# 58一面

### （1）Android四大组件

Android的四大组件除了BroadcastReceiver以外，其他三种组件必须在AndroidManifest中注册，对于BroadcastReceiver来说，既可以在AndroidManifest中注册，也可以在代码中注册。在调用方式上，Activity、Service、和BroadcastReceiver需要借助Intent，二ContentProvider无需借助Intent。

### 1.1.Activity的作用

Activity是一种展示型组件，可以接收用户的输入信息与用户交互。

Activity由Intent触发，有显式和隐式两种。

Activity有四种启动模式，分别为standard、singleTop、singleTask、singleInstance

### 1.2.Activity能否开启多进程

Activity可以通过在AndroidManifest中添加android:process的属性来开启多进程。若该Activity需要被外界调用，则需要设置android:exported = true这个属性，默认是false，除非Activity设置了Intent-Filter。设置了android:exported=true,同时也可以设置android:permission来限制外部应用的调用。

## 2.Service

### 2.1Service的作用

Service是一种计算型组件，用于在后台执行一些列的计算任务。由于Service组件工作在后台，因此用户无法直接感知到它的存在。Service组件和Activity组件不同，Activity只有一种运行模式，即Activity处于启动状态，但Service组件却有两种状态：启动状态和绑定状态，区别在于绑定状态可以和外界进行通信。

1. Service能否开启多线程

同Activity。

1. Service重要知识点总结

https://blog.csdn.net/imxiangzi/article/details/76039978

## 3、BroadcastReceiver

### 3.1.BroadCastReceiver的作用

BroadcastReceiver是一种消息型组件，用于不同组件乃至不同应用间的消息传递，它同样无法被用户感知。

BroadcastReceiver也叫广播，广播的注册方式有两种：静态注册和动态注册。静态注册指在AndroidManifest中注册，这种广播在应用安装时被系统解析，此种形式的广播不需要应用启动就可以接收相应的广播。动态广播需要通过Context.registerReceiver()来实现，并且在不需要的时候通过Context.UnregisterReceiver()解除广播，此种形态的广播要在应用启动后才能注册和接收广播。

它的主要作用是消息的传递，该消息的传递可以在应用内，也可以在应用之间。

### 3.2 BroadcastReceiver能否开启多进程

同Activity。

### 3.3多端调用BroadcastReceiver的例子

发送端

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Intent intent = new Intent("com.test.abc");

intent.putExtra("msg", "hello world");

sendBroadcast(intent, "com.test.mypermission");

}}

发送端代码挺简单的，就使用sendBroadcast发送一个Intent对象，其中的Intent对象的action名为“com.test.abc”，并且携带了key为“msg”，value为“hello world”的信息。注意sendBroadcast方法的第二个参数为"com.test.mypermission"，其实第二个参数代表的是permission，也就是该广播具有permission权限，当接收端的Receiver具有permission权限时，才能接收到该广播。

接收端代码：//自定义的一个广播接收器

public class MyBroadcastReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

String msg = intent.getStringExtra("msg");

Log.d("ABC", "msg from remote:" + msg);

}

}

// AndroidManifest：

<permission android:name="com.test.mypermission"/> <!--声明自定义权限-->

<uses-permission android:name="com.test.mypermission"/> <!--引用自己声明的权限-->

<receiver

android:name=".MyBroadcastReceiver"

android:exported="true">

<intent-filter>

<action android:name="com.test.abc"/>

</intent-filter>

</receiver>

接收端的代码逻辑分为三部分：  
1、首先自定义一个广播接收器，在onReceiver方法中接收发送端发过来的请求信息。  
2、然后我们需要在AndroidManifest中注册该广播接收器，由于发送端和接收端属于两个不同的应用，所以需要声明android:exported=true，并且添加action标签，表明只接收action为"com.test.abc"的广播。  
3、最后由于发送端发送广播时定义了一个permission权限，所以我们需要用<permission />标签声明该权限，并且用<uses-permisssion />标签引用该权限。

## 4、ContentProvider

### 4.1ContentProvider的作用

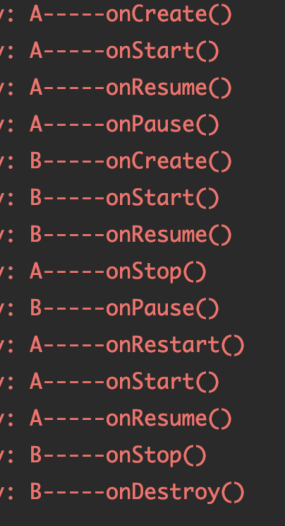
ContentProvider是一种数据共享型组件，用于向其他组件乃至于其他应用共享数据，同样的它也无法被外界感知。它内部需要实现增删改查四种操作。需要注意的是它内部的insert、delete、update和query方法需要处理好线程同步，因为这几个方法都是在Binder线程池中被调用的。ContentProvider不需要手动停止。

ContentProvider的主要作用是作为一个平台，提供数据共享。

### 4.2ContentProvider是否可以开启多线程

ContentProvider可以通过在AndroidManifest中通过添加android:process的属性来开启多进程。若该ContentProvider需要被外界调用，则需要设置android:exported=true这个属性。当android sdk的minSdkVersion或者targetSdkVersion为16或者以下时，android:exported默认是true的。当android sdk的minSdkVersion或者targetSdkVersion为17或者以上时，android:exported默认是false。设置了android:exported=true，同时也可以设置android:permission来限制外部应用调用。

### 问题2：ActivityA启动Activity B，再退出B，他们的生命周期？

答：

### 问题3：MainActivity的启动方式，为什么要这样设计？

答：隐式启动

### 问题4：Service启动方式

答：<https://blog.csdn.net/imxiangzi/article/details/76039978>

### 问题5：LeakCanary原理

答：https://www.jianshu.com/p/bcaab8f0f280