维智地图文档

地图导入

添加模块依赖

添加模块，把 wzMap 导入模块， 在主程序settings.gradle中加入模块编译

```
include ':wzMap'
```

配置 marven 库

在 build.gradle 中配置 marven 库, 需添加 marvenCentral()

```
buildscript {
    repositories {
        google()
        mavenCentral() // 添加 marvenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.2.2'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        mavenCentral()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

导入依赖

```
implementation project(path: ':wzMap')
```

在主应用AndroidManifest.xml中添加权限

```
<!--用于进行网络定位-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION">
</uses-permission>
<!--用于访问GPS定位-->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-
permission>
```

```
<!--用于获取运营商信息，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>
<!--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>
<!--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>
<!--用于访问网络，网络定位需要上网-->
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<!--用于读取手机当前的状态-->
<uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>
<!--用于写入数据到扩展存储卡-->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE">
</uses-permission>
<!--用于发现和通讯蓝牙-->
<uses-permission android:name="android.permission.BLUETOOTH"></uses-permission>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"></uses-permission>
```

布局文件中加入WZMap

```
<com.wz.location.map.view.WZMap
    android:id="@+id/wzMapView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

初始化 MAP

```
Mapbox.getInstance(context, Constants.MAPBOX_KEY); //  找维智提供该 KEY
```

图层样式

修改地图图层

```
if(checkedId == R.id.rb_default){
    wzMap.setMapType(MapType.DEFAULT); // 切换地图样式
}else if(checkedId == R.id.rb_gray_white){
    wzMap.setMapType(MapType.GRAY_WHITE);
}else if(checkedId == R.id.rb_light_blue){
    wzMap.setMapType(MapType.LIGHT_BLUE);
}else if(checkedId == R.id.rb_dark_blue){
    wzMap.setMapType(MapType.DARK_BLUE);
}
```

图层样式切换，需在 assets 目录的 dark_blue_style.json, default_style.json, gray_white_style.json, light_blue_style.json 等文件中，替换您申请的 key。

```
"tiles": [
    "https://api.newayz.com/maps/tilestream/v1/layers/landcover-china-
20210415/tiles/{z}/{x}/{y}?access_key=您的accesskey"
]
```

地图长按事件

```
wzMap.addOnMapLongClickListener(new Map.MapLongClickListener() { // 长按地图事件
    @Override
    public void OnMapLongClickListener(@NonNull LatLng latLng) {
        return;
    }
});
```

Mark 新增

Mark 初始化

```
ng latLng = new LatLng(30.504129,114.413968);
wzMap.addMarker(new MarkerOptions().title("武汉").snippet("武汉市 经纬度
114.413968，30.504129").position(latLng).icon(BitmapFactory.decodeResource(
       MainActivity.this.getResources(), R.mipmap.ic_user)));
```

Marker 点击事件

```
wzMap.setOnMarkerClickListener(new Map.OnMarkerClickListener() {
    @Override
    public boolean onMarkerClick(Marker marker) {

        return false;
    }
});
```

Marker 拖拽事件

```
wzMap.setOnMarkerDragListener(new Map.OnMarkerDragListener() {
    @Override
    public void onMarkerDragStart(Marker marker) { // 开始拖动

    }

    @Override
    public void onMarkerDrag(Marker marker) { // 拖动中

    }

    @Override
    public void onMarkerDragEnd(Marker marker) { //  拖动结束

    }
});
```

绘制 InfoWindow

显示 InfoWindow

InfoWindow 是点标记的一部分，默认的 Infowindow 只显示 Marker 对象的两个属性，一个是 title 和另一个 snippet，如果希望对InfoWindow 的样式或者内容有自定义需求，可以参考如下内容。

```
marker.showInfoWindow();
```

隐藏 InfoWindow

```
marker.hideInfoWindow();
```

InfoWindow 动态配置布局

开发中

绘制多边形

多边形新增

```
// 声明 多边形参数对象
PolygonOptions polygonOptions = new PolygonOptions();

LatLng latLng1 = new LatLng(30.504984,  114.412932);
LatLng latLng2 = new LatLng(30.50163,   114.411925);
LatLng latLng3 = new LatLng(30.501726,  114.417833);
LatLng latLng4 = new LatLng(30.505428,  114.417184);
LatLng latLng5 = new LatLng(30.504984,  114.412932);
// 添加 多边形的每个顶点（顺序添加）
polygonOptions.add(latLng1,latLng2 , latLng3, latLng4, latLng5);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    polygonOptions.strokeWidth(4) // 多边形的边框
            .strokeColor(getColor(R.color.purple_500)) // 边框颜色
            .fillColor(getColor(R.color.teal_200));   // 多边形的填充色
}
polygonOptions.setDottedLine(true);
polygon = wzMap.addPolygon(polygonOptions);
```

多边形删除

两种删除方式

```
polygon.remove(); // 多边形删除
wzMap.removeLayer(polygon.getId());  // 多边形删除
```

多边形更新

```
polygon.setVisible(true); // 设置多边形是否可见
polygon.setColor(getColor(R.color.teal_700)); // 设置多边形填充颜色
polygon.setWidth(4); // 设置多边形宽度
```

绘制线

线新增

```
List<LatLng> latLngs = new ArrayList<>();

latLngs.add(new LatLng(30.504101, 114.414058));
latLngs.add(new LatLng(30.504201, 114.414158));
latLngs.add(new LatLng(30.504301, 114.414258));
latLngs.add(new LatLng(30.504401, 114.414358));
latLngs.add(new LatLng(30.504501, 114.414458));
latLngs.add(new LatLng(30.504601, 114.414558));
latLngs.add(new LatLng(30.504701, 114.414658));

if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    polyline = wzMap.addPolyline(new PolylineOptions().addAll(latLngs).width(4)
            .setDottedLine(true).color(getColor(R.color.teal_200)));
}
```

线删除

两种方式

```
polyline.remove(); // 线删除
wzMap.removeLayer(polygon.getId());  // 线删除
```

线更新

```
polyline.setVisible(true); // 设置线是否可见
polyline.setColor(getColor(R.color.teal_700)); // 设置线填充颜色
polyline.setWidth(4); // 设置线宽度
```

绘制圆

圆新增

圆形由 Circle 类定义实现，构造一个圆形需要确定它的圆心和半径，具体的示例代码如下：

```
LatLng latLng = new LatLng(30.504101, 114.414058);

circle = wzMap.addCircle(new CircleOptions()
                        .center(latLng)
                        .radius(100f)
                        .strokeWidth(5)
                        .setDottedLine(true)
                        .fillColor(getColor(R.color.teal_700))
                        .strokeColor(getColor(R.color.teal_200)));
```

圆删除

```
wzMap.removeLayer(circle.getId()); // 第一种方式
circle.remove(); // 第二种方式
```

圆更新

```
circle.setVisible(true); // 设置圆是否可见
circle.setColor(getColor(R.color.teal_700)); // 设置圆边框颜色
circle.setWidth(4); // 设置圆边框宽度
```

Demo 样例

打开想添加地图的Activity文件，并将以下代码添加到该文件中

```
package com.wz.wifi.maptest;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.Manifest;
import android.content.pm.PackageManager;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.RadioGroup;
import android.widget.Toast;

import com.wayz.location.WzException;
import com.wayz.location.WzLocation;
import com.wayz.location.WzLocationClient;
import com.wayz.location.WzLocationClientOption;
import com.wayz.location.WzLocationListener;
import com.wz.location.map.Map;
import com.wz.location.map.common.MapType;
import com.wz.location.map.model.BaseOverlay;
import com.wz.location.map.model.Circle;
import com.wz.location.map.model.CircleOptions;
import com.wz.location.map.model.LatLng;
import com.wz.location.map.model.Marker;
import com.wz.location.map.model.MarkerOptions;
import com.wz.location.map.model.Polygon;
import com.wz.location.map.model.PolygonOptions;
import com.wz.location.map.model.Polyline;
import com.wz.location.map.model.PolylineOptions;
import com.wz.location.map.view.WZMap;

import java.io.InputStream;
import java.lang.ref.WeakReference;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

```
public class MainActivity extends AppCompatActivity implements
WzLocationListener, Map.MapReadyListener {

    private WzLocationClient wzLocationClient;
    private WzLocationClientOption clientOption;

    private WZMap wzMap;
    private RadioGroup radioGroup;
    private Button button;
    private Button btAddPoly;
    private Button btRemovePoly;
    private Button btnAddCircle;
    private Button btnRemoveCircle;
    private Button btnAddPolygon;
    private Button btnRemovePolygon;
    private WzLocation curLocation;
    double i = 0.00004f;
    private Polygon polygon;
    private Polyline polyline;
    private Circle circle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        wzMap = findViewById(R.id.wzMapView);
        radioGroup = findViewById(R.id.rg_style);
        button = findViewById(R.id.bt_add_mark);
        btAddPoly = findViewById(R.id.bt_add_poly);
        btRemovePoly = findViewById(R.id.bt_remove_poly);
        btnAddCircle = findViewById(R.id.bt_add_circle);
        btnRemoveCircle = findViewById(R.id.bt_remove_circle);
        btnAddPolygon = findViewById(R.id.bt_add_polygon);
        btnRemovePolygon = findViewById(R.id.bt_remove_polygon);

        wzMap.onCreate(savedInstanceState);
        wzMap.setMapReadyListener(this);
        checkPermission();
        initLocationClient();
        radioGroup.check(R.id.rb_default);

        radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {

            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                if (checkedId == R.id.rb_default) {
                    wzMap.setMapType(MapType.DEFAULT); // 切换地图样式
                } else if (checkedId == R.id.rb_gray_white) {
                    wzMap.setMapType(MapType.GRAY_WHITE);
                } else if (checkedId == R.id.rb_light_blue) {
                    wzMap.setMapType(MapType.LIGHT_BLUE);
                } else if (checkedId == R.id.rb_dark_blue) {
                    wzMap.setMapType(MapType.DARK_BLUE);
                }
                wzLocationClient.startLocation(MainActivity.this);
```

```
            }
        });

        wzMap.addOnMapLongClickListener(new Map.MapLongClickListener() { // 长按
地图事件
            @Override
            public void OnMapLongClickListener(@NonNull LatLng latLng) {
                return;
            }
        });

        wzMap.setOnMarkerClickListener(new Map.OnMarkerClickListener() {
            @Override
            public boolean onMarkerClick(Marker marker) {
                if (!marker.isInfoWindowShown()) {
                    marker.showInfoWindow();
                } else {
                    marker.hideInfoWindow();
                }
//                                marker.remove();
                return false;
            }
        });
        btAddPoly.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                List<LatLng> latLngs = new ArrayList<>();

                latLngs.add(new LatLng(30.504101, 114.414058));
                latLngs.add(new LatLng(30.504201, 114.414158));
                latLngs.add(new LatLng(30.504301, 114.414258));
                latLngs.add(new LatLng(30.504401, 114.414358));
                latLngs.add(new LatLng(30.504501, 114.414458));
                latLngs.add(new LatLng(30.504601, 114.414558));
                latLngs.add(new LatLng(30.504701, 114.414658));

                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    polyline = wzMap.addPolyline(new
PolylineOptions().addAll(latLngs).width(4)

.setDottedLine(true).color(getColor(R.color.teal_200)));
                }
            }
        });

        btRemovePoly.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (polyline != null) {
                    polyline.remove();
                } else {
                    Toast.makeText(MainActivity.this, "请先添加绘制线条",
Toast.LENGTH_SHORT).show();
                }
            }
        });
```

```
        btnAddCircle.setOnClickListener(new View.OnClickListener() { // 添加圆形
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    circle = wzMap.addCircle(new CircleOptions().center(new
LatLng(30.504101, 114.414058))

.radius(100f).strokeWidth(5).setDottedLine(true).fillColor(getColor(R.color.teal
_700)).strokeColor(getColor(R.color.teal_200)));
                }
            }
        });

        btnRemoveCircle.setOnClickListener(new View.OnClickListener() { // 删除原
型
            @Override
            public void onClick(View v) {
                if (circle != null) {
                    // wzMap.removeLayer(circle.getId());
                    circle.remove();
                } else {
                    Toast.makeText(MainActivity.this, "请先添加圆",
Toast.LENGTH_SHORT).show();
                }
            }
        });
        btnAddPolygon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 声明 多边形参数对象
                PolygonOptions polygonOptions = new PolygonOptions();

                LatLng latLng1 = new LatLng(30.504984, 114.412932);
                LatLng latLng2 = new LatLng(30.50163, 114.411925);
                LatLng latLng3 = new LatLng(30.501726, 114.417833);
                LatLng latLng4 = new LatLng(30.505428, 114.417184);
                LatLng latLng5 = new LatLng(30.504984, 114.412932);

                // 添加 多边形的每个顶点（顺序添加）
                polygonOptions.add(latLng1, latLng2, latLng3, latLng4, latLng5);
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    polygonOptions.strokeWidth(4) // 多边形的边框
                            .strokeColor(getColor(R.color.purple_500)) // 边框颜
色
                            .fillColor(getColor(R.color.teal_200));   // 多边形的
填充色
                }
                polygonOptions.setDottedLine(true);
                polygon = wzMap.addPolygon(polygonOptions);
            }
        });
        btnRemovePolygon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (polygon != null) {
                    polygon.remove();
                    wzMap.removeLayer(polygon.getId());
```

```
                } else {
                    Toast.makeText(MainActivity.this, "请先添加多边形",
Toast.LENGTH_SHORT).show();
                }

            }
        });

        wzMap.setOnMarkerDragListener(new Map.OnMarkerDragListener() {
            @Override
            public void onMarkerDragStart(Marker marker) {
                Log.i("TAG_TEST", "onMarkerDragStart :  " + marker.getId());
            }

            @Override
            public void onMarkerDrag(Marker marker) {
                Log.i("TAG_TEST", "onMarkerDrag :  " + marker.getId() + "lat: "
+ marker.getPosition().lat + "lon :" + marker.getPosition().lon);
            }

            @Override
            public void onMarkerDragEnd(Marker marker) {
                Log.i("TAG_TEST", "onMarkerDragEnd :  " + marker.getId());
            }
        });


        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (curLocation != null) {
                    i = i + 0.00005;
                    LatLng latLng = new LatLng(curLocation.getLatitude() + i,
curLocation.getLongitude() + i);
                    wzMap.addMarker(new
MarkerOptions().title("title").snippet("经纬度:" + (latLng.lon + "," +
latLng.lat)).position(latLng).icon(BitmapFactory.decodeResource(
                            MainActivity.this.getResources(),
R.mipmap.ic_user)));
                }
            }
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
        wzMap.onStart();
    }

    @Override
    protected void onResume() {
        super.onResume();
        wzMap.onResume();
    }

    @Override
```

```
    protected void onPause() {
        super.onPause();
        wzMap.onPause();
    }

    @Override
    protected void onStop() {
        super.onStop();
        wzMap.onStop();
    }

    @Override
    protected void onDestroy() {
        wzLocationClient.onDestroy();
        super.onDestroy();
        wzMap.onDestroy();
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        wzMap.onLowMemory();
    }

    private void initLocationClient() {
        clientOption = new WzLocationClientOption();
        clientOption.setFastLocation(false);
        clientOption.setOnceLocate(true);
        clientOption.setNeedPosition(true);
        clientOption.setInterval(5000);
        wzLocationClient = new WzLocationClient(this, clientOption);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions,
                                           @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    }

    private void checkPermission() {
        if (checkPermissionImp()) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                this.requestPermissions(new String[]{
                        Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION,
                        Manifest.permission.ACCESS_WIFI_STATE,
Manifest.permission.CHANGE_WIFI_STATE,
                        Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.ACCESS_NETWORK_STATE,
                        Manifest.permission.READ_PHONE_STATE,
                        Manifest.permission.INTERNET}, 0);
```

```
            }
        }
    }

    private boolean checkPermissionImp() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            return this.checkSelfPermission(
                    Manifest.permission.ACCESS_COARSE_LOCATION) !=
                    PackageManager.PERMISSION_GRANTED
                    ||
                    this.checkSelfPermission(
                            Manifest.permission.ACCESS_FINE_LOCATION) !=
                            PackageManager.PERMISSION_GRANTED
                    ||
                    this.checkSelfPermission(
                            Manifest.permission.READ_EXTERNAL_STORAGE) !=
                            PackageManager.PERMISSION_GRANTED
                    ||
                    this.checkSelfPermission(
                            Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
                            PackageManager.PERMISSION_GRANTED
                    ||
                    this.checkSelfPermission(
                            Manifest.permission.READ_PHONE_STATE) !=
                            PackageManager.PERMISSION_GRANTED;
        }
        return false;
    }

    @Override
    public void onLocationReceived(WzLocation wzLocation) {
        if (wzLocation != null) {
            Log.i("TAG_TEST", "lat: " + wzLocation.getLatitude() + "lon : " +
wzLocation.getLongitude());
            curLocation = wzLocation;
            wzMap.showLocationView(wzLocation.getLongitude(),
wzLocation.getLatitude(), wzLocation.getAccuracy()); // 显示定位点
        }
    }

    @Override
    public void onLocationError(WzException exception) {
        String errorMessage = new StringBuilder().append("数据异
常: ").append(exception.getErrorMessage()).toString();
    }

    /**
     * 监听 map 加载完成接口
     */
    @Override
    public void onMapReady() {
        if (wzLocationClient != null) {
            wzLocationClient.startLocation(this);
        }
    }
}
```