1.三大核心

```
SDKInitializer
MapView
BaiduMap
```

2.准备工作

```
获取API Key
建立工程
添加地图引擎到Andoid工程中
添加权限
初始化地图引擎
引入布局（地图控件）
```

2.1获取APIkey

```
获取API Key
地址：http://developer.baidu.com/map/android-mobile-apply-key.htm

    1、打开命令行输入cd .android 进入到android签名目录 如：C:\Users\lenovo\.android
    2、命令 "C:\Program Files\Java\jdk1.6.0_45\bin\keytool.exe" -list -v -keystore debug.keystore 密码是android
    数字签名;包名 分号是英文的
    例如：48:8C:D9:43:56:81:C4:10:10:40:B6:C3:0A:A1:84:A9:65:65:9A:5A;baidumapsdk.demo
    生成的key：V3ITMGba32313GGCLFLALQdP
```

2.2添加地图引擎到Andoid工程中

```
添加引擎到Andoid工程中
添加jar包： baidumapapi_v3_3_0.jar和locSDK_5.0.jar
添加.so文件：拷贝libBaiduMapSDK_v3_3_0_15.so 、 liblocSDK5.so到libs\armeabi目录下
注： locSDK_5.0.jar和liblocSDK5.so为百度定位SDK所使用资源，开发者可根据实际需求自行添加。
```

2.3权限设置

```
<!-- 这个权限用于进行网络定位-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
<!-- 这个权限用于访问GPS定位-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
<!-- 用于访问wifi网络信息，wifi信息会用于进行网络定位-->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-permission>
<!-- 获取运营商信息，用于支持提供运营商信息相关的接口-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
<!-- 这个权限用于获取wifi的获取权限，wifi信息会用来进行网络定位-->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>
<!-- 用于读取手机当前的状态-->
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
<!-- 写入扩展存储，向扩展卡写入数据，用于写入离线定位数据-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
<!-- 访问网络，网络定位需要上网-->
<uses-permission android:name="android.permission.INTERNET" />
<!-- SD卡读取权限，用户写入离线定位数据-->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"></uses-permission>
<!--允许应用读取低级别的系统日志文件 -->
<uses-permission android:name="android.permission.READ_LOGS"></uses-permission>
```

2.4初始化地图引擎

```
增加权限重点是处理位置信息权限
Layout中添加MapView控件用于展示地图
创建管理工具，并初使化
SDKInitializer在initialize时校验key(permission check error)和网络状态(network error)，关于状态码信息我们可以在SDKInitializer查询
注意：
控制MapView的onResume、onPause、onDestroy ；
SDKInitializer对象创建一个就可以；
```

必须校验key，并且key值不能为空；
Initialize方法接受的参数必须是global Application，不能传递Activity。

代码：
SDKInitializer.initialize(getApplicationContext()) ;

3.注册广播接收者，监听网络状态 和 校验结果

```
private void managerInit() {
    // TODO Auto-generated method stub
    SDKInitializer.initialize(getApplicationContext()) ;
    baiduSdkReceiver = new MyBaiduSdkReceiver();
    IntentFilter filter = new  IntentFilter();
    filter.addAction(SDKInitializer.SDK_BROADCAST_ACTION_STRING_NETWORK_ERROR);
    filter.addAction(SDKInitializer.SDK_BROADTCAST_ACTION_STRING_PERMISSION_CHECK_ERROR);
    //注册
    registerReceiver(baiduSdkReceiver, filter);
}

class MyBaiduSdkReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context arg0, Intent arg1) {
        // TODO Auto-generated method stub
        String result = arg1.getAction();
        if(result.equals(SDKInitializer.SDK_BROADCAST_ACTION_STRING_NETWORK_ERROR)){
            //网络错误
            Toast.makeText(HelloWorld.this, "网络连接错误", 0).show();
        }else if(result.equals(SDKInitializer.SDK_BROADTCAST_ACTION_STRING_PERMISSION_CHECK_ERROR)){
            //校验失败
            Toast.makeText(HelloWorld.this, "百度地图服务校验失败", 0).show();
        }
    }

}
```

4 图层

图层分类
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。
实时交通信息图：baiduMap.setTrafficEnabled(true)
卫星图： baiduMap.setMapType(BaiduMap.MAP_TYPE_SATELLITE)
卫星地图是卫星拍摄的真实的地理面貌，所以卫星地图可用来检测地面的信息，你可以了解到地理位置，地形等。

覆盖物的层级压盖关系，具体如下（从下至上的顺序）：
1、基础底图（包括底图、底图道路、卫星图等）；
2、地形图图层（GroundOverlay）；
3、热力图图层（HeatMap）；
4、实时路况图图层（BaiduMap.setTrafficEnabled(true);）；
5、百度城市热力图（BaiduMap.setBaiduHeatMapEnabled(true);）；
6、底图标注（指的是底图上面自带的那些POI元素）；
7、几何图形图层（点、折线、弧线、圆、多边形）；
8、标注图层（Marker），文字绘制图层（Text）；
9、指南针图层（当地图发生旋转和视角变化时，默认出现在左上角的指南针）；
10、定位图层（BaiduMap.setMyLocationEnabled(true);）；
11、弹出窗图层（InfoWindow）；
12、自定义View（MapView.addView(View);）；

5覆盖物

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

覆盖物包括：本地覆盖物和搜索覆盖物
5.1本地覆盖物的抽象基类：OverlayOptions（核心类）
圆形覆盖物： CircleOptions
文字覆盖物： TextOptions

```
marker覆盖物： MarkerOptions
圆点覆盖物：DotOptions
ground 覆盖物：GroundOverlayOptions
圆点覆盖物：DotOptions
多边形覆盖物：PolygonOptions
折线覆盖物：PolylineOptions
弧线覆盖物：ArcOptions

代码：baiduMap.addOverlay(ooCircle);

5.2搜索覆盖物抽象类： OverlayManager （核心类）
本地搜索覆盖物：PoiOverlay
驾车路线覆盖物：DrivingRouteOverlay
步行路线覆盖物：WalkingRouteOverlay
换乘路线覆盖物：TransitOverlay
公交路线覆盖物：BusLineOverlay

添加覆盖物
overlay.setData(result);
overlay.addToMap();
overlay.zoomToSpan();
```

6.百度地图移动版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
核心类： PoiSearch和OnGetPoiSearchResultListener
        RoutePlanSearch和OnGetRoutePlanResultListener
实现思路
初始化PoiSearch类，通过setOnGetPoiSearchResultListener方法注册搜索结果的监听对象OnGetPoiSearchResultListener ，实现异步搜索服务。
通过自定义MySearchListener实现类，处理不同的回调方法，获得搜索结果。
注意， OnGetPoiSearchResultListener只支持一个，以最后一次设置为准
结合覆盖物展示搜索
本地搜索覆盖物：PoiOverlay
驾车路线覆盖物：DrivingRouteOverlay
步行路线覆盖物：WalkingRouteOverlay
换乘路线覆盖物：TransitOverlay
```

6.1 POI(Point of Interest兴趣点)搜索有三种方式

```
根据范围和检索词发起范围检索searchInBound
周边检索searchNearby
城市poi检索searchInCity
poi详细信息检索 searchPoiDetail

例子：查询加油站信息 （ PoiSearch和OnGetPoiSearchResultListener）
多种查询方法，但结果的处理都在OnGetPoiSearchResultListener的onGetPoiResult方法中
处理步骤：
判断服务器结果返回
创建poi覆盖物
将服务器返回数据添加到poi覆盖物中
添加覆盖物到地图addToMap
缩放地图，使所有Overlay都在合适的视野内
```

6.2 Route搜索

```
例子1：驾车路线查询       （RoutePlanSearch和OnGetRoutePlanResultListener）
结果展示： DrivingRouteOverlay
案例：从黑马到传智路线查询
驾车路线查询
查询：RoutePlanSearch.drivingSearch驾乘路线搜索，或者增加途经点.
PlanNode内容的设置：可以使用经纬度和地名，但不支持模糊查询，需要输入准确的名称
可以通过DrivingRoutePlanOption.policy (int policy)  来设置驾车路线规划策略
结果处理：OnGetRoutePlanResultListener. onGetDrivingRouteResult(DrivingRouteResult result)


例子2：步行路线查询
结果展示： WalkingRouteOverlay
RoutePlanSearch. walkingSearch步行路线搜索.
```

结果处理： OnGetRoutePlanResultListener. onGetWalkingRouteResult(WalkingRouteResult result)

例子3：公交路线搜索
结果展示：TransitRouteOverlay
检索： RoutePlanSearch. transitSearch
通过TransitRoutePlanOption.policy (int policy) 设置路线规划策略
结果处理： OnGetRoutePlanResultListener. onGetTransitRouteResult(TransitRouteResult result)

```
驾车路线搜索
private void search() {
    routePlanSearch = RoutePlanSearch.newInstance();
    routePlanSearch.setOnGetRoutePlanResultListener(new MyListener());

    DrivingRoutePlanOption drivingRoutePlanOption = new DrivingRoutePlanOption();
    drivingRoutePlanOption.from(PlanNode.withLocation(latLng));
    drivingRoutePlanOption.to(PlanNode.withLocation(new LatLng(22.812378,108.402389)));
    drivingRoutePlanOption.policy(DrivingPolicy.ECAR_DIS_FIRST);
    routePlanSearch.drivingSearch(drivingRoutePlanOption  );

}

class MyListener implements OnGetRoutePlanResultListener {

    @Override
    public void onGetDrivingRouteResult(DrivingRouteResult result) {
        // TODO Auto-generated method stub
        if(result == null || result.error.equals(SearchResult.ERRORNO.RESULT_NOT_FOUND)){
            Toast.makeText(getApplicationContext(), "未找到结果", 0).show();
            return;
        }

        DrivingRouteOverlay overlay = new MyPoiOverlay(baiduMap);
        overlay.setData(result.getRouteLines().get(0));
        baiduMap.setOnMarkerClickListener(overlay);
        overlay.addToMap();
        overlay.zoomToSpan();
    }

    @Override
    public void onGetTransitRouteResult(TransitRouteResult arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onGetWalkingRouteResult(WalkingRouteResult arg0) {
        // TODO Auto-generated method stub

    }

}

class MyPoiOverlay extends DrivingRouteOverlay{

    public MyPoiOverlay(BaiduMap arg0) {
        super(arg0);
    }


}
```

7地址解析服务

```
GeoCoder.geocode(GeoCodeOption option)
根据地址名获取地址信息 异步函数，返回结果在OnGetGeoCoderResultListener里的onGetGeoCodeResult方法通知
GeoCoder.reverseGeoCode(ReverseGeoCodeOption option)
根据地理坐标点获取地址信息 异步函数，返回结果在OnGetGeoCoderResultListener里的onGetReverseGeoCodeResult方法通知
```

8联想词检索

```
SuggestionSearch . requestSuggestion(SuggestionSearchOption option)
查询一系列与指定key相关的内容，结果中包括城市及包含key的名称
结果处理OnGetSuggestionResultListener . onGetSuggestionResult(SuggestionResult result)
```

公交路线详细信息搜索

```
检索： BusLineSearch. searchBusLine(BusLineSearchOption option)
busLineUid信息获取：公交线路的uid，可以通过poi查询返回的结果中获取MKPoiInfo的uid。
使用poiSearchInCity查询公交线信息，利用PoiInfo的type可以判断poi类型，当类型为公交线路时，记录当前的PoiInfo中的uid信息。
利用获取的uid信息进行公交线的查询
结果处理：在OnGetBusLineSearchResultListener. onGetBusLineResult(BusLineResult result)中进行结果的处理，此时使用到的覆盖物是BusLineOv
```

9 定位

```
LocationClient和BDLocationListener
首先需要打开定位图层BaiduMap.setMyLocationEnabled(true);
设置监听器LocationClient. registerLocationListener(BDLocationListener)
设置定位模式baiduMap. setLocationMode(LocationMode)
Hight_Accuracy，高精度定位模式：这种定位模式下，会同时使用网络定位和GPS定位，优先返回最高精度的定位结果；
Battery_Saving，低功耗定位模式：这种定位模式下，不会使用GPS，只会使用网络定位（Wi-Fi和基站定位）
Device_Sensors，仅用设备定位模式：这种定位模式下，不需要连接网络，只使用GPS进行定位，这种模式下不支持室内环境的定位
设置定位显示模式BaiduMap.setMyLocationConfigeration(MyLocationConfiguration)
定位数据获取：在BDLocationListener. onReceiveLocation(BDLocation result)方法中设置定位数据，
baiduMap.setMyLocationData(MyLocationData);
```

> 在百度地图移动版API中，提供一个重要的特色功能：定位，通过这个功能，能获取到用户当前所在位置。 在程序中，如果使用此功能，必须注册GPS和网络的使用权限。 在获取用户位置时，优先使用GPS进行定位；如果GPS定位没有打开或者没有可用位置信息，则会通过判断网络是否连接（即确认手机是否能上网，不论是连接2G/3G或Wi-Fi网络），如果是，则通过请求百度网络定位服务，返回网络定位结果。为了使获得的网络定位结果更加精确，请打开手机的Wi-Fi开关。 目前系统自带的网络定位服务精度低，且服务不稳定、精度低，并且从未来的趋势看，基站定位是不可控的（移动公司随时可能更改基站编号以垄断定位服务），而Wi-Fi定位则不然，它是一种精度更高、不受管制的定位方法。国内其它使用Wi-Fi定位的地图软件，Wi-Fi定位基本不可用，百度的定位服务量化指标优秀，网络接口返回速度快（服务端每次定位响应时间50毫秒以内），平均精度70米，其中Wi-Fi精度40米左右，基站定位精度200米左右，覆盖率98%，在国内处于一枝独秀的地位。
>
> 注意：
> 关于经纬度的说明：该经纬度信息是经过加密处理，所以在其它地图工具中测得的经纬度 信息不适合百度的坐标系统。 使用百度经纬度坐标，可以通过http://api.map.baidu.com/lbsapi/getpoint/index.html查询地理坐标如果需要在百度地图上显示使用其他坐标系统的位置，请发邮件至mapapi@baidu.com申请坐标转换接口

```
http://developer.baidu.com/map/index.php?title=android-locsdk/guide/v5-0

mLocationClient = new LocationClient(getApplicationContext());
myListener = new MylocationListener();
mLocationClient.registerLocationListener(myListener);

LocationClientOption option = new LocationClientOption();
option.setLocationMode(LocationMode.Hight_Accuracy);// 设置定位模式
option.setCoorType("bd09ll");// 返回的定位结果是百度经纬度,默认值gcj02
option.setScanSpan(5000);// 设置发起定位请求的间隔时间为5000ms
option.setIsNeedAddress(true);// 返回的定位结果包含地址信息
option.setNeedDeviceDirect(true);// 返回的定位结果包含手机机头的方向
mLocationClient.setLocOption(option);
baiduMap.setMyLocationEnabled(true);
baiduMap.setMyLocationConfigeration(new MyLocationConfiguration(
MyLocationConfiguration.LocationMode.COMPASS, true,
BitmapDescriptorFactory.fromResource(R.drawable.icon_geo)));

public void onReceiveLocation(BDLocation result) {
if (result != null) {
double latitude2 = result.getLatitude();
double longitude2 = result.getLongitude();
```

```
MyLocationData data = new MyLocationData.Builder()
.latitude(latitude2).longitude(longitude2).build();
baiduMap.setMyLocationData(data);

}
}

}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
switch (keyCode) {
case KeyEvent.KEYCODE_1:
baiduMap.setMyLocationConfigeration(new MyLocationConfiguration(
MyLocationConfiguration.LocationMode.NORMAL, true,
BitmapDescriptorFactory.fromResource(R.drawable.icon_geo)));
break;
case KeyEvent.KEYCODE_2:
baiduMap.setMyLocationConfigeration(new MyLocationConfiguration(
MyLocationConfiguration.LocationMode.COMPASS, true,
BitmapDescriptorFactory.fromResource(R.drawable.icon_geo)));
break;
case KeyEvent.KEYCODE_3:
baiduMap.setMyLocationConfigeration(new MyLocationConfiguration(
MyLocationConfiguration.LocationMode.FOLLOWING, true,
BitmapDescriptorFactory.fromResource(R.drawable.icon_geo)));
break;

default:
break;
}
return super.onKeyDown(keyCode, event);
}
```

我的Demo:

```
    private void locate() {
           //声明LocationClient类
        mLocationClient = new LocationClient(getApplicationContext());
        System.out.println("初始化mLocationClient："+mLocationClient.toString());
        BDLocationListener myListener = new MyLocationListener();
        mLocationClient.registerLocationListener( myListener );    //注册监听函数

          LocationClientOption option = new LocationClientOption();
            option.setLocationMode(LocationMode.Hight_Accuracy);//可选，默认高精度，设置定位模式，高精度，低功耗，仅设备
            option.setCoorType("bd09ll");//可选，默认gcj02，设置返回的定位结果坐标系
            int span=1000;
            option.setScanSpan(span);//可选，默认0，即仅定位一次，设置发起定位请求的间隔需要大于等于1000ms才是有效的
            option.setIsNeedAddress(true);//可选，设置是否需要地址信息，默认不需要
            option.setOpenGps(true);//可选，默认false,设置是否使用gps
            option.setLocationNotify(true);//可选，默认false，设置是否当gps有效时按照1S1次频率输出GPS结果
            option.setIsNeedAddress(true);//可选，默认false，设置是否需要位置语义化结果，可以在BDLocation.getLocationDescribe里得到，结
            option.setNeedDeviceDirect(true);//可选，默认false，设置是否需要POI结果，可以在BDLocation.getPoiList里得到
            option.setIgnoreKillProcess(false);//可选，默认true，定位SDK内部是一个SERVICE，并放到了独立进程，设置是否在stop的时候杀死这个
            option.SetIgnoreCacheException(false);//可选，默认false，设置是否收集CRASH信息，默认收集
            mLocationClient.setLocOption(option);

            baiduMap.setMyLocationEnabled(true);
            baiduMap.setMyLocationConfigeration(new MyLocationConfiguration(
                    MyLocationConfiguration.LocationMode.FOLLOWING, true,
                    BitmapDescriptorFactory.fromResource(R.drawable.icon_geo)));


    }
    @Override
    protected void onStart() {
        // TODO Auto-generated method stub
        super.onStart();
        System.out.println("mLocationClient.start()");
        mLocationClient.start();
```

```
    }
    @Override
    protected void onPause() {
        // TODO Auto-generated method stub
        mLocationClient.stop();
        super.onPause();
    }

    public class MyLocationListener implements BDLocationListener {

        @Override
        public void onReceiveLocation(BDLocation result) {
            //Receive Location

            if (result != null) {
                double latitude2 = result.getLatitude();
                double longitude2 = result.getLongitude();

                MyLocationData data = new MyLocationData.Builder()
                .latitude(latitude2).longitude(longitude2).build();
                baiduMap.setMyLocationData(data);
            }


        }
    }
}
```