Android SDK 开发指南

目录

[简介 5](#_Toc430268843)

[关于HDmapAndroid SDK 5](#_Toc430268844)

[面向的用户 5](#_Toc430268845)

[基本功能 5](#_Toc430268846)

[兼容性 5](#_Toc430268847)

[配置环境及发布 6](#_Toc430268848)

[Eclipse工程配置方法 6](#_Toc430268849)

[基础地图 7](#_Toc430268850)

[简介 7](#_Toc430268851)

[加载地图 7](#_Toc430268852)

[地图类型切换 8](#_Toc430268853)

[标注覆盖物 8](#_Toc430268854)

[几何图形覆盖物 9](#_Toc430268855)

[折线 9](#_Toc430268856)

[多边形 10](#_Toc430268857)

[检索功能 11](#_Toc430268858)

[城市检索 11](#_Toc430268859)

[周边检索 12](#_Toc430268860)

[区域检索 13](#_Toc430268861)

[检索错误 14](#_Toc430268862)

[地理编码 14](#_Toc430268863)

[正向地理编码 14](#_Toc430268864)

[逆地理编码 15](#_Toc430268865)

[行政区划检索 16](#_Toc430268866)

[行政区划地理范围 16](#_Toc430268867)

[行政区划编码 17](#_Toc430268868)

[经纬度点所在行政区划搜索 17](#_Toc430268869)

[工具 18](#_Toc430268870)

[简介 18](#_Toc430268871)

[罗盘 18](#_Toc430268872)

[比例尺 19](#_Toc430268873)

[定位功能 20](#_Toc430268874)

[使用定位 20](#_Toc430268875)

[事件监听 20](#_Toc430268876)

[地图类 20](#_Toc430268877)

[地图单击事件 20](#_Toc430268878)

[地图双击事件 20](#_Toc430268879)

[地图长按事件 20](#_Toc430268880)

[地图状态改变事件 20](#_Toc430268881)

[Marker类 20](#_Toc430268882)

[Marker显示 20](#_Toc430268883)

[Marker隐藏 20](#_Toc430268884)

[Marker点击 21](#_Toc430268885)

[搜索类 21](#_Toc430268886)

[POI搜索 21](#_Toc430268887)

[行政区划搜索 21](#_Toc430268888)

[地理编码 22](#_Toc430268889)

# 简介

## 关于HDmapAndroid SDK

Android SDK是一套基于Android4.0及以上版本设备的应用程序接口，您可以通过该接口实现丰富的LBS功能：

* 地图：提供地图的展示和缩放、平移、旋转、改变视角等地图操作；
* POI检索：可根据关键字，对POI数据进行周边、区域和城市内三种检索；
* 地理编码：提供地理坐标和地址之间相互转换的能力；
* 覆盖物：提供多种地图覆盖物（自定义标注、几何图形等），满足开发者的各种需求；
* 定位：采用多种定位模式，使用定位SDK获取位置信息，使用SDK的位置图层进行位置展示；

## 面向的用户

SDK是提供给具有一定Android编程经验和了解面向对象概念的读者使用。此外，读者还应该对地图的基本知识有一定的了解。

## 基本功能

* 基础地图：包括基本地图、卫星图、各种地图覆盖物。此外还包括各种与地图相关的操作和事件监听；
* 检索功能：包括POI检索（周边、区域、城市内），Place详情检索，地理编码/反地理编码等；

注：本套开发指南是针对默认的全功能包编写，开发者在使用定制化的SDK进行开发时，请参考开发指南中相对应的章节获取更多信息。

## 兼容性

支持Android4.0及以上系统。

# 配置环境及发布

我们推荐各位开发者使用Eclipse作为自己的开发工具。下面我们将针对Eclipse介绍一下地图SDK的工程配置方法。

### Eclipse工程配置方法

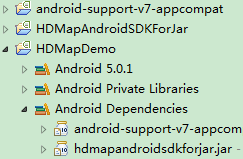
第一步：在新建开发工程里选择导入已存在的安卓代码,找到开发包里的HDMapAndroidSDKForJar工程并导入到新建开发工程下。

第二步：在开发工程属性->Java Build Path中选择Configure Build Path，进入以下页面：



在页面中选择“Android”，然后添加HDMapAndroidSDKForJar到“Library”，确定后返回。

通过以上两步操作后，工程目录如下图所示：



您就可以在开发工程中正常使用HDmapAndroid SDK为您提供的全部功能了。

# 基础地图

## 简介

开发者可利用SDK提供的接口，使用已部署的本地服务为您提供的基础地图数据。目前SDK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、多边形等）、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。

## 加载地图

进行初始化设置，实现地图的加载显示。核心代码如下：

mMapView = (MapView) mView.findViewById(R.id.*mapView*);

ITileLayer maptile\_service\_source;

//创建地图数据源

maptile\_service\_source = **new** WebSourceTileLayer("service", "http://190.0.0.29/v1/rastertiles/land\_map/{z}/{x}/{y}.png").setName("Service").setAttribution("Service contributors.").setMinimumZoomLevel(1).setMaximumZoomLevel(18);

//添加地图源

mMapView.addTileSource(maptile\_service\_source);

//设置地图滚动范围

BoundingBox box = maptile\_service\_source.getBoundingBox();

mMapView.setScrollableAreaLimit(box);

//设置地图缩放范围

mMapView.setMinZoomLevel(mMapView.getTileProvider().getMinimumZoomLevel());

mMapView.setMaxZoomLevel(mMapView.getTileProvider().getMaximumZoomLevel());

//设置当前地图中心

mMapView.setCenter(**new** LatLng(38.919184, 121.616318));

//设置当前缩放级别

mMapView.setZoom(15);

//设置瓦片大小

mMapView.setTileSize(256);

## 地图类型切换

HDmapAndroid SDK为您提供了三种类型的地图资源（海图，陆图，融合图），开发者可以利用MapView中的addTileSource ()和removeTileSource()方法来改变地图源。核心代码如下：

ITileLayer maptile\_service\_source;

//创建地图数据源

maptile\_service\_source = **new** WebSourceTileLayer("service", http://190.0.0.29/v1/rastertiles/sea\_map/{z}/{x}/{y}.png").setName("Service").setAttribution("Service contributors.").setMinimumZoomLevel(1).setMaximumZoomLevel(18);

//添加地图源

mMapView.addTileSource(maptile\_service\_source);

//移除上一个地图源

mMapView.removeTileSource(0);

## 标注覆盖物

开发者可根据自己实际的业务需求，利用标注覆盖物，在地图指定的位置上添加标注信息。具体实现方法如下：

String title = "Example marker";

String description = "This is an example of marker";

//创建Marker

Marker mark = **new** Marker(mMapView, title, description, new LatLng(38.919184, 121.616318));

//添加Marker

mMapView.addMarker(mark);

## 几何图形覆盖物

HDmapAndroid SDK提供多种结合图形覆盖物，利用这些图形，可帮助您构建更加丰富多彩的地图应用。目前提供的几何图形有：折线（Polyline）、多边形（Polygon）。

### 折线

具体实现方法如下：

//初始化五个点

LatLng pt1 = new LatLng(39.93923, 116.357428);

LatLng pt2 = new LatLng(39.91923, 116.327428);

LatLng pt3 = new LatLng(39.89923, 116.347428);

LatLng pt4 = new LatLng(39.89923, 116.367428);

LatLng pt5 = new LatLng(39.91923, 116.387428);

//创建Path

PathOverlay Path = new PathOverlay();

//添加点到Path

Path.addPoint(pt1);

Path.addPoint(pt2);

Path.addPoint(pt3);

Path.addPoint(pt4);

Path.addPoint(pt5);

//创建画笔

Paint linePaint = new Paint();

//设置为划线样式

linePaint.setStyle(Paint.Style.*STROKE*);

//设置颜色

linePaint.setColor(Color.*BLUE*);

//设置线宽

linePaint.setStrokeWidth(5);

//把画笔设置给Path

Path.setPaint(linePaint);

//添加Path

mMapView.addOverlay(Path);

### 多边形

具体实现方法如下：

//初始化五个点

LatLng pt1 = new LatLng(39.93923, 116.357428);

LatLng pt2 = new LatLng(39.91923, 116.327428);

LatLng pt3 = new LatLng(39.89923, 116.347428);

LatLng pt4 = new LatLng(39.89923, 116.367428);

LatLng pt5 = new LatLng(39.91923, 116.387428);

//创建Path

PathOverlay Path = new PathOverlay();

//添加点到Path

Path.addPoint(pt1);

Path.addPoint(pt2);

Path.addPoint(pt3);

Path.addPoint(pt4);

Path.addPoint(pt5);

//创建画笔

Paint linePaint = new Paint();

//设置为填充样式

linePaint.setStyle(Paint.Style.FILL);

//设置颜色

linePaint.setColor(Color.*BLUE*);

//设置线宽

linePaint.setStrokeWidth(5);

//把画笔设置给Path

Path.setPaint(linePaint);

//添加Path

mMapView.addOverlay(Path);

## 检索功能

目前HDmapAndroid SDK所集成的POI检索服务包括：城市检索、周边检索、区域检索。

### 城市检索

具体实现方法如下：

//创建检索参数

PoiCitySearchOption option = new PoiCitySearchOption();

//设置检索关键字

option.setKeyword(“故宫”);

//设置检索城市

option.setCity(“北京”);

//设置每页个数

option.setPageSize(10);

//设置检索第几页

option.setPageNum(1);

//设置检索范围

option.setSearchScope(0);

//创建检索引擎

PoiSearch search = PoiSearch.*newInstance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnPoiSearchListener poiListener = new OnPoiSearchListener(){

//重载的监听方法

public void onGetPoiResult(PoiResult result){

//获取POI检索结果

}

};

//把监听对象传递给检索引擎

search.setListerner(poiListener);

//传入检索参数，开始检索

search.searchInCity(option);

### 周边检索

具体实现方法如下：

//创建检索参数

PoiBoundSearchOption option = new PoiBoundSearchOption();

//设置检索关键字

option.setKeyword(“港口”);

//设置检索区域矩形坐标

option.setBound(new BoundingBox(41.001178,124.264514,37.702655,114.081617));

//设置每页个数

option.setPageSize(10);

//设置检索第几页

option.setPageNum(0);

//设置检索范围

option.setSearchScope(1);

//创建检索引擎

PoiSearch search = PoiSearch.*newInstance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnPoiSearchListener poiListener = new OnPoiSearchListener(){

//重载的监听方法

public void onGetPoiResult(PoiResult result){

//获取POI检索结果

}

};

//把监听对象传递给检索引擎

search.setListerner(poiListener);

//传入检索参数，开始检索

search.searchInBound(option);

### 区域检索

具体实现方法如下：

//创建检索参数

PoiNearbySearchOption option = new PoiNearbySearchOption();

//设置检索关键字

option.setKeyword(“电影院”);

//设置中心坐标

option.setLocation(new LatLng(38.919184,121.616318));

//设置区域半径

option.setRaduis(1000);

//设置每页个数

option.setPageSize(10);

//设置检索第几页

option.setPageNum(0);

//设置检索范围

option.setSearchScope(0);

//创建检索引擎

PoiSearch search = PoiSearch.*newInstance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnPoiSearchListener poiListener = new OnPoiSearchListener(){

//重载的监听方法

public void onGetPoiResult(PoiResult result){

//获取POI检索结果

}

};

//把监听对象传递给检索引擎

search.setListerner(poiListener);

//传入检索参数，开始检索

search.searchNearby(option);

### 检索错误

检索结果中包含status，是本次访问状态，如果成功返回0，如果失败返回为其他数字。

检索结果中包含message，是对访问状态值的英文说明，如果成功返回"ok"，并返回结果字段，如果失败返回错误说明。

## 地理编码

地理编码指的是将地址信息建立空间坐标关系的过程。可分为正向地图编码和反向地图编码。

### 正向地理编码

正向地理编码指的是由地址信息转换为坐标点的过程，核心代码如下：

//创建检索参数

GeoCodeOption option = **new** GeoCodeOption();

//设置地址信息

option.set\_address(text);

//设置检索城市

option.set\_city(text);

//创建检索引擎

GeoCoder search = GeoCoder.*newInstance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnGetGeoCoderResultListener geoCoderListener = new OnGetGeoCoderResultListener(){

//重载的监听方法

public void onGetGeoCodeResult(GeoCodeResult result) {

//获取地理编码

}

};

//把监听对象传递给检索引擎

search.setOnGetGeoCodeResultListener(geoCoderListener);

//传入检索参数，开始检索

search.geocode(option);

### 逆地理编码

逆地理编码服务实现了将地球表面的地址坐标转换为标准地址的过程，核心代码如下：

//创建检索参数

ReverseGeoCodeOption option = new ReverseGeoCodeOption();

//设置地址坐标

option.set\_location(new LatLng(38.919184,121.616318));

//创建检索引擎

ReverseGeoCoder search = ReverseGeoCoder.newInstance("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnGetReverseGeoCoderResultListener reversGeoCoderListener = new OnGetReverseGeoCoderResultListener(){

//重载的监听方法

public void onGetReverseGeoCodeResult(ReverseGeoCodeResult result) {

//获取逆地理编码

}

};

//把监听对象传递给检索引擎

search.setOnGetReverseGeoCoderResultListener(reversGeoCoderListener);

//传入检索参数，开始检索

search.reverseGeoCode(option);

## 行政区划检索

行政区划检索帮助用户查询得到某个行政区划的编码，名称以及地理范围。

### 行政区划地理范围

具体实现方法如下：

//创建检索参数

DistrictCodeOption option = new DistrictCodeOption();

//设置区划编码

option.set\_code(“210200”);

//创建检索引擎

DistrictCodeSearch search = DistrictCodeSearch.newInst*ance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnGetDistrictGeometryResultListener districtListener = new OnGetDistrictGeometryResultListener(){

//重载的监听方法

public void OnGetDistrictGeometryResult(DistrictGeometryResult result) {

//获取行政区划地理范围

}

};

//把监听对象传递给检索引擎

search.setOnGetDistrictGeometryResultListener(districtListener);

//传入检索参数，开始检索

search.district(option);

### 行政区划编码

具体实现方法如下：

//创建检索参数

DistrictNameOption option = **new** DistrictNameOption();

//设置区划名称

option.set\_districtname(text);

//创建检索引擎

DistrictNameSearch search = DistrictNameSearch.*newInstance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnGetMultiDistrictInfoResultListener districtListener = new OnGetMultiDistrictInfoResultListener(){

//重载的监听方法

public void OnGetMultiDistrictInfoResult(MultiDistrictInfoResult result) {

//获取行政区划编码

}

};

//把监听对象传递给检索引擎

search.setOnGetMultiDistrictInfoResultListener(districtListener);

//传入检索参数，开始检索

search.district(option);

### 经纬度点所在行政区划搜索

具体实现方法如下：

//创建检索参数

DistrictLocationOption option = **new** DistrictLocationOption();

//设置地理坐标

option.set\_location(**new** LatLng(lat, lon));

//创建检索引擎

DistrictLocationSearch search = DistrictLocationSearch.*newInstance*("http://190.0.0.29", 8080);

//创建检索结果监听对象

OnGetDistrictInfoResultListener districtListener = new OnGetDistrictInfoResultListener(){

//重载的监听方法

public void OnGetDistrictInfoResult(DistrictInfoResult result) {

//获取行政区划

}

};

//把监听对象传递给检索引擎

search.setOnGetDistrictInfoResultListener(districtListener);

//传入检索参数，开始检索

search.district(option);

# 工具

## 简介

地图SDK目前提供的工具有：罗盘，比例尺功能，可以开启和关闭。

## 罗盘

罗盘用于指示地图旋转方向，具体实现方法如下：

//显示开启

mMapView.showUserCompass(true);

//显示关闭

mMapView.showUserCompass(false);

## 比例尺

比例尺用于指示地图当前显示比例，具体实现方法如下：

//字体大小

int fontsize = 18;

//划线样式

Style linestyle = Style.STROKE;

//颜色

int color = Color.rgb(0, 0, 0);

//线宽

int linewidth = 1;

//起始点屏幕坐标X

int screenX = 50;

//起始点屏幕坐标y

int screenY = 50;

//显示开启

mMapView.showScaleBar(fontsize, linestyle, color, linewidth, screenX, screenY, true);

//显示关闭

mMapView.showScaleBar(fontsize, linestyle, color, linewidth, screenX, screenY, false);

# 定位功能

## 使用定位

# 事件监听

HDmapAndroid SDK提供了丰富的回调接口，供用户对于不同的事件进行处理。

## 地图类

继承或者实例化接口MapViewListener，重载实现以下方法：

### 地图单击事件

public void onTapMap(final MapView pMapView, final ILatLng pPosition);

### 地图双击事件

public void onDoubleTapMap(final MapView pMapView, final ILatLng pPosition);

### 地图长按事件

public void onLongPressMap(final MapView pMapView, final ILatLng pPosition);

### 地图状态改变事件

public void onMapChangeStart();

public void onMapChangeEnd()；

## Marker类

继承或者实例化接口MapViewListener，重载实现以下方法：

### Marker显示

**public** **void** onShowMarker(**final** MapView pMapView, **final** Marker pMarker);

### Marker隐藏

**public** **void** onHideMarker(**final** MapView pMapView, **final** Marker pMarker);

### Marker点击

**public** **void** onTapMarker(**final** MapView pMapView, **final** Marker pMarker);

## 搜索类

### POI搜索

继承或者实例化接口OnPoiSearchListener，重载实现以下方法：

#### 按城市搜索

**public** **void** onGetPoiResult(PoiResult result);

#### 按区域搜索

**public** **void** onGetPoiResult(PoiResult result);

#### 按范围搜索

**public** **void** onGetPoiResult(PoiResult result);

### 行政区划搜索

#### 行政区划编码搜索

继承或者实例化接口OnGetMultiDistrictInfoResultListener，重载实现以下方法：

**public** **void** OnGetMultiDistrictInfoResult(MultiDistrictInfoResult result);

#### 行政区划地理范围搜索

继承或者实例化接口OnGetDistrictGeometryResultListener，重载实现以下方法：

**public** **void** OnGetDistrictGeometryResult(DistrictGeometryResult result);

#### 经纬度点所在行政区划搜索

继承或者实例化接口OnGetDistrictInfoResultListener，重载实现以下方法：

**public** **void** OnGetDistrictInfoResult(DistrictInfoResult result);

### 地理编码

#### 正向地理编码

继承或者实例化接口OnGetGeoCoderResultListener，重载实现以下方法：

**public** **void** onGetGeoCodeResult(GeoCodeResult result);

#### 逆地理编码

继承或者实例化接口OnGetReverseGeoCoderResultListener，重载实现以下方法：

**public** **void** onGetReverseGeoCodeResult(ReverseGeoCodeResult result);

以地图单击事件为例，响应事件时，在地图被单击的地方添加一个标注，具体实现方法如下：

//创建事件监听对象

MapViewListener listener = new MapViewListener() {

//重载监听方法

Override

**public** **void** onTapMap(MapView pMapView, ILatLng pPosition) {

String title = "Example marker";

String description = String.*format*("%s", "地图被点击");

//创建Marker

Marker mark = new Marker(pMapView, title, description, (LatLng)pPosition);

//添加Marker

pMapView.addMarker(mark);

}

};