IDO-BLE SDK

Quick Access

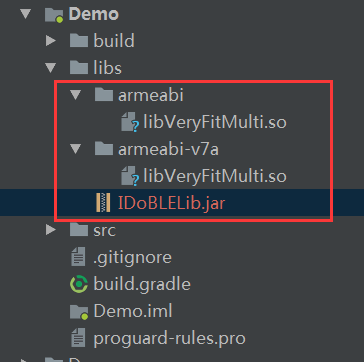
## Revision Recording

|  |  |
| --- | --- |
| Version | V1 |
| Time | 2017.11.06 |
| Maker | Mr. Zhou |

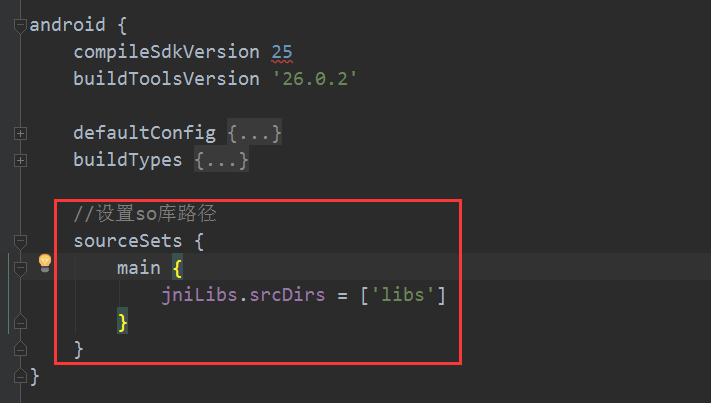
## 一 Environment configuration

1. Start android studio, set up or open your app project

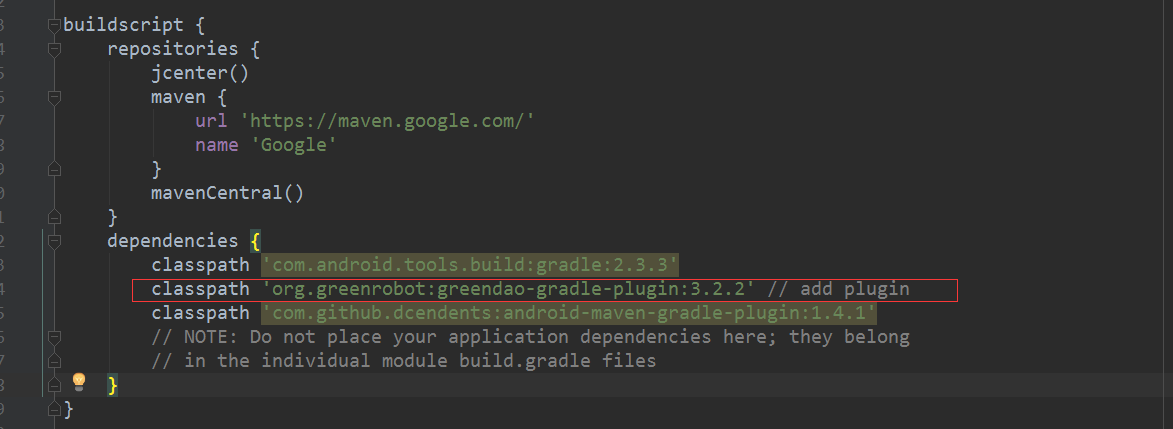
2. Copy the SO library and JAR pack we provided to libs category, see the area in below with red frame.



3. Add below configuration in build.gradle under moudle to set up the path of SO, see the area in below with red frame.



Add below configuration in build.gradle under project, see the area in below with red frame.



classpath 'org.greenrobot:greendao-gradle-plugin:3.2.2'

Open Source：

//SDK--need  
 compile 'org.greenrobot:greendao:3.2.2'

compile 'com.alibaba:fastjson:1.2.35'

compile 'com.squareup.okhttp3:okhttp:3.4.1'

compile 'com.squareup.okio:okio:1.9.0'

compile 'com.google.code.gson:gson:2.8.0'

compile 'com.android.support:support-core-utils:27.0.2'

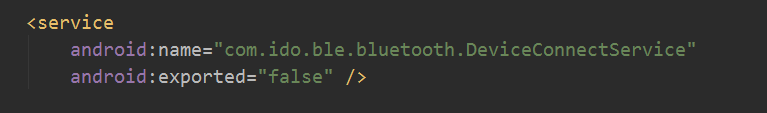
compile 'no.nordicsemi.android:dfu:1.6.1'

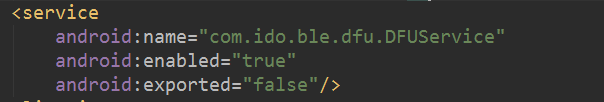
4. Add below configuration in AndroidManifest.xml under moudle,

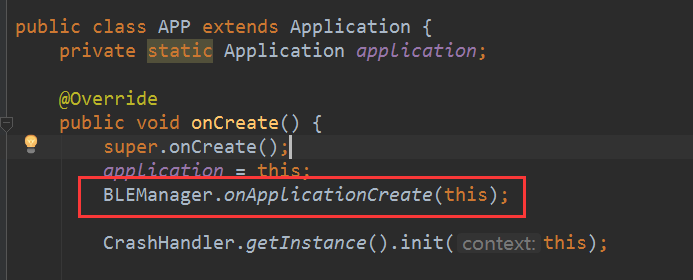
Permission：

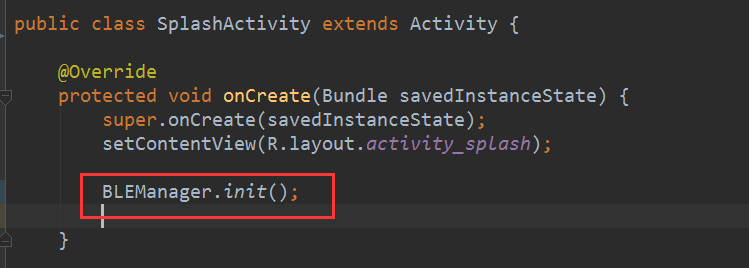
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.BLUETOOTH" />  
<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />  
<uses-permission android:name="android.permission.INTERNET" />

service：





1. Add below code in your application page, see the area in below with red frame:
2. 

6. Add SDK initial code on you first Activity page, see the area in below with red frame****

**For 6.0 or above system, you should get the permission first and then do initialization.**

## 二 Initialization Flow

**I：Using APP for first time**

1. scan device

2. connect device

3. device binding

4. setup user information

5. setup units

6. sets target for each day

7. synchronize configuration information

**II：Unbind**

1. scan device

2. connect device

3. device binding

4. synchronize configuration information

**Note：**

All above steps with red letters must be executed according to the specified steps.

While for those steps with black letters, you can choose to execute or not per your own idea.

You can refer to our Demo for the particular example.

## 三 Callback Mechanism

Callback mechanism is mainly used to monitor “connection status” and “feedback on interaction status between APP and bracelet”

For example:

When you scan the device, you can see which device are scanned by setting the scan Callback.

When you bind the device, you can see if it is bound successfully or not by binding Callback.

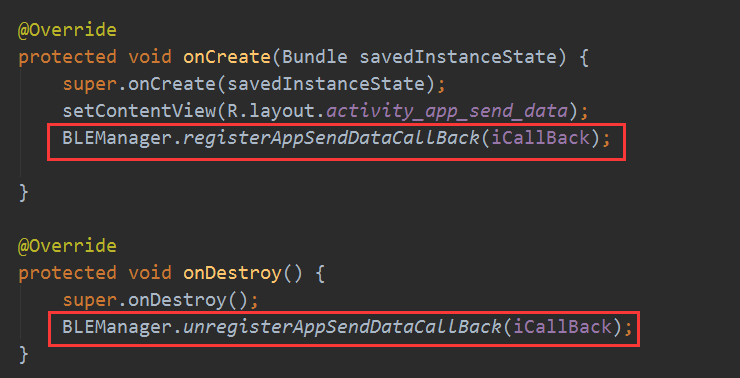
When you setup parameters, you will know if it is setup successfully by setup Callback.

Callback registration and logout should occur in pairs, otherwise there will be a risk of memory leak.

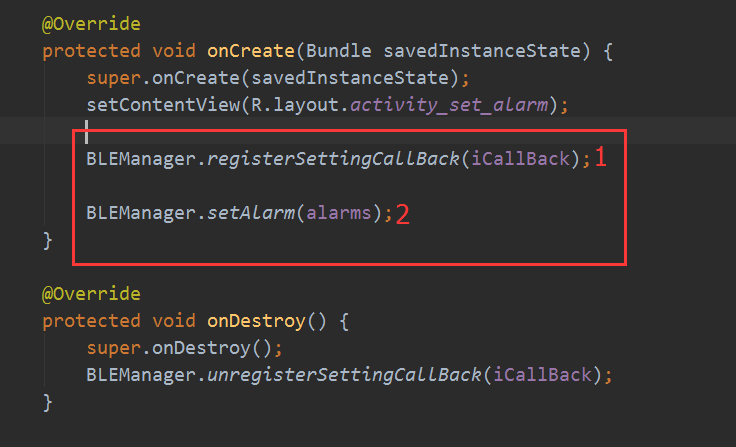
All Callbacks are in the main thread, so don’t do any time-consuming (blocking) operations in callback.

Correct operation way:

Register at onCreate and logout at onDestroy



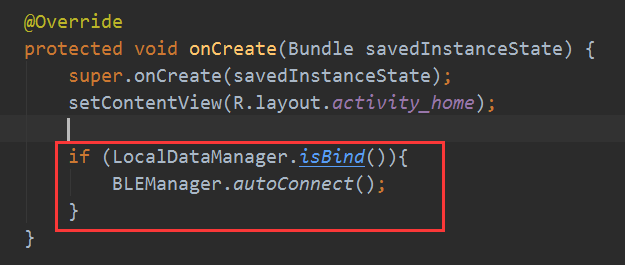
Before interaction with the bracelet, you should setup the callback first and then do the operation; Such as setting the alarm clock, you should set the alarm clock callback, and then send the alarm data to bracelets, so that you can receive feedback from bracelet normally to judge whether it is set successfully, see below



## 四 Connection

After you use App or unbinding at the first time, you should scan the device first and then call BLEManger.connect() interface to connect device.

As long as it is bound sucessfully at the first time, you can call BLEManager.autoConnect() interface to connect device automatically at your initial page, see below



SDK has reconnecting mechanism, means when APP is disconnected with smarts band, SDK will carry the reconnect operation till successfully connect to the last binding of bracelet.

**Reconnecting mechanism will stop under any of following conditions**

1. The Bluetooth switch in phone is off.
2. App has been unbound from bracelet.

3. App process is killed.

4. Already connected bracelet.

## 五 Health data

**Get healthy data:**

You cannot get health data from bracelet, but can only get from the local database in SDK. So before you get health data, you should call BLEManager.startSyncHealthData() to synchronize data, and after the data is synchronized successfully, you can call LocalDataMangaer.XXX() interface to get data.

**Data synchronization flow** 

1. SDK send synchronization command, the bracelet will feedback all data after get the command.
2. After receiving the health data, SDK will store the data in the local database DB, and then the APP can access the health data through API provided by SDK.
3. Once the synchronization is successful, the bracelet will delete all health data outside the day automatically.

**Data storage time:**

Although the bracelet will delete the data outside the day after synchronization, the data is stored in the SDK database, so there will be no problems.

If you unbind, the SDK will clear all health data;

If the user unloads the App, the health data in the SDK will also be erased, and the data will never be recovered.

You can do data permanent storage operation, namely "when synchronization is completed, access to health data from the SDK, and put these health data permanently in a remote server", then even if the user uninstalled your application, you can also obtain previous health data from the remote server.

## 六 Local data

SDK local data management mainly saves two types of data:

1. Health data

2. Setting parameter data (for example, after you successfully set daily goal, the SDK will save this parameter to local database)

Specifically, you can get the data you want by calling LocalDataManger.XXX().

## 七 Key class file

For SDK, you will mainly use the following two classes

1. BLEManager

2. LocalDataManger

## 八 Notes

1. You should judge whether the phone is connected to the device before you interact with the bracelet (judge through BLEManager. IsConnected ())

2. If it is android 6.0 or above, you will not only need to apply for permission in androidmanifest.xml, bu also need to apply for dynamic permission.

3. If you need to determine whether the health data in the SDK is correct, in the Debug stage of your APP, you can use the " Stetho plug-in in facebook" to view local database data; For specific way, pls search 'Stetho' from Google.