百度地图 iOS SDK 开发教程

第二部分 进阶篇

第五章 检索

发布日期：2013 年 11 月

百度在线网络技术（北京）有限公司

目录

第二部分 进阶篇

第五章 检索

百度地图 API 提供的检索服务包括兴趣点检索，地理编码，路径规划，公交线路查询，在线建议查询，短串分享等 6 大检索服务。

在使用检索服务之前，需注意一下两点注意事项：

（1） 所有检索请求接口均为异步接口，您必须实现 BMKSearchDelegate 协议，在一个时刻只能有一个 BMKSearch 接受回调消息，因此如果在不同的 viewController 中使用多个 BMKSearch，需要在页面切换对 BMKSearch 的 delegate 做处理，代码如下：

```
-(void)viewWillAppear:(BOOL)animated {
    _search.delegate = self; // 此处记得不用的时候需要置 nil，否则影响内存的释放
}

-(void)viewWillDisappear:(BOOL)animated {
    _search.delegate = nil; // 不用时，置 nil
}
```

（2） 在检索到结果后，API 会回调 BMKSearchDelegate 对应的接口，通知调用者检索结果数据，开发者可以按照自己的需求来实现相关接口方法，具体接口如下：

```
// ----------------------兴趣点检索----------------------

/**
 *返回 POI 搜索结果
 *@param poiResultList 搜索结果列表，成员类型为 BMKPoiResult
 *@param type 返回结果类型： BMKTypePoiList,BMKTypeAreaPoiList,BMKAreaMultiPoiList
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error;

// ----------------------地理编码----------------------

/**
 *返回地址信息搜索结果
 *@param result 搜索结果
```

```
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetAddrResult:(BMKAddrInfo*)result errorCode:(int)error;

// ----------------------路径规划----------------------

/**
 *返回驾乘搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetDrivingRouteResult:(BMKPlanResult*)result errorCode:(int)error;

/**
 *返回公交搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetTransitRouteResult:(BMKPlanResult*)result errorCode:(int)error;

/**
 *返回步行搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetWalkingRouteResult:(BMKPlanResult*)result errorCode:(int)error;

// ----------------------公交线路查询----------------------

/**
 *返回 busdetail 搜索结果
 *@param busLineResult 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetBusDetailResult:(BMKBusLineResult*)busLineResult
errorCode:(int)error;

// ----------------------在线建议查询----------------------

/**
 *返回 suggestion 搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
```

```
 */
- (void)onGetSuggestionResult:(BMKSuggestionResult*)result errorCode:(int)error;

// ----------------------短串分享----------------------

/**
 *返回 poi 详情分享 url
 *@param url 返回结果 url
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetShareUrl:(NSString*) url withType:(BMK_SHARE_URL_TYPE) urlType
errorCode:(int)error;
```

更多内容详见：inc/BMKSearch.h

1. 兴趣点检索

百度地图 API 提供的兴趣点（Point of Interest，POI）检索包括以下几种类型：城市检索、周边检索、范围检索。

1.1. 城市检索

城市检索是指，检索一个城市内的 POI 信息，例如“北京市”，具体参数设置和检索方法如下：

```
/**
 *城市 POI 检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetPoiResult 通知
 *@param city 城市名，如果为 nil 则在当前底图范围搜索
 *@param key 关键词
 *@param index 页码，如果是第一次发起搜索，填 0，根据返回的结果可以去获取第 n 页的结果，页码从 0 开始
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)poiSearchInCity:(NSString*)city withKey:(NSString*)key pageIndex:(int)index;
```

城市检索的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/PoiSearchDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface PoiSearchDemoViewController : UIViewController<BMKMapViewDelegate, BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _cityText;
    IBOutlet UITextField* _keyText;
    IBOutlet UIButton* _nextPageButton;
    BMKSearch* _search;
    int curPage;
}
```

```
-(IBAction)onClickOk;
-(IBAction)onClickNextPage;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 poiSearchInCity 方法发起范围检索，并实现 BMKSearchDelegate 协议中获取检索结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    // 搜索城市
    _cityText.text = @"北京";
    // 搜索关键字
    _keyText.text  = @"餐厅";
}

// 点击开始搜索按钮
-(IBAction)onClickOk
{
    curPage = 0;
    // 城市检索，请求发送成功返回 YES，请求发送失败返回 NO
    BOOL flag = [_search poiSearchInCity:_cityText.text withKey:_keyText.textpage
Index:curPage];
    if (flag) {
        _nextPageButton.enabled = true;
        NSLog(@"search success.");
    }
    else{
        _nextPageButton.enabled = false;
        NSLog(@"search failed!");
    }
}

// 搜索结果回调
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error
{
    // 在此处添加对城市检索结果的处理
```

```
}
```

示例代码中相关截图如下图所示：

1.2. 范围检索

范围检索是指，检索一个矩形区域内的 POI 信息，具体参数设置和检索方法如下：

```
/**
 *根据范围和检索词发起范围检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetPoiResult 通知
 *@param key 关键词
 *@param ptLB 地理坐标，范围的左下角
 *@param ptRT 地理坐标，范围的右下角
 *@param index 页码，如果是第一次发起搜索，填 0，根据返回的结果可以去获取第 n 页的结果，页码从 0 开始
 *@return 成功返回 YES，否则返回 NO
 */
```

```
- (BOOL)poiSearchInbounds:(NSString*)key leftBottom:(CLLocationCoordinate2D)ptLB
rightTop:(CLLocationCoordinate2D)ptRT pageIndex:(int)index;
```

范围检索的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/PoiBoundSearchDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface PoiBoundSearchDemoViewController : UIViewController<BMKMapViewDelegate,
BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _keyText;
    IBOutlet UIButton* _nextPageButton;
    BMKSearch* _search;
    int curPage;
}

-(IBAction)onClickOk;
-(IBAction)onClickNextPage;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 poiSearchInbounds 方法发起范围检索，并实现 BMKSearchDelegate 协议中获取检索结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    // 搜索关键字
    _keyText.text  = @"餐厅";
    //初始化搜索条件——范围左下点
    leftBottomPt.latitude = 39.920038;
    leftBottomPt.longitude = 116.403596;
    //初始化搜索条件——范围右上点
```

```
    rightTopPt.latitude = 39.96196;
    rightTopPt.longitude = 116.44067;
}

// 点击开始搜素按钮
-(IBAction)onClickOk
{
    curPage = 0;
    // 范围内搜索，请求发送成功返回 YES，请求发送失败返回 NO
    BOOL flag = [_search poiSearchInbounds:_keyText.text leftBottom:leftBottomPt
rightTop:rightTopPt pageIndex:curPage];
    if (flag) {
        _nextPageButton.enabled = true;
        NSLog(@"search success.");
    }
    else{
        _nextPageButton.enabled = false;
        NSLog(@"search failed!");
    }
}

//搜索结果回调
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error
{
    // 在此处添加您对范围检索结果的处理
}
```

示例代码中相关截图如下图所示：

1.3. 周边检索

周边检索是指，以一个点为中心检索一定半径内的 POI 信息 ，例如检索当前位置 3 公里范围内的餐厅，具体参数设置和检索方法如下：

```
/**
 *根据中心点、半径和检索词发起周边检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetPoiResult 通知
 *@param key 关键词
 *@param ptCenter 中心点地理坐标
 *@param radius 半径，单位：米 必须大于 0
 *@param index 页码，如果是第一次发起搜索，填 0，根据返回的结果可以去获取第 n 页的结果，页码从 0 开始
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)poiSearchNearBy:(NSString*)key center:(CLLocationCoordinate2D)ptCenter radius:(int)radius pageIndex:(int)index;
```

周边检索的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/PoiNearbySearchDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface PoiNearbySearchDemoViewController :
UIViewController<BMKMapViewDelegate, BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _cityText;
    IBOutlet UITextField* _keyText;
    IBOutlet UIButton* _nextPageButton;
    BMKSearch* _search;
    int curPage;
}

-(IBAction)onClickOk;
-(IBAction)onClickNextPage;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 poiSearchNearBy 方法发起范围检索，并实现 BMKSearchDelegate 协议中获取检索结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    //初始化搜索条件——关键字
    _keyText.text  = @"餐厅";
    //初始化搜索条件——搜索中心点
    ptCenter.latitude = 39.920038;
    ptCenter.longitude = 116.403596;
    //初始化搜索条件——搜索半径
    radius = 1000;
    _cityText.text = [NSString stringWithFormat:@"%d",radius];
```

```
}

//点击开始搜索按钮
-(IBAction)onClickOk
{
    if(_cityText.text.intValue<=0)
    {
        radius=1000;
    }
    else
    {
        radius = _cityText.text.intValue;
    }
    curPage = 0;
    //城市内检索，请求发送成功返回 YES，请求发送失败返回 NO
    BOOL flag = [_search poiSearchNearBy:_keyText.text center:ptCenter radius:radius
pageIndex:curPage];
    if (flag) {
        _nextPageButton.enabled = true;
        NSLog(@"search success.");
    }
    else{
        _nextPageButton.enabled = false;
        NSLog(@"search failed!");
    }
}

//搜索结果回调
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error
{
    // 在此处添加您对周边检索结果的处理
}
```

示例代码中相关截图如下图所示：

1.4. 展示检索结果

步骤 1：

在 onGetPoiResult 回调中批量添加 annotation，代码如下：

```
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error
{
    // 清除屏幕中所有的 annotation
    NSArray* array = [NSArray arrayWithArray:_mapView.annotations];
    [_mapView removeAnnotations:array];

    if (error == BMKErrorOk) {
        BMKPoiResult* result = [poiResultList objectAtIndex:0];
        //批量添加 Annotation
        for (int i = 0; i < result.poiInfoList.count; i++) {
            BMKPoiInfo* poi = [result.poiInfoList objectAtIndex:i];
```

```
            BMKPointAnnotation* item = [[BMKPointAnnotation alloc]init];
            item.coordinate = poi.pt;
            item.title = poi.name;
            [_mapView addAnnotation:item];
            if(i == 0)
            {
                // 将第一个点的坐标移到屏幕中央
                _mapView.centerCoordinate = poi.pt;
            }
            [item release];
        }
    } else{
        NSLog(@"error=%d",error);
    }
}
```

步骤 2：

实现回调 viewForAnnotation 根据 anntation 生成对应的 View,代码如下：

```
/**
 *根据 anntation 生成对应的 View
 *@param mapView 地图 View
 *@param annotation 指定的标注
 *@return 生成的标注 View
 */
- (BMKAnnotationView *)mapView:(BMKMapView *)view viewForAnnotation:(id
<BMKAnnotation>)annotation
{
    // 生成重用标示 identifier
    NSString *AnnotationViewID = @"xidanMark";

    // 检查是否有重用的缓存
    BMKAnnotationView* annotationView = [view
dequeueReusableAnnotationViewWithIdentifier:AnnotationViewID];

    // 缓存没有命中，自己构造一个，一般首次添加 annotation 代码会运行到此处
    if (annotationView == nil) {
        annotationView = [[[BMKPinAnnotationView alloc]
initWithAnnotation:annotation reuseIdentifier:AnnotationViewID] autorelease];
        ((BMKPinAnnotationView*)annotationView).pinColor =
BMKPinAnnotationColorRed;
        // 设置重天上掉下的效果(annotation)
        ((BMKPinAnnotationView*)annotationView).animatesDrop = YES;
    }
```

```
    // 设置位置
    annotationView.centerOffset = CGPointMake(0,
-(annotationView.frame.size.height * 0.5));
    annotationView.annotation = annotation;
    // 单击弹出泡泡，弹出泡泡前提 annotation 必须实现 title 属性
    annotationView.canShowCallout = YES;
    // 设置是否可以拖拽
    annotationView.draggable = NO;

    return annotationView;
}
```

步骤 3：

为 marker 添加点击事件，实现回调 didSelectAnnotationView 来定义 marker 的点击时间,代码如下：

```
- (void)mapView:(BMKMapView *)mapView didSelectAnnotationView:(BMKAnnotationView
*)view
{
    [mapView bringSubviewToFront:view];
    [mapView setNeedsDisplay];
    NSLog(@"didSelectAnnotationView");
}
```

1.5. 事件监听

1、 兴趣点搜索：<BMKSearchDelegate>

（1）返回 POI 搜索结果

```
/**
 *返回 POI 搜索结果
 *@param poiResultList 搜索结果列表，成员类型为 BMKPoiResult
 *@param type 返回结果类型： BMKTypePoiList,BMKTypeAreaPoiList,BMKAreaMultiPoiList
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error;
```

2、BMKPoiInfo：POI 信息类

属性名称	类型	说明
name	NSString	名称

uid	NSString	唯一标识
address	NSString	地址
city	NSString	所在城市
phone	NSString	电话号码
postcode	NSString	邮编
epoitype	int	类型： 0：普通点，1：公交站， 2：公交线路，3：地铁站 4：地铁线
pt	CLLocationCoordinate2D	坐标

更多信息详见：inc/BMKPoiSearchType.h

2. 地理编码

地理编码(Geocoding)指的是将统计资料或是地址信息建立空间坐标关系的过程，它包含正向地理编码和反向地理编码，其中正向地理编码实现了将中文地址或地名描述转换为地球表面上相应位置的功能，反向地理编码实现了将地球表面的地址坐标转换为标准地址的过程的功能。

2.1. 正向地理编码

正向地理编码具体参数设置和检索方法如下：

```
/**
 *根据地址名称获取地理信息-正向
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetAddrResult 通知
 *@param addr       地址信息
 *@param city       城市名称
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)geocode:(NSString*)addr withCity:(NSString*)city;
```

正向地理编的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/GeocodeDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface GeocodeDemoViewController : UIViewController<BMKMapViewDelegate,
BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _cityText;
    IBOutlet UITextField* _addrText;
    BMKSearch* _search;
}

-(IBAction)onClickGeocode;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 geocode 方法发起正向地理编码检索，并实现 BMKSearchDelegate 协议中获取地理编码结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    _cityText.text = @"北京";
    _addrText.text = @"海淀区上地十街 10 号";
}

// 根据地址名称获取地理信息（发起正向地理编码搜索，结果在 onGetAddrResult 回调中返回）
-(IBAction)onClickGeocode
{
    BOOL flag = [_search geocode:_addrText.text withCity:_cityText.text];
    if (flag) {
        NSLog(@"Geocode search success.");
    }
    else{
        NSLog(@"Geocode search failed!");
    }
}

// 获取结果
- (void)onGetAddrResult:(BMKAddrInfo*)result errorCode:(int)error
{
    if (error == 0) {
        _mapView.centerCoordinate = result.geoPt;

        NSString* titleStr;
        NSString* showmeg;
        titleStr = @"正向地理编码";
        showmeg = [NSString stringWithFormat:@"经度:%f,纬
度:%f",result.geoPt.latitude,result.geoPt.longitude];

        UIAlertView *myAlertView = [[UIAlertView alloc] initWithTitle:titleStr
message:showmeg delegate:self cancelButtonTitle:nil otherButtonTitles:@"确定
",nil];
        [myAlertView show];
        [myAlertView release];
```

```
    }
}
```

示例代码中相关截图如下图所示：

2.2. 反向地理编码

反向地理编码具体参数设置和检索方法如下：

```
/**
 *根据地理坐标获取地址信息-反向
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetAddrResult 通知
 *@param center 点坐标
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)reverseGeocode:(CLLocationCoordinate2D)center;
```

反向地理编码的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/ReverseGeocodeDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface ReverseGeocodeDemoViewController : UIViewController<BMKMapViewDelegate,
BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _coordinateXText;
    IBOutlet UITextField* _coordinateYText;
    BMKSearch* _search;
}

-(IBAction)onClickReverseGeocode;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 reverseGeocode 方法发起反向地理编码检索，并实现 BMKSearchDelegate 协议中获取地理编码结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    _coordinateXText.text = @"116.403981";
    _coordinateYText.text = @"39.915101";
}

// 根据经纬度坐标获取地址名称（发起反向地理编码搜索，结果在 onGetAddrResult 回调中返回）
-(IBAction)onClickReverseGeocode
{
    CLLocationCoordinate2D pt = (CLLocationCoordinate2D){0, 0};
    if (_coordinateXText.text != nil&&_coordinateYText.text != nil) {
        pt = (CLLocationCoordinate2D){[_coordinateYText.textfloatValue],
[_coordinateXText.textfloatValue]};
    }
    BOOL flag = [_search reverseGeocode:pt];
    if (flag) {
```

```
        NSLog(@"ReverseGeocode search success.");
    }
    else{
        NSLog(@"ReverseGeocode search failed!");
    }
}

// 获取结果
- (void)onGetAddrResult:(BMKAddrInfo*)result errorCode:(int)error
{
    if (error == 0) {
        _mapView.centerCoordinate = result.geoPt;

        NSString* titleStr;
        NSString* showmeg;
        titleStr = @"反向地理编码";
        showmeg = [NSString stringWithFormat:@"%@",result.strAddr];

        UIAlertView *myAlertView = [[UIAlertView alloc] initWithTitle:result.strAddr
message:showmeg delegate:self cancelButtonTitle:nil otherButtonTitles:@"确定
",nil];
        [myAlertView show];
        [myAlertView release];
    }
}
```

示例代码中相关截图如下图所示：

2.3. 事件监听

1、 地理编码：<BMKSearchDelegate>

（1） 返回地理信息搜索结果

```
/**
 *返回地址信息搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetAddrResult:(BMKAddrInfo*)result errorCode:(int)error;
```

3. 路径规划

支持公交和地铁查询的城市：目前支持全国 250 多个城市公交换乘查询，支持全国 18 个城市地铁换乘查询，分别为北京、上海、广州、深圳、南京、成都、香港、重庆、天津、沈阳、杭州、武汉、苏州、大连、长春、西安、昆明、佛山。

百度地图 API 提供的路径规划包括：驾车规划、公交规划和步行规划，其中驾车规划还提供了多途经点路径规划。

包含驾车、公交和步行规划的路径规划示例代码详见：

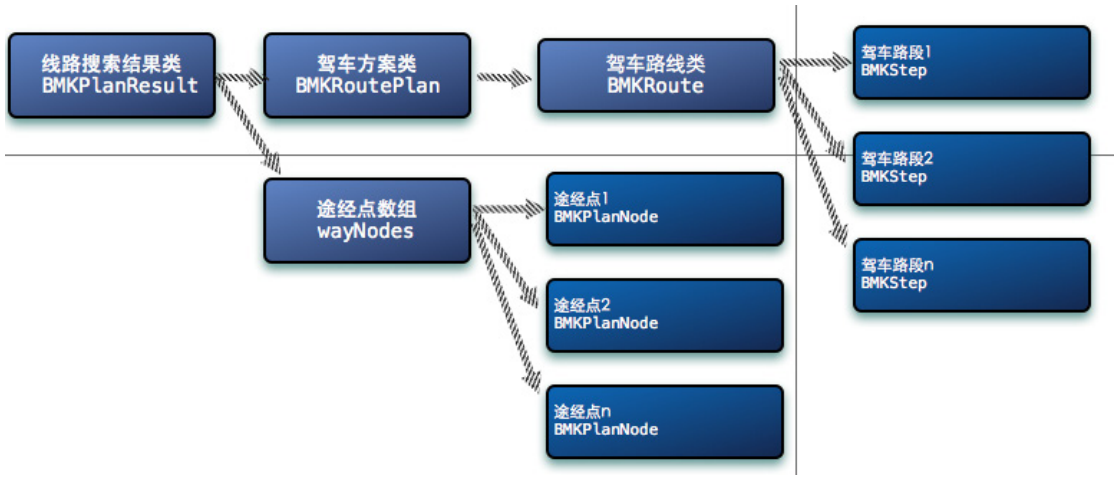
BaiduMapApiTutorial/src/Chapter5/RouteSearchViewController

示例代码中相关截图如下图所示：

3.1. 驾车规划

百度地图 SDK 为大家提供了两种驾车规划的方法，一种适用于无途经点的驾车规划，另一种适用于多途经点的驾车规划。

驾车规划返回结果的数据结构如下图所示：



下面针对有无途经点来讲解驾车规划。

1、 无途经点

无途经点驾车规划参数设置和检索方法如下：

```
/**
 *驾乘路线检索-无途经点
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetDrivingRouteResult 通知
 *@param startCity 起点所在城市，起点为坐标时可不填
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param endCity 终点所在城市，终点为坐标时可不填
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)drivingSearch:(NSString*)startCity startNode:(BMKPlanNode*)start
endCity:(NSString*)endCity endNode:(BMKPlanNode*)end;
```

无途经点驾车规划的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/RouteSearch4DrivingDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface RouteSearch4BusTransitDemoViewController :
UIViewController<BMKMapViewDelegate, BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
```

```
    IBOutlet UITextField* _startCityText;
    IBOutlet UITextField* _startAddrText;
    IBOutlet UITextField* _endCityText;
    IBOutlet UITextField* _endAddrText;
    BMKSearch* _search;
}

-(IBAction)onClickBusSearch;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 drivingSearch 方法发起驾车规划检索，并实现 BMKSearchDelegate 协议中获取驾车规划结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    _startCityText.text = @"北京";
    _startAddrText.text = @"天安门";
    _endCityText.text = @"北京";
    _endAddrText.text = @"百度大厦";
}

//点击［驾乘］进行步行规划检索，结果在 onGetDrivingRouteResult 回调中返回
-(IBAction)onClickDriveSearch
{
    BMKPlanNode* start = [[[BMKPlanNode alloc] init] autorelease];
    start.name = _startAddrText.text;
    BMKPlanNode* end = [[[BMKPlanNode alloc] init] autorelease];
    end.name = _endAddrText.text;

    BOOL flag = [_search drivingSearch:_startCityText.text startNode:start
endCity:_endCityText.text endNode:end];
    if (flag) {
        NSLog(@"search success.");
    }
    else{
        NSLog(@"search failed!");
    }
```

```
}

// 获取结果
- (void)onGetDrivingRouteResult:(BMKPlanResult*)result errorCode:(int)error {
    // 在此处添加您对驾车规划结果的处理
    // 具体获取结果和添加路线的方法，详见示例代码中的方法
}
```

示例代码中相关截图如下图所示：

2、 多途经点

多途经点驾车规划参数设置和检索方法如下：

```
/**
 *驾乘路线检索-多途经点
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetDrivingRouteResult 通知
 *@param startCity 起点所在城市，起点为坐标时可不填
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param endCity 终点所在城市，终点为坐标时可不填
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
```

```
 *@param wayPointsArray 途经点数组，存储 BMKPlanNode 信息的节点。
wayPointsArray 为 nil 或空时，表示没有途经点。最多支持 10 个途经点，超过 10 个
请求发送不成功。
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)drivingSearch:(NSString*)startCity
startNode:(BMKPlanNode*)start endCity:(NSString*)endCity
endNode:(BMKPlanNode*)end
throughWayPoints:(NSArray*)wayPointsArray;
```

多途经点驾车规划的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/WayPointRouteSearchDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface WayPointRouteSearchDemoViewController : UIViewController
<BMKMapViewDelegate, BMKSearchDelegate> {
    IBOutlet UITextField* _startAddrText;
    IBOutlet UITextField* _wayPointAddrText;
    IBOutlet UITextField* _endAddrText;
    IBOutlet BMKMapView* _mapView;
    BMKSearch* _search;

}

-(IBAction)onDriveSearch;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 drivingSearch 方法发起驾车规划（含途经点）检索，并实现 BMKSearchDelegate 协议中获取驾车规划结果的方法，代码如下：

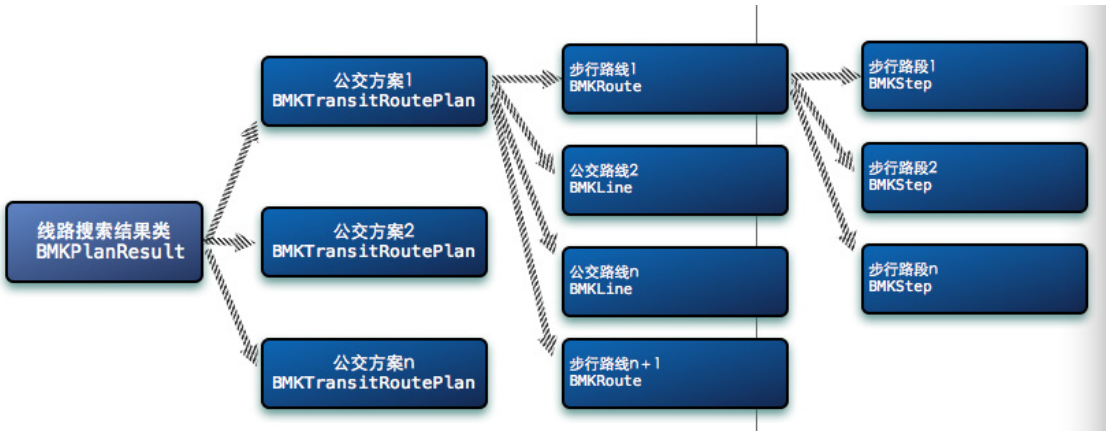
```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
```

```
    _search.delegate = self;
    startAddrText.text = @"天安门";
    _wayPointAddrText.text = @"东直门";
    _endAddrText.text = @"百度大厦";
}

// 点击［驾乘］进行步行规划检索，结果在 onGetDrivingRouteResult 回调中返回
-(IBAction)onDriveSearch
{
    BMKPlanNode* start = [[[BMKPlanNode alloc] init] autorelease];
    start.name = _startAddrText.text;
    BMKPlanNode* end = [[[BMKPlanNode alloc] init] autorelease];
    end.name = _endAddrText.text;
    // 定义途经点数组，存储 BMKPlanNode 信息的节点。wayPointsArray 为 nil 或空时，表示没有途经点。最多支持 10 个途经点，超过 10 个请求发送不成功。
    NSMutableArray * array = [[[NSMutableArray alloc] initWithCapacity:10] autorelease];
    BMKPlanNode* wayPointItem1 = [[[BMKPlanNode alloc] init] autorelease];
    wayPointItem1.cityName = @"北京";
    wayPointItem1.name = _wayPointAddrText.text;
    [array addObject:wayPointItem1];
    // 发起驾车（含途经点）规划搜索
    BOOL flag = [_search drivingSearch:@"北京" startNode:start endCity:@"北京" endNode:end throughWayPoints:array];
    if (flag) {
        NSLog(@"search success.");
    }
    else{
        NSLog(@"search failed!");
    }
}

// 获取结果
- (void)onGetDrivingRouteResult:(BMKPlanResult*)result errorCode:(int)error{
    // 在此处添加您对驾车规划结果的处理
    // 具体获取结果和添加路线的方法，详见示例代码中的方法
}
```

示例代码中相关截图如下图所示：

3.2. 公交规划

公交规划返回结果的数据结构如下图所示：



公交规划具体参数设置和检索方法如下：

```
/**
```

```
 *公交路线检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetTransitRouteResult 通知
 *@param city 城市名，用于在哪个城市内进行检索
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)transitSearch:(NSString*)city startNode:(BMKPlanNode*)start
endNode:(BMKPlanNode*)end;
```

公交规划的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/ RouteSearch4BusTransitDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface RouteSearch4BusTransitDemoViewController :
UIViewController<BMKMapViewDelegate, BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _startCityText;
    IBOutlet UITextField* _startAddrText;
    IBOutlet UITextField* _endCityText;
    IBOutlet UITextField* _endAddrText;
    BMKSearch* _search;
}

-(IBAction)onClickBusSearch;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 transitSearch 方法发起公交规划检索，并实现 BMKSearchDelegate 协议中获取公交规划结果的方法，代码如下：

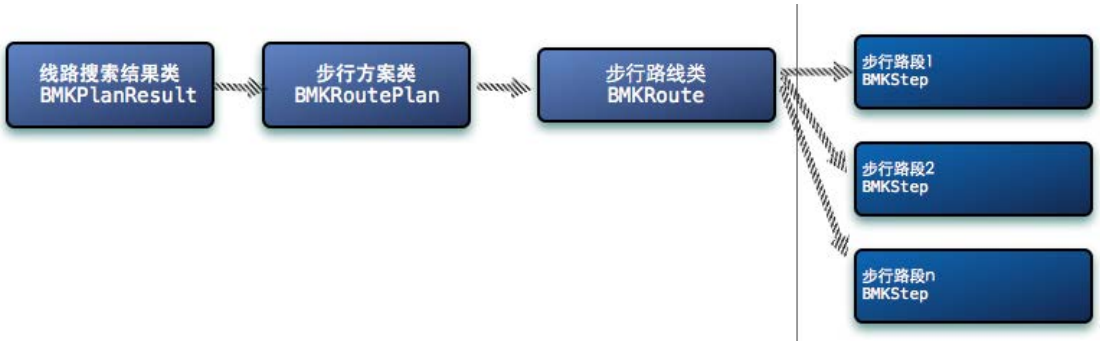
```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
```

```
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    _startCityText.text = @"北京";
    _startAddrText.text = @"天安门";
    _endCityText.text = @"北京";
    _endAddrText.text = @"百度大厦";
}

// 点击［公交］进行步行规划检索，结果在 onGetTransitRouteResult 回调中返回
-(IBAction)onClickBusSearch
{
    BMKPlanNode* start = [[[BMKPlanNode alloc] init] autorelease];
    start.name = _startAddrText.text;
    BMKPlanNode* end = [[[BMKPlanNode alloc] init] autorelease];
    end.name = _endAddrText.text;
    // 公交路线检索
    BOOL flag = [_search transitSearch:_startCityText.text startNode:start
endNode:end];
    if (flag) {
        NSLog(@"search success.");
    }
    else{
        NSLog(@"search failed!");
    }
}

// 获取结果
- (void)onGetTransitRouteResult:(BMKPlanResult*)result errorCode:(int)error {
    // 在此处添加您对公交规划结果的处理
    // 具体获取结果和添加路线的方法，详见示例代码中的方法
}
```

示例代码中相关截图如下图所示：

3.3. 步行规划

步行规划返回结果的数据结构如下图所示：



步行规划具体参数设置和检索方法如下：

```
/**
 *步行路线检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetWalkingRouteResult 通知
```

```
 *@param startCity 起点所在城市，起点为坐标时可不填
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param endCity 终点所在城市，终点为坐标时可不填
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)walkingSearch:(NSString*)startCity startNode:(BMKPlanNode*)start
endCity:(NSString*)endCity endNode:(BMKPlanNode*)end;
```

步行规划的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/RouteSearch4WalkingDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface RouteSearch4WalkingDemoViewController :
UIViewController<BMKMapViewDelegate, BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _startCityText;
    IBOutlet UITextField* _startAddrText;
    IBOutlet UITextField* _endCityText;
    IBOutlet UITextField* _endAddrText;
    BMKSearch* _search;
}

-(IBAction)onClickWalkSearch;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 walkingSearch 方法发起步行规划检索，并实现 BMKSearchDelegate 协议中获取步行规划结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
```

```
    _startCityText.text = @"北京";
    _startAddrText.text = @"天安门";
    _endCityText.text = @"北京";
    _endAddrText.text = @"百度大厦";}

// 点击［步行］进行步行规划检索，结果在 onGetWalkingRouteResult 回调中返回
-(IBAction)onClickWalkSearch
{
    BMKPlanNode* start = [[[BMKPlanNode alloc] init] autorelease];
    start.name = _startAddrText.text;
    BMKPlanNode* end = [[[BMKPlanNode alloc] init] autorelease];
    end.name = _endAddrText.text;
    // 发起步行规划检索
    BOOL flag = [_search walkingSearch:_startCityText.text startNode:start
endCity:_endCityText.text endNode:end];
    if (flag) {
        NSLog(@"search success.");
    }
    else{
        NSLog(@"search failed!");
    }
}

// 获取结果
- (void)onGetWalkingRouteResult:(BMKPlanResult*)result errorCode:(int)error
{
    // 在此处添加您对步行规划结果的处理
    // 具体获取结果和添加路线的方法，详见示例代码中的方法
}
```

示例代码中相关截图如下图所示：

3.4. 事件监听

1、 路径规划：<BMKSearchDelegate>

（1） 返回驾乘搜索结果

```
/**
 *返回驾乘搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetDrivingRouteResult:(BMKPlanResult*)result errorCode:(int)error;
```

（2） 返回公交换乘搜索结果

```
/**
 *返回公交搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
```

```
 */
- (void)onGetTransitRouteResult:(BMKPlanResult*)result errorCode:(int)error;
```

（3） 返回步行搜索结果

```
/**
 *返回步行搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetWalkingRouteResult:(BMKPlanResult*)result errorCode:(int)error;
```

4. 公交线路查询

百度地图API提供的公交线路查询功能，可以查询一个城市的公交线路详情，包括上行和下行两个方向的线路详情。

4.1. 公交线路查询

公交线路查询具体参数设置和检索方法如下：

```
/**
 *公交详情检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetBusDetailResult 通知
 *@param city 城市名，用于在哪个城市内进行检索
 *@param busLineUid 公交线路的 uid
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)busLineSearch:(NSString*)city withKey:(NSString*)busLineUid;
```

公交线路查询的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/BusLineSearchViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import "BMapKit.h"

@interface BusLineSearchViewController : UIViewController<BMKMapViewDelegate,
BMKSearchDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UITextField* _cityText;
    IBOutlet UITextField* _busLineText;

    NSMutableArray* _busPoiArray;
    int currentIndex;
    BMKSearch* _search;
    BMKPointAnnotation* _annotation;
}

-(IBAction)onClickBusLineSearch;
-(IBAction)onClickNextSearch;
```

```
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

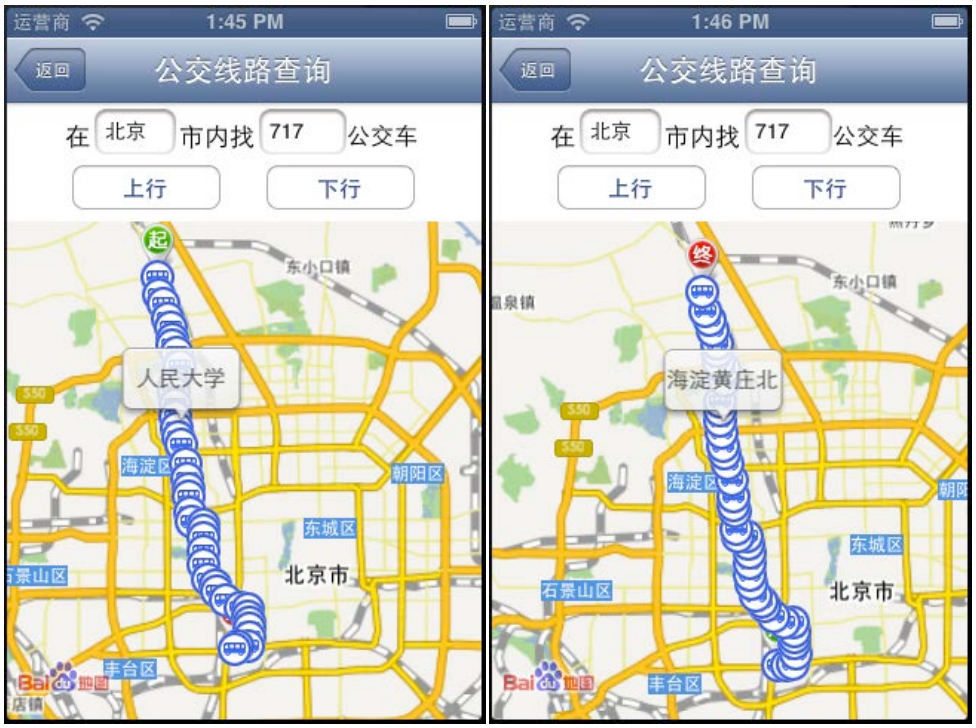
步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 busLineSearch 方法发起公交线路检索，并实现 BMKSearchDelegate 协议中获取公交线路详情的方法。发起公交详情搜索前，先进行 POI 检索，检索结果中 poi.epoitype == 2 时表示该类型为公交线路，才可发起公交搜索，代码如下：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    _cityText.text = @"北京";
    _busLineText.text  = @"717";
}

// 首先发起 poi 检索
-(IBAction)onClickBusLineSearch {
    BOOL flag = [_search poiSearchInCity:_cityText.text withKey:_busLineText.text
pageIndex:0];
    if (flag) {
        NSLog(@"search success.");
    }
    else{
        NSLog(@"search failed!");
    }
}

// 在 poi 检索结果回调里发起具体线路检索
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error
{
    if (error == BMKErrorOk) {
        BMKPoiResult* result = [poiResultList objectAtIndex:0];
        BMKPoiInfo* poi = nil;
        BOOL findBusline = NO;
        for (int i = 0; i < result.poiInfoList.count; i++) {
            poi = [result.poiInfoList objectAtIndex:i];
            if (poi.epoitype == 2) {
```

```
                findBusline = YES;
                [_busPoiArray addObject:poi];
            }
        }
        // 开始 bueline 详情搜索
        if(findBusline)
        {
            currentIndex = 0;
            NSString* strKey = ((BMKPoiInfo*) [_busPoiArray
objectAtIndex:currentIndex]).uid;
            BOOL flag = [_search busLineSearch:_cityText.text withKey:strKey];
            if (flag) {
                NSLog(@"search success.");
            }
            else{
                NSLog(@"search failed!");
            }
        }
    }
}

// 获取结果
-(void)onGetBusDetailResult:(BMKBusLineResult *)busLineResult
errorCode:(int)error
{
    // 在此处添加您对线路搜索结果的处理
    // 具体获取结果和添加路线的方法，详见示例代码中的方法
}
```

示例代码中相关截图如下图所示：



4.2. 事件监听

1、 公交线路查询：<BMKSearchDelegate>

（1） 返回公交详细搜索结果

```
/**
 *返回 busdetail 搜索结果
 *@param busLineResult 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetBusDetailResult:(BMKBusLineResult*)busLineResult
errorCode:(int)error;
```

5. 在线建议查询

在线建议查询是指根据关键词查询在线建议词，例如输入“西单”，返回在线建议词列表：“西单”、“西单商场”、“西单商场超市”……

5.1. 在线建议查询

在线建议查询具体参数设置和检索方法如下：

```
/**
 *搜索建议检索
 *@param key       搜索建议的检索关键字
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetSuggestionResult 通知
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)suggestionSearch:(NSString*)key;

/**
 *特定城市内搜索建议检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetSuggestionResult 通知
 *@param key      搜索建议的检索关键字
 *@param cityname 城市名，用于指定在特定城市内检索
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)suggestionSearch:(NSString*)key inCity:(NSString*)cityname;
```

在线建议查询的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/SuggestionSearchDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate 协议，代码如下：

```
#import <Foundation/Foundation.h>
#import "BMapKit.h"

@interface
SuggestionSearchDemoViewController :UIViewController<BMKMapViewDelegate,BMKSearchDelegate>
{
    BMKSearch* _search;
    IBOutlet BMKMapView* _mapView;//.xib 里要有 BMKMapView 类用于初始化数据驱动
```

```
    IBOutlet UIButton* searchBtn;
    IBOutlet UITextField* sugText;
    IBOutlet UITableView* plainTableView;
    NSArray * _suggestionTitle;//sug 搜索结果
}

-(IBAction)search:(id)sender;
-(IBAction)textFiledReturnEditing:(id)sender;

@end
```

步骤 2：

在 ViewController.m 中创建 BMKSearch 对象，设置对应的 delegate，调用 suggestionSearch 方法进行在线建议搜索，并实现 BMKSearchDelegate 协议中获取 sug 搜索结果的方法，代码如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];
    _mapView.delegate = self;
    _search = [[BMKSearch alloc] init];
    _search.delegate = self;
    sugText.text = @"西单";
}

// sug 搜索
- (IBAction)search:(id)sender
{
    [sugText resignFirstResponder];
    // 搜索建议检索
    BOOL flag = [_search suggestionSearch:sugText.text];
    if (flag) {
        NSLog(@"suggestionSearch success.");
    }
    else{
        NSLog(@"suggestionSearch failed!");
    }
}

// 返回 suggestion 搜索结果
- (void)onGetSuggestionResult:(BMKSuggestionResult*)result errorCode:(int)error
{
    // 在此处添加您对 suggestion 搜索结果的处理
}
```

示例代码中相关截图如下图所示：

5.2. 事件监听

1、 在线建议查询：<BMKSearchDelegate>

（1）返回 suggestion 搜索结果

```
/**
 *返回 suggestion 搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetSuggestionResult:(BMKSuggestionResult*)result errorCode:(int)error;
```

6. 短串分享

自 **V2.0.2** 开始，百度地图 API 提供短串分享功能。短串分享是指，用户搜索查询后得到的每一个地理位置结果将会对应一条短串（短链接），用户可以通过短信、邮件或第三方分享组件（如微博、微信等）把短串分享给其他用户从而实现地理位置信息的分享。当其他用户收到分享的短串后，点击短串即可打开手机上的百度地图客户端或者手机浏览器进行查看。

例如，用户搜索"百度大厦"后通过短信使用短串分享功能把该地点分享给好友，好友点击短信中的短串“http://j.map.baidu.com/BkmBk”后可以调起百度地图客户端或者手机浏览器查看"百度大厦"的地理位置信息。

目前，短串分享功能仅暂时开放了“POI 搜索结果分享”和“反向地理编码结果分享”2 个功能。

6.1. POI 点分享

POI 点分享具体参数设置和检索方法如下：

```
/**
 *获取 poi 详情短串分享 url
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetShareUrl 通知
 *@param uid poi 的 uid
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)poiDetailShareUrl:(NSString*) uid;
```

POI 点分享的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/PoiDetailShortUrlShareDemoViewController

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKSearchDelegate、MFMessageComposeViewControllerDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import <MessageUI/MessageUI.h>
#import <MessageUI/MFMessageComposeViewController.h>
#import "BMapKit.h"


@interface PoiDetailShortUrlShareDemoViewController :
UIViewController<BMKMapViewDelegate,
BMKSearchDelegate,MFMessageComposeViewControllerDelegate> {
```

```
    IBOutlet BMKMapView* _mapView;
    IBOutlet UIButton* _poiShortUrlButton;
    BMKSearch* _search;

}
-(IBAction)poiShortUrlSearch;

@end
```

步骤 2：

根据 POI 检索结果的 UID，生成一个短连接，用于分享。方法为：- (BOOL) poiDetailShareUrl:(NSString*) uid;参数 uid 为待分享 POI 点的 UID，代码如下：

```
// 点击［poi 点分享］根据 uid 发起短串搜索
-(IBAction)poiShortUrlSearch
{
    // 根据 uid(06d2dffda107b0ef89f15db6)发起短串搜索获取 poi 分享的 url
    BOOL flag = [_search poiDetailShareUrl:@"06d2dffda107b0ef89f15db6"];
    if (flag) {
        NSLog(@"poiDetailShareUrl search success.");

    }
    else{
        NSLog(@"poiDetailShareUrl search failed!");
    }
}

// 返回短串分享 url
- (void)onGetShareUrl:(NSString*) url withType:(BMK_SHARE_URL_TYPE) urlType
errorCode:(int)error
{
    if (error == BMKErrorOk)
    {
        // poi 点的名字
        NSString* geoName=@"故宫博物院";
        // poi 点的地址信息
        NSString* addr =@"北京市景山前街 4 号";
        // showmeg 是要分享的信息,除了包含短串 url 外还可以包含 poi 点的名字，地址等其它信息
        showmeg = [NSString stringWithFormat:@"这里
是:%@\r\n%@\r\n%@",geoName,addr,url];
        // 显示 Alert 视图
        UIAlertView *myAlertView = [[UIAlertView alloc] initWithTitle:@"短串分享"
message:showmeg delegate:self cancelButtonTitle:nil otherButtonTitles:@"分享",@"
取消",nil];
```

```
        myAlertView.tag = 1000;
        [myAlertView show];
        [myAlertView release];
    }
    else
    {
        NSLog(@"获取短串分享 url 错误,errorCode=%d",error);
    }
}
```

示例代码中相关截图如下图所示：

6.2. 反 Geo 点分享

反 Geo 点具体参数设置和检索方法如下：

```
/**
 *获取反 geo 短串分享 url
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetShareUrl 通知
```

```
 *@param coor 经纬度坐标
 *@param name 标题
 *@param address 地址
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)reverseGeoShareUrl:(CLLocationCoordinate2D)coor poiName:(NSString*)name
poiAddress:(NSString*)address;
```

反 Geo 点的示例代码详见：

BaiduMapApiTutorial/src/Chapter5/ReverseGeoShortUrlShareDemoViewControl
ler

步骤 1：

在 ViewController.h 中声明 BMKSearch 对象，并将 ViewController 实现 BMKS
earchDelegate、MFMessageComposeViewControllerDelegate 协议，代码如下：

```
#import <UIKit/UIKit.h>
#import <MessageUI/MessageUI.h>
#import <MessageUI/MFMessageComposeViewController.h>
#import "BMapKit.h"

@interface ReverseGeoShortUrlShareDemoViewController :
UIViewController<BMKMapViewDelegate,
BMKSearchDelegate,MFMessageComposeViewControllerDelegate> {
    IBOutlet BMKMapView* _mapView;
    IBOutlet UIButton* _reverseGeoShortUrlButton;
    BMKSearch* _search;
    bool isPoiShortUrlShare;

}
-(IBAction)reverseGeoShortUrlSearch;

@end
```

步骤 2：

根据反向地理编码结果，生成一个用于分享的短连接。方法为：-(BOOL)rev
erseGeoShareUrl:(CLLocationCoordinate2D)coorpoiName:(NSString*)namepoiAddres
s:(NSString*)address;参数 location：共享点经纬度坐标；name：共享点的名称；a
ddress：共享点的地址。代码如下：

```
@interface ReverseGeoShortUrlShareDemoViewController ()
{
    //名称
    NSString* geoName;
```

```
    //地址
    NSString* addr;
    //坐标
    CLLocationCoordinate2D pt;
    //短串
    NSString* shortUrl;
    //分享字符串
    NSString* showmeg;
}
@end

// 初始化变量数据
-(void)initPoiData
{
    // 反 geo 点的经纬度坐标
    pt.latitude = 39.920038;
    pt.longitude = 116.403596;
    // 反 geo 点的名字——泡泡上显示的名字，可以自定义
    geoName = @"自定义泡泡名字";
    // 反 geo 点的地址信息
    addr =@"北京市景山前街 4 号";
}

// 点击［反向地理编码结果分享］先发起反 geo 点分享短串搜索
-(IBAction)reverseGeoShortUrlSearch
{
    // 根据经纬度坐标,名称，地址发起短串搜索获取反 geo 点分享的 url
    BOOL flag = [_search reverseGeoShareUrl:pt poiName:geoName poiAddress:addr];
    if (flag) {
        NSLog(@"reverseGeoShortUrl search success.");

    }
    else{
        NSLog(@"reverseGeoShortUrl search failed!");
    }
}

// 返回短串分享 url
- (void)onGetShareUrl:(NSString*) url withType:(BMK_SHARE_URL_TYPE) urlType
errorCode:(int)error
{
    if (error == BMKErrorOk)
    {
```

```
        showmeg = [NSString stringWithFormat:@"这里
是:%@\r\n%@\r\n%@",geoName,addr,url];
        UIAlertView *myAlertView = [[UIAlertView alloc] initWithTitle:@"短串分享"
message:showmeg delegate:self cancelButtonTitle:nil otherButtonTitles:@"分享",@"
取消",nil];
        myAlertView.tag = 1000;
        [myAlertView show];
        [myAlertView release];
    }
    else
    {
        NSLog(@"获取反 geo 点短串分享 url 错误,errorCode=%d",error);
    }
}
```

示例代码中相关截图如下图所示：

6.3. 事件监听

1、 短串分享：<BMKSearchDelegate>

（1） 返回短串分享结果

```
/**
 *返回分享 url
 *@param url 返回结果 url
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetShareUrl:(NSString*) url withType:(BMK_SHARE_URL_TYPE) urlType
errorCode:(int)error;
```

7. 小结

7.1. 检索方法和事件监听

1、 兴趣点检索

```
//-----------------------------------检索方法-----------------------------------

/**
 *城市 POI 检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetPoiResult 通知
 *@param city 城市名，如果为 nil 则在当前底图范围搜索
 *@param key 关键词
 *@param index 页码，如果是第一次发起搜索，填 0，根据返回的结果可以去获取第 n 页的结果，页码从 0 开始
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)poiSearchInCity:(NSString*)city withKey:(NSString*)key pageIndex:(int)index;

/**
 *根据范围和检索词发起范围检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetPoiResult 通知
 *@param key 关键词
 *@param ptLB 地理坐标，范围的左下角
 *@param ptRT 地理坐标，范围的右下角
 *@param index 页码，如果是第一次发起搜索，填 0，根据返回的结果可以去获取第 n 页的结果，页码从 0 开始
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)poiSearchInbounds:(NSString*)key leftBottom:(CLLocationCoordinate2D)ptLB rightTop:(CLLocationCoordinate2D)ptRT pageIndex:(int)index;

/**
 *根据中心点、半径和检索词发起周边检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetPoiResult 通知
 *@param key 关键词
 *@param ptCenter 中心点地理坐标
 *@param radius 半径，单位：米 必须大于 0
 *@param index 页码，如果是第一次发起搜索，填 0，根据返回的结果可以去获取第 n 页的结果，页码从 0 开始
 *@return 成功返回 YES，否则返回 NO
```

```
 */
- (BOOL)poiSearchNearBy:(NSString*)key center:(CLLocationCoordinate2D)ptCenter
radius:(int)radius pageIndex:(int)index;

//-----------------------------------事件监听-----------------------------------

/**
 *返回 POI 搜索结果
 *@param poiResultList 搜索结果列表，成员类型为 BMKPoiResult
 *@param type 返回结果类型： BMKTypePoiList,BMKTypeAreaPoiList,BMKAreaMultiPoiList
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetPoiResult:(NSArray*)poiResultList searchType:(int)type
errorCode:(int)error;
```

2、 地理编码

```
//-----------------------------------检索方法-----------------------------------

/**
 *根据地址名称获取地理信息-正向
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetAddrResult 通知
 *@param addr       地址信息
 *@param city       城市名称
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)geocode:(NSString*)addr withCity:(NSString*)city;

/**
 *根据地理坐标获取地址信息-反向
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetAddrResult 通知
 *@param center 点坐标
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)reverseGeocode:(CLLocationCoordinate2D)center;

//-----------------------------------事件监听-----------------------------------

/**
 *返回地址信息搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetAddrResult:(BMKAddrInfo*)result errorCode:(int)error;
```

3、 路径规划

```
//-----------------------------------检索方法-----------------------------------

/**
 *驾乘路线检索-无途经点
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetDrivingRouteResult 通知
 *@param startCity 起点所在城市，起点为坐标时可不填
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param endCity 终点所在城市，终点为坐标时可不填
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)drivingSearch:(NSString*)startCity startNode:(BMKPlanNode*)start
endCity:(NSString*)endCity endNode:(BMKPlanNode*)end;

/**
 *驾乘路线检索-多途经点
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetDrivingRouteResult 通知
 *@param startCity 起点所在城市，起点为坐标时可不填
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param endCity 终点所在城市，终点为坐标时可不填
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@param wayPointsArray 途经点数组，存储 BMKPlanNode 信息的节点。wayPointsArray 为 nil 或
空时，表示没有途经点。最多支持 10 个途经点，超过 10 个请求发送不成功。
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)drivingSearch:(NSString*)startCity startNode:(BMKPlanNode*)start
endCity:(NSString*)endCity endNode:(BMKPlanNode*)end
throughWayPoints:(NSArray*)wayPointsArray;

/**
 *公交路线检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetTransitRouteResult 通知
 *@param city 城市名，用于在哪个城市内进行检索
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)transitSearch:(NSString*)city startNode:(BMKPlanNode*)start
endNode:(BMKPlanNode*)end;

/**
 *步行路线检索
```

```
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetWalkingRouteResult 通知
 *@param startCity 起点所在城市，起点为坐标时可不填
 *@param start 检索的起点，可通过关键字、坐标两种方式指定
 *@param endCity 终点所在城市，终点为坐标时可不填
 *@param end 检索的终点，可通过关键字、坐标两种方式指定
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)walkingSearch:(NSString*)startCity startNode:(BMKPlanNode*)start
endCity:(NSString*)endCity endNode:(BMKPlanNode*)end;

//-----------------------------------事件监听-----------------------------------

/**
 *返回驾乘搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetDrivingRouteResult:(BMKPlanResult*)result errorCode:(int)error;

/**
 *返回公交搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetTransitRouteResult:(BMKPlanResult*)result errorCode:(int)error;

/**
 *返回步行搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetWalkingRouteResult:(BMKPlanResult*)result errorCode:(int)error;
```

4、 公交线路查询

```
//-----------------------------------检索方法-----------------------------------

/**
 *公交详情检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetBusDetailResult 通知
 *@param city 城市名，用于在哪个城市内进行检索
 *@param busLineUid 公交线路的 uid
 *@return 成功返回 YES，否则返回 NO
 */
```

```
- (BOOL)busLineSearch:(NSString*)city withKey:(NSString*)busLineUid;

//-----------------------------------事件监听-----------------------------------

/**
 *返回 busdetail 搜索结果
 *@param busLineResult 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetBusDetailResult:(BMKBusLineResult*)busLineResult
errorCode:(int)error;
```

5、 在线建议查询

```
//-----------------------------------检索方法-----------------------------------

/**
 *搜索建议检索
 *@param key       搜索建议的检索关键字
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetSuggestionResult 通知
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)suggestionSearch:(NSString*)key;

/**
 *特定城市内搜索建议检索
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetSuggestionResult 通知
 *@param key      搜索建议的检索关键字
 *@param cityname 城市名，用于指定在特定城市内检索
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)suggestionSearch:(NSString*)key inCity:(NSString*)cityname;

//-----------------------------------事件监听-----------------------------------

/**
 *返回 suggestion 搜索结果
 *@param result 搜索结果
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetSuggestionResult:(BMKSuggestionResult*)result errorCode:(int)error;
```

6、 短串分享

```
//-----------------------------------检索方法-----------------------------------
```

```
/**
 *获取 poi 详情短串分享 url
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetShareUrl 通知
 *@param uid poi 的 uid
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)poiDetailShareUrl:(NSString*) uid;

/**
 *获取反 geo 短串分享 url
 *异步函数，返回结果在 BMKSearchDelegate 的 onGetShareUrl 通知
 *@param coor 经纬度坐标
 *@param name 标题
 *@param address 地址
 *@return 成功返回 YES，否则返回 NO
 */
- (BOOL)reverseGeoShareUrl:(CLLocationCoordinate2D)coor poiName:(NSString*)name
poiAddress:(NSString*)address;

//-----------------------------------事件监听-----------------------------------

/**
 *返回分享 url
 *@param url 返回结果 url
 *@param error 错误号，@see BMKErrorCode
 */
- (void)onGetShareUrl:(NSString*) url withType:(BMK_SHARE_URL_TYPE) urlType
errorCode:(int)error;
```

7.2. 检索常量

常量名称	值	说明
BMKInvalidCoordinate	-1	无效坐标
兴趣点检索策略常量		
BMKTypeCityList	7	城市列表
BMKTypePoiList	11	城市内搜索 POI 列表
BMKTypeAreaPoiList	21	范围搜索、周边搜索 POI 列表
BMKTypeAreaMultiPoiList	45	多关键字范围搜索、周边搜索 POI 列表

驾车规划策略常量		
BMKCarTrafficFIRST	60	躲避拥堵， 若无事实路况，默认按照时间优先策略
BMKCarTimeFirst	0	时间优先
BMKCarDisFirst	1	最短距离
BMKCarFeeFirst	2	较少费用
公交规划策略常量		
BMKBusTimeFirst	3	时间优先
BMKBusTransferFirst	4	最少换乘
BMKBusWalkFirst	5	最小步行距离
BMKBusNoSubway	6	不含地铁

更多内容详见：inc/BMKSearch.h

7.3. 错误码

错误名称	错误码	说明
BMKErrorOk	0	正确，无错误
BMKErrorConnect	2	网络连接错误
BMKErrorData	3	数据错误
BMKErrorRouteAddr	4	起点或终点选择有歧义
BMKErrorResultNotFound	100	搜索结果未找到
BMKErrorParse	310	数据解析失败

更多错误码详见：inc/BMKTypes.h 中 BMKErrorCode