android四大组件

1. **Activity：一个Acticity是一个应用程序组件，提供一个界面（屏幕）。**

它是 android 应用程序的基本功能单元。Activity 本身是没有界面的。所以activity类创建了一个窗口，开发人员可以通过setContentView(View)接口把UI放到activity创建的窗口上，当activity指向全屏窗口时，activity是单独的，用于处理用户操作。

一个应用程序中有多个Activity，这些Activity组成了Activity栈（Stack）,当前活动的Activity位于栈顶，之前的Activity被压入下面，成为非活动Activity，等待是否可能被恢复为活动状态。

Activity的四个重要状态：

1. 活动状态：当前的Activity，位于Activity栈顶，用户可见，并且可以获得焦点。
2. 暂停状态：失去焦点的Activity，仍然可见，但是在内存低的情况下，不能被系统的killed（杀死）。
3. 停止状态：该Activity被其他Activity所覆盖，不可见，但是它仍然保持所有的状态和信息，当内存低的情况下，它将要被系统skill（杀死）。
4. 销毁状态：该Activity结束，或Activity所在的**Dalvik**进程结束。
   1. **Activity的生命周期**

包括：

**onCreate()：**创建Activity时调用。

**onStart()：**启动Activity是被调用，也就是当一个Activity变为显示时被调用。

**onRestart()：**重新启动Activity时被调用，该方法总时在onStart()方法以后执行。

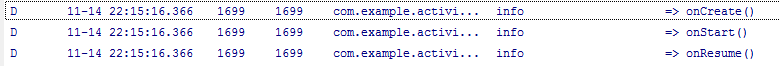
**onPause()：**暂停Activity时被调用（该方法需要被执行的非常迅速，因为直到该方法执行完毕后，下一个Activity才能被恢复），在次方法中通常用于持久保存数据。

**onResume()：**当Activity由暂停状态恢复为活动状态时被调用。调用该方法后，该Activity位于栈顶。该方法总是在onPause()之后执行（这句话要理解为必须是Activity初次被创建的情况除外）。

**onStop()：**停止Activity时被回调。

**onDestory()：**销毁Activity时被调用。

打开一个应用（显示）：



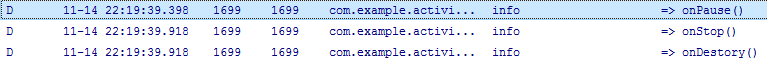
显示其它应用（不可见）：



恢复显示：



退出该应用：



(1) 一个Acticity的正常启动过程：onCreate创建一个Activity ——> onStart() ——> onResume()；当Activity被kill掉时调用的方法顺序是：onPause() ——> onStop() ——> onDestory().以上为一个完整的生命周期。

(2) 当Other applications need memory(中断)时，则是：onPause() ——> onStop()，恢复时：onRestart() ——> onStart() ——> onResume();



* 1. **启动Activity和关闭Activity**

**startActivity（Intent intent）；//启动Activity**

在Android中启动Activity的标准Activity动作最常见的：ACTION\_MAIN和ACTION\_EDIT。

-》在Intent对象中的动作使用setAction()方法设置，使用getAction()方法获取。

**finish()；//关闭Activity**

* 1. **横竖屏之间变换时Activity的变化（生命周期）**

原理就是在横屏变为竖屏时，先销毁横屏在重新创建一个合适的竖屏Activity，所以需要设置2个layout.xml布局文件。

1. **用Intent在Activity之间的通信（传数据）：Intent是系统各组件之间数据传递的负载者**

Android中通过Intent对象来表示一条信息，Intent主要负责对操作的动作、动作涉及的数据、附件数据进行描述（装起来），Android则根据Intent

的描述，负责去找对对应的组件，将Intent传递给调用的组件，并完成组件之间的调用。

所以在用Intent传递信息时，Intent是作为一个“中介”作用，专门提供组件互相调用的相关信息，实现调用者和被调用者之间的解耦。

//将数据存入到Intent对象中

Intent intent = new Intent();

intent.putExtra(String key, XXX Value);

//获取Intent中的数据

Intent intent = getIntent();

XXX Value = intent.getXXXExtra(String key);//XXX表示传入数据的类型（int，String等等）

**2.1. Intent有2种形式**：显式Intent和隐式Intent（说白就是一个在代码文件中直接写出，另一个则是在xml配置文件中用action,category,intent-filter等）

**2.1.1、显式Intent调用方法（有以下几种方式）：通过组件名称来指定目标组件**

Intent intent = new Intent(this,otherAcitvity.class);//方法1

Intent intent2 = new Intent();

intent2.setClass(this,otherAcitvity.class);//方法2

intent2.setClassName(this,"<包名>.otherAcitvity")//方法3

intent2.setComponent(new ComponentName(this,otherAcitvity.class);//方法4

startActivity(intent2);//调用

**2.1.2、 隐式**Intent调用方法:在代码中只有与<action><category>中的内容同时能够匹配上Intent中指定的action和categary时，这个activity才会响应。通常用于激活其他应用程序中的组件。

<acticity>

<intent-filter>//如果没有此过滤器，组件等只能接受显示Intent；若有过滤器，都可以接受

<action .../>

<category .../>

...

</intent-filter>

</acticity>

**2.1.3、以上2种方**式都是直接传输数据过去，但是不需要有返回数据，

当需要将数据返回时，第一个Activity传输数据用startActivityForResult(Intent,RequestCode).跳转到下一个Activity，其中第一个参数为传入的意图对象，第二个为设置的请求码;

跳转到第二个Activity后，调用setResult(ResultCode, intent)方法可返回上一个Activity，其中第一个参数为结果码（为了让程序知道返回的数据是来自哪一个Activity），第二个为传入的意图对象；

在第一个Activity通过onActivityResult()方法获得返回的数据。

！！！注意：在setResult后，要调用finish()销毁当前的Activity，否则无法返回到原来的Activity，就无法执行原来Activity的onActivityResult函数，看到当前的Activity没反应。

1. **用Bundle在Activity之间交换数据**

Bundle传递数据和Intent传递数据在本质上没有什么区别。在Android中，可以将要保存的数据存放在Bundle对象中，然后通过Intent提供的putExtras()方法将要携带的数据保存到Intent中。

/\*\*\*\*\*\*\*\*将数据存入Bundle \*\*\*\*\*\*/

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

**Bundle bundle = new Bundle();**

**bundle.putCharSequence(“user”, user);**

**….**

**intent.putExtras(bundle);//将数据添加的Bundle中**

**startActivity(intent);**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*/**

/\*\*\*\*\*\*\*\*将数据从Bundle中取出来 \*\*\*\*\*\*/

**Intent intent = getIntent();//获取Intent对象**

**Bundle bundle = intent.getExtras();//获取传递的数据包**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

1. **Android用栈管理Activity**
2. **Activity的4种加载方式**

4.1 standard （默认模式）：就是intent将发送给新的实例，所以每次跳转都生成一个新的activity。

4.2 singleTop :也是发送新的实例，但不同与standard，在请求的activity正好处于栈顶时，则不会创建新的实例。

4.3 singleTask ：只创建一个实例，当Intent到来，需要创建设置为singleTask的Activity时，系统会检测栈里是否有该Activity的实例，若有则直接将Intent发送给它。

4.4 singleInstance : 只创建一个实例，singleInstance模式就是将该Activity单独放入一个栈中，这样这个栈中只有这一个Activity，不同应用的intent都由这个Activity接收和展示。

**“拿来主义”standard模式**。哪里需要调用我我就去哪里，可以多次实例化，可以几个相同的Activity重叠。

 **“拒绝堆叠”singleTop模式**。可以多次实例化，但是不可以多个相同的Activity重叠，当堆栈的顶部为相同的Activity时，会调用onNewIntent函数。

**“独立门户”singleTask模式**。同一个应用中调用该Activity时，如果该Activity没有被实例化，会在本应用程序的Task内实例 化，如果已经实例化，会将Task中其上的Activity销毁后，调用onNewIntent；其它应用程序调用该Activity时，如果该 Activity没有被实例化，会创建新的Task并实例化后入栈，如果已经实例化，会销毁其上的Activity，并调用onNewIntent。一句 话，singleTask就是“独立门户”，在自己的Task里，并且启动时不允许其他Activity凌驾于自己之上。

**“孤独寂寞”singleInstance模式**。加载该Activity时如果没有实例化，他会创建新的Task后，实例化入栈，如果已经存在，直接调用 onNewIntent，该Activity的Task中不允许启动其它的Activity，任何从该Activity启动的其他Activity都将被 放到其他task中，先检查是否有本应用的task，没有的话就创建。

/\*\*\*onCreate()\*\*\*/

(1),setContentView(R.layout.main);

(2),LayoutInflater inflater = LayoutInflater from(this);

View view = inflater.inflate(R.layout.main);

setContentView(view);

以上2者的区别为：(1)一旦调用，layout就会显示UI，而(2)只会把layout形成一个View类实现成的对象，有需要时才会再用setContentView(view)显示。