百度地图移动版 API for android
开发指南

2012.06.20

1 简介

什么是百度地图 API?

百度地图移动版 API（Android）是一套基于 Android 1.5 及以上设备的应用程序接口，通过该接口，您可以轻松访问百度服务和数据，构建功能丰富、交互性强的地图应用程序。百度地图移动版 API 不仅包含构建地图的基本接口，还提供了诸如地图定位、本地搜索、路线规划等数据服务，你可以根据自己的需要进行选择。

面向的读者

API 是提供给那些具有一定 Android 编程经验和了解面向对象概念的读者使用。此外，读者还应该对地图产品有一定的了解。

您在使用中遇到任何问题，都可以通过 API 贴吧或交流群反馈给我们。

获取 API Key

用户在使用 API 之前需要获取百度地图移动版 API Key，该 Key 与你的百度账户相关联，您必须先有百度帐户，才能获得 API KEY。并且，该 KEY 与您引用 API 的程序名称有关，具体流程请参照获取密钥。

兼容性

支持 Android 1.5 及以上系统。

2 在您的地图中显示

如何把 API 添加到我的 Android 工程中？

首先将 API 包括的两个文件 baidumapapi.jar 和 libBMapApiEngine_v1_3_3.so 拷贝到工程根目录及 libs\armeabi 目录下，并在工程属性->Java Build Path->Libraries 中选择“Add JARs”，选定 baidumapapi.jar，确定后返回，这样您就可以在您的程序中使用 API 了。

百度地图的“Hello,World”

在 Manifest 中添加使用权限

```
0.<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-per
    mission>
1.<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-per
    mission>
2.<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

```
3.<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-p
    ermission>
4.<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permis
    sion>
5.<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permis
    sion>
6.<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permiss
    ion>
```

在 Manifest 中添加 Android 版本支持

```
0.<supports-screens android:largeScreens="true"  android:normalScreens="true" androi
    d:smallScreens="true"  android:resizeable="true" android:anyDensity="true"/>
1.<uses-sdk android:minSdkVersion="3"></uses-sdk>
```

让创建的地图 Activity 继承 com.baidu.mapapi.MapActivity，并 import 相关类

```
0.          import java.util.ArrayList;
1.          import java.util.List;
2.          import android.content.Context;
3.          import android.graphics.Canvas;
4.          import android.graphics.Paint;
5.          import android.graphics.Point;
6.          import android.graphics.drawable.Drawable;
7.          import android.location.Location;
8.          import android.os.Bundle;
9.          import android.util.Log;
10.         import android.view.View;
11.         import android.widget.Toast;
12.         import com.baidu.mapapi.BMapManager;
13.         import com.baidu.mapapi.GeoPoint;
14.         import com.baidu.mapapi.ItemizedOverlay;
15.         import com.baidu.mapapi.LocationListener;
16.         import com.baidu.mapapi.MKAddrInfo;
17.         import com.baidu.mapapi.MKDrivingRouteResult;
18.         import com.baidu.mapapi.MKGeneralListener;
19.         import com.baidu.mapapi.MKLocationManager;
20.         import com.baidu.mapapi.MKPlanNode;
21.         import com.baidu.mapapi.MKPoiResult;
22.         import com.baidu.mapapi.MKSearch;
23.         import com.baidu.mapapi.MKSearchListener;
```

```
import com.baidu.mapapi.MKTransitRouteResult;
import com.baidu.mapapi.MKWalkingRouteResult;
import com.baidu.mapapi.MKSuggestionResult;
import com.baidu.mapapi.MapActivity;
import com.baidu.mapapi.MapController;
import com.baidu.mapapi.MapView;
import com.baidu.mapapi.MyLocationOverlay;
import com.baidu.mapapi.Overlay;
import com.baidu.mapapi.OverlayItem;
import com.baidu.mapapi.PoiOverlay;
import com.baidu.mapapi.RouteOverlay;
import com.baidu.mapapi.TransitOverlay;

public class MyMapActivity extends MapActivity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }

        @Override
        protected boolean isRouteDisplayed() {
                return false;
        }
}
```

在布局 xml 中添加地图控件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent">
  <TextView android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="@string/hello" />
  <com.baidu.mapapi.MapView android:id="@+id/bmapsView" android:layout_width="fill_parent" android:layout_height="fill_parent" android:clickable="true" />
</LinearLayout>
```

初始化地图 Activity

在地图 Activity 中定义变量： BMapManager mBMapMan = null; 在 onCreate 方法中增加以下代码，并将您申请的 Key 替换“我的 Key”：

```
0.mBMapMan = new BMapManager(getApplication());
1.mBMapMan.init("我的 Key", null);
2.super.initMapActivity(mBMapMan);
3.MapView mMapView = (MapView) findViewById(R.id.bmapsView);
4.mMapView.setBuiltInZoomControls(true);  //设置启用内置的缩放控件
5.MapController mMapController = mMapView.getController();  // 得到 mMapView 的控制权,可
    以用它控制和驱动平移和缩放
6.GeoPoint point = new GeoPoint((int) (39.915 * 1E6),     (int) (116.404 * 1E6));  //用
    给定的经纬度构造一个 GeoPoint，单位是微度 (度 * 1E6)
7.mMapController.setCenter(point);  //设置地图中心点
8.mMapController.setZoom(12);    //设置地图 zoom 级别
```

Override 以下方法,管理 API:

```
0.@Override
1.protected void onDestroy() {
2.    if (mBMapMan != null) {
3.        mBMapMan.destroy();
4.        mBMapMan = null;
5.    }
6.    super.onDestroy();
7.}
8.@Override
9.protected void onPause() {
10.             if (mBMapMan != null) {
11.                 mBMapMan.stop();
12.             }
13.             super.onPause();
14.         }
15.         @Override
16.         protected void onResume() {
17.             if (mBMapMan != null) {
18.                 mBMapMan.start();
19.             }
20.             super.onResume();
21.         }
```

完成上述步骤后，运行程序，结果如下：

3 地图图层

地图图层概念

地图可以包含一个或多个图层，每个图层在每个级别都是由若干张图块组成的，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。

3.1 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。

3.2 实时交通信息

在以下 11 个城市中，支持实时交通信息：北京，上海，广州，深圳，南京，南昌，成都，重庆，武汉，大连，常州。在地图中显示实时交通信息示例如下：

```
mMapView.setTraffic(true);
```

运行程序，结果如下：

3.3 卫星图

```
mMapView.setSatellite(true);
```

3.4 实景图

在此版本 API 中暂不支持。

```
mMapView.setStreetView(true);
```

4 覆盖物

地图覆盖物概述

所有叠加或覆盖到地图的内容，我们统称为地图覆盖物。如标注、矢量图形元素(包括：折线和多边形和圆)、定位图标等。覆盖物拥有自己的地理坐标，当您拖动或缩放地图时，它们会相应的移动。

地图 API 提供了如下几种覆盖物：

- Overlay：覆盖物的抽象基类，所有的覆盖物均继承此类的方法，实现用户自定义图层显示。
- MyLocationOverlay：一个负责显示用户当前位置的 Overlay。
- ItemizedOverlay<Item extends OverlayItem>：Overlay 的一个基类，包含了一个 OverlayItem 列表，相当于一组分条的 Overlay，通过继承此类，将一组兴趣点显示在地图上。
- PoiOverlay：本地搜索图层，提供某一特定地区的位置搜索服务，比如在北京市搜索“公园”，通过此图层将公园显示在地图上。
- RouteOverlay：步行、驾车导航线路图层，将步行、驾车出行方案的路线及关键点显示在地图上。

- TransitOverlay：公交换乘线路图层，将某一特定地区的公交出行方案的路线及换乘位置显示在地图上。

4.1 覆盖物的抽象类 ：Overlay

一般来说，在 MapView 中添加一个 Overlay 需要经过以下步骤：

- 自定义类继承 Overlay，并 Override 其 draw()方法，如果需要点击、按键、触摸等交互操作，还需 Override onTap()等方法。

```
0.          public class MyOverlay extends Overlay {
1.              GeoPoint geoPoint = new GeoPoint((int) (39.915 * 1E6), (int) (116.404
    * 1E6));
2.              Paint paint = new Paint();
3.              @Override
4.              public void draw(Canvas canvas, MapView mapView, boolean shadow) {
5.                  //在天安门的位置绘制一个 String
6.                  Point point = mMapView.getProjection().toPixels(geoPoint, null);
7.                  canvas.drawText("★这里是天安门", point.x, point.y, paint);
8.              }
9.          }
```

添加到 MapView 的覆盖物中：

```
mMapView.getOverlays().add(new MyOverlay());
```

运行结果如下： 

4.2 当前位置：MyLocationOverlay

将 MyLocationOverlay 添加到覆盖物中，能够实现在地图上显示当前位置的图标以及指南针：

- 初始化 Location 模块

```
0.          // 初始化 Location 模块
1.          mLocationManager = mBMapMan.getLocationManager();
2.          // 通过 enableProvider 和 disableProvider 方法，选择定位的 Provider
```

```
3.          //
    mLocationManager.enableProvider(MKLocationManager.MK_NETWORK_PROVIDER);
4.          // mLocationManager.disableProvider(MKLocationManager.MK_GPS_PROVIDER);
5.          // 添加定位图层
6.          MyLocationOverlay mylocTest = new MyLocationOverlay(this, mMapView);
7.          mylocTest.enableMyLocation(); // 启用定位
8.          mylocTest.enableCompass();    // 启用指南针
9.          mMapView.getOverlays().add(mylocTest);
```

运行结果如下：

4.3 分条目覆盖物：ItemizedOverlay

某个类型的覆盖物，包含多个类型相同、显示方式相同、处理方式相同的项时，使用此类：

- 自定义类继承 ItemizedOverlay<OverlayItem>，并 Override 其 draw()方法，如果需要点击、按键、触摸等交互操作，还需 Override onTap()等方法。

```
0.          class OverItemT extends ItemizedOverlay<OverlayItem> {
1.              private List<OverlayItem> GeoList = new ArrayList<OverlayItem>();
2.              private Context mContext;
3.              private double mLat1 = 39.90923;//39.9022; // point1 纬度
4.              private double mLon1 = 116.397428;//116.3822; // point1 经度
5.              private double mLat2 = 39.9022;
6.              private double mLon2 = 116.3922;
7.              private double mLat3 = 39.917723;
8.              private double mLon3 = 116.3722;
9.              public OverItemT(Drawable marker, Context context) {
10.               super(boundCenterBottom(marker));
11.               this.mContext = context;
12.             // 用给定的经纬度构造 GeoPoint，单位是微度 (度 * 1E6)
13.               GeoPoint p1 = new GeoPoint((int) (mLat1 * 1E6), (int) (mLon1 * 1E6));
14.               GeoPoint p2 = new GeoPoint((int) (mLat2 * 1E6), (int) (mLon2 * 1E6));
15.               GeoPoint p3 = new GeoPoint((int) (mLat3 * 1E6), (int) (mLon3 * 1E6));
16.               GeoList.add(new OverlayItem(p1, "P1", "point1"));
17.               GeoList.add(new OverlayItem(p2, "P2", "point2"));
```

```
18.             GeoList.add(new OverlayItem(p3, "P3", "point3"));
19.              populate();  //createItem(int)方法构造item。一旦有了数据，在调用其它方法前，
    首先调用    这个方法
20.            }
21.            @Override
22.            protected OverlayItem createItem(int i) {
23.              return GeoList.get(i);
24.            }
25.            @Override
26.            public int size() {
27.             return GeoList.size();
28.            }
29.           @Override
30.           // 处理当点击事件
31.           protected boolean onTap(int i) {
32.             Toast.makeText(this.mContext, GeoList.get(i).getSnippet(),
33.                   Toast.LENGTH_SHORT).show();
34.               return true;
35.                }
36.         }
```

添加到 MapView 的覆盖物中：

```
0.          Drawable marker = getResources().getDrawable(R.drawable.iconmark);  //得
  到需要标在地图上的资源
1.          mMapView.getOverlays().add(new OverItemT(marker, this)); //添加
      ItemizedOverlay 实例到 mMapView
```

4.4 本地搜索覆盖物：PoiOverlay

详见 POI 搜索及 PoiOverlay 章节。

4.5 驾车路线覆盖物：RouteOverlay

详见驾车路线搜索及 RouteOverlay 和步行路线搜索及 RouteOverlay 章节。

4.6 换乘路线覆盖物：TransitOverlay

详见公交换乘路线搜索及 TransitOverlay 章节。

5 服务类

5.1 搜索服务

百度地图移动版 API 集成搜索服务包括：位置检索、周边检索、范围检索、公交检索、驾乘检索、步行检索、在线建议搜索，通过初始化 MKSearch 类，注册搜索结果的监听对象 MKSearchListener，实现异步搜索服务。首先自定义 MySearchListener 实现 MKSearchListener 接口，通过不同的回调方法，获得搜索结果：

```
0.      public class MySearchListener implements MKSearchListener {
1.          @Override
2.          public void onGetAddrResult(MKAddrInfo result, int iError) {
3.          }
4.          @Override
5.          public void onGetDrivingRouteResult(MKDrivingRouteResult result, int
    iError) {
6.          }
7.          @Override
8.          public void onGetPoiResult(MKPoiResult result, int type, int iError) {
9.          }
10.         @Override
11.         public void onGetTransitRouteResult(MKTransitRouteResult result, int
    iError) {
12.         }
13.         @Override
14.         public void onGetWalkingRouteResult(MKWalkingRouteResult result, int
    iError) {
15.         }
16.         @Override
17.         public void onGetBusDetailResult(MKBusLineResult result, int iError)
18.        {
19.        }
20.         @Override
21.         public void onGetSuggestionResult(MKSuggestionResult res, int iError)
22.        {
23.         }
24.     }
```

然后初始化 MKSearch 类:

```
0.      mMKSearch = new MKSearch();
1.      mMKSearch.init(mBMapMan, new MySearchListener());
```

5.2 POI 搜索及 PoiOverlay

POI 搜索有三种方式，根据范围和检索词发起范围检索 poiSearchInbounds，城市 poi 检索 poiSearchInCity，周边检索 poiSearchNearBy，以下以周边检索为例介绍如何进行检索并显示覆盖物 PoiOverlay：

- 检索天安门周边 5000 米之内的 KFC 餐厅：

```
mMKSearch.poiSearchNearBy("KFC", new GeoPoint((int) (39.915 * 1E6), (int) (116.404 * 1E6)), 5000);
```

- 实现 MySearchListener 的 onGetPoiResult，并展示检索结果：

```
@Override
public void onGetPoiResult(MKPoiResult result, int type, int iError) {
    if (result == null) {
        return;
    }
    PoiOverlay poioverlay = new PoiOverlay(MyMapActivity.this, mMapView);
    poioverlay.setData(result.getAllPoi());
    mMapView.getOverlays().add(poioverlay);
}
```

运行结果如下：

5.3 驾车路线搜索及 RouteOverlay

- 检索从天安门到百度大厦的驾车路线：

```
0.          MKPlanNode start = new MKPlanNode();
1.          start.pt = new GeoPoint((int) (39.915 * 1E6), (int) (116.404 * 1E6));
2.          MKPlanNode end = new MKPlanNode();
3.          end.pt = new GeoPoint(40057031, 116307852);
4.          // 设置驾车路线搜索策略，时间优先、费用最少或距离最短
5.          mMKSearch.setDrivingPolicy(MKSearch.ECAR_TIME_FIRST);
6.          mMKSearch.drivingSearch(null, start, null, end);
```

实现 MySearchListener 的 onGetDrivingRouteResult，并展示检索结果：

```
@Override
```

```
0.          public void onGetDrivingRouteResult(MKDrivingRouteResult result, int iEr
    ror) {
1.              if (result == null) {
2.                  return;
3.              }
4.              RouteOverlay routeOverlay = new RouteOverlay(MyMapActivity.this, mMap
    View);
5.              // 此处仅展示一个方案作为示例
6.              routeOverlay.setData(result.getPlan(0).getRoute(0));
7.              mMapView.getOverlays().add(routeOverlay);
8.          }
```

运行结果如下

5.4 步行路线搜索及 RouteOverlay

方式与驾车路线搜索类似，只需将 mMKSearch.drivingSearch(null, start, null, end)修改为 mMKSearch.walkingSearch(null, start, null, end), 实现的方法改为 onGetWalkingRouteResult 即可，不再赘述。

5.5 公交换乘路线搜索及 TransitOverlay

- 检索从天安门到百度大厦的公交换乘路线：

```
0. MKPlanNode start = new MKPlanNode();
1. start.pt = new GeoPoint((int) (39.915 * 1E6), (int) (116.404 * 1E6));
2. MKPlanNode end = new MKPlanNode();
3. end.pt = new GeoPoint(40057031, 116307852);
4. // 设置乘车路线搜索策略，时间优先、最少换乘、最少步行距离或不含地铁
5. mMKSearch.setTransitPolicy(MKSearch.EBUS_TRANSFER_FIRST);
6. mMKSearch.transitSearch("北京", start, end); // 必须设置城市名
```

实现 MySearchListener 的 onGetTransitRouteResult(MKTransitRouteResult, int)，并展示检索结果：

```
0.   @Override
1.   public void onGetTransitRouteResult(MKTransitRouteResult result, int
  iError) {
2.       if (result == null) {
3.           return;
4.       }
5.       TransitOverlay transitOverlay = new
  TransitOverlay(MyMapActivity.this, mMapView);
6.       // 此处仅展示一个方案作为示例
7.       transitOverlay.setData(result.getPlan(0));
8.       mMapView.getOverlays().add(transitOverlay);
9.   }
```

5.6 公交路线详情搜索

- 检索北京市公交路线 717 的 poi，获取公交路线的 uid：

```
0. mSearch.poiSearchInCity("北京", "717");
```

实现 MySearchListener 的 onGetPoiResult (MKPoiResult res, int type, int error)，获得公交线路 poi 的 uid，并根据此 uid 发起公交线路详情的检索：

```
10.   @Override
```

```
11.   public void onGetPoiResult(MKPoiResult res, int type, int error) {
12.       if (error != 0 || res == null) {
13.   Toast.makeText(BusLineSearch.this, "抱歉，未找到结果",
  Toast.LENGTH_LONG).show();
14.           return;
15.       }
16.       // 找到公交路线poi node
17.       MKPoiInfo curPoi = null;
18.       int totalPoiNum  = res.getNumPois();
19.       for( int idx = 0; idx < totalPoiNum; idx++ ) {
20.             curPoi = res.getPoi(idx);
21.            if ( 2 == curPoi.ePoiType ) {
22.                break;
23.            }
24.         }
25.       mSearch.busLineSearch(mCityName, curPoi.uid);
26.   }
```

实现 MySearchListener 的 onGetBusDetailResult(MKBusLineResult result, int iError)，并展示搜索结果：

```
0. @Override
1. public void onGetBusDetailResult(MKBusLineResult result, int iError)
2.     if (error != 0 || res == null) {
3. Toast.makeText(BusLineSearch.this, "抱歉，未找到结果",
Toast.LENGTH_LONG).show();
4.         return;
5.     }
6.     RouteOverlay routeOverlay = new RouteOverlay(BusLineSearch.this,
mMapView);
7.     // 此处仅展示一个方案作为示例
8.     routeOverlay.setData(result.getBusRoute());
9.     mMapView.getOverlays().clear();
10.    mMapView.getOverlays().add(routeOverlay);
11.    mMapView.invalidate();
12.    mMapView.getController().animateTo(result.getBusRoute().getStart(
));
13.    }
```

运行结果如下：

5.7 地址信息查询

根据地理坐标查询地址信息：

```
0. mMKSearch.reverseGeocode(new GeoPoint(40057031, 116307852));
```

实现 MySearchListener 的 onGetAddrResult，得到查询结果。

5.8 在线建议查询

根据关键词查询在线建议词：

```
1. mMKSearch.suggestionSearch("天安门");        //输入关键词
```

实现 MySearchListener 的 onGetSuggestionResult，得到查询结果：

```
14.   ListView mSuggestionList = (ListView) findViewById(R.id.listView1);
15.   @Override
16.   public void onGetSuggestionResult(MKSuggestionResult res, int iError)
17.   {
18.       if (iError!= 0 || res == null) {
19.       Toast.makeText(PoiSearch.this, "抱歉，未找到结果",
Toast.LENGTH_LONG).show();
20.       return;
21.       }
22.       int nSize = res.getSuggestionNum();
23.       String[] mStrSuggestions = new String[nSize];
24.       for (int i = 0; i < nSize; i++)
25.       {
26.          mStrSuggestions[i] = res.getSuggestion(i).city +
res.getSuggestion(i).key;
27.        }
28.       ArrayAdapter<String> suggestionString = new
ArrayAdapter<String>(PoiSearch.this,
android.R.layout.simple_list_item_1,mStrSuggestions);
29.       mSuggestionList.setAdapter(suggestionString);
30.
31.   }
```

6 事件

6.1 定位监听

实现方式与系统的定位监听类似，通过 MKLocationManager 注册或者移除定位监听器：

```
0. mLocationManager = mBMapMan.getLocationManager();
1. LocationListener listener = new LocationListener() {
2.     @Override
3.     public void onLocationChanged(Location location) {
4.         // TODO 在此处处理位置变化
5.     }
6. };
7. // 注册监听
8. mLocationManager.requestLocationUpdates(listener);
9. // 不需要时移除监听
```

```
10.         mLocationManager.removeUpdates(listener);
```

6.2 一般事件监听

在初始化地图 Activity 时，注册一般事件监听，并实现 MKGeneralListener 的接口处理相应事件，将 mBMapMan.init("我的 Key", null)替换为下面的代码：

```
0. mBMapMan.init("我的 key", new MKGeneralListener() {
1.     @Override
2.     public void onGetPermissionState(int iError) {
3.         // TODO 返回授权验证错误，通过错误代码判断原因，MKEvent 中常量值。
4.     }
5.     @Override
6.     public void onGetNetworkState(int iError) {
7.         // TODO 返回网络错误，通过错误代码判断原因，MKEvent 中常量值。
8.     }
9. });
```

7 离线地图

7.1 初始化

```
1. mOffline = new MKOfflineMap();
2. mOffline.init(mBMapMan, new MKOfflineMapListener() {
3.     @Override
4.     public void onGetOfflineMapState(int type, int state) {
5.         switch (type) {
6.             case MKOfflineMap.TYPE_DOWNLOAD_UPDATE:
7.                 {
8.                     MKOLUpdateElement update = mOffline.getUpdateInfo(state);
9.                 }
10.                Break;
11.           case MKOfflineMap.TYPE_NEW_OFFLINE:
12.               Log.d("OfflineDemo", String.format("add offlinemap num:%d", state));
13.               break;
14.           case MKOfflineMap.TYPE_VER_UPDATE:
```

```
15.              Log.d("OfflineDemo", String.format("new offlinemap ver"));
16.              break;
17.    }
18.};
```

7.2 导入离线包

SDK 支持导入离线包，将从官方渠道下载的离线包(只支持老版)解压，把其中的 Mapdata 文件夹拷入 SD 卡根目录下的 BaiduMapSdk 文件夹内。

```
1. int num = mOffline.scan();
2. if (num != 0)
3.     mText.setText(String.format("已安装%d 个离线包", num));
```

7.3 WIFI 下载离线包

SDK 支持在 WIFI 网络情况下，下载离线包。提供如下功能：

1. 返回热门城市列表。
2. 城市名搜索离线地图信息。
3. 启动下载。
4. 暂停下载。
5. 删除离线地图。
6. 多个 APP 共享一份离线地图数据。

详见官网 Demo 中 OfflineDemo.java 文件。