# 高德地图 Android
# 室内定位 SDK
# 开发指南

V1.0

高德软件有限公司
2015 年 12 月·北京

# 法律声明



高德地图

联系邮箱：api@autonavi.com

技术交流论坛：bbs.amap.com

官方微博：http://weibo.com/gaodedituapi

商务合作联系人：张先生　电话：010-84107170　电子邮箱： shiyue.zhang@autonavi.com

高德地图 API 欢迎用户的任何建议或意见。

# 目录

# 1 简介

## 1.1 室内定位 SDK

Android 室内定位 SDK 是一套简单的 LBS 服务的室内定位接口，通过室内定位 SDK 开发者可以迅速为应用程序实现室内定位功能。您可以单独使用室内定位 SDK，也可以结合高德地图 Android 室内地图 SDK。

## 1.2 面向的读者

高德地图 Android 室内定位 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的。此外，读者还应该对地图产品有一定的了解。用户在使用中遇到任何问题，都可以通过官网提供的 QQ 群或问答社区反馈给我们。

## 1.3 兼容性

支持 Android 4.0 以上系统，蓝牙定位需 Android 4.3 以上系统。

## 1.4 申请 API Key

为保证服务可以正常使用，您需要注册成为开发者并申请 Key。每个帐户，最多可以申请 10 个 Key。申请流程请参照获取密钥。

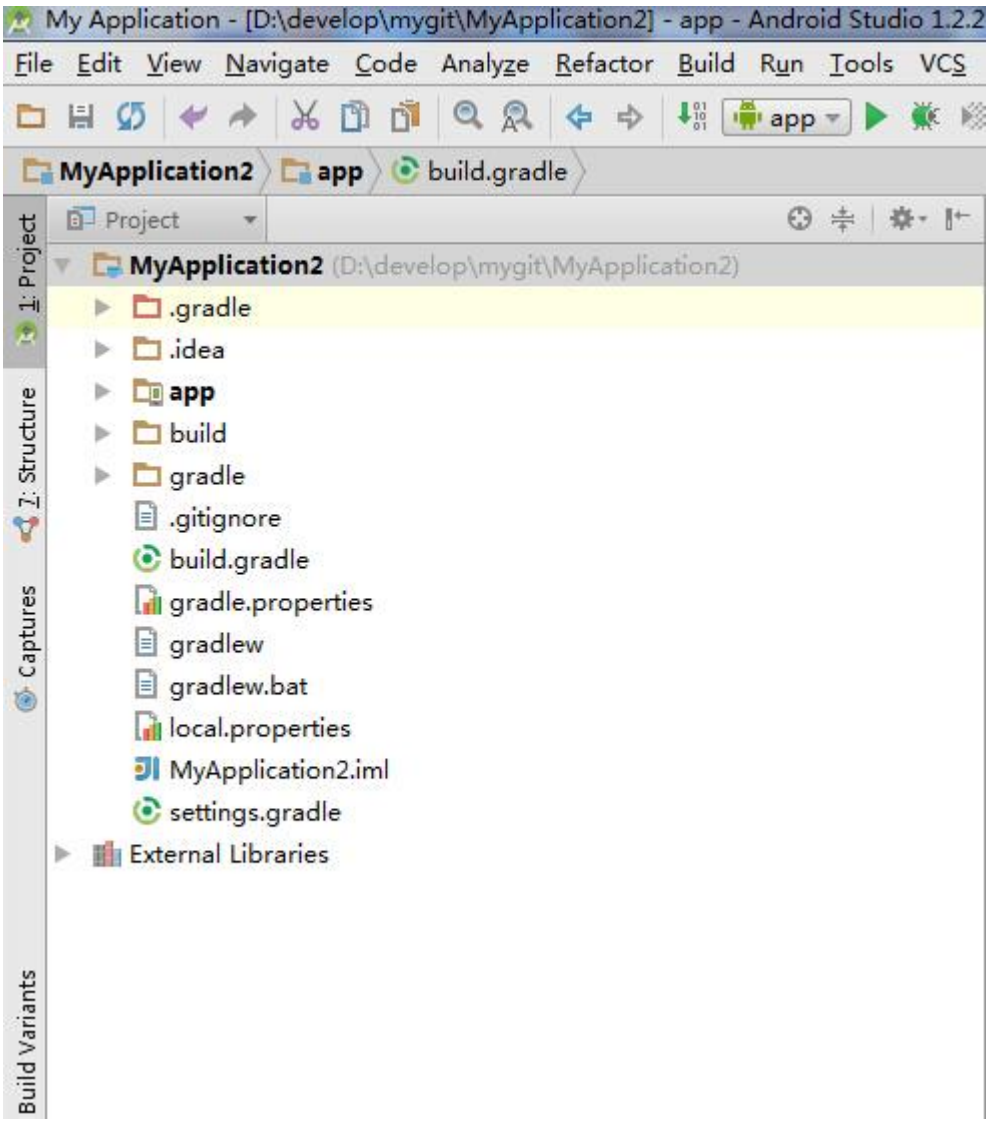
# 2 工程配置

## 2.1 下载

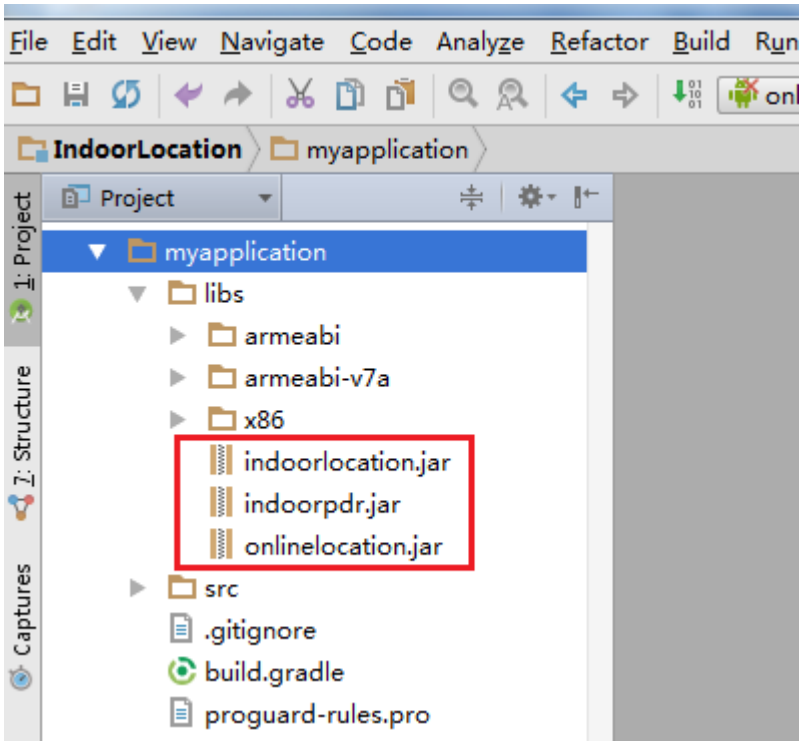
从网站下载并解压得到 3 个定位包"indoorlocation.jar"、"indoorpdr.jar"、"onlinelocation.jar", 分别是混合定位包、PDR 包和在线定位包。

## 2.2 Android Studio 配置工程

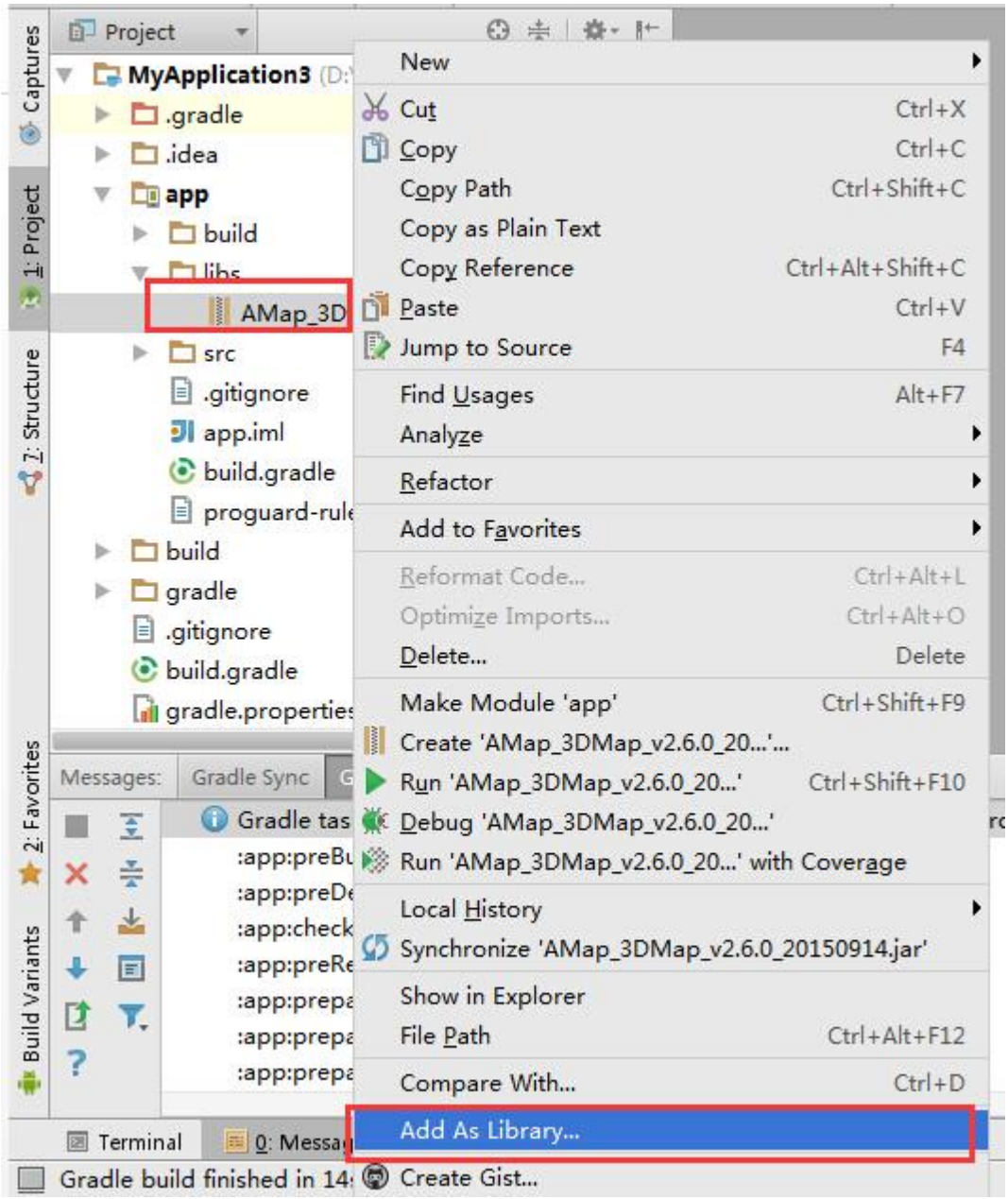
1. 新建工程如图所示：



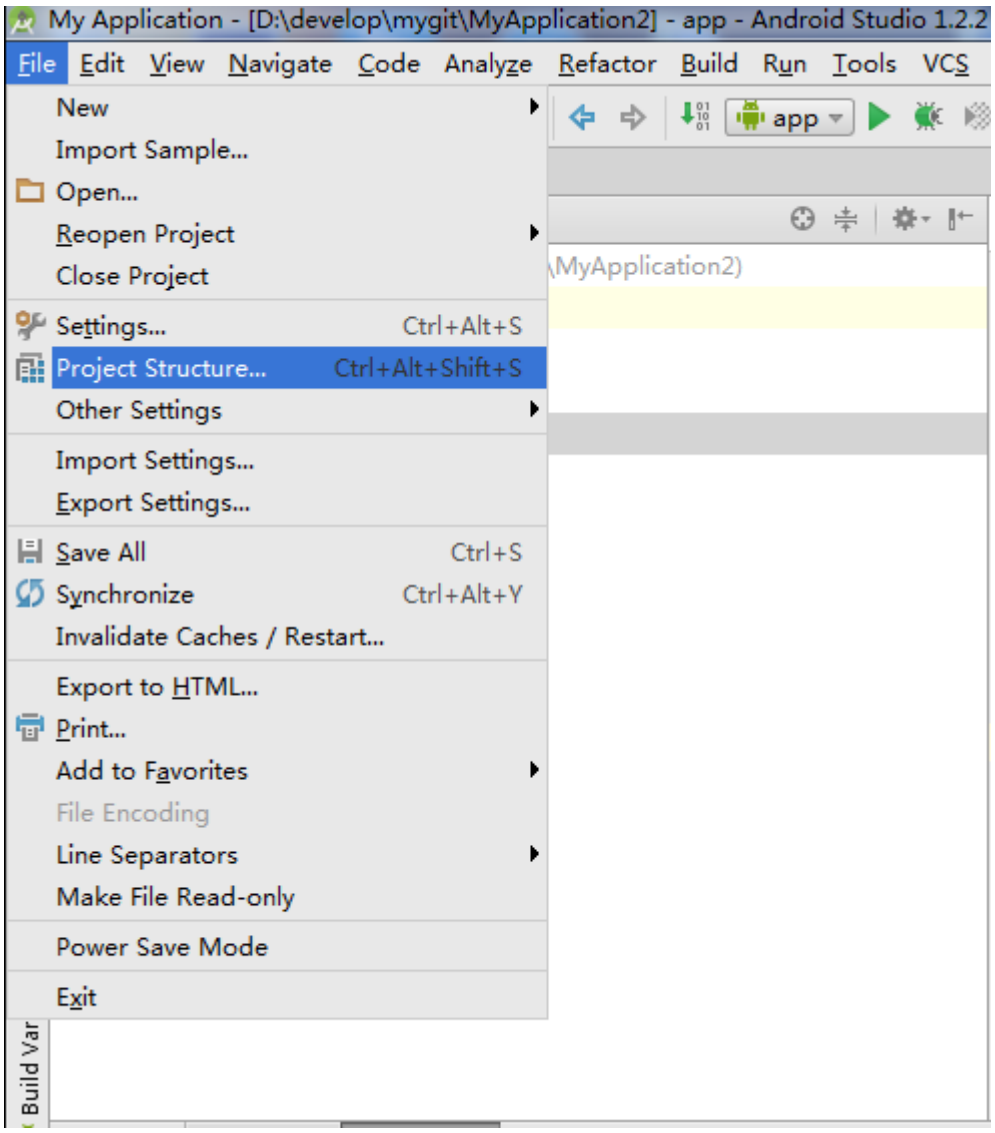

2. 在工程的 app/libs 目录下放入已经下载的开发包,将开发包中的 jar 包加到 libs 目录下，如图所示：



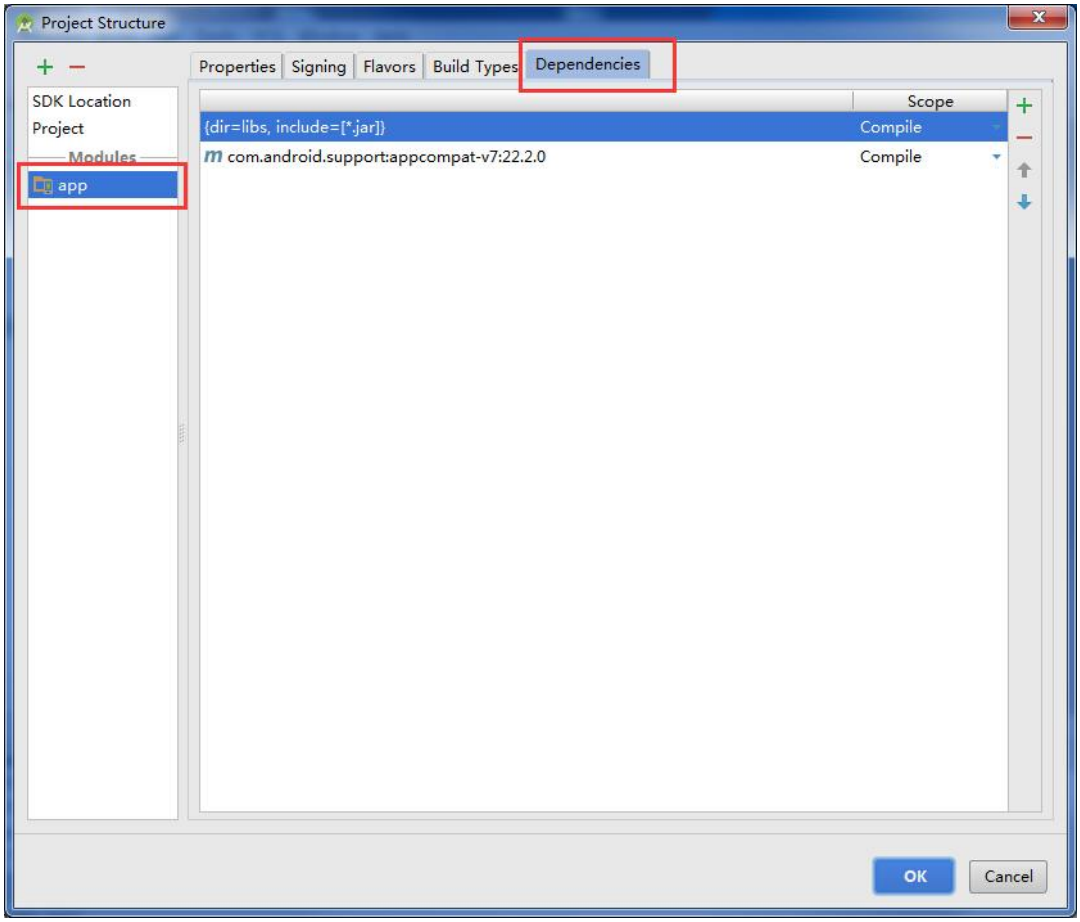

3. 选择放到 libs 下的 jar 包，右击选择 Add As Library



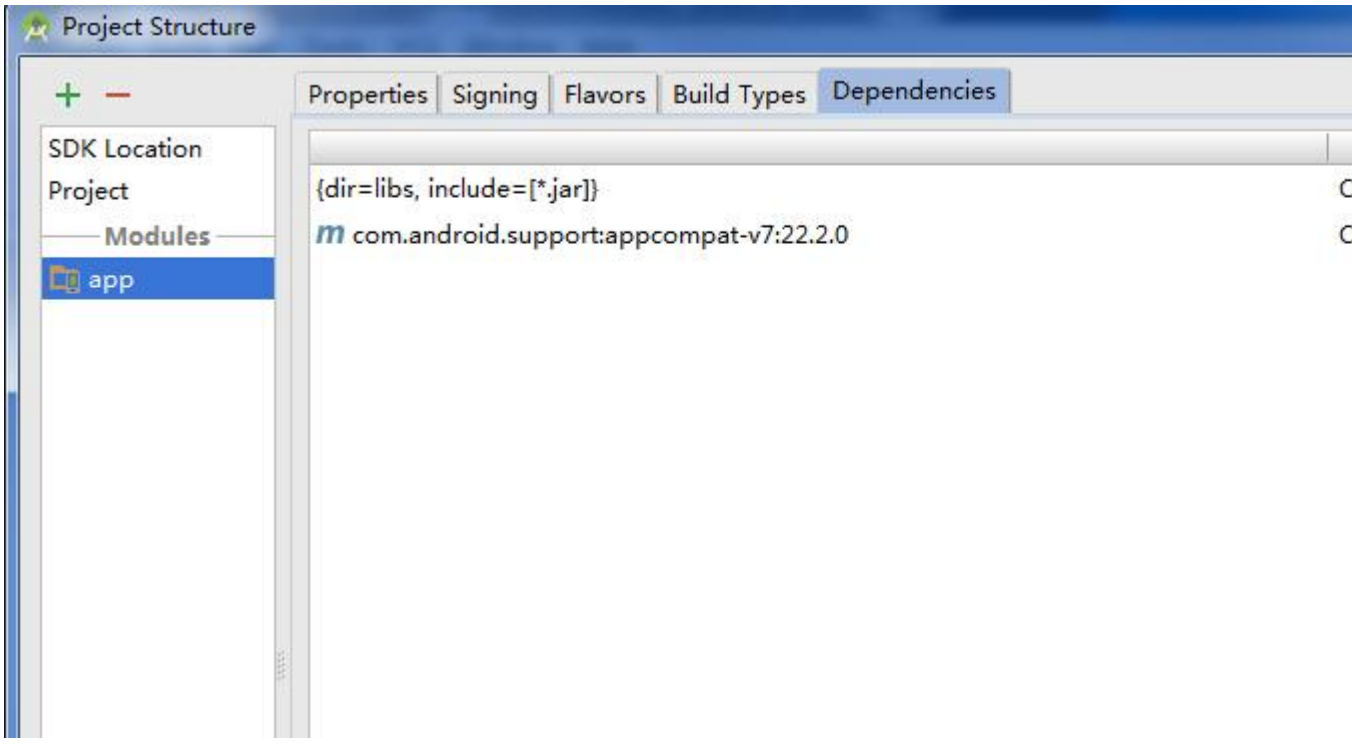

或者进行以下操作： <1>菜单栏选择 File —>Project Structure.




<2>在弹出的 Project Structure 对话框中，选择 module，然后点击 Dependencies 选项卡.

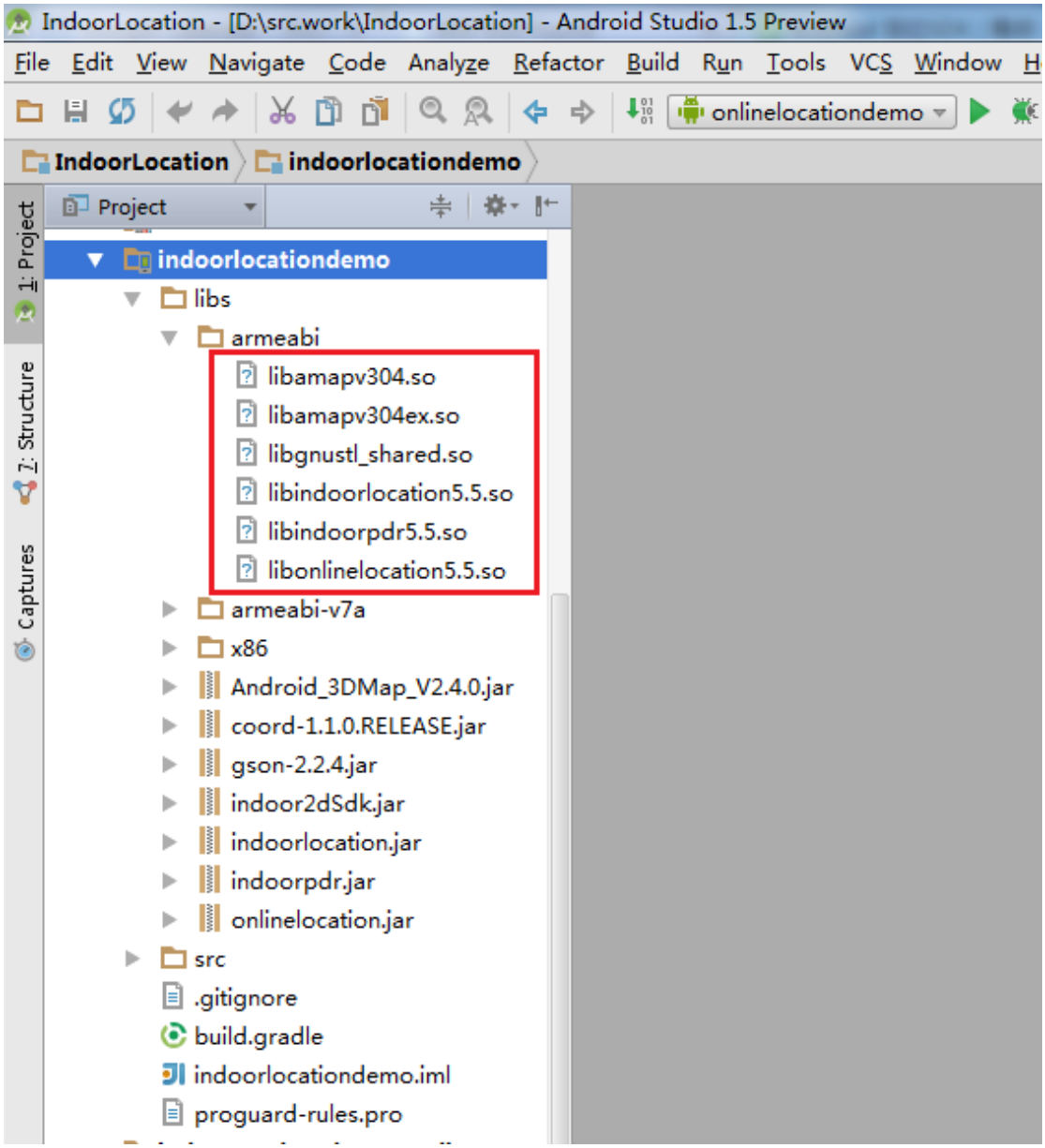



<3>点击绿色的加号选择 File dependency.



<4>然后选择要添加的 jar 包即可 完成上边的操作后在 app 目录下的 build.gradle 文件中，会有引入的类库

注意：因为 SDK 需要引入 so 库文件，所有需要再 app/src/main/目录下新建 jniLibs 目录，将 so 放到此目录下，如图所示：




## 2.3 添加用户 Key

添加您的用户 Key。 示例代码如下：

```
Configuration.Builder mConfigBuilder= new Configuration.Builder(context);
mConfigBuilder.setLBSParam("265d27445a2fec86fa53f93be85084e5");
```

## 2.4 添加权限

添加权限，请直接拷贝。

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-feature android:name="android.hardware.bluetooth_le" android:required="true
"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
```

# 3 配置模块

运用定位引擎实现室内定位，就必须提前设置好一些定位需要的参数。配置模块独立出来，必须在定位最开始的时候设置进去，一旦设置之后配置将不能再修改，直到用户销毁定位模块。

配置模块使用 Builder 模式用来构建，首先创建 Configuration.Builder 的对象，然后调用对象的一系列 set 方法设置各种参数，最后调用 build() 方法构造最后的配置对象，设置给定位引擎。

设置选项：

| 函数名 | 参数类型 | 说明 |
| --- | --- | --- |
| setSqlitePath | String | 设置数据库文件名 |
| setNetworkType | Int | 设置下载指纹的网络类型 |

示例代码：

```
Configuration.Builder mConfigBuilder=new Configuration.Builder(context);
LocationManager.getInstance().init(mConfigBuilder.build());
```

# 4 下载模块

下载模块用于从服务器端下载指纹数据等，默认情况下，在定位开始的时候，系统会自动启动下载服务，不断的下载当前定位需要的指纹数据。如果用户需要提前预先下载指纹，可以调用此模块实现指纹预下载功能。

下载模块以单体的形式提供出来，用户获取实例之后直接调用相应接口即可。

## 3.1. 获取实例

函数体：

public static DownloadManager getInstance();

功能诠释：

获取定位模块实例单体。

## 3.2. 初始化定位模块

函数体：

public void init(Configuration config);

功能诠释：

使用提前配置好的配置信息初始化下载模块，配置文件的主要配置选项参考上一节内容。

## 3.3. 按建筑下载指纹

函数体：

public DownloadCode download(String bid, int network, Handler handler);

功能诠释：

此函数可以一次性的下载 bid 指定建筑的所有指纹，network 指定下载时候需要的网络类型，handler 是下载消息处理接口，下载模块在下载过程中通过 handler 通知客户端下载所处的状态，返回值是错误码。

### 3.4. 下载模块返回码

```
	public enum DownloadCode
	{
		OK,				//正确
		ERROR,			//错误
		COMPLETE,		//完成
		PARTIAL,		//部分
		CANCEL,			//取消
		PROGRESS,		//进行中
		NOTHING,		//没有
	}
```

### 3.5. 网络类型

```
	public static final int NETWORK_NONE = 0;
	public static final int NETWORK_WIFI = 1;
	public static final int NETWORK_2G = 2;
	public static final int NETWORK_3G = 4;
	public static final int NETWORK_4G = 8;
	public static final int NETWORK_234G = NETWORK_2G | NETWORK_3G | NETWORK_4
G;
	public static final int NETWORK_34GWIFI = NETWORK_3G | NETWORK_4G | NETWOR
K_WIFI;
	public static final int NETWORK_234GWIFI = NETWORK_2G | NETWORK_34GWIFI;
	public static final int NETWORK_MOBILE = NETWORK_234G;
	public static final int NETWORK_ALL = NETWORK_234GWIFI;
```

### 3.6. 取消预下载

函数体：

public DownloadCode cancelDownload();

功能诠释：

取消正在下载的下载任务。

# 5 定位模块

定位模块根据客户端采集的信息进行定位服务。
下载模块以单体的形式提供出来，用户获取实例之后直接调用相应接口即可。主要接口有：

## 4.1. 引擎初始化

函数体：

public void init(String bid, final Configuration config, final Handler initHandler);

功能诠释：

初始化的室内定位模块。

参数诠释：

| 参数 | 数据类型 | 用处 |
| --- | --- | --- |
| bid | String | 要进行的室内定位建筑，如果是纯在线定位这个参数可以为空，为空的时候引擎做粗定位，成功之后会把建筑 bid 返回。如果不为空则表示在本建筑之内进行室内定位；如果不是 |
| config | Configuration | 配置信息，如上一节所述。 |
| initHandler | Handler | 初始化结果 handler，初始化过程中的一些错误和处理结果都通过这个回调函数返回过来。可能的消息有：MessageCode.MSG_THREAD_PREPARED 、MSG_WIFI_NOT_ENABLED 等，具体的错误消息参考其他章节。 |

用法备注：

```
示例代码：
ILocationManager mLocationManager;
// 获取到实例
mLocationManager = LocationManager.getInstance();
SDKInitHandler mSDKInitHandler = new SDKInitHandler(this);
// 初始化定位引擎
mLocationManager.init(this.strBuildNameId, mConfigBuilder.build(), mSDKInitHandle
r);
```

## 4.2. 销毁定位模块

函数体：

public void destroy();

功能诠释：

释放所有定位相关资源，并停止、销毁室内定位引擎。

用法备注：

销毁之后，室内定位引擎将不能再被使用，用户需要重新初始化定位引擎。

## 4.3. 监听/停止监听定位

函数体：

public void requestLocationUpdates(Handler handler);

public void removeUpdates(Handler handler)

功能诠释：

注册或删除定位监听，参数 handler 是回调接口，定位引擎发送定位结果、定位错误等信息到这个 handler。在第一次添加监听的时候，引擎会自动启动定位服务，当最后一个监听被删除之后，引擎将自动停止定位模块。

```
示例代码：
设置监听：
mLocationManager.requestLocationUpdates(mInnerHandler);
取消监听：
mLocationManager.removeUpdates(mInnerHandler);
```

## 4.4. 消息类型

类 MessageCode 中定义了定位引擎所需要的所有消息类型。主要的消息有：

| 消息名 | 含义 |
| --- | --- |
| MSG_THREAD_PREPARED | 定位模块初始化完成 |
| MSG_LOCATED_NO_MATCH_MAC | 定位失败。扫描到的 mac 地址都是无效的。 |
| MSG_LOCATED_FEW_MATCH_MAC | 定位失败。扫描到的 mac 地址太少，不足以定位 |
| MSG_REPORT_ONLINE_LOCATION | 在线定位成功返回。此时 msg.obj 是一个 LocationResult 对象 |
| MSG_REPORT_LOCATION | 离线或混合定位成功返回。此时 msg.obj 是一个 LocationResult |

| | 对象 |
|---|---|
| MSG_SENSOR_MISSING | 手机缺少步导需要的传感器：加速度、磁力计、重力计等 |
| MSG_BLE_NO_SCAN | 一段时间内没有蓝牙扫描 |
| MSG_WIFI_NO_SCAN | 一段时间内没有 WIFI 扫描 |
| MSG_NETWORK_ERROR | 网络错误 |
| MSG_NETWORK_NOT_SATISFY | 当前网络和用户设置的不符，不能下载数据 |
| MSG_SERVER_ERROR | 服务器端错误 |
| MSG_PRESSURE_CHANGED | 气压计有变化。此时 msg.obj 是一个 PressData 对象 |
| MSG_REPORT_PED | 步导信息返回值。此时 msg.obj 是一个 PedData 对象 |

示例代码：

```
// 创建 Handler 对象，处理回调消息
public void handleMessage(Message msg)
{
                switch (msg.what) {
                        // 获取到了在线定位结果
                        case MessageCode.MSG_REPORT_ONLINE_LOCATION: {
                                onLocated(msg, true);
                                break;
                        }
                        // 获取到了离线或混合定位结果
                        case MessageCode.MSG_REPORT_LOCATION: {
                                onLocated(msg, false);
                                break;
                        }
                        // 传感器检测到错误
                        case MessageCode.MSG_SENSOR_MISSING: {
                                AlertDlg.show(mParent, "MSG_SENSOR_MISSING",
"手机缺少步导需要的传感器：加速度、磁力计、重力计等");
                                break;
                        }
                        // 蓝牙检测到错误
                        case MessageCode.MSG_BLE_NO_SCAN: {
                                AlertDlg.show(mParent, "MSG_BLE_NO_SCAN", "一
段时间内没有蓝牙扫描");
                                break;
                        }
                        // wifi 检测到错误
                        case MessageCode.MSG_WIFI_NO_SCAN: {
```

```
                                AlertDlg.show(mParent, "MSG_WIFI_NO_SCAN", "
一段时间内没有 WIFI 扫描");
                                break;
                        }
                        // 网路错误
                        case MessageCode.MSG_NETWORK_ERROR: {
                                AlertDlg.show(mParent, "MSG_NETWORK_ERROR", "
网络错误");
                                break;
                        }
                        case MessageCode.MSG_NETWORK_NOT_SATISFY: {
                                AlertDlg.show(mParent, "MSG_NETWORK_NOT_SATIS
FY", "当前网络和用户设置的不符，不能下载数据");
                                break;
                        }
                        case MessageCode.MSG_SERVER_ERROR: {
                                AlertDlg.show(mParent, "MSG_SERVER_ERROR", "
服务器端错误");
                                break;
                        }
                        // 气压计变化
case MessageCode.MSG_PRESSURE_CHANGED: {
                                PressData pressure = (PressData) msg.obj;
                                L.d("MSG_PRESSURE_CHANGED, press:" + pressur
e.mPress);
                                break;
                        }
                        // 步导模块返回
                        case MessageCode.MSG_REPORT_PED: {
                                PedData pedData = (PedData) msg.obj;
                                L.d("MSG_REPORT_PED, step:" + pedData.mStep +
 " angle:" + pedData.mAngle);
                                break;
                        }
                        }
                }
```

## 4.5. 定位结果

定位成功之后，引擎发送消息 MSG_REPORT_LOCATION 或 MSG_REPORT_ONLINE_LOCATION 给上层 其中定位结果以 LocationResult 对象方式保存在 msg.obj 中 具体 LocationResult 的成员主要有：

| 参数 | 数据类型 | 说明 |
| --- | --- | --- |

| | | |
|---|---|---|
| x | double | 经度 |
| y | double | 纬度 |
| z | int | 楼层 |
| r | float | 精度，单位米 |
| a | float | 角度，单位度(0-360) |
| o | int | 定位结果类型是在线还是离线 |