationManager;

/**
* 定位服务配置选项
* @ note 为nil时所有参数为默认
*/
@property (strong, nonatomic) CLLocationManagerOptions *options;
```

```
/**
* 定位授权状态
* @ note 状态值为kCLAuthorizationStatusDenied说明定位服务开启被用户拒绝了或者系统定位服务没开
*    此时可以弹窗提示用户去设置中开启定位服务，最好直接跳转至设置定位服务页面，增强用户体验
*/
@property (assign, nonatomic) CLAuthorizationStatus authorizationStatus;

/**
* 定位、反地理编码回调
*/
@property (copy, nonatomic) MYLocationFinshBlock locationFinishBlock;

/**
* 获取定位服务管理对象实例
*
*/
+ (instancetype)defaultManager;

/**
* 系统是否允许定位判断
*
*/
- (BOOL)canLocation;

/**
* 打开设置定位服务界面
*
*/
- (void)turnToSettingLocationServicePage;
```

```
/**
 * 开启定位
 *
 * @ param isGeo 定位结果是否需要反编译
 */
- (void)startUpdatingLocationWithReverseGeo:(BOOL)isGeo;

/**
 * 关闭定位
 *
 * @ note 不用的时候记得关闭定位
 */
- (void)stopUpdatingLocation;

/**
 * 启用后台定位
 *
 * @ note 可以在后台或者前台都能监视到用户位置的移动，即使程序没有启动
 */
- (void)startMonitoringSignificantLocationChanges:(BOOL)isGeo;

/**
 * 关闭后台定位
 *
 * @ note 可以在后台或者前台都能监视到用户位置的移动，即使程序没有启动
 */
- (void)stopMonitoringSignificantLocationChanges;
```

```
/**
* 应用挂起时启用后台定位
*
*/
- (void)applicationDidEnterBackground;

/**
* 应用回到前台时禁用后台定位
*
*/
- (void)applicationDidBecomeActive;

/**
* 应用强退时禁用后台定位
*
* @ note 防止应用被kill了还进行不必要的耗电的定位操作
*/
- (void)applicationWillTerminate;

/**
* 获取指定坐标的地址详情
*
* @ param location 指定经纬度坐标
*/
- (void)getAddressDetailWithLocation:(CLLocationCoordinate2D)location;
```

```
/**
* 计算两坐标点距离
*
* @ param location1 指定经纬度坐标
* @ param location2 指定经纬度坐标
*/
- (CLLocationDistance)getDistanceBetweenLocation1:(CLLocation *)location1 andLocation2:(CLLocation *)location2;

/**
* 判断位置是否在指定区域内
*
*/
- (BOOL)location:(CLLocation *)location isInCircleArea:(CLLocation *)center circleRadius:(CLLocationDistance)radius;
```

```
//----------------------------------------------------------CLLocationManagerOptions----------------------------------------------------------//

@implementation CLLocationManagerOptions

- (void)setDistanceFilter:(CLLocationDistance)distanceFilter{
    [MYLocationManager defaultManager].locationManager.distanceFilter = distanceFilter;
}

- (void)setDesiredAccuracy:(CLLocationAccuracy)desiredAccuracy{
    [MYLocationManager defaultManager].locationManager.desiredAccuracy = desiredAccuracy;
}

@end
```

```
//----------------------------------------------------------MYLocationManager----------------------------------------------------------//

static MYLocationManager *instance;

@interface MYLocationManager ()<CLLocationManagerDelegate>

@property (strong, nonatomic) CLGeocoder *geoCoder;
@property (assign, nonatomic) BOOL isGeo;//是否需要逆向解析
@property (assign, nonatomic) UIBackgroundTaskIdentifier backTaskIdentifier;

@end

@implementation MYLocationManager

+ (instancetype)defaultManager{
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        instance = [[MYLocationManager alloc] init];
        instance.locationManager = [[CLLocationManager alloc] init];
        if([CLLocationManager locationServicesEnabled]){
            instance.locationManager.delegate = instance; // 设置代理
            instance.locationManager.distanceFilter = 100;
            instance.locationManager.desiredAccuracy = kCLLocationAccuracyBest;
            //在ios 8.0下要求用户主动请求对程序授权,授权状态改变就会通知代理
            if ([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0)
                [instance.locationManager requestAlwaysAuthorization];  //调用了这句,就会弹出允许框了.
            if([instance.locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)])
                instance.locationManager.allowsBackgroundLocationUpdates = NO;
            instance.locationManager.pausesLocationUpdatesAutomatically = YES;//设置iOS设备是否可暂停定位来节省电池的电量
        }
    });
    return instance;
```

```
+ (instancetype)allocWithZone:(struct _NSZone *)zone{
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        instance = [super allocWithZone:zone];
    });
    return instance;
}

#pragma mark Public Methods

- (BOOL)canLocation{
    BOOL serviceEnable = [CLLocationManager locationServicesEnabled];//用户是否开启了系统定位
    CLAuthorizationStatus authorizationStatus = [CLLocationManager authorizationStatus];
    _authorizationStatus = authorizationStatus;

    if ((authorizationStatus==kCLAuthorizationStatusAuthorized || authorizationStatus==kCLAuthorizationStatusAuthorizedWhenInUse || authorizationStatus ==
        = kCLAuthorizationStatusAuthorizedAlways) && serviceEnable) {//用户是否允许应用定位
        return YES;
    } else if (authorizationStatus == kCLAuthorizationStatusNotDetermined) {//用户还没允许定位
        return YES;
    }

    return NO;
}

- (void)turnToSettingLocationServicePage{
    NSString *openUrlStr = @"prefs:root=LOCATION_SERVICES";
    NSURL *openUrl = [NSURL URLWithString:openUrlStr];
    [[UIApplication sharedApplication] openURL:openUrl];
}
```

```
- (void)startUpdatingLocationWithReverseGeo:(BOOL)isGeo{
    _isGeo = isGeo;
    if([self canLocation])
    [_locationManager startUpdatingLocation];
}

- (void)stopUpdatingLocation{
    [_locationManager stopUpdatingLocation];
}

- (void)startMonitoringSignificantLocationChanges:(BOOL)isGeo{
    _locationManager.pausesLocationUpdatesAutomatically = NO;
    if([_locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)])
        _locationManager.allowsBackgroundLocationUpdates = YES;


    _isGeo = isGeo;
     if([self canLocation])
    [_locationManager startMonitoringSignificantLocationChanges];
}

- (void)stopMonitoringSignificantLocationChanges{
    _locationManager.pausesLocationUpdatesAutomatically = YES;
    if([_locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)])
        _locationManager.allowsBackgroundLocationUpdates = NO;


    [_locationManager stopMonitoringSignificantLocationChanges];
}
```

```
- (void)applicationDidEnterBackground{
    if([self canLocation] == NO)
        return;

    if ([CLLocationManager significantLocationChangeMonitoringAvailable]){//表明设备能否报告基于significant location changges的更新
        [self stopUpdatingLocation];
        [self startMonitoringSignificantLocationChanges:NO];
    }
    else{
        NSLog(@"Significant location change monitoring is not available.");
    }

    __weak typeof(self) weakSelf = self;
    if(_backTaskIdentifier)
        [[UIApplication sharedApplication] endBackgroundTask:self.backTaskIdentifier];

    _backTaskIdentifier =  [[UIApplication sharedApplication] beginBackgroundTaskWithExpirationHandler:^{
        [[UIApplication sharedApplication] endBackgroundTask:weakSelf.backTaskIdentifier];
        weakSelf.locationManager.pausesLocationUpdatesAutomatically = YES;
        if([weakSelf.locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)])
            weakSelf.locationManager.allowsBackgroundLocationUpdates = NO;

    }];
}
```

```
- (void)applicationDidBecomeActive{
    if([self canLocation] == NO)
        return;

    if(_backTaskIdentifier){
        [[UIApplication sharedApplication] endBackgroundTask:self.backTaskIdentifier];
        _backTaskIdentifier = 0;
    }

    if ([CLLocationManager significantLocationChangeMonitoringAvailable]){
        [self stopMonitoringSignificantLocationChanges];
        [self startUpdatingLocationWithReverseGeo:_isGeo];
    }
    else{
        NSLog(@"Significant location change monitoring is not available.");
    }
}
```

```
- (void)applicationWillTerminate{//由于程序被kill前还会调用一次applicationDidEnterBackground，导致applicationDidBecomeActive禁用的后台定位线程再次开启，所以需要在此禁止后台定位服务
    if([self canLocation] == NO)
        return;

    if(self.backTaskIdentifier)
        [[UIApplication sharedApplication] endBackgroundTask:self.backTaskIdentifier];

    if ([CLLocationManager significantLocationChangeMonitoringAvailable]){
        [self stopMonitoringSignificantLocationChanges];
        [self stopUpdatingLocation];
    }
    else{
        NSLog(@"Significant location change monitoring is not available.");
    }
}
```

```
- (void)getAddressDetailWithLocation:(CLLocationCoordinate2D)location{
    if(!_geoCoder){
    _geoCoder = [[CLGeocoder alloc] init];
    }
    [_geoCoder reverseGeocodeLocation:[[CLLocation alloc] initWithLatitude:location.latitude longitude:location.longitude] completionHandler:^(NSArray *array,
        NSError *error){
        if(error){
            if(_locationFinishBlock){
                _locationFinishBlock(location, nil, error);
            }
            return ;
        }

        if (array.count >0){
            CLPlacemark *placeMark = [array firstObject];
            if(_locationFinishBlock){
                _locationFinishBlock(location, placeMark, nil);
            }
        }else{
            NSError *emptyError = [NSError errorWithDomain:@"找不到对应地址" code:-1234 userInfo:nil];
            if(_locationFinishBlock){
                _locationFinishBlock(location, [[CLPlacemark alloc] init], emptyError);
            }
        }
    }];
}
```

```
- (CLLocationDistance)getDistanceBetweenLocation1:(CLLocation *)location1 andLocation2:(CLLocation *)location2{
    CLLocationDistance meters= [location1 distanceFromLocation:location2];
    return meters;
}

- (BOOL)location:(CLLocation *)location isInCircleArea:(CLLocation *)center circleRadius:(CLLocationDistance)radius{
    if([location distanceFromLocation:center] < radius){
        return YES;
    }else
        return NO;
}

#pragma mark CLLocationManagerDelegate

/** 获取到新的位置信息时调用*/
-(void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations{
    CLLocation *location = [locations firstObject];
    if(_isGeo){
        [self getAddressDetailWithLocation:location.coordinate];
    }else{
        if(_locationFinishBlock){
            _locationFinishBlock(location.coordinate, nil, nil);
        }
    }
}

/** 不能获取位置信息时调用*/
-(void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *)error{
    if(_locationFinishBlock){
        _locationFinishBlock(kCLLocationCoordinate2DInvalid, nil, error);
    }
}
```

```objc
/** 定位服务状态改变时调用*/
-(void)locationManager:(CLLocationManager *)manager didChangeAuthorizationStatus:(CLAuthorizationStatus)status{
    switch (status) {
        case kCLAuthorizationStatusNotDetermined:{
            NSLog(@"用户还未决定授权");
            break;
        }
        case kCLAuthorizationStatusRestricted:{
            NSLog(@"访问受限");
            break;
        }
        case kCLAuthorizationStatusDenied:{
            // 类方法，判断是否开启定位服务
            if ([CLLocationManager locationServicesEnabled]) {
                NSLog(@"定位服务开启，被拒绝");
            } else {
                NSLog(@"定位服务关闭，不可用");
            }
            break;
        }
        case kCLAuthorizationStatusAuthorizedAlways:{
            NSLog(@"获得前后台授权");
            break;
        }
        case kCLAuthorizationStatusAuthorizedWhenInUse:{
            NSLog(@"获得前台授权");
            break;
        }
        default:
            break;
    }
}
```