百度地图移动版 API for android
开发指南

2012.04.13

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.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

```
1.<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-per
    mission>
2.<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-per
    mission>
3.<uses-permission android:name="android.permission.INTERNET"></uses-permission>
4.<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-p
    ermission>
5.<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permis
    sion>
```

```
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
```

在 Manifest 中添加 Android 版本支持

```
<supports-screens android:largeScreens="true"  android:normalScreens="true" android:smallScreens="true"  android:resizeable="true" android:anyDensity="true"/>
<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

```
import java.util.ArrayList;
import java.util.List;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.drawable.Drawable;
import android.location.Location;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Toast;
import com.baidu.mapapi.BMapManager;
import com.baidu.mapapi.GeoPoint;
import com.baidu.mapapi.ItemizedOverlay;
import com.baidu.mapapi.LocationListener;
import com.baidu.mapapi.MKAddrInfo;
import com.baidu.mapapi.MKDrivingRouteResult;
import com.baidu.mapapi.MKGeneralListener;
import com.baidu.mapapi.MKLocationManager;
import com.baidu.mapapi.MKPlanNode;
import com.baidu.mapapi.MKPoiResult;
import com.baidu.mapapi.MKSearch;
import com.baidu.mapapi.MKSearchListener;
import com.baidu.mapapi.MKTransitRouteResult;
import com.baidu.mapapi.MKWalkingRouteResult;
import com.baidu.mapapi.MapActivity;
import com.baidu.mapapi.MapController;
```

```
29. import com.baidu.mapapi.MapView;
30. import com.baidu.mapapi.MyLocationOverlay;
31. import com.baidu.mapapi.Overlay;
32. import com.baidu.mapapi.OverlayItem;
33. import com.baidu.mapapi.PoiOverlay;
34. import com.baidu.mapapi.RouteOverlay;
35. import com.baidu.mapapi.TransitOverlay;
36.
37. public class MyMapActivity extends MapActivity {
38.         @Override
39.         public void onCreate(Bundle savedInstanceState) {
40.                 super.onCreate(savedInstanceState);
41.                 setContentView(R.layout.main);
42.         }
43.
44.         @Override
45.         protected boolean isRouteDisplayed() {
46.                 return false;
47.         }
48. }
```

在布局 xml 中添加地图控件

```
1.<?xml version="1.0" encoding="utf-8"?>
2.<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"android:or
    ientation="vertical" android:layout_width="fill_parent" android:layout_height="f
    ill_parent">
3.  <TextView android:layout_width="fill_parent" android:layout_height="wrap_content
    " android:text="@string/hello" />
4.  <com.baidu.mapapi.MapView android:id="@+id/bmapsView" android:layout_width="fill
    _parent" android:layout_height="fill_parent" android:clickable="true" />
5.</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”：

```
1.mBMapMan = new BMapManager(getApplication());
2.mBMapMan.init("我的 Key", null);
3.super.initMapActivity(mBMapMan);
```

```
4.MapView mMapView = (MapView) findViewById(R.id.bmapsView);
5.mMapView.setBuiltInZoomControls(true);  //设置启用内置的缩放控件
6.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
7.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)
8.mMapController.setCenter(point);  //设置地图中心点
9.mMapController.setZoom(12);    //设置地图 zoom 级别
```

Override 以下方法,管理 API:

```
1.@Override
2.protected void onDestroy() {
3.    if (mBMapMan != null) {
4.        mBMapMan.destroy();
5.        mBMapMan = null;
6.    }
7.    super.onDestroy();
8.}
9.@Override
10. protected void onPause() {
11.     if (mBMapMan != null) {
12.         mBMapMan.stop();
13.     }
14.     super.onPause();
15. }
16. @Override
17. protected void onResume() {
18.     if (mBMapMan != null) {
19.         mBMapMan.start();
20.     }
21.     super.onResume();
22. }
```

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。

```
1.  public class MyOverlay extends Overlay {
2.      GeoPoint geoPoint = new GeoPoint((int) (39.915 * 1E6), (int) (116.404 * 1E6));
3.      Paint paint = new Paint();
4.      @Override
5.      public void draw(Canvas canvas, MapView mapView, boolean shadow) {
6.          //在天安门的位置绘制一个 String
7.          Point point = mMapView.getProjection().toPixels(geoPoint, null);
8.          canvas.drawText("★这里是天安门", point.x, point.y, paint);
9.      }
10. }
```

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

```
1.  // 初始化 Location 模块
2.  mLocationManager = mBMapMan.getLocationManager();
3.  // 通过 enableProvider 和 disableProvider 方法，选择定位的 Provider
4.  // mLocationManager.enableProvider(MKLocationManager.MK_NETWORK_PROVIDER);
5.  // mLocationManager.disableProvider(MKLocationManager.MK_GPS_PROVIDER);
6.  // 添加定位图层
7.  MyLocationOverlay mylocTest = new MyLocationOverlay(this, mMapView);
```

```
8.  mylocTest.enableMyLocation(); // 启用定位
9.  mylocTest.enableCompass();    // 启用指南针
10. 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。

```
1.  class OverItemT extends ItemizedOverlay<OverlayItem> {
2.     private List<OverlayItem> GeoList = new ArrayList<OverlayItem>();
3.     private Context mContext;
4.     private double mLat1 = 39.90923;//39.9022; // point1 纬度
5.     private double mLon1 = 116.397428;//116.3822; // point1 经度
6.     private double mLat2 = 39.9022;
7.     private double mLon2 = 116.3922;
8.     private double mLat3 = 39.917723;
9.     private double mLon3 = 116.3722;
10.    public OverItemT(Drawable marker, Context context) {
11.      super(boundCenterBottom(marker));
12.      this.mContext = context;
13.   // 用给定的经纬度构造 GeoPoint，单位是微度 (度 * 1E6)
14.     GeoPoint p1 = new GeoPoint((int) (mLat1 * 1E6), (int) (mLon1 * 1E6));
15.     GeoPoint p2 = new GeoPoint((int) (mLat2 * 1E6), (int) (mLon2 * 1E6));
16.     GeoPoint p3 = new GeoPoint((int) (mLat3 * 1E6), (int) (mLon3 * 1E6));
17.     GeoList.add(new OverlayItem(p1, "P1", "point1"));
18.     GeoList.add(new OverlayItem(p2, "P2", "point2"));
19.     GeoList.add(new OverlayItem(p3, "P3", "point3"));
20.      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
21.   }
22.   @Override
23.   protected OverlayItem createItem(int i) {
```

```
24.      return GeoList.get(i);
25.    }
26.    @Override
27.    public int size() {
28.     return GeoList.size();
29.    }
30.   @Override
31.   // 处理当点击事件
32.   protected boolean onTap(int i) {
33.     Toast.makeText(this.mContext, GeoList.get(i).getSnippet(),
34.          Toast.LENGTH_SHORT).show();
35.      return true;
36.       }
37. }
```

添加到 MapView 的覆盖物中：

```
38.         Drawable marker = getResources().getDrawable(R.drawable.iconmark);  //得到需要标在地图上的资源
39.     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，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：

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

然后初始化 MKSearch 类:

```
1.       mMKSearch = new MKSearch();
2.  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：

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

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

```
@Override
public void onGetDrivingRouteResult(MKDrivingRouteResult result, int iError) {
    if (result == null) {
        return;
    }
    RouteOverlay routeOverlay = new RouteOverlay(MyMapActivity.this, mMapView);
    // 此处仅展示一个方案作为示例
    routeOverlay.setData(result.getPlan(0).getRoute(0));
    mMapView.getOverlays().add(routeOverlay);
}
```

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：

```
1. MKPlanNode start = new MKPlanNode();
2. start.pt = new GeoPoint((int) (39.915 * 1E6), (int) (116.404 * 1E6));
3. MKPlanNode end = new MKPlanNode();
4. end.pt = new GeoPoint(40057031, 116307852);
5. // 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
6. mMKSearch.setTransitPolicy(MKSearch.EBUS_TRANSFER_FIRST);
7. 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：

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

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：

```
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：

```
@Override
public void onGetPoiResult(MKPoiResult res, int type, int error) {
    if (error != 0 || res == null) {
Toast.makeText(BusLineSearch.this, "抱歉，未找到结果", Toast.LENGTH_LONG).show();
        return;
    }
    // 找到公交路线 poi node
    MKPoiInfo curPoi = null;
    int totalPoiNum  = res.getNumPois();
    for( int idx = 0; idx < totalPoiNum; idx++ ) {
          curPoi = res.getPoi(idx);
         if ( 2 == curPoi.ePoiType ) {
             break;
         }
      }
    mSearch.busLineSearch(mCityName, curPoi.uid);
```

```
16.    }
```

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：

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

实现 MySearchListener 的 onGetAddrResult，得到查询结果。

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：

```
1. mLocationManager = mBMapMan.getLocationManager();
2. LocationListener listener = new LocationListener() {
```

```
    @Override
    public void onLocationChanged(Location location) {
        // TODO 在此处处理位置变化
    }
};
// 注册监听
mLocationManager.requestLocationUpdates(listener);
// 不需要时移除监听
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：

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

7 离线地图

7.1 初始化

```
mOffline = new MKOfflineMap();
mOffline.init(mBMapMan, new MKOfflineMapListener() {
    @Override
    public void onGetOfflineMapState(int type, int state) {
        switch (type) {
            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。