链滴

Android：进程间通信第一发----AIDL

作者：wind

原文链接：https://ld246.com/article/1444577246884

来源网站：链滴

最近来看研究一下神奇的AIDL通信，结果发现网上的教程都是很老很老的啊，全是在Eclipse上进的，放到Android Studio有些方法就不行嘛.....

其实感觉网上的教程是有点过时，很多都是11，12年的。。现在的少年不太愿意分享咩= =

先发一下自己的代码链接吧：

https://github.com/buptljy/aidl_demo-service

https://github.com/buptljy/aidl_demo-client

这个demo呢，由service和client两个程序组成，把这两个程序下载到手机里，运行就OK了。

现在呢，我们来看看怎么做吧。好可惜不能上图，所以我会尽量的说的详细一点。

Android和Eclipse最大的不同是gradle的引入吧，虽然Eclipse也有，但没有Android Studio那么视。

在Eclipse中，当你建立aidl文件之后，ADT会自动帮你生成java文件，而在Android Studio 中却要自己手动操作啦！也不是很麻烦。下面说详细步骤！（代码贴在最后）

建立两个工程，package name不能一样！否则无法安装！我里service的package是com.styling，client的package是com.style

将两个工程以Project的形式展开（这样你也可以看到generatd文件夹里的生成的java文件），以下都为两个工程都要进行的操作：在main下建立一个名为aidl文夹，在aidl下建立package，此时两个package的名字都要一样，如com.style，在com.style中建立mnterface.aidl文件。

打开client工程，在client工程里的MainActivity中写下绑定Sevice的代码，打开service工程，新建一个service，并在manifest.xml文件中定义service。

<pre class="prettyprint lang-js">AIDL代码:
package com.style;

```
interface mInterface {
    void invokeTest();
}</pre>
<p>
    <br />
</p>
<pre class="prettyprint lang-js">MainActivity代码:
public class MainActivity extends Activity {


private static final String TAG = "AIDLActivity";
private Button btnOk;
private Button btnCancel;
private Button btnCallBack;

private void Log(String str){
    Log.d(TAG,"----------" + str + "----------");
}


mInterface mService;
private ServiceConnection mConnection = new ServiceConnection(){
    public void onServiceConnected(ComponentName className,
                                   IBinder service){
        Log("connect service");
        mService = mInterface.Stub.asInterface(service);
    }

    public void onServiceDisconnected(ComponentName className){
        Log("disconnect service");
        mService = null;
    }
};



/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btnOk = (Button)findViewById(R.id.btn_ok);
    btnCancel = (Button)findViewById(R.id.btn_cancel);
    btnCallBack = (Button)findViewById(R.id.btn_callback);

    btnOk.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
            Bundle args = new Bundle();
            Intent intent = new Intent("com.styling.service");
            intent.putExtras(args);
            bindService(intent,mConnection,Context.BIND_AUTO_CREATE);
        }
    });
```

```
    btnCancel.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
            unbindService(mConnection);
        }
    });
    btnCallBack.setOnClickListener(new OnClickListener(){
        public void onClick(View v){
            try{
                Log.i(TAG,"current Thread id = " + Thread.currentThread().getId());
                mService.invokeTest();
            }
            catch(RemoteException e){
                e.printStackTrace();
            }
        }
    });


}

}</pre>

<p>
    <br />
</p>
<p>
    这里新建的ServiceConnection对象mConnection重写了两个方法，其中onServiceConnected方
，在Service连接之后会自动调用，这里是调用之后，实例化接口对象。
</p>
<p>
    然后在看下面的Service代码，注意新建的mInterface.Stub对象mBinder，client在连接Service的
候，传入onServiceConnected函数的IBinder对象就是这个mBinder！然后在client端利用mBinder
立了一个mInterface的实例。这就实现了进程间的通信，有没有感觉这里有个Proxy，client并没有
接和Service通信，而是间接的！由于两个aidl文件所在的包名必须相同，所以我推测，进程间通信是
据aidl文件所在包来做选择的。
</p>
<p>
    <br />
</p>
<pre class="prettyprint lang-js">Service代码:
public class mService extends Service{
    private static final String TAG = "AIDLService";


private void Log(String str){
    Log.i(TAG, "----------" + str + "----------");
}

public void onCreate(){
    Log("service created");
}

public void onStart(Intent intent, int startId){
```

```
    Log("service started id = " + startId);
}

public IBinder onBind(Intent t){
    Log("service on bind");
    return mBinder;
}

public void onDestroy(){
    Log("service on destroy");
    super.onDestroy();
}

public boolean onUnbind(Intent intent){
    Log("service on unbind");
    return super.onUnbind(intent);
}

public void onRebind(Intent intent){
    Log("service on rebind");
    super.onRebind(intent);
}


private final mInterface.Stub mBinder = new mInterface.Stub() {
    public void invokeTest() throws RemoteException {
        // TODO Auto-generated method stub
        Log.e(TAG, "remote call from client! current thread id = " + Thread.currentThread().getId(
);
    }
};

}</pre>

<p>
    <br />
</p>
<p>
    <br />
</p>
<p>
    <br />
</p>
<p>
    <br />
</p>
```