**[Android 多个APK共享数据(Shared User ID)](http://bingoit.iteye.com/blog/457891)**

http://bingoit.iteye.com/blog/457891

Android给每个APK进程分配一个单独的用户空间,其manifest中的userid就是对应一个Linux用户  
(Android 系统是基于Linux)的.  
所以不同APK(用户)间互相访问数据默认是禁止的.  
但是它也提供了2种APK间共享数据的形式:  
1. Share Preference. / Content Provider  
APK可以指定接口和数据给任何其他APK读取. 需要自己实现接口和Share的数据.  
本文对于这个不做详细解释  
  
2. Shared User id  
通过Shared User id,拥有同一个User id的多个APK可以配置成运行在同一个进程中.所以默认就是  
可以互相访问任意数据. 也可以配置成运行成不同的进程, 同时可以访问其他APK的数据目录下的  
数据库和文件.就像访问本程序的数据一样.  
比如某个公司开发了多个Android 程序, 那么可以把数据,图片等资源集中放到APK  A中去. 然后  
这个公司的所有APK都使用同一个User ID, 那么所有的资源都可以从APK A中读取.  
  
举个例子:  
APK A 和APK B 都是C公司的产品,那么如果用户从APK A中登陆成功.那么打开APK B的时候就不用  
再次登陆. 具体实现就是 A和B设置成同一个User ID:  
    \* 在2个APK的AndroidManifest.xml 配置User ID:  
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"   
    package="com.android.demo.a1"  
    android:sharedUserId="com.c">  
   这个"com.c" 就是user id, 然后packagename APK A就是上面的内容,  APK B可能  
   是"com.android.demo.b1" 这个没有限制  
  
这个设定好之后, APK B就可以像打开本地数据库那样 打开APK A中的数据库了.  
APK A把登陆信息存放在A的数据目录下面. APK B每次启动的时候读取APK A下面的数据库  
判断是否已经登陆:  
APK B中的代码:  
            friendContext = this.createPackageContext(  
                    "com.android.demo.a1",  
                    Context.CONTEXT\_IGNORE\_SECURITY);  
通过A的package name 就可以得到A的 packagecontext  
通过这个context就可以直接打开数据库.

[android 类似QQ 换皮肤 实现思路 apk资源共享](http://blog.csdn.net/csy172775922/article/details/6714703)

http://blog.csdn.net/csy172775922/article/details/6714703

1、首先在AndroidManifest.xml中的<manifest>中加入android:sharedUserId="共享id"----具有相同id的apk可以不受限制访问。

2、根据包名创建Context，在Activity下有提供方法createPackageContext能够依据包名创建。

3、获取共享的Apk资源。

注意：

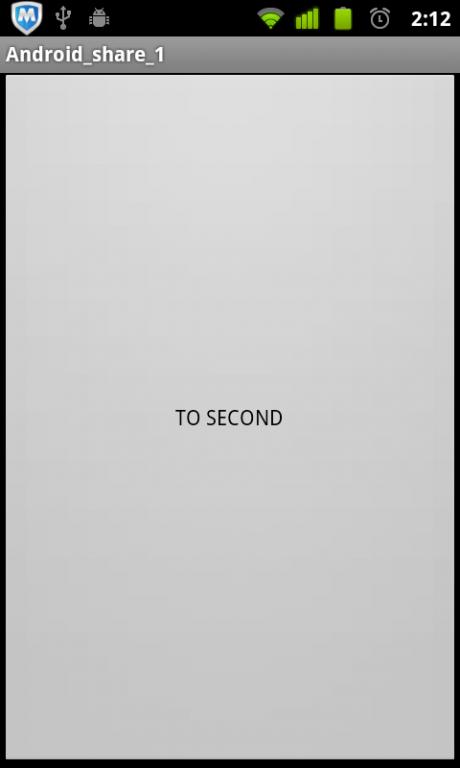
1、Activity中的findViewById()与View中的findViewById()区别在于：

Activity需要对setContextView()后的Layout才能由findViewById()获得View。

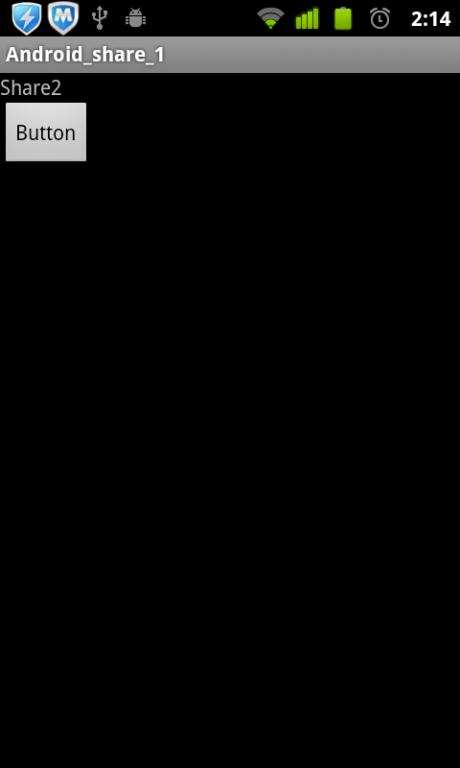
View只需要有对象即可由findViewById()获得在View中包含id的View。

2、在资源获取方面主要获取那个apk下的资源主要是由Context决定。

在资源apk没安装前：



资源文件安装后：



主apk包名：com.app.share;

资源apk包名：com.app.share2;

主Activity中获取资源apk中R,并且生成Layout对id为button1的按钮设置监听。

**[java]** [view plaincopy](http://blog.csdn.net/csy172775922/article/details/6714703)

**public** **class** StartAct **extends** Activity {

    /\*\* Called when the activity is first created. \*/

    @Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

//        setContentView(R.layout.main);

        Button btn = **new** Button(**this**);

        btn.setText("TO SECOND");

        btn.setOnClickListener(**new** OnClickListener() {

            @Override

**public** **void** onClick(View v) {

                // TODO Auto-generated method stub

                Intent intent = **new** Intent(StartAct.**this**, SecontAct.**class**);

                StartAct.**this**.startActivity(intent);

            }

        });

**this**.setContentView(btn);

**try** {

            Context other = **this**.createPackageContext("com.app.share2", CONTEXT\_IGNORE\_SECURITY|CONTEXT\_INCLUDE\_CODE);

            Class<?> c = other.getClassLoader().loadClass("com.app.share2.R");

            Class<?>[] cl = c.getClasses();

**int** b =0;

**for** (**int** i = 0; i < cl.length; i++) {

                Log.d("TAG", cl[i].getSimpleName());

                Field field[] = cl[i].getFields();

**for** (**int** j = 0; j < field.length; j++) {

                    Log.d("TAG", "NAME:"+field[j].getName()+"--VALUE:"+field[j].getInt(field[j].getName()));

**if**(field[j].getName().equals("button1")) {

                        b = field[j].getInt(field[j].getName());

                        Log.d("TAG", "--------id");

                    }

                };

            }

            View v = LayoutInflater.from(other).inflate(R.layout.main, **null**);

            Button btn1 = (Button) v.findViewById(b);

            btn1.setOnClickListener(**new** OnClickListener() {

                @Override

**public** **void** onClick(View v) {

                    // TODO Auto-generated method stub

                    Log.d("TAG", "BUTTON FROM Share2");

                    Toast.makeText(StartAct.**this**, "BUTTON FROM Share2", Toast.LENGTH\_SHORT).show();

                }

            });

**this**.setContentView(v);

        } **catch** (NameNotFoundException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        } **catch** (ClassNotFoundException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        } **catch** (IllegalArgumentException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        } **catch** (IllegalAccessException e) {

            // TODO Auto-generated catch block

            e.printStackTrace();

        }

    }

}

资源apk包Layout

**[html]** [view plaincopy](http://blog.csdn.net/csy172775922/article/details/6714703)

**<?xml** version="1.0" encoding="utf-8"**?>**

**<LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android"

    android:orientation="vertical"

    android:layout\_width="fill\_parent"

    android:layout\_height="fill\_parent"

**>**

**<TextView**

    android:layout\_width="fill\_parent"

    android:layout\_height="wrap\_content"

    android:text="Share2"

**/>**

**<Button** android:text="Button" android:id="@+id/button1" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"**></Button>**

**</LinearLayout>**

Android QQ皮肤制作 apk之间资源共享

在android系统中，apk之间可以相互读取数据。但是必须具备以下条件

1.有同样的签名(\*.keystoe)

2.在AndroidManifest.xml文件中配置的android:sharedUserId属性值相同

3.二个apk运行在同一个进程中.

主程序及读取的apk中AndroidManifest.xml中配置

        android:sharedUserId="com.why.android"

访问资源的方法

        Context mContext = this.createPackageContext("com.why.android",Context.CONTEXT\_IGNORE\_SECURITY);

        Drawable image = mContext.getResources().getDrawable(R.drawable.icon);

    Button mButton = (Button)findViewById(R.id.btn);

    mButton.setBackgroundDrawable(image);

利用这原理，咱们可以为应用程序制作像QQ那样的皮肤。

## shareUserId介绍：

Android给每个APK进程分配一个单独的空间,manifest中的userid就是对应一个分配的Linux用户ID，并且为它创建一个沙箱，以防止影响其他应用程序（或者其他应用程序影响它）。用户ID 在应用程序安装到设备中时被分配，并且在这个设备中保持它的永久性。

通常，不同的APK会具有不同的userId，因此运行时属于不同的进程中，而不同进程中的资源是不共享的，在保障了程序运行的稳定。然后在有些时候，我们自己开发了多个APK并且需要他们之间互相共享资源，那么就需要通过设置shareUserId来实现这一目的。

通过Shared User id,拥有同一个User id的多个APK可以配置成运行在同一个进程中.所以默认就是可以互相访问任意数据. 也可以配置成运行成不同的进程, 同时可以访问其他APK的数据目录下的数据库和文件.就像访问本程序的数据一样。

## shareUserId设置：

在需要共享资源的项目的每个AndroidMainfest.xml中添加shareuserId的标签。

android:sharedUserId="com.example"

id名自由设置，但必须保证每个项目都使用了相同的sharedUserId。一个mainfest只能有一个Shareuserid标签。

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.shareusertesta"

android:versionCode="1"

android:versionName="1.0"

android:sharedUserId="com.example">

## 

## \data\data\自定义的package\ 路径下的互相访问

每个安装的程序都会根据自己的包名在手机文件系统的data\data\your package\建立一个文件夹（需要su权限才能看见），用于存储程序相关的数据。

在代码中，我们通过context操作一些IO资源时，相关文件都在此路径的相应文件夹中。比如默认不设置外部路径的文件、DB等等。

正常情况下，不同的apk无法互相访问对应的app文件夹。但通过设置相同的shareUserId后，就可以互相访问了。代码如下。

[复制代码](javascript:void(0);)

//程序A：

public class MainActivityA extends Activity {

TextView textView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

textView = (TextView)findViewById(R.id.textView1);

WriteSettings(this, "123");

}

public void WriteSettings(Context context, String data) {

FileOutputStream fOut = null;

OutputStreamWriter osw = null;

try {

//默认建立在data/data/xxx/file/

fOut = openFileOutput("settings.dat", MODE\_PRIVATE);

osw = new OutputStreamWriter(fOut);

osw.write(data);

osw.flush();

Toast.makeText(context, "Settings saved", Toast.LENGTH\_SHORT)

.show();

} catch (Exception e) {

e.printStackTrace();

Toast.makeText(context, "Settings not saved", Toast.LENGTH\_SHORT)

.show();

} finally {

try {

osw.close();

fOut.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

//程序B：

public class MainActivityB extends Activity {

TextView textView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

textView = (TextView) this.findViewById(R.id.textView1);

try {

//获取程序A的context

Context ctx = this.createPackageContext(

"com.example.shareusertesta", Context.CONTEXT\_IGNORE\_SECURITY);

String msg = ReadSettings(ctxDealFile);

Toast.makeText(this, "DealFile2 Settings read" + msg,

Toast.LENGTH\_SHORT).show();

WriteSettings(ctx, "deal file2 write");

} catch (NameNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public String ReadSettings(Context context) {

FileInputStream fIn = null;

InputStreamReader isr = null;

char[] inputBuffer = new char[255];

String data = null;

try {

//此处调用并没有区别，但context此时是从程序A里面获取的

fIn = context.openFileInput("settings.dat");

isr = new InputStreamReader(fIn);

isr.read(inputBuffer);

data = new String(inputBuffer);

textView.setText(data);

Toast.makeText(context, "Settings read", Toast.LENGTH\_SHORT).show();

} catch (Exception e) {

e.printStackTrace();

Toast.makeText(context, "Settings not read", Toast.LENGTH\_SHORT)

.show();

} finally {

try {

isr.close();

fIn.close();

} catch (IOException e) {

e.printStackTrace();

}

}

return data;

}

public void WriteSettings(Context context, String data) {

FileOutputStream fOut = null;

OutputStreamWriter osw = null;

try {

fOut = context.openFileOutput("settings.dat", MODE\_PRIVATE);

//此处调用并没有区别，但context此时是从程序A里面获取的

osw = new OutputStreamWriter(fOut);

osw.write(data);

osw.flush();

Toast.makeText(context, "Settings saved", Toast.LENGTH\_SHORT)

.show();

} catch (Exception e) {

e.printStackTrace();

Toast.makeText(context, "Settings not saved", Toast.LENGTH\_SHORT)

.show();

} finally {

try {

osw.close();

fOut.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

[复制代码](javascript:void(0);)

如果A和B的mainfest中设置了相同的shareuserId，那么B的read函数就能正确读取A写入的内容。否则，B无法获取该文件IO。

通过这种方式，两个程序之间不需要代码层级的引用。之间的约束是，B需要知道A的file下面存在“settings.dat”这个文件以及B需要知道A的package的name。