高德地图 Windows Phone API V1.1.1 开发指南

高德软件有限公司
2013 年 6 月·北京

法律声明

高德地图 API 欢迎用户的任何建议或意见。

目录

第1章 简介

1.1 什么是高德地图 Windows Phone API？

高德地图 Windows Phone API 是一套基于 Windows Phone 及以上版本的地图应用程序开发接口。通过该接口，用户可使用高德地图数据和服务轻松构建功能丰富、交互性强的地图应用。高德地图 Windows Phone API 不仅包含构建地图的基本接口，还提供了诸如本地搜索、路线规划等服务，用户可以根据自己的需要进行选择。

1.2 面向的读者

高德地图 Windows Phone API 是提供给具有一定 Windows Phone 开发经验和了解面向对象概念的读者使用的。此外，读者还应该对地图产品有一定的了解。

用户在使用中遇到任何问题，都可以在高德地图 API 网站 http://api.amap.com/ 的问答社区进行反馈。

1.3 兼容性

支持 Windows Phone 7.5 （发行版本） 或 Windows Phone OS 7.1 （操作系统版本）。

1.4 申请 Key

用户必须先注册 AMap 帐户 ,才能获得 API KEY。申请 KEY 的用户需要在 AMap API 网站 http://api.amap.com/Account/login.shtml 注册用户。每个账户只允许申请 10 个 Key。

第2章 开发环境

2.1.1 开发平台

Microsoft Visual Studio 2010 或 Microsoft Visual Studio 2010 Express。

2.1.2 开发语言

支持 C# 语言。

2.1.3 软件开发工具包

- Microsoft Visual Studio 2010。
- Windows Phone Software Development Kit (SDK)。
- Microsoft Visual Studio 2010 Service Pack 1 (SP1)。

2.1.4 硬件配置

- 硬盘空间：驱动器上保留 3GB 硬盘空间。
- 内存：2GB。

- 显卡：Windows Phone 模拟器必须符合 DirectX 10 版定义的规格，并配备 WDDM 1.1 版的驱动程序。

2.2 运行环境

2.2.1 操作系统

适用于 Windows Phone 7.5 （发行版本） 或 Windows Phone OS 7.1 （操作系统版本）。

第3章 地图显示

在开始编码应用程序前，首先在 Microsoft Visual Studio 2010 Express for Windows Phone 内创建项目。具体步骤如下：

3.1 新建 Windows Phone 设备工程

在创建编码程序前，首先在 Microsoft Visual Studio 2010 Express for Windows Phone 内创建项目，具体步骤如下：

（1） 启动 Microsoft Visual Studio 2010 Express for Windows Phone，点击菜单栏**文件>新建项目，**在弹出的对话框中选择 **已安装模板>Visual C#>Silverlight for Windows Phone**，在中间栏选择 **Windows Phone 应用程序，**输入工程名：**AutoNavi_gettingStarted**，如图 3-1 所示：

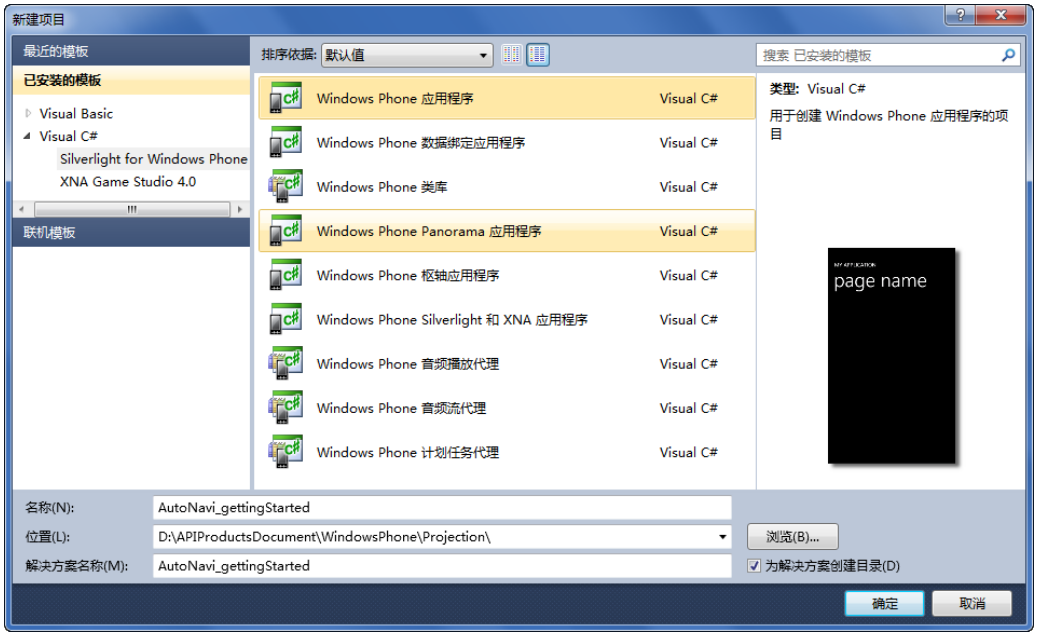

图 3-1 新建项目

（2） 点击**确定**，在弹出对话框中选择目标 Windows Phone OS 版本为 Windows Phone OS 7.1，如图 3-2 所示：

图 3-2 选择 Windows Phone OS 版本

（3） 单击确定，完成项目创建。在项目显示的 MainPage.xaml 文件结构的预览窗口和代码窗口如图 3-3 所示，在项目的**解决方案资源管理器**中，可找到与 MainPage.xaml 对应的 CS 文件，打开即可查看 C#代码。

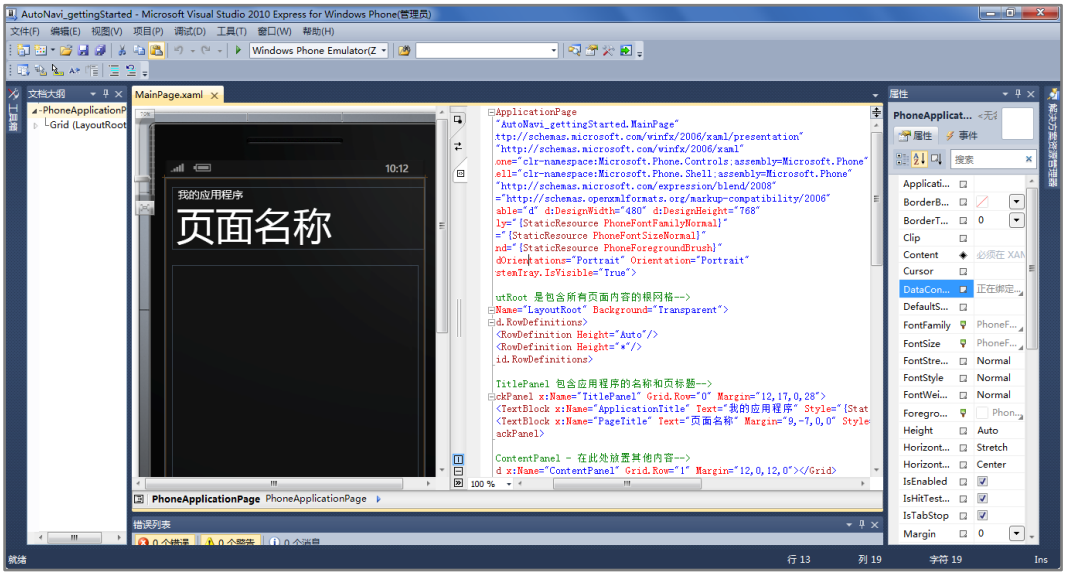

图 3-3 文件结构

3.2 引入地图 Key 和开发库

（1） 从网站下载 AMap_WP7_Api_Lib 文件，在该项目的解决方案资源管理器中依次选择**项目**>**添加引用**，弹出**添加引用**对话框，单击**浏览**选项卡，选择 AMap.WP7.Map.API.dll 和 AMap.WP7.Search.API.dll 文件，这样就可以添加高德地图 API Windows Phone 版的 dll 文件的引用，如图 3-4 所示：

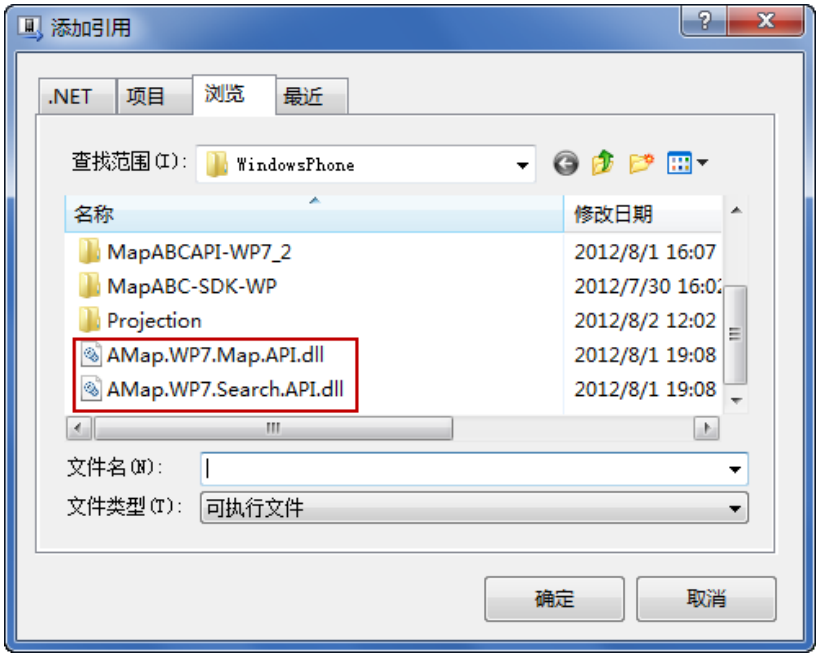

图 3-4 引入地图开发库

2. 将用户申请的 Key 添加到 Windows Phone7 应用中。在 MainPage.xaml.cs 的 MainPage()中添加代码 AMapConfig.Key = "[用户 Key]";

3.3 地图显示

引入高德地图 Windows Phone API 的地图开发库之后，添加相关代码来初始化地图，步骤如下所示：

（1） 在项目的 MainPage.xaml 中引入高德地图 Windows Phone API 的命名空间，添加引用的命名空间示例代码如下：

```
xmlns:AMap="clr-namespace:Com.AMap.Maps.Api;assembly=AMap.WP7.Map.API"
```

（2） 在 MainPage.xaml 中添加地图标题为**显示地图**，中心点为"**116,39**"的地图应用，代码如下所示：

```
<Grid x:Name="LayoutRoot" Background="Transparent">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
        <TextBlock x:Name="ApplicationTitle" Text="显示地图"
                   Style="{StaticResource PhoneTextNormalStyle}"/>
    </StackPanel>
    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,-10,12,0">
        <AMap:MMap x:Name="map" Center="116,39" ToolBar="Visible"/>
    </Grid>
</Grid>
```

（3） 单击 按钮，编译 Windows Phone 应用程序，在弹出的模拟器中运行程序如图 3-5 所示：

图 3-5 地图显示

第4章 地图手势操作

4.1 地图移动和缩放

高德地图 API 里已经针对地图的移动缩放做了处理，在 Windows Phone 上采用高德地图 API 开发地图应用可以直接使用地图的移动缩放功能，手指在屏幕上滑动可拖动地图，双击屏幕上的某点则地图放大，如图 4-1 所示。

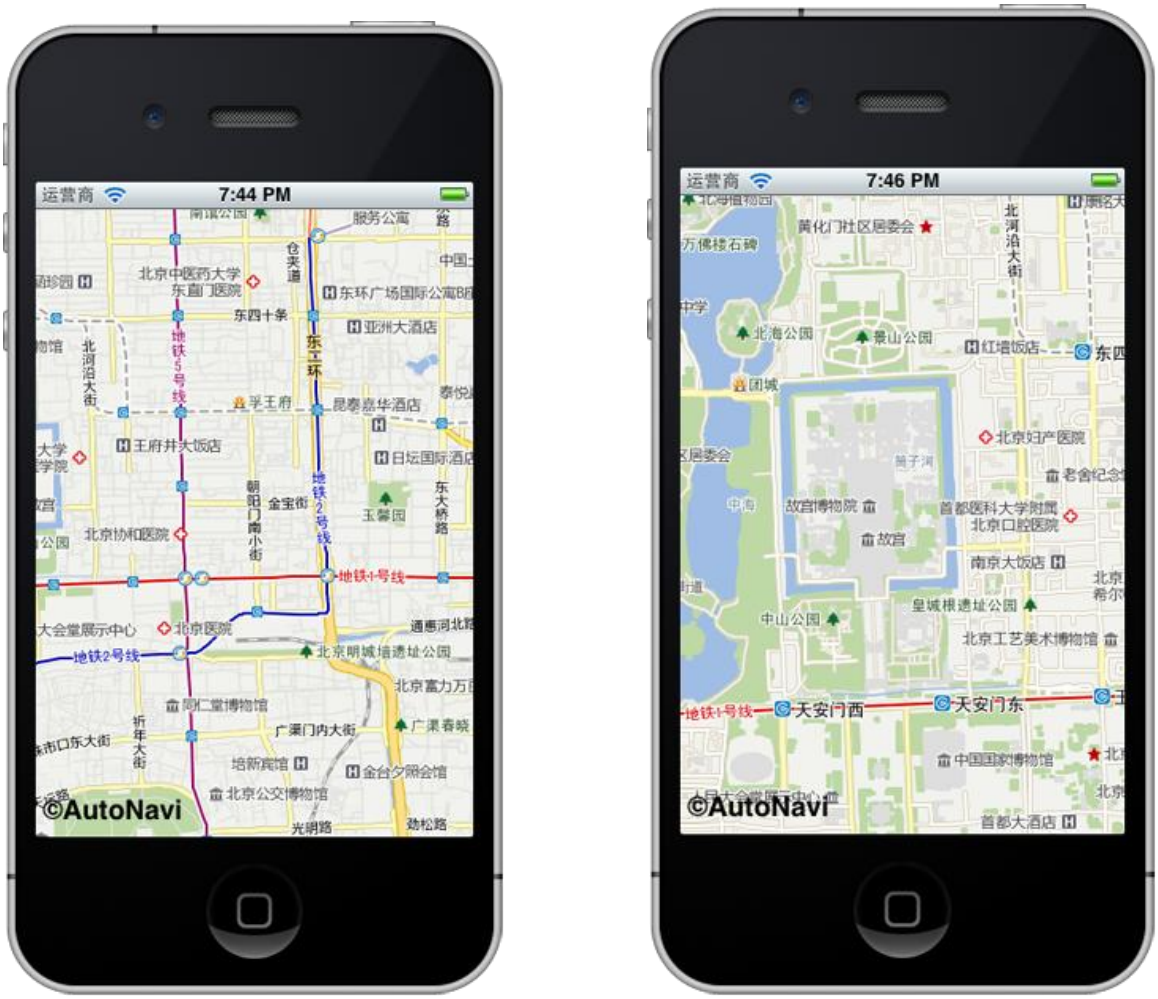

图 4-1 地图移动和缩放

4.2 手势添加覆盖物

高德地图 Windows Phone API 支持通过手势单击地图，在地图上添加点、线、面、矩形、圆、测距等操作。Map 类的属性 GestureType 为当前手势动作类型，通过给该属性赋值，设置手指点击来执行的操作。以下代码显示通过手指点击屏幕，在地图上画点、线、面、圆等操作，可以点击删除按钮来删除添加的覆盖物：

```
//map 为实例化的地图对象
//鼠标画点
map.GestureType = Com.AMap.Maps.Api.Enums.MapGestureType.AddMarker;
…
//鼠标画线
map.GestureType = Com.AMap.Maps.Api.Enums.MapGestureType.DrawLine;
…
//鼠标画多边形
map.GestureType = Com.AMap.Maps.Api.Enums.MapGestureType.DrawPolygon;
…
//鼠标画矩形
map.GestureType = Com.AMap.Maps.Api.Enums.MapGestureType.DrawRectangle;
…
//鼠标测直线距离
map.GestureType = Com.AMap.Maps.Api.Enums.MapGestureType.Ruler;
…
//鼠标画圆
 map.GestureType = Com.AMap.Maps.Api.Enums.MapGestureType.DrawCircle;
```

显示效果如图 4-2 所示：

图 4-2 手势添加和删除覆盖物

第5章 地图图层

5.1 地图图层的概念

地图是由图层组成的，地图可以包括一个或多个图层，每个图层在每个级别都是由若干张图块组成，它们覆盖了地球的整个表面。例如您所看到包括街道、兴趣点、学校、公园等内容的地图展现就是一个图层，另外交通流量的展现也是通过图层来实现的。

5.2 底图

基本的地图图层，包括若干个缩放级别，显示基本的地图信息，包括道路、街道、学校、公园等内容。

5.3 实时交通

实时交通功能支持 25 个城市的实时交通信息查询，支持的城市有：北京，上海，广州，深圳，天津，南京，成都，沈阳，重庆，武汉，长沙，宁波，青岛，杭州，石家庄，太原，常州，大连，西安，长春，无锡，厦门，福州，珠海，东莞。在地图中显示实时交通信息示例代码如下：

```
...
MTileLayer tilelayer = new MTileLayer(MTileLayerType.Traffic);
map.AddLayer(tilelayer);
```

实时交通在地图上显示如图 5-1 所示：

图 5-1 实时交通

第6章 地图覆盖物

6.1 覆盖物概述

覆盖物是指覆盖到地图上的所有事物，如自定义的标注、折线、多边形和圆等。覆盖物有自己的地理属性，通过设置地理属性来控制覆盖物在地图中的显示位置。

高德地图 Windows Phone API 中包括的覆盖物主要有：标注、线（直线/折线）、多边形、矩形和圆。

6.2 地图点标注

可以在地图上添加某个点的标注信息，对该点进行移动、删除等操作。

高德地图 Windows Phone API 提供的地图标注为 MMarker 类。

添加地图点标注，代码如下所示：

```
MMarker mk = new MMarker(new MLngLat(116.397945, 39.90817));// 构造 MMarker
map.Children.Add(mk);//添加到地图
map.Center = mk.LngLat;//设置地图中心点
Pivot.SelectedIndex = 1;//显示效果
```

运行效果如图 6-1 所示：

图 6-1 添加标注

6.3 添加多折线

在地图上添加多折线，类为 MPolyline，可以通过该类设置线的粗细、透明度、显示信息窗体、删除此多折线等操作，添加多折线代码如下：

```
MLngLatCollection mc = new MLngLatCollection();//构造经纬度序列集合
mc.Add(new MLngLat(116,39));//集合添加点经纬度坐标
mc.Add(new MLngLat(116.2, 39.3));//集合添加点经纬度坐标
mc.Add(new MLngLat(116.12, 39.45));//集合添加点经纬度坐标
MPolyline ml=new MPolyline();//构造线对象
ml.LngLats = mc;//mc 为组成线的经纬度坐标串
ml.LineColor = Utilities.HexToColor("#022672") ;//线的颜色
ml.LineThickness = 3;//线的粗细
map.Children.Add(ml);//添加到地图
```

```
Pivot.SelectedIndex = 1;//显示效果
```

运行结果如图 6-2 所示：

图 6-2 添加多折线

6.4 添加多边形

添加多边形是通过 MPolygon 类来实现。通过该类设置多边形线的宽度，大小，透明度等，绘制多边形代码如下所示：

```
MLngLatCollection mp = new MLngLatCollection();//构造经纬度序列集合
mp.Add(new  MLngLat(116.3225715637207,39.909209536859834));//集合添加点经纬度坐标
mp.Add(new MLngLat(116.3725715637207, 39.95920953685983));//集合添加点经纬度坐标
mp.Add(new  MLngLat(116.42257156372072,  39.95920953685983));//集合添加点经纬度坐标
MPolygon polygon = new MPolygon();//构造多边形对象
```

```
polygon.LngLats = mp ;//mp 为组成多边形的经纬度坐标串
polygon .FillColor = Utilities.HexToColor("#a3b4ff");//填充颜色
polygon .FillOpacity =0.8;//填充透明度
polygon.LineColor = Utilities.HexToColor("#022672");//线的颜色
polygon.LineThickness = 3;//线的粗细
map.Children.Add(polygon);//添加到地图
Pivot.SelectedIndex = 1;//显示效果
```

运行结果如图 6-3 所示：

图 6-3 添加多变形

6.5 添加圆

使用 MCircle 类实现圆的添加。通过该类实现圆的添加、删除等操作，设置圆的大小，边框线的粗细等属性设置。

添加圆代码如下所示：

```
MLngLat mc = new MLngLat(116.3225715637207, 39.909209536859834);//构造经纬度坐标
MCircle circle = new MCircle();//构造圆对象
circle.SetCenterAndRadius(mc, 1000);//设置圆的中心和半径，半径单位为米
circle.FillColor = Utilities.HexToColor("#a3b4ff");//填充颜色
circle.FillOpacity = 0.8;//填充透明度
circle.LineColor = Utilities.HexToColor("#022672");//线的颜色
circle.LineThickness = 3;//线的粗细
map.Children.Add(circle);//添加到地图
Pivot.SelectedIndex = 1;//显示效果
```

运行结果如图 6-4 所示：

图 6-4 绘制圆

6.6 添加矩形

使用 MRectangle 类可以添加、删除矩形覆盖物，通过该类可设置边框线的颜色，大小，透明度等属性。

添加矩形示例代码如下：

```
MLngLatCollection mp = new MLngLatCollection();//构造经纬度序列集合
mp.Add(new MLngLat(116.3225715637207, 39.909209536859834));//集合添加点经纬度坐标
mp.Add(new  MLngLat(116.3725715637207,  39.95920953685983));//集合添加点经纬度坐标
MRectangle rectangle = new MRectangle();//构造矩形对象
rectangle.LngLats = mp;//mp 里的前两个坐标为矩形的左上角和右下角坐标
rectangle.FillColor = Utilities.HexToColor("#a3b4ff");//填充颜色
rectangle.FillOpacity = 0.8;//填充透明度
rectangle.LineColor = Utilities.HexToColor("#022672");//线的颜色
rectangle.LineThickness = 3;//线的粗细
map.Children.Add(rectangle);//添加到地图
Pivot.SelectedIndex = 1;//显示效果
```

运行结果如图 6-5 所示：

6-5 添加矩形

第7章 自动定位

高德地图 Windows Phone API 可以实现移动客户端的自动定位功能，使用系统提供的类 GeoCoordinateWatcher 获取 GPS 坐标，通过 MRGCSearch.GPSToOffset()方法将 GPS 获取的坐标转化为高德地图经纬度坐标，然后在地图上显示定位信息。具体示例代码如下所示：

初始化一个 GeoCoordinateWatcher 对象，定位成功后调用 _watcher_PositionChanged()方法进行坐标转换。

```
public void RunGeoCoordinateWatcher()
{
    GeoCoordinateWatcher _watcher = new
                GeoCoordinateWatcher(GeoPositionAccuracy.Default);
    _watcher.PositionChanged += new
                EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(_
                watcher_PositionChanged);
    _watcher.Start();
}
```

在地图上添加圆覆盖物以标识当前位置，圆心为当前位置的经纬度坐标，半径是当前位置的 500m。

```
void MRGCSearchResultAction(MRGCSearchResult o)
{
     if (o.Erro == null)
     {
          if (centerMarker == null)
          {
               map.Children.Add(centerMarker = new MMarker()
               {
                    LngLat = o.RGCItemList[0],
                    IconURL = "/location_on.png",
                    //(0.5, 0.5)是以图片的比例，表示此图片的中心点和经纬度坐标点对齐。
                    Anchor = new Point(0.5, 0.5)
               });
               map.Children.Add(centerCircle = new MCircle() { });
               centerCircle.SetCenterAndRadius(o.RGCItemList[0], 500);
          }
          else
          {
               centerMarker.LngLat = o.RGCItemList[0];
               centerCircle.SetCenterAndRadius(o.RGCItemList[0], 500);
```

```
                }
                //地图中心点为转换后的经纬度点位置
                map.Center = o.RGCItemList[0];
            }
        }
```

第8章 地图查询

在 Windows Phone 平台上使用高德地图 Windows Phone API 提供的查询接口实现关键字查询、关键字周边关键字查询、坐标点周边关键字查询功能。

实现查询的类是 MPOISearch，该类提供三个静态函数 PoiSearchByKeywords()、PoiSearchByCenKeywords()、PoiSearchByCenLngLat() 分别实现 3 种查询，或是结合 MPOISearchOption 类来实现 3 种查询。MPOISearchOption 类是通过设置查询关键字，查询坐标点、查询类型等属性参数，其中 Config 属性用来设置查询类型，当 config=BESN 时，表示关键字查询；当 config=BELSBN 时，表示坐标点周边关键字查询，即对某一地理坐标的周边进行指定关键字的查询；当 config=BELSBXY 时，表示关键字周边的关键字查询，然后调用 MPOISearch.PoiSearchWithOption() 静态方法来实现查询。下面以关键字查询为例来介绍这两种查询方法。

8.1 关键字查询

该查询功能根据关键字，查询该关键字对应点的位置描述，包括该点所在省市区，周边的 POI 和道路等信息。

方法一：直接调用 MPOISearch.PoiSearchByKeywords() 静态函数，输入关键字，城市，回调函数进行查询。回调函数 searchCallBack 显示查询信息。示例代码如下所示：

```
MPOISearch.PoiSearchByKeywords("肯德基", "010", searchCallBack);
```

方法二：实例化一个 MPOISearchOption 对象，设置查询参数，将该对象传入静态函数 MPOISearch.PoiSearchWithOption()，其中回调函数 searchCallBack 显示查询信息来实现查询，示例代码如下：

```
// <summary>
// 根据 MPOISearchOption 对象设置查询参数进行 POI 查询。
```

```
// </summary>
private void POISearchWithOption()
{
        MPOISearchOption mPOISearchOption = new MPOISearchOption();
        mPOISearchOption.Batch = 1;
        mPOISearchOption.CenName = "肯德基";
        mPOISearchOption.CenX = 0.0;
        mPOISearchOption.CenY = 0.0;
        mPOISearchOption.CityCode = "010";
        mPOISearchOption.Config = "BELSBN";  //“BELSBN”是关键字查询
        mPOISearchOption.Encode = "GBK";//utf-8
        mPOISearchOption.Naviflag = 1;
        mPOISearchOption.Number = 20;
        mPOISearchOption.Range = 3000;
        mPOISearchOption.SearchName = "肯德基";
        mPOISearchOption.SearchType = "";
        mPOISearchOption.Sr = 0; //1 按距离排序
        mPOISearchOption.SrcType = "POI";
        MPOISearch.PoiSearchWithOption(mPOISearchOption, searchCallBack);
}
```

回调函数 searchCallBack()用来设置对查询结果的处理，如下所示示例为将查询城市为“北京”，关键字为“肯德基”的地址用标注点显示在地图上，点击标注会弹出信息窗口显示该点的名称和地址信息。

```
void searchCallBack(MPOISearchResult sender)
{
      if (sender.Erro == null)
      {
            //初始化一个覆盖物数组
            List<MOverlay> list = new List<MOverlay>();
            for (int i = 0; i < sender.POIs.Count; i++)
            {
                  MMarker mk;
                  map.Children.Add(mk = new MMarker()
                  {
                        LngLat = new MLngLat(sender.POIs[i].X, sender.POIs[i].Y),
                        TipFrameworkElement = new MTip() { Title = sender.POIs[i].Name,
                                    ContentText = sender.POIs[i].Address }
                   });
                   //将标注点添加到 list 数组中
                   list.Add(mk);
            }
            //设置覆盖物到合适的地图视野级别
            map.SetFitview(list);
      }
}
```

8.2 坐标点周边关键字查询

该功能实现对某一地理坐标点进行该点周边的关键字查询。查询使用的方法可以是MPOISearch.PoiSearchByCenLngLat()方法，也可以通过 MPOISearchOption 和MPOISearch.PoiSearchWithOption()进行查询。具体实现方法请参考关键字查询示例代码。

8.3 关键字周边关键字查询

该功能实现对某一关键字进行该关键字周边的关键字查询 ,如查询关键字“方恒国际中心”的周边以“肯德基”为关键字的信息。查询使用的方法可以是MPOISearch.PoiSearchByCenKeywords()，也可以通过 MPOISearchOption 和MPOISearch.PoiSearchWithOption()进行查询。具体实现方法请参考关键字查询示例代码。

第9章 路径规划

9.1 公交查询

公交查询包括公交名称查询、公交 ID 查询，公交站点查询。实现查询有两种方法实现，一种是直接使用 MBusLineSearch 的静态函数 BusSearchByBusId()、BusSearchByBusName()、BusSearchByBusStation()来实现公交 ID、公交名称、公交站点查询，一种是使用 MBusLineSearchOption 和 MBusLineSearch.BusLineSearchWithOption()方法来实现查询。

9.1.1 公交名称查询公交信息

该功能实现通过公交名称查询公交起点、终点、途径站点、公交名称、长度、公交类型等公交信息。方法之一是通过实例化一个 MBusLineSearchOption 对象来设置相关信息，其中属性 StationName 的值设置公交名称，将该对象作为参数传给 MBusLineSearch.BusLineSearchWithOption()静态函数，实现此查询。也可以通过 MBusLineSearch.BusSearchByBusName() 直接输入公交名称、城市、回调函数实现此查询，然后通过回调函数 CallBack() 来设置查询结果显示方式，示例代码如下：

```
private void BusLineSearchWithOption()
{
    MBusLineSearchOption busLineSearchOpt = new MBusLineSearchOption();
    busLineSearchOpt.Batch = 1;
    busLineSearchOpt.CityCode = "";
    busLineSearchOpt.Config = "BusLine"; //如果是公交查询，设置其值为 BusLine。
    busLineSearchOpt.Encode = "GBK";
    busLineSearchOpt.Number = 5;
    busLineSearchOpt.ResData = 0;
    busLineSearchOpt.StationName = "二里庄";
    MBusLineSearch.BusLineSearchWithOption(busLineSearchOpt, CallBack);
}
void CallBack(MBusLineSearchResult sender)
{
    MessageBox.Show("通过查询参数类搜索公交引导信息 如下：\nCount：" +
        sender.Count.ToString() + " Record：" + sender.Record.ToString() + "
```

```
            SearchTime：" + sender.SearchTime.ToString() + " Total：" +
            sender.Total.ToString() + " BusLineList.count：" +
            sender.BusLineList.Count);

        if (sender.Erro == null)
        {
            //初始化一个覆盖物数组
            List<MOverlay> list = new List<MOverlay>();
            for (int i = 0; i < sender.BusLineList.Count; i++)
            {
                //显示查询的公交信息。
                string url = "Air：" + sender.BusLineList[i].Air + "\n" + "AutoTicket：" +
                    sender.BusLineList[i].AutoTicket + "\n" +……+ "TerminalName：" +
                    sender.BusLineList[i].TerminalName + "\n" + …+ "\n" +
                    "TimeDesc：" + sender.BusLineList[i].TimeDesc + "\n" +
                    "TimeInterval1：" + sender.BusLineList[i].TimeInterval1 + "\n" +
                    "TimeInterval2：" +…+ sender.BusLineList[i].TimeInterval8 + "\n" +
                    "TotalPrice：" + sender.BusLineList[i].TotalPrice + "\n" + "Type：" +
                    sender.BusLineList[i].Type;
                MessageBox.Show("第" + i + "个 BusLine 信息如下：\n" + url);
            }

            //根据覆盖物来调整视野
            map.SetFitview(list);
        }
    }
```

9.1.2 公交 ID 查询公交信息

该功能实现通过公交 ID 查询公交信息。该方法通过设置 MBusLineSearchOption 类的 Ids 属性，将该对象传入 MBusLineSearch.BusLineSearchWithOption()静态函数，实现查询。也可以通过 MBusLineSearch.BusSearchByBusId() 实现公交 ID 查询。然后通过回调函数 CallBack() 来设置查询结果显示方式，示例代码请参考公交名称查询示例代码。

9.1.3 公交站点查询公交信息

该功能实现通过公交站点名称查询公交信息。该方法通过将实例化 MBusLineSearchOption 对象的 StationName 属性设置为具体值，将该对象传入 MBusLineSearch.BusLineSearchWithOption()静态函数，实现查询。也可以通过

MBusLineSearch.BusSearchByBusStation() 静态函数实现公交站点查询。然后通过回调函数 CallBack() 来设置查询结果显示方式，示例代码请参考公交名称查询示例代码。

9.2 公交导航查询

该功能实现从起点到终点的公交路径的查询。该方法使用的类是公交导航搜索类 MBusRouteSearch。其中有两种方法可以实现该功能，一种是使用 BusSearchByTwoPoi() 静态函数，输入起始点经纬度坐标、城市代码和回调函数实现查询。一种是使用 BusRouteSearchWithOption()实现查询。下面以使用 BusRouteSearchWithOption()方法为例来介绍具体实现方法。

首先，实例化一个 MBusRouteSearchOption 对象，输入公交导航查询的相关参数，包括起始点经纬度坐标，城市代码等，将该对象作为参数传递给 MBusRouteSearch.BusRouteSearchWithOption()函数，其中包括回调函数名称 callback 作为另外参数。callback 函数实现在地图上显示查询结果信息，此示例是将查询的坐标点以多折线的形式显示在地图上。

```
// MBusRouteSearchOption 实例化
private void BusSearchWithOption()
{
    MBusRouteSearchOption m_BusRouteSearchOption = new
    MBusRouteSearchOption();
    m_BusRouteSearchOption.CityCode = "010";
    m_BusRouteSearchOption.Config = "BR";
    m_BusRouteSearchOption.Encode = "GBK";
    m_BusRouteSearchOption.RouteType = 0;
    //设置起始点经纬度坐标信息
    m_BusRouteSearchOption.X1 = 116.38969318005;
    m_BusRouteSearchOption.Y1 = 39.9048015978361;
    m_BusRouteSearchOption.X2 = 116.323902107975;
    m_BusRouteSearchOption.Y2 = 39.907648620322;


    MBusRouteSearch.BusRouteSearchWithOption(m_BusRouteSearchOption,
    CallBack);
}
```

```
void CallBack(MBusRouteSearchResult sender)
{
        if (sender.Erro == null)
        {
            //初始化覆盖物数组
            List<MOverlay> list = new List<MOverlay>();

            //获取查询路线各路段的经纬度坐标，以多折线的形式显示在地图上
            for (int routescount = 0; routescount < sender.Routes.Count;
                    routescount++)
            {
                for (int segmentArraycount = 0; segmentArraycount <
                    sender.Routes[routescount].SegmentArray.Count;
                    segmentArraycount++)
                {
                    //初始化一个多折线
                    MPolyline mp;
                    //将该多折线添加到地图上
                    map.Children.Add(mp = new MPolyline()
                    {
                        //获取路线的经纬度坐标
                        LngLats =
                            sender.Routes[routescount].SegmentArray[segmentArray
                            count].LngLats,
                        LineColor = Colors.Green,
                        LineThickness = 6
                    });
                    list.Add(mp);
                }
            }
            map.SetFitview(list);
    }
   }
```

9.3 驾车导航查询

该功能实现从起点到终点的驾车路径的查询。该方法使用的类是 MRouteSearch。有两种方法实现该查询类，一种是使用 RouteSearchByTwoPoi()输入起始点经纬度坐标和回调函数实现查询，一种是使用 RouteSearchWithOption()实现查询。具体实现方法请参考公交导航查询。

第10章 地理编码

该功能实现地理编码服务，即地址匹配，从已知的地址描述到对应的经纬度坐标的转换，即根据地址信息，查询该地址所对应的点坐标。

有两种方法实现该功能，一种是直接使用 MGeoCode 类的 AddressToGeoCode()静态方法，输入地址信息和回调函数来实现。一种是使用 MGeoCode 类的 AddressToGeoCodeWithOption()方法来实现。

下面来介绍使用MGeoCode类的AddressToGeoCodeWithOption()方法实现地理编码的方法。

首先实例化一个 MGeoCodingOption 对象来设置地理编码相关参数。然后将该对象和回调函数 CallBack 作为参数值传给 MGeoCode.AddressToGeoCodeWithOption()静态方法，CallBack()函数实现地理编码结果的显示。具体示例代码如下所示：

```
private void GeoCodeToAddressWithOption()
{
    MGeoCodingOption geoCodingOpt = new MGeoCodingOption();
    geoCodingOpt.Address = "北京市海淀区中关村";
    MGeoCode.AddressToGeoCodeWithOption(geoCodingOpt, CallBack);
}

void CallBack(MGeoCodingResult sender)
{
    //如果没有返回错误，就执行将地理编码结果显示在地图上
    if (sender.Erro == null)
    {
        //实例化一个覆盖物数组
        List<MOverlay> list = new List<MOverlay>();

        for (int i = 0; i < sender.GeoCodingList.Count; i++)
        {
            MMarker mk;
            //在地图上添加该结构的坐标点
            map.Children.Add(mk = new MMarker()
            {
                LngLat = new MLngLat(sender.GeoCodingList[i].X,
                        sender.GeoCodingList[i].Y),
                 //弹出对话框的信息显示
```

```
                    TipFrameworkElement = new MTip() { Title =
                            sender.GeoCodingList[i].Name, ContentText =
                            sender.GeoCodingList[i].Address }
                });
                list.Add(mk);
            }
            //根据覆盖物来调整视野上
            map.SetFitview(list);
        }
    }
```

第11章 逆地理编码

该功能实现逆地理编码服务，即地址解析服务，具体是指从已知的经纬度坐标到对应的地址描述（如省市、街区、楼层、房间等）的转换服务。

有两种方法实现该功能，一种是直接调用 MReGeoCode 类的 MulPoiToAddress()或 PoiToAddress()静态方法。MulPoiToAddress()为查询多个位置点的的详细地址信息，PoiToAddress()为查询一个点的位置信息。另一种是使用 GeoCodeToAddressWithOption()方法来实现。具体使用方法请参照地理编码示例。