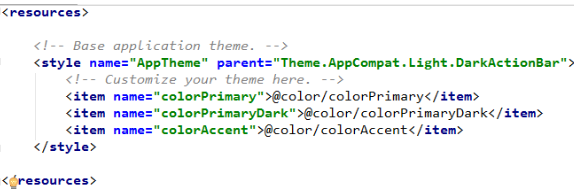
PPT：

习题：

1、下面的对自定 style 的方式正确的是（ ）





2、onSaveInstanceState()   
当你的程序中某一个Activity A在运行时，主动或被动地运行另一个新的Activity B，这个时候A会执行onSaveInstanceState()。B完成以后又会来找A，这个时候就有两种情况：一是A被回收，二是A没有被回收，被回收的A就要重新调用onCreate()方法，不同于直接启动的是这回onCreate()里是带上了参数savedInstanceState；而没被收回的就直接执行onResume()，跳过onCreate()了。

3、显式启动 是明确指定了需要启动的Activity 或 service 的类名或包名。

隐式启动 不明确制定需要启动哪个Activity，而是通过设置action、data、 Category 等让系统来匹配出合适的目标

4下列不属于 service 生命周期的方法是 \_\_\_\_

由于Android Service的生命周期并不像Activity那么复杂，它只继承了onCreate(),onStart()（onStartCpmmand()）,onDestroy()三个方法，当我们第一次启动Service时，先后调用了onCreate(),onStart()这两个方法，当停止Service时，则执行onDestroy()方法，需要注意的是，如果Service已经启动了，当我们再次启动Service时，不会在执行onCreate()方法，而是直接执行onStart()方法。

5、Android中MVC模式 C层指的是？

M是逻辑模型

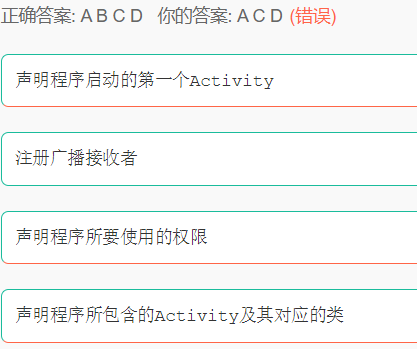
V是视图模型，对应于android里面的View

c是控制器，android中的activity实现了这个职责

6、在滴滴打车点击到支付宝支付，出现密码输入框，到此时相关的Activity会发生的生命周期回调依次为（ ）

onPause->onCreate->onStart->onResume

7、下列哪一项是AndroidManifest 文件的作用（ ）









handler是跨线程！

四种方式：

1.**访问其他应用程序的Activity**

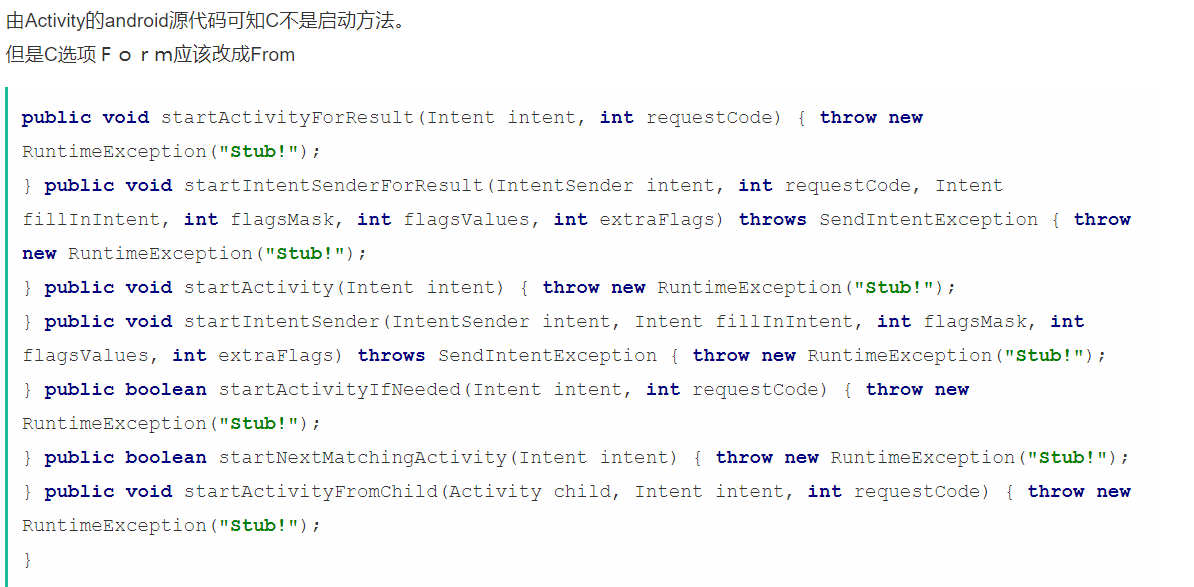
**2.Content Provider**

**3.广播（Broadcast）**

**4.AIDL服务**

11、

Activity启动方法







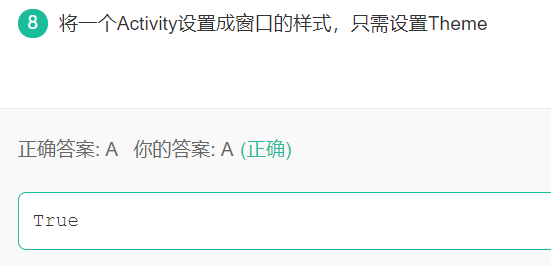
Android中不允许开启一个新新线程更新UI，handler属于创建它的线程，所以BD错误

C Handler采用的是消息队列的方式，每一个Handler都会有一个与之相关联的消息队列，而且都是以先进先出的方式执行

下面关于andriod dvm的进程和Linux的进程，应用程序的进程说法正确的是



ndroid系统用的就是Linux的内核，DVM可以理解成在Linux上跑的程序，每个应用程序运行时，都会启动一个DVM实例，也就是一个Linux的进程，因此DVM进程和Linux进程实际上是一回事。

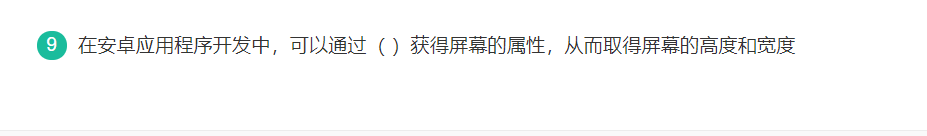


 将一个Activity设置成窗口的样式有两种方式

  AndroidManifest.xml 中加上：

（1）android:theme=@android:style/Theme.Dialog

（2）android:theme=@android:style/Theme.Translucent就变成半透明的



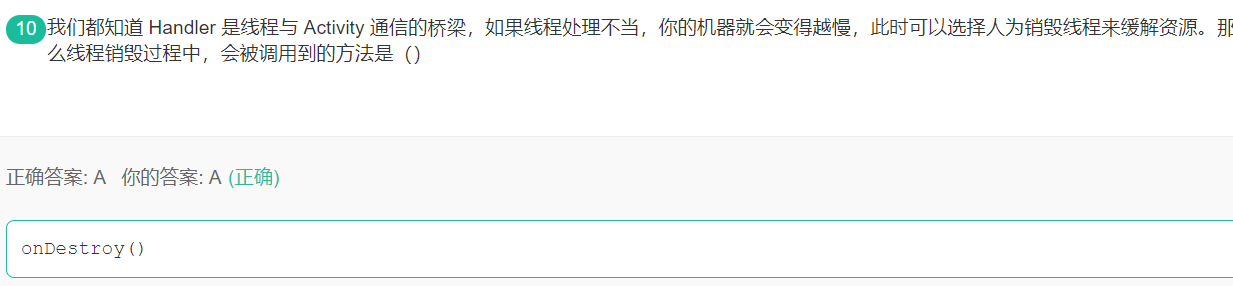
获取屏幕密度  getScreen

获取设备屏幕大小 getDisplay

获得屏幕的属性，从而取得屏幕的高度和宽度 getMetrics

获得window对象 getWindows

在Activity中创建的线程，在app退出时如果没有结束掉线程，那么每次打开app，都会创建一个新的线程，导致线程越来越多。因此，需要在app退出的时候，销毁掉创建的线程。问题是，应该在Activity中的哪个方法中操作、销毁。

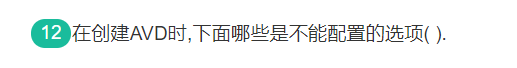


Parcelable和Serializable 俩者异同

1、Serializable在序列化的时候会产生大量的临时变量，从而引起频繁的GC；

2、在使用内存的时候，Parcelable比Serializable性能高，所以推荐使用Parcelable。

3、Parcelable不能使用在要将数据存储在磁盘上的情况，因为Parcelable不能很好的保证数据的持续性在外界有变化的情况下。尽管Serializable效率低点，但此时还是建议使用Serializable 。





ixiaomo

Uri的使用场景：

1，调web浏览器

Uri myBlogUri = Uri.parse(" http://xxxxx.com ");

returnIt = new Intent(Intent.ACTION\_VIEW, myBlogUri);

2，地图

Uri mapUri = Uri.parse("geo:38.899533,-77.036476");

returnIt = new Intent(Intent.ACTION\_VIEW, mapUri);

3，调拨打电话界面

Uri telUri = Uri.parse("tel:100861");

returnIt = new Intent(Intent.ACTION\_DIAL, telUri);

4，直接拨打电话

Uri callUri = Uri.parse("tel:100861");

returnIt = new Intent(Intent.ACTION\_CALL, callUri);

5，卸载

Uri uninstallUri = Uri.fromParts("package", "xxx", null);

returnIt = new Intent(Intent.ACTION\_DELETE, uninstallUri);

6，安装

Uri installUri = Uri.fromParts("package", "xxx", null);

returnIt = new Intent(Intent.ACTION\_PACKAGE\_ADDED, installUri);

7，播放

Uri playUri = Uri.parse("file:///sdcard/download/everything.mp3");

returnIt = new Intent(Intent.ACTION\_VIEW, playUri);

8，调用发邮件

Uri emailUri = Uri.parse("mailto:xxxx@gmail.com");

returnIt = new Intent(Intent.ACTION\_SENDTO, emailUri);

9，发邮件

returnIt = new Intent(Intent.ACTION\_SEND);

String[] tos = { "xxxx@gmail.com" };

String[] ccs = { "xxxx@gmail.com" };

returnIt.putExtra(Intent.EXTRA\_EMAIL, tos);

returnIt.putExtra(Intent.EXTRA\_CC, ccs);

returnIt.putExtra(Intent.EXTRA\_TEXT, "body");

returnIt.putExtra(Intent.EXTRA\_SUBJECT, "subject");

returnIt.setType("message/rfc882");

Intent.createChooser(returnIt, "Choose Email Client");

10，发短信

Uri smsUri = Uri.parse("tel:100861");

returnIt = new Intent(Intent.ACTION\_VIEW, smsUri);

returnIt.putExtra("sms\_body", "yyyy");

returnIt.setType("vnd.android-dir/mms-sms");

11，直接发邮件

Uri smsToUri = Uri.parse("smsto://100861");

returnIt = new Intent(Intent.ACTION\_SENDTO, smsToUri);

returnIt.putExtra("sms\_body", "yyyy");

12，发彩信

Uri mmsUri = Uri.parse("content://media/external/images/media/23");

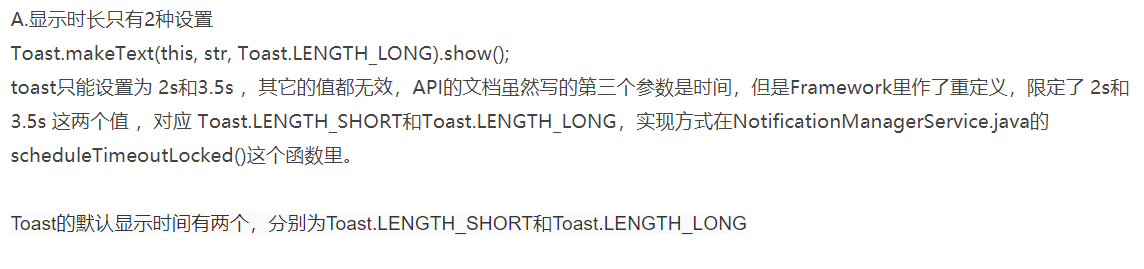
returnIt = new Intent(Intent.ACTION\_SEND);

returnIt.putExtra("sms\_body", "yyyy");

returnIt.putExtra(Intent.EXTRA\_STREAM, mmsUri);

returnIt.setType("image/png");



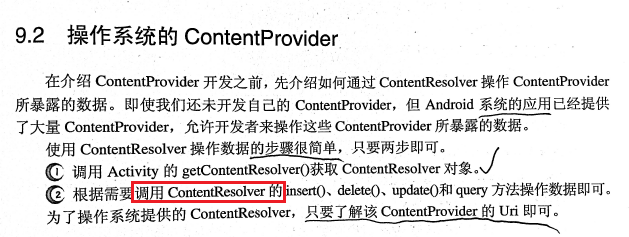




个人觉得上边说的多少都有问题，我来总结吧：

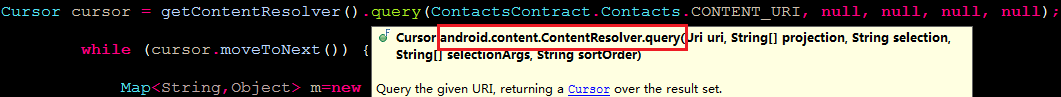
1、无论是ContentProvider还是ContentResolver都有各自的insert(),delte(),uapdte(),query(),四个方法。

2、使用ContentResolver是用来操作数据的。贴一张李刚的《疯狂Android讲义》第9章9.2开头对这部分ContentResolver的原话.



注意红色框起来的部分，调用的是**ContentResolver的的的**

3、再不行上一小段代码证明吧：我们在avtivity的onCreate()里边写一句代码就可以。把鼠标放在query()函数上，可以看到解释的是ContentResolver的query(),不是ContentProvider的



阅读代码回答运行结果

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | public classMainActivity extends Activity implements OnClickListener  {     private Button mBtnLogin = (Button) findViewById(R.id.btn\_login);     private TextView mTextViewUser;       @Override     protected void onCreate(BundlesavedInstanceState)     {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);          mTextViewUser = (TextView) findViewById(R.id.textview\_user);          mBtnLogin.setOnClickListener(this);          new Thread()          {              @Override              public void run()              {                  mTextViewUser.setText(10);              }          }.start();     }       @Override     public void onClick(View v)     {          mTextViewUser.setText(20);     }  } |

代码实测：

1、首先会报错NullPointerException，就是privateButton mBtnLogin = (Button) findViewById(R.id.btn\_login);这个位置，要先加载了layout后才能成功获取到相应的按钮组件对象；

2、修改NullPointerException错误后再运行，报错 Resources$NotFoundException，在mTextViewUser.setText(10);这个位置（原本以为会先检查onclick方法里的setText（），但实际是run（）里的setText（）），要改成字符串形式；

3、修改上面的错误后再运行，报错Resources$NotFoundException，这次就轮到mTextViewUser.setText(20);这个位置了；

4、修改上面的错误后再运行，没有报错，程序成功运行，点击按钮后TextView由10变为20，说好的不能在非UI线程里更新UI组件呢？翻看别人的博客后，终于找到答案了，其实非UI线程是可以刷新UI的，前提是它要拥有自己的ViewRoot，ViewRoot是在onResume（）里addview（）创建的，所以是在 onResume（）检查是否为UI线程，一般在onCreate（）中通过子线程可以更新UI，但官方不建议这样做，因为 Android UI操作并不是线程安全的。

PS：而且，可以试下在上面代码的run（）中setText（）前加一句Thread.sleep(2000)，先让线程休眠个2到3秒，就会报错ViewRootImpl$CalledFromWrongThreadException，说明已经检查到在非UI线程里更新UI。

Android数据持久化有五种方式：

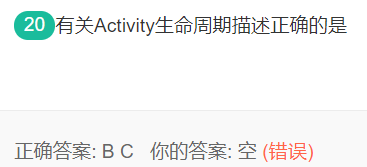
1、SharedPreferences

2、内部存储（例如通过openFileOutput()打开一个文件输入输出流）

3、SQLite Database

4、网络连接（将数据存储到服务器上）

5、外部存储（SD卡）



**A 设置Activity的android:screenOrientation="portrait"属性时,无法切换横竖屏，因此不但不会重新调用各个生命周期方法，而且onConfigurationChanged()方法也不会执行。**

**B 未设置Activity的android:configChanges属性，API上这样说"the activity will be restarted if any of these configuration changes happen in the system.";如果配置有改变，就会重启activity**

**C launchMode为singleTask的时候，通过Intent启到一个Activity, 如果系统已经存在一个实例，系统就会将请求发送到这个实例上，但这个时候，系统就不会再调用通常情况下我们处理请求数据的onCreate方法，而是调用onNewIntent方法**

**D 用户正在操作某个Activity，这时如果其他应用程序需要内存。 此时的Activity是Foreground process,应该按照Empty process，Background process，Service process，Visible process顺序kill，最后才是前台进程。**

Android5.0之前，当启动一个Activity时，系统将首先检查Activity的launchMode，如果为A页面设置为SingleInstance或者B页面设置为singleTask或者singleInstance，则会在LaunchFlags中加入FLAG\_ACTIVITY\_NEW\_TASK标志，而如果含有FLAG\_ACTIVITY\_NEW\_TASK标志的话，onActivityResult将会立即接收到一个cancle的信息。

在Android5.0做出来修改，系统即便启动的页面设置launchMode为singleTask或singleInstance，onActivityResult依旧可以正常工作。

AlphaAnimation

渐变透明度动画效果

ScaleAnimation

渐变尺寸伸缩动画效果

TranslateAnimation

画面转换位置移动动画效果

RotateAnimation

画面转移旋转动画效果

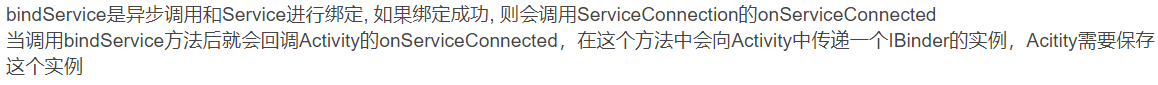
Toast.makeText(this, str, Toast.LENGTH\_LONG).show();

toast只能设置为 2s和3.5s ，其它的值都无效，API的文档虽然写的第三个参数是时间，但是Framework里作了重定义，限定了 2s和3.5s 这两个值 ，对应 Toast.LENGTH\_SHORT和Toast.LENGTH\_LONG，实现方式在NotificationManagerService.java的scheduleTimeoutLocked()这个函数里。

Toast的默认显示时间有两个，分别为Toast.LENGTH\_SHORT和Toast.LENGTH\_LONG









B:

AlertDialog的构造方法被声明为protected

，所以不能直接使用new关键字来创建AlertDialog类的对象实例。要想创建AlertDialog对话框，需要使用Builder类，该类是AlertDialog类中定义的一个内嵌类。因此必须创建AlertDialog.Builder类的对象实例，然后再调用show()来显示对话框。

例如：

       AlertDialog.Builder db= new Builder(this);

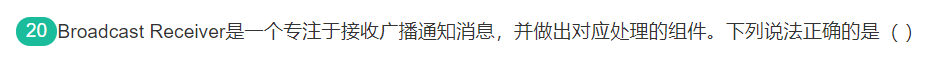
                    db..create().show();

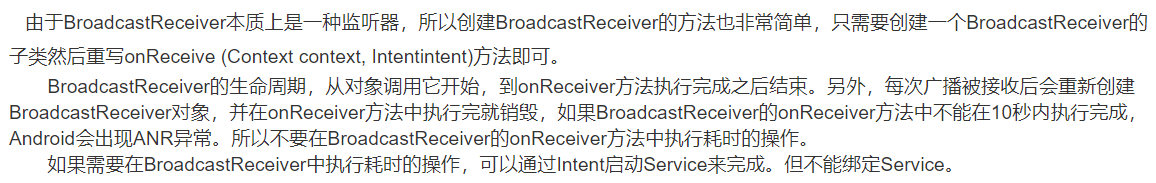
-------------------------------------

C:

|  |  |
| --- | --- |
| 1  2 | ProgressDialog pDialog;//进度条对话框对象  pDialog = new ProgressDialog(Context); //构造进度条对话框 |







IntentFilter与隐式Intent  
android系统处理隐式Intent时, 会比较Intent和IntentFilter的action, data, category属性, 如果**以上3个属性全都相符的话**, 则IntentFilter所属的component就可以作为目标组件的**候选**(存在多个符合条件的component时).   
1. 测试action属性. intent最多只能定义1个action, 而filter可以定义1个或多个action.   
通过action测试的条件为:**filter定义了intent的action**. 例如intent的action为"android.intent.action.MAIN", 则定义了"android.intent.action.MAIN"这个action的filter都能通过action测试(当然, filter还可以包含更多额外的action).  
如果filter没有定义action, 则这个filter将阻塞所有intent. 如果intent没有定义action, 那么只要filter定义了action就可以通过action测试.   
2. 测试category属性. intent可以任意多个category, filter也可以任意个category. 通过category测试的条件为: **filter定义了intent的所有category**. 例如intent定义了"android.intent.category.DEFAULT"和"cn.xing.intent.category.UPLOAD"这2个category, 则定义了以上2个category属性的filter才能通过测试(当然, filter还可以包含更多额外的category).  
根据上面的规则, **如果一个intent没有定义category, 则所有filter都可以通过category测试**. 但是有一种例外: 以startActivity(intent)方式启动一个activity时, 系统为会intent增加一个值为"android.intent.category.DEFAULT"的category, 这就意味着每一个期望通过category测试的activity, 都要在其filter中定义"android.intent.category.DEFAULT"(除了作为程序入口的activity).  
3. 测试data属性. intent最多只能定义1个data, filter则可以定义多个data.  
通过data测试的条件为:  
a. 如果intent没有指定data和data type, 则只有没有定义data和date type的filter才能通过测试;  
b. 如果intent定义了data没有定义data type, 则只有定义了相同data且没有定义date type的filter才能通过测试;  
c. 如果intent没有定义data却定义了data type, 则只有未定义data且定义了相同的data type的filter才能通过测试;  
d. 如果intent既定义了data也定义了data type, 则只有定义了相同的data和data type的filter才能通过测试.  
data属性是一个URI, URI中包含scheme, host, post和path, 典型的URI为:  
scheme://host:port/path  
scheme, host, post和path都是可选的. 比较2个data时, 只比较filter中包含的部分. 比如filter的一个data只是指定了scheme部分, 则测试时只是比较data的scheme部分, 只要两者的scheme部分相同, 就视为"相同的data".